Asiorowi

45 lat WNT

Wydawnictwa Naukowo-Techniczne
Warszawa

RYSZARD TADEUSIEWICZ
PIOTR CHRZĄSTOWSKI

INFORMATYKA?

Wydanie drugie

ależ to bardzo proste!

Opiniodawca i redaktor naukowy *dr Leszek Rudak*
Skład komputerowy *Ryszard Tadeusiewicz*
Projekt okładki i stron tytułowych *Wojciech J. Steifer*
Okładkę przygotowano do druku na komputerze firmy OPTIMUS SA

681.3

Książka jest bogato ilustrowanym albumem, uczącym w sposób poglądowy czym jest komputer i jak go używać.
Oferuje w atrakcyjnej i ciekawej formie minimum niezbędnej wiedzy o komputerach, programach i zastosowaniach informatyki. Wszystkie wiadomości są zilustrowane kolorowymi fotografiami (obrazami, jakie użytkownik widzi na ekranie podczas pracy z opisywanymi programami), co ułatwia wiązanie treści książki z codzienną praktyką.
Książka, choć w zasadzie przeznaczona dla „nowicjuszy", którzy znaleźli się w przymusowej sytuacji pracy z komputerem, może też być literaturą uzupełniającą dla zaawansowanych użytkowników komputerów i słuchaczy wykładu „wstęp do informatyki".

Książka wydana staraniem Fundacji „Książka Naukowo-Techniczna"
z pomocą finansową Komitetu Badań Naukowych

ISBN 83-204-1766-X

WNT. Wydanie II. Symbol Mk/83091/WNT
Ark. wyd. 16,5. Ark. druk. 20,0
Cieszyńska Drukarnia Wydawnicza

Spis treści

1. Czy lubisz komputery?

Jeśli **tak**, to powinieneś przeczytać tę książkę, ponieważ dowiesz się z niej wielu ciekawych rzeczy na temat tych wspaniałych maszyn, które tak bardzo zmieniły świat i mają szansę zmienić go jeszcze bardziej.

Jeśli jednak nie lubisz komputerów, jeśli napełniają cię lękiem, bo ich nie rozumiesz, jeśli obawiasz się utraty pracy, ponieważ nie potrafisz znaleźć wspólnego języka z maszyną, która nieuchronnie zjawi się wkrótce także w Twojej firmie – PRZECZYTAJ TĘ KSIĄŻKĘ KONIECZNIE! Przekonasz się, że "nie taki diabeł straszny". Informatyka jest w istocie bardzo prosta, komputerem potrafi się posługiwać nawet dziecko, a poznawszy podstawowe zasady można się dogadać z **każdym** komputerem i na **każdy** temat.

Oczywiście, **są** w informatyce zagadnienia trudne. Nikt nie twierdzi, że – na przykład – budowa komputera jest prosta. Jednak wcale nie trzeba wiele o tym wiedzieć, by skutecznie używać komputera. Czy musisz znać budowę telewizora, żeby obejrzeć film? Wystarczy, że masz podstawowe wiadomości: jak włączyć aparat, gdzie wetknąć antenę, jak wybrać potrzebny kanał – to wszystko. Z komputerem jest podobnie – możesz wiedzieć o nim bardzo mało i już z dużym pożytkiem korzystać z jego pracy, chociaż im więcej się o nim dowiesz, tym skuteczniej będziesz mógł mu stawiać zadania. W tej książce znajdziesz właśnie to **minimum** wiadomości na temat informatyki. Nie przejmuj się, jeśli natkniesz się w tekście na jakieś niezrozumiałe słowo (np. **bit). Wszystkie** fachowe terminy **będą** w książce objaśnione, tylko nie zawsze da się to zrobić od razu, prawda?

Przyglądaj się pilnie **wszystkim** fotografiom. Pokazują one zawsze w sposób konkretny to, o czym napisano w sąsiadującym z fotografią tekście i są równie ważne jak tekst (a czasem nawet ważniejsze!). Dlatego najszybciej dowiesz się **podstawowych** rzeczy, jeśli po prostu **obejrzysz obrazki** i potraktujesz sąsiedni tekst jako podpisy do tych obrazków. Natomiast mając mało czasu nie zwracaj zupełnie uwagi na przypisy u dołu stronicy – są tam zawarte głównie ciekawostki lub uzupełnienia zasadniczego tekstu, z których naprawdę możesz na początku bez szkody zrezygnować.

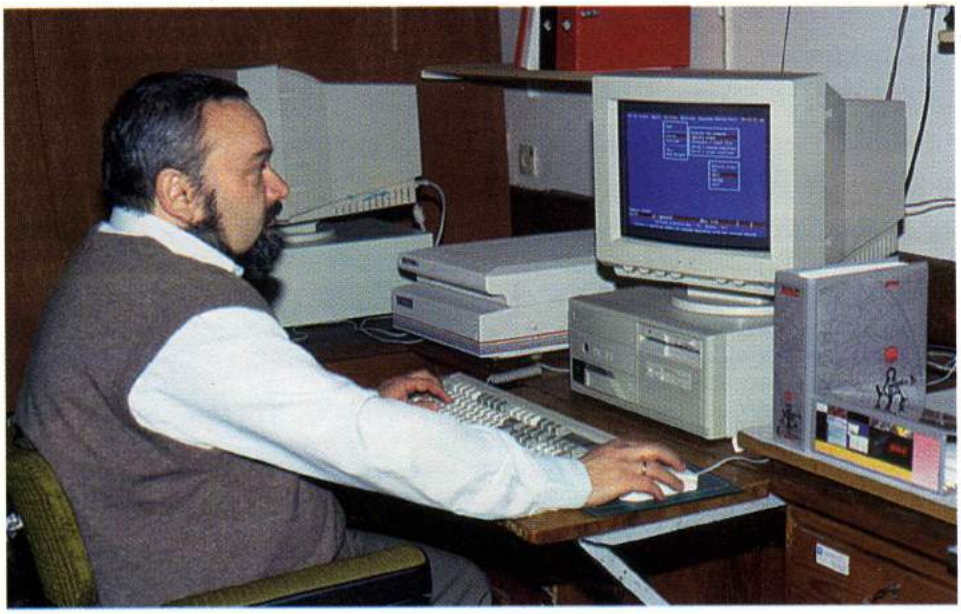
Informatyka rozwija się bardzo szybko, dlatego jeśli zdołamy Cię przekonać, że jest ona dziedziną ciekawą, nie nazbyt trudną i wartą wysiłku – powinieneś uzupełniać i aktualizować podane tu wiadomości, czytając specjalistyczne pisma (*PC–Kurier, Komputer, Enter* i wiele innych). Nawyk stałego śledzenia nowości musi być podstawowym odruchem każdego informatyka – jeśli oczywiście nie zamierza specjalizować się w historii informatyki. **Żadna książka nie jest w stanie dostarczyć wiedzy, która wystarczy na całe życie!**

Kraków, październik 1993.

2. Co jest w środku?

2.1. Uwagi wstępne

Celem tej książki jest dostarczenie Ci tak dobranego minimum wiadomości, abyś potrafił **stosować** metody i środki techniki obliczeniowej w rozwiązywaniu zagadnień praktycznych. Oczywiście w tej sytuacji najważniejsze jest, by umiejętnie i sprawnie **używać** komputera. Nie sądzimy, aby w Twojej praktyce zawodowej zachodziła potrzeba samodzielnego naprawiania komputerów, a tym bardziej nie przewidujemy, abyś musiał własnoręcznie **budować** te maszyny. W związku z tym w tej książce dominować będą oczywiście zagadnienia związane z zastosowaniami komputerów, poszerzone o pewne wiadomości dotyczące ich programowania.



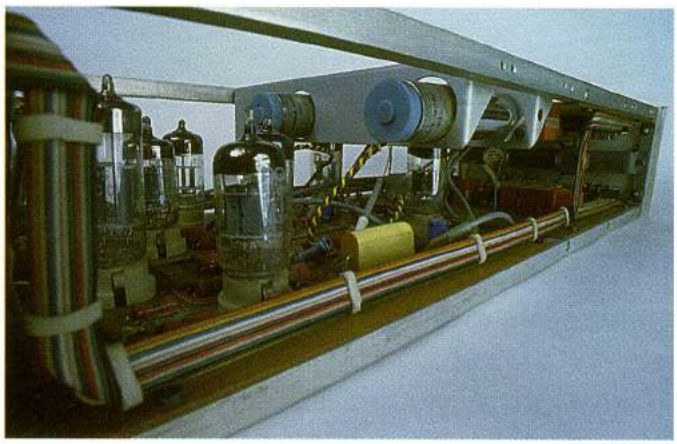
Jednak od prawdziwego **specjalisty** zajmującego się tak nowoczesną dziedziną, jak informatyka – można wymagać, by miał przynajmniej **ogólne** pojęcie o tym, czym jest komputer[1] i dlaczego działa. W końcu zanim zaczniesz stosować komputer – musisz go fizycznie **mieć**, a także odróżnić ten komputer od stołu, na którym go postawiono ...

Podane niżej wiadomości musisz traktować jako wstępne, wręcz encyklopedyczne ujęcie rozważanego zagadnienia, jednak nawet taka powierzchowna wiedza znacznie ułatwi "obcowanie z komputerem" oraz – co ważniejsze – umożliwi Ci bieżące uzupełnianie swojej wiedzy poprzez swobodne studiowanie specjalistycznych periodyków i literatury. Niesłychanie szybki postęp dokonujący się w informatyce zmusza bowiem do traktowania podanych tu wiadomości jedynie jako **elementarnego wprowadzenia.**

---

[1] Środków techniki obliczeniowej nie musi się utożsamiać **wyłącznie** z elektronicznymi komputerami. Od wieków ludzie budowali rozmaite maszyny, by ułatwić i przyspieszyć procesy obliczeniowe. Prawdopodobnie najstarszą maszyną matematyczną jest chińskie liczydło nazywane **Abakus** lub **Suan-Pan.** Potwierdzone historycznie jest używanie tej maszyny już w 25. stuleciu przed naszą erą. Urządzenia te były z powodzeniem stosowane w obsłudze obliczeniowej handlu na Dalekim Wschodzie, a w Chinach liczydła są stosowane z powodzeniem do dziś – między innymi w sklepach sprzedających ... komputery (w Singapurze).

Skalę rozwoju techniki komputerowej najłatwiej prześledzić odwołując się do pojęcia tak zwanych **generacji maszyn cyfrowych**. Rozwój sprzętu komputerowego zachodzi oczywiście w sposób nieprzerwany i ciągły[1]. Od czasu do czasu jednak pojawiają się w tym rozwoju innowacje na tyle znaczące, że powszechnie uważa się je za graniczne dla pewnych etapów rozwoju sprzętu, zapoczątkowując nowe generacje. Granice czasowe poszczególnych generacji są dość niepewne, ponieważ regułą było długotrwałe współistnienie maszyn należących do różnych generacji. Nowo konstruowane komputery bowiem wolno i z oporami wypierają starsze, ale znane użytkownikom, dobrze oprogramowane i z powodzeniem stosowane komputery wcześniejszych generacji. Niemniej z grubsza można zarysować następujące granice.



Za początek **pierwszej generacji** komputerów uznać można rok **1944,** ponieważ pierwszym szeroko znanym komputerem (w nowoczesnym znaczeniu tego słowa[2]) był **Mark I,** zbudowany w latach **1939 – 1944** przez **Howarda Aikena** na Uniwersytecie Harvarda. Mark I był ogromną maszyną: długą na 17 metrów, na 2 metry wysoką i na metr szeroką. Zawierał 3000 przełączników (mechanicznych), 750.000 lamp elektronowych i 800 kilometrów przewodów. Pracował w systemie dziesiętnym (a nie dwójkowo!) z dokładnością do 23 cyfr znaczących. Wykonywał do 3 dodawań w ciągu sekundy, potrzebował 6 sekund do wykonania mnożenia i 12 do dzielenia. Warto podkreślić, że był on zadowalająco eksploatowany przez ponad 10 lat.

Druga generacja powstała w latach pięćdziesiątych wraz z masowym rozwojem półprzewodników i elektronicznych **układów tranzystorowych**, chociaż my jeszcze w 1970 roku pracowaliśmy na lampowym komputerze (maszyna **UMC–1**), wykonując (nie bez trudu[3]) bardzo skomplikowane obliczenia. Tranzystory[4] pozwoliły zminiaturyzować komputery, a także umożliwiły budowę maszyn na tyle niezawodnych, że znalazły zastosowanie w instytucjach gospodarczych, w przemyśle i w wojsku. Od tego momentu datuje się burzliwy rozwój informatyki, który trwa do dziś.

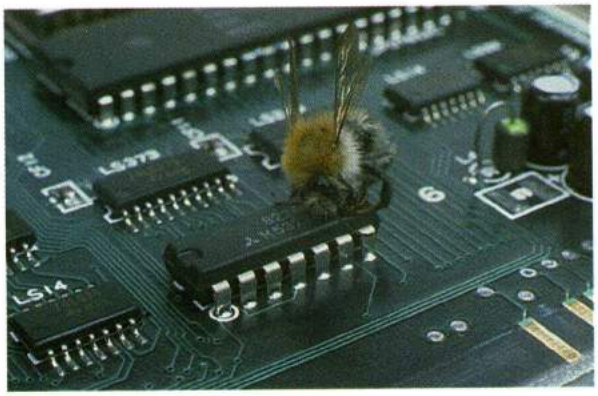
---

[1] Uważa się obecnie, że "dziadkami" współczesnych komputerów byli: **John V. Atansoff** (twórca idei) i **Clifford Berry** (konstruktor) pierwszego komputera **ABC**, zbudowanego z wykorzystaniem 1.5 tys. lamp elektronowych w latach **1939 – 1942**. Na bazie tego komputera **John Mauchly** i **Prosper Eckert** zbudowali w **1948** roku pierwszy szeroko znany elektroniczny komputer ENIAC, a w **1949** roku założyli pierwszą firmę produkującą (i sprzedającą!) elektroniczne komputery UNIVAC I (*Remington Rand*). Słynny IBM w owym czasie lekceważył komputery, gdyż produkował mechaniczne urządzenia liczące, chlubiąc się tradycją firmy istniejącej od **1896** roku i wynalazkami swego założyciela, **Hermana Holleritha**, które w **1890** roku zrewolucjonizowały prace biurowe.

[2] Komputer ma dłuższą historię, niż się to powszechnie przypuszcza. Pierwszy projekt komputera (mechanicznego, lecz możliwego do programowania) opracował w **1830** roku **Charles Babbege,** profesor Uniwersytetu w Cambridge. Pierwszy program napisała w **1842** roku **Ada Lovelace,** córka poety Byrona. Niestety maszyna Babbege'a nigdy nie została ukończona. Działający komputer, zbudowany dokładnie według planów Charlesa Babbage'a z 1847 roku,zbudowali po 140 latach pracownicy **Muzeum Nauki** w Londynie. Ta ogromna (2.1 x 3.4 x 0.5 metra) maszyna waży 3 tony i składa się z ponad 4000 części, wykonanych z zegarmistrzowską precyzją z brązu, żelaza i stali. Podczas testów (polegających na obliczeniu siódmych potęg stu liczb całkowitych) "komputer" ten dostarczał poprawnych wyników o dokładności 31 cyfr znaczących, których sprawdzanie za pomocą zwykłych, elektronicznych komputerów sprawiało trochę kłopotów. Jednak prawdziwym problemem podczas testów był napęd maszyny Babbage'a: dla uzyskania jednego wyniku trzeba było wykonać ponad **27 tysięcy** obrotów korbką napędzającą mechanizm!

[3] Bardzo ważnym narzędziem informatyka w tamtych czasach był gumowy młotek, którym po otwarciu stalowych drzwi wielkiej szafy, w której żarzyły się setki lamp, opukiwało się pakiety celem usunięcia wadliwie pracujących styków. Komputer ten miał jednak bardzo istotną zaletę: grzał się tak, że nawet w największe mrozy było przy nim bardzo ciepło! Dlatego w słynną "zimę stulecia" pierwszą czynnością przychodzących do pracy badaczy było załączenie komputera i uruchomienie jakichś "dużych obliczeń". Zebrana tą drogą kolekcja rozwiązań numerycznych problemów związanych ze słynną **teorią chaosu** budziła potem uznanie na międzynarodowych konferencjach i wprawiała w zdumienie amerykańskich badaczy, wyposażonych w kilkaset razy szybsze komputery.

[4] Wynalazek tranzystora zrewolucjonizował cywilizację w sposób wręcz trudny do przecenienia. To, że mamy dziś tanie i łatwo dostępne telewizory i magnetofony, sieć telefoniczną obejmującą cały glob no i oczywiście nowoczesne komputery – zawdzięczamy właśnie temu wynalazkowi. Tym bardziej dziwna i niesłuszna jest powszechnie panująca ignorancja co do nazwisk ludzi, którym ten wiekopomny wynalazek zawdzięczamy. Pora to zaniedbanie skorygować! Przy badaniu półprzewodników pracowało wielu ludzi (*I. Tamm, J. Bardeen, W.H. Brattain* i inni), jednak bezwarunkowo największe zasługi położył **William Bradford Shockley,** którego doświadczenia w 1947 i praca teoretyczna w **1949 roku** otworzyły drogę do budowy tranzystora, a samego twórcę zaprowadziły do Nagrody Nobla (w 1956 roku).



Trzecia generacja komputerów pojawiła się w połowie lat sześćdziesiątych i związana była z zastosowaniem tak zwanych **układów scalonych**. Układy scalone mogą obejmować większe lub mniejsze fragmenty struktury komputera, z czym wiąże się zarówno klasyfikacja tych układów, jak i numeracja następnych generacji maszyn cyfrowych. Układy scalone są bardzo małe, a w ich wnętrzu można zobaczyć (na fotografii obok) jeszcze mniejsze kryształy krzemu, zawierające wszystkie niezbędne elementy elektroniczne. Pierwsze układy scalone obejmowały poszczególne proste moduły, stanowiące podstawowe "cegiełki" w strukturze komputera: "bramki", układy wykonujące operacje logiczne, przerzutniki, wzmacniacze, generatory itp. Dziś tę skalę scalania określa się skrótem **SSI** (*small scale integration*) i wiąże się ją z trzecią generacją komputerów. Trwała ona mniej więcej do początku lat siedemdziesiątych.

Po niej przyszła możliwość scalania w formie pojedynczego układu całych jednostek funkcjonalnych: liczników, rejestrów, dekoderów, multiplekserów itp. Ta skala scalania nazywana jest dziś średnią skalą integracji, w skrócie **MSI** (*midle scale integration*). Komputery budowane z takich elementów mogły być znacznie mniejsze i tańsze (wtedy właśnie pojawiły się minikomputery) – a także bardziej niezawodne w działaniu i szybsze. Wreszcie stale zwiększana skala scalania objęła cały procesor, w wyniku czego powstał produkt zwany dziś **mikroprocesorem**.

Wynalazek mikroprocesora był kluczem do komputerów **czwartej generacji**. O mikroprocesorach będzie obszernie mowa dalej, w podrozdz. 2.4, dlatego poprzestaniemy tutaj wyłącznie na tym zdawkowym stwierdzeniu. Dodając do mikroprocesora kilka stosunkowo tanich układów elektronicznych (pamięć **RAM**[1] i **ROM**, zegar, sterowniki urządzeń zewnętrznych), otrzymuje się bardzo tani i bardzo efektywny komputer, zwany ze względu na miniaturowe rozmiary i ze względu na przystępną cenę ***mikrokomputerem***[2]. Z urządzeniem tym zapoznamy się dokładniej w następnym podrozdziale.

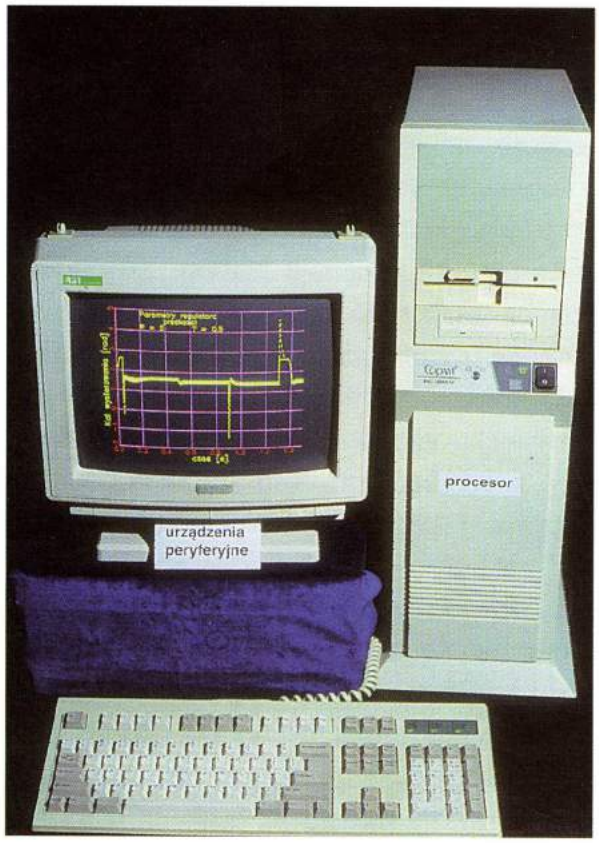
Mikroprocesory oraz bardziej od nich rozbudowane mikrokontrolery, a także moduły, które zawierają w jednym układzie scalonym płaty pamięci **RAM** lub **ROM** mieszczące kilka tysięcy bitów, stanowią wyroby dużej skali integracji **LSI** (*large scale integration*). To one ostatecznie wyznaczyły czwartą generację, której wyroby wciąż jeszcze pracują w licznych ośrodkach obliczeniowych.

Dalszy rozwój skali scalania powoduje przejście do **piątej generacji** komputerów, budowanych z tak zwanych układów bardzo wielkiej skali integracji **VLSI** (*very large scale integration*). Tutaj efekt rozwoju technologii jest jeszcze bardziej doniosły, niż przy pokonywaniu wcześniej wymienionych progów, lecz trudniejszy do nazwania. Jeśli powiemy, że procesory 8–bitowe zastąpione

---

[1] Czy pamiętasz naszą solenną obietnicę, że wszystkie niezrozumiałe w tej chwili nazwy i terminy fachowe będą dalej wyjaśnione?

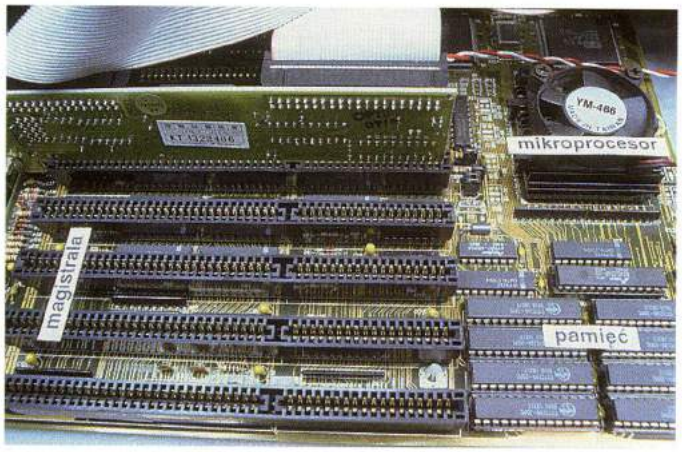
[2] Pierwszym szerzej znanym mikrokomputerem był zapewne ALTAIR, zbudowany w 1974 roku przez **Lesa Solmana**. Odniósł on niespodziewany sukces: maszyna, która przewidywana była do użytku dla kilku osób i traktowana jako ciekawostka techniczna została ostatecznie sprzedana w ponad stu egzemplarzach i była wykorzystywana w najbardziej nieoczekiwanych miejscach (między innymi w Białym Domu). Jednak mikrokomputerem, który autentycznie zapoczątkował nową epokę, był **Apple**, zbudowany w **1976** roku przez **Stephena G. Wozniaka** i **Stevena J. Jobsa.** Mikrokomputer ten, zbudowany w garażu (!) za pieniądze uzyskane ze sprzedaży starego Volkswagena, stał się ogromnym sukcesem rynkowym: do 1984 roku sprzedano ponad dwa miliony tych komputerów, co przyniosło jego twórcom ponad miliard dolarów wpływów.



zostały przez 16, a potem 32–bitowe – to dla elektroników jest to cała epoka, z punktu widzenia wydajności jest to równoważne przesiadce z konnego wozu do Jumbo Jeta[1], ale dla laika jest to mało zauważalne: i tu mikroprocesor i tu mikroprocesor... Podobnie jest ze stwierdzeniem, że układy pamięciowe o pojemności **1 kB** zastępowane są obecnie przez moduły o pojemności **1 MB** – praktyczny efekt jest taki, że rozmiary komputera można tysiąckrotnie (!) zmniejszyć, ale dla niefachowca nie ma w tym nic istotnie nowego – tu pamięć i tam pamięć...

Podejdźmy więc do wskazania zalet **VLSI** z innej strony. Potężne pod względem mocy obliczeniowej i miniaturowe w rozmiarach procesory powodują, że możliwe jest używanie **struktur wieloprocesorowych.** To właśnie do nich należy piąta generacja komputerów, które pojawiwszy się w połowie lat osiemdziesiątych w formie tak zwanych **superkomputerów** schodzą teraz "w dół" stając się dostępne za cenę, na jaką możesz sobie swobodnie pozwolić, a równocześnie zajmują zaledwie pół biurka.

2.2. Szkic budowy mikrokomputera



Mikrokomputer składa się z **procesora** (nazywanego też jednostką centralną) i **urządzeń peryferyjnych**. Procesor jest zbudowany z wielu układów scalonych[2], jednak **funkcjonalnie** można mówić, że składa się on z trzech podstawowych elementów:

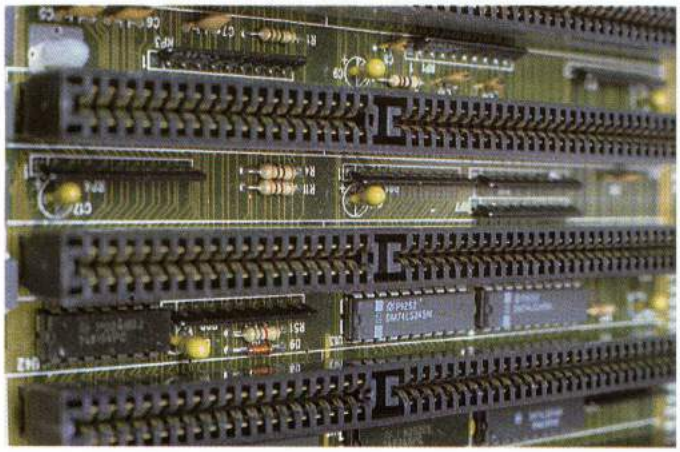
- mikroprocesora,
- pamięci,
- magistrali.

---

[1] Fascynujący wzrost mocy obliczeniowej komputerów wymyka się wprost wyobraźni. **Christopher Evans** napisał w **1979** roku, że gdyby podobny wzrost wydajności dotyczył przemysłu samochodowego, to ***Rolls–Royce*** kosztowałby 2,75$ i pozwalał przejechać 3 miliony mil na jednym galonie benzyny. Cytując te dane **David Oven Arnold** napisał w **1988** roku, ze obecnie cena tego (najlepszego na świecie) samochodu spadłaby poniżej jednego dolara, a połowa uncji benzyny (około 15 g) wystarczyłoby na cały czas "życia" samochodu, szacowany na około 20 lat. Od tego czasu efektywność systemów komputerowych wzrosła jeszcze tysiące razy!

[2] Układy scalone stanowią dziś zasadniczy budulec, z którego wytwarza się cały sprzęt informatyczny – nie tylko komputery zresztą, ale także wszystkie urządzenia peryferyjne. Warto więc poznać historię tych układów, gdyż trudno sobie wyobrazić kogoś, kto mieni się informatykiem, a równocześnie nie ma w omawianej kwestii elementarnego pojęcia. Historia wynalazku układu scalonego wiąże się z firmą **Texas Instruments** i z osobą inżyniera tej firmy, posiadacza historycznego patentu na układ scalony, **Jacka Kilby**. Wspomniany wynalazca pracował początkowo w firmie w Milwaukee przy budowie układów elektronicznych dla potrzeb korpusu łączności Armii Stanów Zjednoczonych. W tej firmie zapoznał się z technologią półprzewodnikową, gdyż od 1952 roku wytwarzał (na licencji firmy Bell) tranzystory germanowe. W 1958 roku Jack Kilby przeszedł do pracy w potężnej już wówczas firmie elektronicznej Texas Instruments, gdzie w lipcu tegoż roku wytworzył pierwszy układ scalony. Był to przerzutnik bistabilny, do dziś jeden z podstawowych elementów każdego komputera. **28 sierpnia 1958** roku przeprowadzono pierwszą demonstrację pracy układu scalonego wytworzonego przez Kilby'ego. Jego szef, **Willis Adcock,** poparł wynalazcę i lawina ruszyła. Trzeba było jednak blisko 10 lat na to, by epokowy wynalazek Kilby'ego został należycie doceniony i wykorzystany.

Mikroprocesor składający się z układów arytmetyczno–logicznych oraz układu sterowania jest elementem organizującym pracę komputera. Służy on bezpośrednio do przetwarzania danych, wykonuje obliczenia, porównuje liczby, itp.



Dane i **wyniki** są przechowywane w **pamięci**. Tam też zapisany jest **program**, opisujący sposób przetwarzania informacji. Program jest zbudowany z wielu rozkazów, z których każdy stanowi dla procesora polecenie wykonania pojedynczej operacji.

Komunikacja pomiędzy procesorem i pamięcią, a także przesyłanie danych od i do urządzeń peryferyjnych możliwe są dzięki istnieniu **magistrali.** Łączy ona wszystkie elementy komputera, a ponadto pozwala dołączać do niego tzw. **karty rozszerzeń.**

Aby wykonać zadanie, komputer musi czasem gromadzić większe ilości danych, musi otrzymywać od człowieka dane wejściowe, a także powinien przekazywać na zewnątrz wyniki. Zadań tych nie wykonuje sam procesor, potrzebne są więc dodatkowe **urządzenia peryferyjne.** Do gromadzenia dużych ilości danych służą **pamięci masowe,** zaś do wprowadzania i wyprowadzania informacji **urządzenia wejścia/wyjścia** (np. **klawiatura** i **monitor**).

Wszystkie przedstawione tu skrótowo elementy komputera będą dalej dokładniej omówione[1], zanim to jednak nastąpi poznajmy budowę mikrokomputera w sposób ogólny.

Oto komputer klasy **PC**[2] – obecnie najpopularniejszy na całym świecie. Możliwości takiego mikrokomputera osobistego[3] można zwykle rozszerzać i dopasowywać do indywidualnych potrzeb i wymagań użytkownika poprzez stosowanie tak zwanych **kart rozszerzeń**. Karty te, wkładane do specjalnych gniazd w procesorze, pozwalają dopasowywać komputer do specjalnych zadań: wczytywania obrazów, generowania mowy, sterowania różnych urządzeń lub łączenia komputerów pomiędzy sobą. Będzie o tym mowa w kolejnych podrozdziałach.

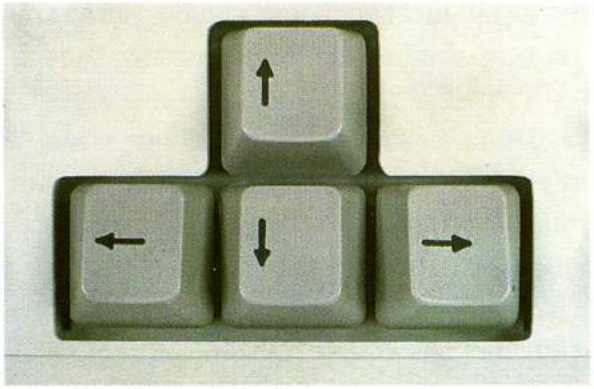
Do podstawowego wyposażenia mikrokomputera należy **klawiatura i monitor** – monochromatyczny lub kolorowy. Ekran monitora to podstawowe urządzenie służące do

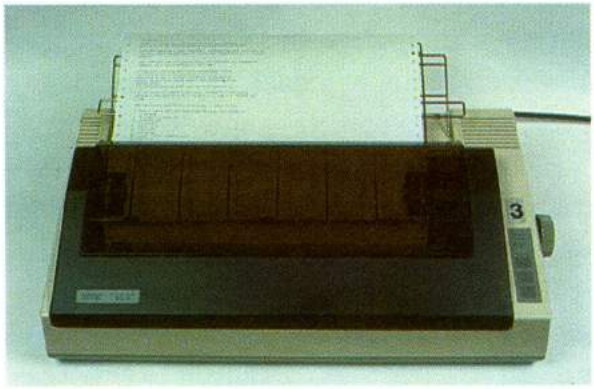
---

[1] Niezbędnym elementem każdego komputera są też elementy wyposażenia technicznego, na przykład zasilacz zaopatrujący układy komputera w energię elektryczną i obudowa mechanicznie wiążąca wszystkie elementy, ale nimi nie będziemy się zajmować.

[2] W klasie komputerów PC niekwestionowany prymat (ponad sto milionów sprzedanych maszyn!) dzierży **IBM PC**. Jego twórcą był **Don Estridge,** przy czym fenomenem jedynym w swoim rodzaju było tempo opracowania tej maszyny: prace rozpoczęto w 1980 roku,a już **12 sierpnia 1981** zaprezentowano ten mikrokomputer jako gotowy produkt. Niestety, Don Estridge nie dożył światowego sukcesu swej maszyny: zginął 2 sierpnia 1985 roku w katastrofie samolotu Delta Airlines L–1011 w Dallas...

[3] Uważa się, że **mikrokomputerem osobistym** jest komputer przeznaczony dla pojedynczego użytkownika, mający małe wymiary i mały pobór mocy, tani, prosty w obsłudze, o pamięci kilku milionów bajtów, z klawiaturą i ekranem jako podstawowymi urządzeniami zewnętrznymi, ale z możliwością współpracy z **drukarką** i innymi urządzeniami peryferyjnymi. Najczęściej spotyka się konfiguracje z dwoma **stacjami dyskietek** (wygodne kopiowanie!) i z tak zwanym **dyskiem twardym**.





wyprowadzania wyników pracy komputera. Wyświetlane są na nim **znaki alfanumeryczne** (litery i cyfry) przedstawiające wprowadzane z klawiatury teksty i liczby oraz odpowiedzi komputera na postawione zadania. Znaki wprowadzane są na ekran w miejscu pojawienia się migającego znacznika, tzw. **kursora,** którego położenie może być zmieniane za pomocą **klawiszy kursora** lub za pomocą **myszki**.

Podstawowa pamięć komputera (nazywana zwykle **operacyjną** lub określana skrótowo jako **RAM**) ma niewielką pojemność, a informacje w niej zapisane znikają w chwili wyłączenia zasilania. Konieczne jest zatem stosowanie **pamięci masowej,** pozwalającej trwale zapisać duże zbiory informacji. Najpopularniejszym rodzajem pamięci masowej jest **pamięć dyskowa** – stała lub na **dyskach elastycznych (dyskietkach).** Zapis i odczyt informacji jest dokonywany w urządzeniu zwanym **stacją** (*drive*). W czasie gdy dyskietka się obraca, na obudowie zapala się lampka. Nie wolno wtedy wyjmować dyskietki – próba taka może spowodować uszkodzenie stacji. Pojemność dyskietki wynosi zazwyczaj od **360 KB** (w najtańszym wariancie dyskietek o średnicy 5,25 cala) do **1,44 MB** (w nowocześniejszych dyskietkach 3,5 calowych formatowanych z tzw. podwójną gęstością).

Drukarka jest jednym z najczęściej wykorzystywanych urządzeń zewnętrznych. Najpopularniejsze są drukarki mozaikowe, w których obraz znaku powstaje z szeregu drukowanych kolejno rzędów kropek. W ten sposób można także tworzyć rysunki. Oczywiście im więcej kropek składa się na jeden znak, tym ładniej wygląda i tym lepsze wrażenie robi cały wydrukowany tekst. Naturalnym kierunkiem rozwoju drukarek było więc stosowanie coraz większej liczby coraz cieńszych drukujących igieł. Wreszcie postanowiono zastąpić ruchome igły promieniem światła lasera – cieńszym i szybszym niż wolframowe igły. Tak powstała drukarka laserowa – znacznie doskonalsza, ale i droższa (w momencie zakupu, a także w eksploatacji).

2.3. Zasada działania komputera

2.3.1. Dane i sposób ich kodowania

Współczesne komputery to urządzenia elektroniczne działające na zasadzie wytwarzania, przesyłania i przetwarzania impulsów elektrycznych. Obecność jednego typu impulsu w określonym momencie oznaczana jest cyfrą "1", a obecność innego utożsamiana jest z cyfrą "0". To wystarcza do odwzorowania dowolnych sygnałów i dowolnych informacji, ponieważ stosując odpowiednio ciągi zer i jedynek można zakodować dowolne dane.

2.3.1.1. Zalety systemu dwójkowego

Istnieje wiele powodów wybierania systemu dwójkowego jako podstawy funkcjonowania układów współczesnych komputerów. Najważniejsze z nich mają charakter techniczny: układy elektroniczne operujące na liczbach dwójkowych są szczególnie łatwe w budowie, odznaczają się bardzo dużą odpornością na zakłócenia, a także pozwalają budować systemy obliczeniowe z wykorzystaniem szczególnie małej liczby elementów składowych, czyli są proste, miniaturowe i tanie. Innym argumentem jest urzekająca prostota arytmetyki dwójkowej. Na przykład reguły dodawania dwójkowego obejmują jedną tylko konieczną do zapamiętania formułę[1]:

1 + 1 = 10

co należy odczytywać w następujący sposób: dodanie do siebie dwóch jednostek niższego rzędu daje w wyniku jedną jednostkę wyższego rzędu[2]. Zauważ, o ile prostsza jest ta reguła od obszernego zbioru koniecznych do zapamiętania reguł dla systemu dziesiętnego; uczniowie szkół podstawowych od stuleci mozolą się nad zapamiętaniem, że 7+5=12, 4+9=13 itd.

Jeszcze bardziej uderzająca jest prostota reguł mnożenia. Cała tabliczka mnożenia w systemie dwójkowym sprowadza się do jednej tylko formuły:

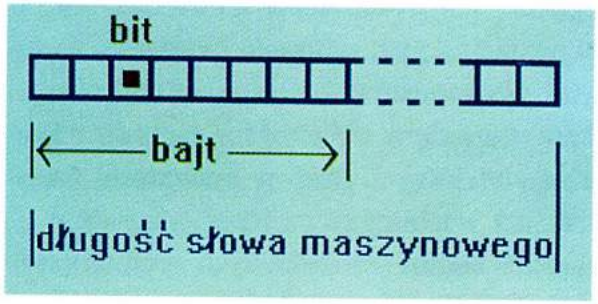
1 × 1 = 1

ponieważ mnożenie przez zero w każdym systemie daje w wyniku zero, a innych cyfr poza **0** i **1** w systemie dwójkowym nie ma.

2.3.1.2. Kodowanie liczb

Kodowanie binarne jest szczególnie łatwe w odniesieniu do liczb: istnieje wszak dwójkowy system wyrażania dowolnych wielkości liczbowych i zawsze można wartości zapisane w sposób dziesiętny zamienić jednoznacznie na wartości zapisane w sposób dwójkowy[3]. Przykładowo cyfrę 5 możesz w systemie dwójkowym zapisać jako 101, a liczbę 1465 jako 10110111001.

Dwójkowe (złożone wyłącznie z symboli 0 i 1) kody określonych liczb wpisywane są w komputerze do tak zwanych **rejestrów** lub umieszczane są w **pamięci**. W jednym i w drugim przypadku wykorzystuje się fakt, że odpowiednie układy elektroniczne, tak zwane przerzutniki, mogą przybierać dwa łatwo rozróżnialne stany, a większe struktury budowane z nich mogą rejestrować dłuższe ciągi zer i jedynek zapamiętując – na przykład – dowolne liczby. Na rysunku obok przedstawiono schemat rejestru lub tak zwanej **komórki pamięci**. Przedstawiono na nim tak zwane **bity**, to znaczy jednostki mogące znajdować się w jednym z dwóch wyróżnionych stanów (0 lub 1), **bajty** stanowiące wydzielone grupy po 8 bitów każda oraz przedstawiono **długość słowa maszynowego** jako liczbę bitów zawartych w całym słowie.



W słowach o budowie podanej na rysunku bez trudu można rozmieszczać liczby dwójkowe. Po prostu odpowiednie kolejne bity liczby dwójkowej wpisuje się (od prawej strony) do słowa maszynowego, uzupełniając ewentualnie od lewej zawartość słowa zerami. Pewne komplikacje

---

[1] W systemie dwójkowym są używane tylko dwie cyfry: **0** i **1**, a dodawanie zera zawsze polega na pozostawieniu bez zmian liczby, do której dodawano, dlatego rozważamy wyłącznie regułę dla dodawania jedynek.

[2] Identyczna reguła obowiązuje w systemie dziesiętnym: dodanie dziesięciu (czyli tylu, ile wynosi podstawa systemu liczenia) jednostek niższego rzędu (na przykład **setek**) daje jedną jednostkę wyższego rzędu (**tysiąc**).

[3] Jeśli Cię ten aspekt szczególnie interesuje – przypomnij sobie podstawowe wiadomości na temat systemu dwójkowego i sposobu zamiany liczb pomiędzy systemami – dziesiętnym i dwójkowym. Nie jest to jednak konieczne, bo śledzenie przedstawionych dalej wywodów możliwe jest nawet bez elementarnej wiedzy na temat systemów liczbowych – tylko trzeba wtedy przyjąć pewne stwierdzenia "na wiarę".

wynikają, jeśli chce się zapisywać liczby mające **znak**. Tu już trzeba się umówić, że jedna z cyfr dwójkowych nie będzie wchodziła w skład liczby, lecz będzie symbolizowała znak. Najczęściej umowa przewiduje, że taka wyróżniona cyfra (tak zwany **bit znakowy**) znajdować się będzie na początku zapisywanej liczby. Zakłada się zazwyczaj, że cyfra "**0**" na tej wyróżnionej pozycji odpowiadać będzie znakowi "**+**", zaś cyfra "**1**" znakowi "**–**".

W rzeczywistości opisana wyżej zasada przedstawiania liczb w pamięci komputera musi być nieco zmodyfikowana, gdyż dla prawidłowego prowadzenia obliczeń na liczbach **ujemnych** trzeba zmodyfikować ich zapis wprowadzając tzw. **kod uzupełnieniowy,** natomiast w odniesieniu do liczb zawierających ułamki (tzw. liczb **rzeczywistych**) dodatkowo trzeba wprowadzić tak zwany **zapis zmiennoprzecinkowy.** O szczegółach tych zapisów można się dowiedzieć z innych podręczników (wykaz niektórych z nich podaliśmy na końcu książki), jednak z punktu widzenia **użytkownika** komputera taka ogólna wiedza, jak podano wyżej, jest całkowicie wystarczająca, gdyż pozwala ona uświadomić sobie, jak komputer operuje liczbami w swojej pamięci – i co z tego wynika.

Warto dodać, że wiedza o tym, jak komputer rozróżnia liczby dodatnie od liczb ujemnych może czasem ułatwić zrozumienie zachowania maszyny – na przykład nieostrożnie wykonując działania na bardzo dużych liczbach całkowitych można doprowadzić do zjawiska tzw. **nadmiaru** (*overflow*), w wyniku czego w sposób pozornie zupełnie tajemniczy dodając do siebie dwie liczby dodatnie można dostać w wyniku liczbę ujemną, co jest oczywistym błędem. Dobre oprogramowanie zabezpiecza użytkownika przed powstawaniem takich paradoksów, ale czasem mamy do czynienia z kiepskim oprogramowaniem i wtedy mogą się zdarzać podobne "cuda". Nie należy w takim przypadku przypuszczać, że komputer odkrył nowe prawa matematyczne, tylko uświadomić sobie, jak prymitywnie odróżnia on liczby dodatnie od ujemnych i – oczywiście – poszukać lepszego programu do wykonania potrzebnych obliczeń.

Podobnie, znajomość rozróżnienia między sposobem reprezentacji w pamięci komputera liczb **całkowitych** i **rzeczywistych** może czasami być przydatna w celu zrozumienia, dlaczego liczby zawierające ułamki zajmują o 100% więcej miejsca w pamięci i dlaczego obliczenia na nich prowadzone dostarczają **zwykle** wyników niedokładnych, z lepszym albo gorszym **przybliżeniem** realnych matematycznych wartości.

2.3.1.3. Zapis liczb dwójkowych w systemie ósemkowym i szesnastkowym

Operowanie liczbami binarnymi jest nieco niewygodne, ponieważ nawet niewielkie (co do wartości bezwzględnej) liczby dwójkowe wymagają wielu pozycji[1] do ich zapisania (porównaj wyżej liczbę 1465 i jej dwójkowy odpowiednik). Dlatego na użytek programistów, zmuszonych do częstego wyrażania w swoich programach wartości binarnych, wykorzystuje się w informatyce zapis ósemkowy (**oktalny**) lub szesnastkowy (**heksadecymalny**). Wartości niektórych liczb w systemach: dziesiętnym, dwójkowym, ósemkowym i szesnastkowym podaliśmy w tablicy na następnej stronicy.

Zauważ, że w systemie ósemkowym używa się wyłącznie **ośmiu** rozróżnialnych symboli (cyfr od **0** do **7**), a w systemie szesnastkowym aż 16 symboli, zatem obok cyfr od **0** do **9** używa się (traktowanych jako cyfry) symboli **A, B, ..., F.**

Sposób zamiany liczb dziesiętnych na dwójkowe i odwrotnie znasz zapewne ze szkoły[2]. Gorzej natomiast – pozornie – jest z zamianą na system **ósemkowy** lub **szesnastkowy**. Tymczasem systemy te możesz traktować jako skrócony sposób zapisywania liczb dwójkowych, nie wymagają zatem żadnego dodatkowego wysiłku. Zapis liczby w systemie ósemkowym powstaje po podzieleniu liczby dwójkowej na grupy po trzy bity i zastąpieniu każdej trójki bitów odpowiadającą jej cyfrą ósemkową.

---

[1] W przybliżeniu (z niedomiarem) można przyjąć, że do zapisu wartości wyrażanej **n**–cyfrową liczbą dziesiętną trzeba użyć **3n**–cyfrowej liczby dwójkowej (dokładny przelicznik wynosi **lg 2 = 3,321928095**).

[2] Jeśli nie znasz i w dodatku nic Cię to nie obchodzi – to po prostu pomiń te wszystkie nudziarstwa i zacznij czytać od początku następnego podrozdziału. Uczciwie mówiąc – niewiele stracisz!

Sposób zapisu liczby							
dziesiętny	dwójkowy	ósemkowy	szesnastkowy	**dziesiętny**	dwójkowy	ósemkowy	szesnastkowy
0	0	0	0	**13**	1,101	15	D
1	1	1	1	**14**	1,110	16	E
2	10	2	2	**15**	1,111	17	F
3	11	3	3	**16**	10,000	20	10
4	100	4	4	**17**	10,001	21	11
5	101	5	5	**18**	10,010	22	12
6	110	6	6	**19**	10,011	23	13
7	111	7	7	**20**	10,100	24	14
8	1,000	10	8	**...**	...	...	...
9	1,001	11	9	**100**	1,100,100	144	64
10	1,010	12	A	**...**	...	...	...
11	1,011	13	B	**256**	100,000,000	400	FF
12	1,100	14	C	**65,535**	1,111,111,111,111,111	177,777	FFFF

Rozważmy liczbę dziesiętną **1000.** Odpowiadająca jej liczba dwójkowa ma postać: **1111101000.** Dzieląc tę liczbę na grupy trzybitowe, otrzymujesz:

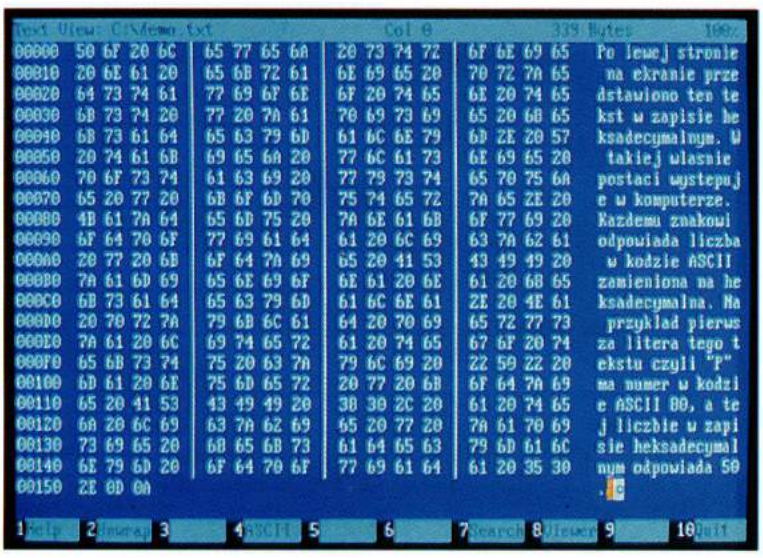
001 111 101 000
1 7 5 0

Pod każdą grupą bitów dopisaliśmy odpowiadającą jej cyfrę ósemkową; zbierając te cyfry razem otrzymasz **oktalny** (czyli ósemkowy) zapis rozważanej liczby:

1750

Zamiana "w drugą stronę" zachodzi według analogicznej zasady: każdą cyfrę liczby ósemkowej rozpisuje się na trzy cyfry binarne, a potem analizuje się liczbę binarną w ogólnie znany sposób. Zasada postępowania przy korzystaniu z układu szesnastkowego jest podobna, jednak kodowaniu podlegają czterobitowe fragmenty liczby dwójkowej (ponieważ $16 = 2^4$, natomiast $8 = 2^3$). Prześledźmy to na tym samym przykładzie liczby 1000, która w formie dwójkowej pogrupowanej w czterobitowe "paczki" ma postać:

0011 1110 1000



Znajdując (w tablicy) szesnastkowe odpowiedniki dla poszczególnych wydzielonych grup bitów, otrzymujesz łącznie szesnastkową wartość **3E8**, wyglądającą nieco egzotycznie, ale stanowiącą bardzo krótki i wygodny zapis odpowiedniej liczby binarnej. Powinieneś chociaż trochę znać i rozumieć te kody, ponieważ czasem możesz chcieć zajrzeć do pamięci komputera, a wtedy on ci pokaże co w niej ma – ale właśnie szesnastkowo, jak na fotografii obok. I jak sobie poradzisz?

Zauważ, że system ósemkowy jest nieco łatwiejszy w użyciu, natomiast system szesnastkowy daje lepsze upakowanie prezentowanych liczb, a ponadto lepiej pasuje do struktury pamięci komputera (każdy bajt to dwie cyfry szesnastkowe). Z tego powodu w początkowych latach rozwoju informatyki stosowano głównie system ósemkowy, a obecnie najczęściej stosuje się zapis szesnastkowy – **heksadecymalny.**

Zauważ też, że liczby "okrągłe" w systemie dziesiętnym (10, 100, 1000) z reguły mają bardzo skomplikowaną budowę w systemie dwójkowym (i automatycznie także w ósemkowym i szesnastkowym), zaś liczby wygodnie wyrażane w systemie dwójkowym (postaci 2^n) mają bardzo "brzydkie" rozwinięcia dziesiętne.

2.3.1.4. Binarny zapis tekstów i innych informacji

D	H	Ch	D	H	Ch
64	40	@	80	50	P
65	41	A	81	51	Q
66	42	B	82	52	R
67	43	C	83	53	S
68	44	D	84	54	T
69	45	E	85	55	U
70	46	F	86	56	V
71	47	G	87	57	W
72	48	H	88	58	X
73	49	I	89	59	Y
74	4A	J	90	5A	Z
75	4B	K	91	5B	[
76	4C	L	92	5C	\
77	4D	M	93	5D	]
78	4E	N	94	5E	^
79	4F	O	95	5F	_

W dwójkowej postaci można przedstawić dowolne inne rodzaje danych wykorzystywanych w technice komputerowej. Na przykład dowolne **teksty** używane w komputerach zapisuje się w ten sposób, że każdej literze (lub innemu znakowi, na przykład symbolowi **&** czy kropce) przypisuje się jakiś numer, a potem te numery można już łatwo zapisać w sposób dwójkowy. Jeden z częściej używanych sposobów ponumerowania liter znany jest jako tak zwany kod **ASCII**. Kod ASCII (*American Standard Code for Information Interchange*), czyli **Amerykański Standardowy Kod do Wymiany Informacji,** jest specjalnym kodem, w którym każdej dużej literze, małej literze, cyfrze, znakowi interpunkcyjnemu i symbolowi specjalnemu zostaje przypisana jedna liczba z zakresu od 0 do 127[1]. Na przykład literze **A** odpowiada numer **65,** więc w pamięci komputera dla tej litery zapisane są następujące bity:

01000001

Kolejnym literom tekstu odpowiadają w pamięci komputera kolejne takie "paczki" bitów. Na przykład słowo **ALA** wygląda w sposób następujący:

01000001 01001100 01000001

W podobny sposób, to znaczy binarnie (jako ciągi zer i jedynek), zapisywane są w komputerze wszelkie inne dane nienumeryczne, na przykład warunki logiczne związane z oceną przez komputer prawdziwości lub fałszywości określonych stwierdzeń, rysunki lub dowolne inne obrazy, dźwięki mowy lub muzyki, sygnały sterowanego przez komputer procesu przemysłowego itp.

2.3.2. Programy i ich znaczenie

2.3.2.1. Czym jest program?

W pamięci komputera oprócz danych zawarte są także **programy**. Programy komputerowe składają się z oddzielnych **rozkazów**[2], opisujących dokładnie, szczegółowo i jednoznacznie wszystkie czynności, jakie komputer ma wykonać, aby rozwiązać postawione zadanie. Jedne rozkazy nakazują wykonanie konkretnych działań, inne natomiast organizują pracę maszyny wskazując, jakie

---

[1] W IBM–PC stosowany jest rozszerzony kod ASCII (EXTENDED ASCII CODE) wykorzystujący dodatkowo liczby z zakresu od 128 do 255. Są one używane do kodowania symboli matematycznych, prostych symboli graficznych i niektórych znaków specjalnych.

[2] Twórcą większości podstawowych koncepcji, na których do dziś oparte jest działanie komputerów, był **John von Neumann**, Węgier pracujący od **1930** roku w *Uniwersytecie Priceton* (USA). W szczególności jego dziełem jest koncepcja zapisywania w pamięci komputera w tej samej formie programów i danych, a także inne podstawowe rozwiązania – struktury rozkazu, adresacji pamięci itp.

rozkazy i w jakiej kolejności mają być wykonywane. Więcej wiadomości na ten temat znajdziesz w książkach, których wykaz podaliśmy na końcu.

Jak wspomnieliśmy wyżej, komputer przechowuje programy w swojej pamięci tak samo, jak i dane, co powoduje, że programy mogą być w nim łatwo zmieniane, poprawiane i uzupełniane – zależnie od potrzeb. Komputer, dzięki działaniu według programu zawartego w swojej pamięci, jest w trakcie wykonywania postawionych mu zadań całkowicie autonomiczny: interwencja człowieka w czasie wykonywania rozkazów jest całkowicie niepotrzebna. Równocześnie dzięki łatwości modyfikowania programu komputer może bardzo elastycznie dostosowywać swoje działanie do Twoich potrzeb, a w wybranych warunkach może także uczyć się i doskonalić swoje zachowanie.

2.3.2.2. Struktura rozkazu

Struktura rozkazów wchodzących w skład programu może być różna, zależnie od typu i budowy rozważanego komputera, możliwe jest jednak podanie pewnej liczby ogólnych prawideł. Zwykle rozkaz zawiera **kod operacji** oraz **część adresową.**

kod operacji	**część adresowa**

Kod operacji określa, co komputer ma zrobić, żeby wykonać dany rozkaz. Jest on zwykle numerem polecenia, wydanego w danym rozkazie – według pewnej arbitralnie ustalonej i związanej z architekturą danego komputera **listy rozkazów**. Lista ta może być dłuższa lub krótsza – zależnie od tego, jaką "filozofię" wyznają twórcy danego komputera.

Przez długi czas rozwój techniki komputerowej przebiegał w ten sposób, że listy rozkazów kolejno budowanych komputerów ciągle wzbogacano i rozszerzano. Powstawały w ten sposób komputery, które wprawdzie potrafiły wykonywać – za pomocą pojedynczego rozkazu – coraz bardziej skomplikowane operacje, ale równocześnie ich programowanie stawało się coraz bardziej uciążliwe, a szybkość działania – mimo coraz doskonalszej technologii elektronicznego wykonania tych komputerów – była coraz mniejsza.

Wielkie firmy wytwarzające sprzęt komputerowy przeprowadziły badania, z których wynikało, że to całe bogactwo skomplikowanych list rozkazów zwyczajnie się nie opłaca. Stwierdzono, że **ponad 90% linii tekstu używanych w praktyce programów wykorzystuje zaledwie 10% dostępnych instrukcji.** Gdyby komputer wyposażyć w listę instrukcji zawierającą wyłącznie te najczęściej używane 10% rozkazów, wówczas można by zbudować maszynę wykonującą te (nieliczne) rozkazy ponad 10 razy szybciej. Oczywiście, jeśli w takim komputerze trzeba wykonać skomplikowany rozkaz, wówczas należy go "rozpisać" na czynności elementarne i traci się czas – w sumie jednak się to bardzo opłaca. Ta droga rozwoju komputerów znana jest pod nazwą **RISC** (*Reduced Instruction Set Computer*).

Część adresowa rozkazu musi wskazać, na jakim obiekcie ma być wykonana zadana rozkazem operacja. W zasadzie powinno się podawać aż trzy adresy:

- adres pierwszego argumentu wykonywanej operacji;
- adres drugiego argumentu;
- adres wyniku operacji.

Przykładowo, jeśli kod operacji nakazuje dodawanie należałoby określić: co dodać, do czego dodać oraz co zrobić z wynikiem dodawania. W rzeczywistości podaje się tylko jeden adres, ponieważ pierwszy argument oraz wynik operacji na zasadzie powszechnie akceptowanej umowy umieszczane są w specjalnym rejestrze procesora, tak zwanym **akumulatorze.**

2.3.3. Zasada działania komputera, a jego uniwersalność

Jak wynikało z przytoczonych rozważań, zasada działania komputera jest bardzo prosta. Spróbujmy ją zapisać w sposób zwarty:

Działanie komputera polega na tym, by maszyna wykonując rozkazy zawarte w jej programie przekształcała jedne informacje (dane) w inne informacje, stanowiące wyniki.

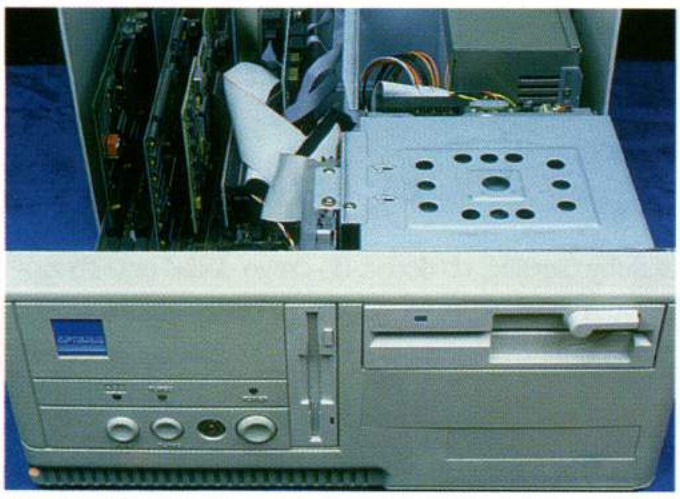
Jak wspomniano na początku tego rozdziału, na poziomie elektroniki podzespołów, składających się na strukturę komputera, mamy do czynienia z różnymi impulsami (oznaczającymi jedynki i zera). Układy, z których składa się komputer, pozwalają w sposób celowy zamieniać jedne serie impulsów na inne. Impulsy mogą być zliczane w licznikach, magazynowane w rejestrach i łączone według założonych reguł w jednostkach arytmetyczno–logicznych. Możliwe jest także trwałe rejestrowanie (pamiętanie) pojedynczych impulsów i całych ich serii. I to wszystko!

Doprawdy trudno uwierzyć, lecz niezliczone zastosowania komputerów, ich szeroko znane możliwości i zalety – sprowadzają się na poziomie sprzętu do takich właśnie elementarnych zjawisk: jest impuls lub go nie ma, następuje przesłanie sygnału dalej – lub jest on wygaszany, zostaje zarejestrowany nowy zbiór zer i jedynek lub odtwarzana jest poprzednia zawartość rejestru czy pamięci. O szeroko znanej uniwersalności komputera, o niezliczonych zastosowaniach, o niewiarygodnych niemal możliwościach maszyny decydują w rezultacie dwa czynniki.

Pierwszy związany jest z **szybkością** pracy układów elektronicznych. Czynności, które wykonują układy komputera, są bardzo proste i sprowadzają się do przepuszczania lub zatrzymywania impulsów, ale przebiegają niesłychanie szybko. Pobranie, przesłanie lub przetworzenie impulsu wymaga zaledwie ułamków milionowej części sekundy. W rezultacie jest więc wystarczająco dużo czasu, by z tysięcy takich prostych, elementarnych operacji złożyć czynności bardziej zbliżone do realnie stawianych komputerowi zadań – w najprostszym przypadku polegających na wykonywaniu działań arytmetycznych.

Drugi czynnik, który powoduje, że komputer jest tak bardzo użyteczną i uniwersalną maszyną, jest związany z **rozmaitością możliwych interpretacji** powstających w nim, przetwarzanych i przesyłanych impulsów. Zależnie od chęci i wyobraźni osoby korzystającej z maszyny możliwe jest utożsamienie impulsowo reprezentowanych w komputerze zer i jedynek – z liczbami, elementami rysunku, fragmentami tekstu, dźwiękami muzyki, sygnałami sterującymi wysyłanymi do kontrolowanych przez komputer podsystemów... Możliwości jest tu bardzo wiele, a wciąż pojawiają się nowe!

2.4. Budowa mikrokomputera

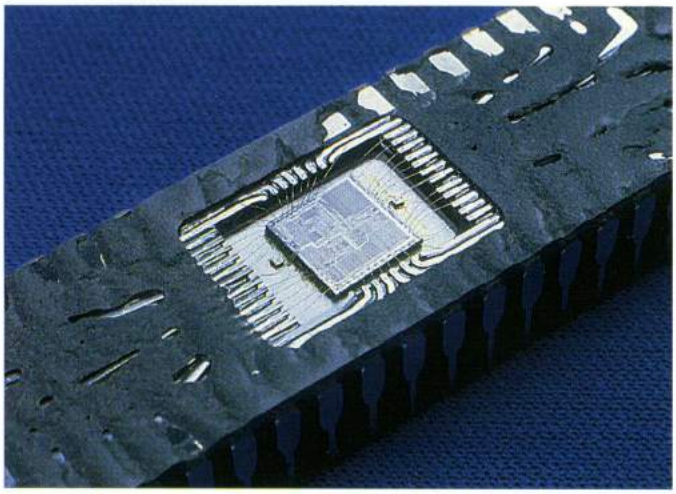


Szkic budowy mikrokomputera podano w podrozdziale 2.2. Jak z tego wstępnego opisu wynikało, wszystkie zasadnicze dla działania komputera operacje i procesy wykonywane są przez **procesor**, czyli jednostkę centralną. To tutaj mieści się **pamięć** przechowująca programy i dane, tutaj znajduje się **mikroprocesor** zawierający układ sterowania i arytmometr, dokonujący wszystkich niezbędnych wyliczeń i tu mieści się **magistrala,** spinająca wszystkie najważniejsze elementy procesora. Poza procesorem w skład komputera

wchodzą **pamięci masowe** oraz inne **urządzenia peryferyjne** – głównie **urządzenia wejścia/wyjścia.**

Scharakteryzujmy teraz w sposób systematyczny wyżej wymienione, podstawowe składniki jednostki centralnej, a następnie pamięci i ważniejsze urządzenia peryferyjne.

2.4.1. Mikroprocesor



Mikroprocesor[1] jest układem wykonanym w odpowiedniej technice (tzw. układów scalonych bardzo wielkiej skali integracji, oznaczonych z angielska **VLSI**) i w odpowiednio zaawansowanej technologii elektronowej (zwykle **CMOS**), wykonującym wszystkie nakazane programem obliczenia. W strukturze komputera mikroprocesor występuje jako pojedynczy mikromoduł, o rozmiarach kilku centymetrów, lecz o trudnej do wyobrażenia wewnętrznej złożoności (32– bitowy mikroprocesor mieści w jednym krysztale krzemu **kilka milionów** odpowiednio połączonych tranzystorów!). Bywają komputery, które mają kilka mikroprocesorów używanych do różnych celów, my jednak skupimy uwagę na **typowej** strukturze przewidującej użycie jednego mikroprocesora w jednostce centralnej.

2.4.1.1. Podstawowe elementy mikroprocesora

Elementem mikroprocesora jest zawsze **arytmometr**. Jest to oczywiste: nazwa procesor pochodzi od anglojęzycznego terminu *data processing*, co znaczy przetwarzanie danych, a proces przetwarzania informacji zachodzi właśnie nie gdzie indziej, tylko w arytmometrze[2]. W skład struktury wewnętrznej arytmometru (której oczywiście nie będziemy tu szczegółowo omawiali) wchodzą tak zwane **rejestry**. Służą one do chwilowego przechowywania argumentów i wyników wykonywanych operacji (arytmetycznych, logicznych i innych). Celowość stosowania rejestrów wynika z faktu, że odpowiednie działania wykonywane są z użyciem rejestrów wielokrotnie szybciej, niż w przypadku używania tylko komórek pamięci. Odpowiednie techniki programowania pozwalają zresztą na wykorzystanie tego atutu i wyliczanie całych złożonych wyrażeń wyłącznie przy użyciu rejestrów. Wśród rejestrów jest zwykle jeden wyróżniony, nazywany **akumulatorem**. Był on już wcześniej wzmiankowany przy omawianiu struktury rozkazów (rozdz. 2.3.2.2). To w nim umieszcza arytmometr wynik wykonywanej operacji (na przykład dodawania lub mnożenia)[3].

Innym elementem mikroprocesora jest **układ sterujący**. Zadaniem tego układu (nazywanego także – zamiennie – układem sterowania) jest wykonywanie rozkazów składających się na program i sterowanie pracą wszystkich pozostałych urządzeń, w tym głównie arytmometru i pamięci, a także nadzorowanie pracy urządzeń peryferyjnych i kontrolowanie wszelkich operacji przesyłania danych.

---

[1] Twórcą mikroprocesora jest **Ted Hoff**, pracownik firmy *Intel*, który wynalazł ten rewelacyjny układ elektroniczny w **1969** roku. Pierwszy mikroprocesor oznaczony był jako **Intel 4004.**

[2] Nie należy przy tym sugerować się zbytnio nazwą tego modułu, pozwalającą przypuszczać, że wykonuje on tylko działania arytmetyczne. W istocie wykonuje on **różne** operacje, w tym także operacje arytmetyczne i logiczne (stąd używana niekiedy nazwa **ALU** od *Arithmetical & Logical Unit*), a także uczestniczy w przetwarzaniu wszelkich typów danych (między innymi tekstów i dowolnych sygnałów). Ważną sferą działalności arytmometru są też **porównania** określonych danych (na przykład sprawdzenie, która z dwu przedstawionych liczb jest większa lub ustalenie, czy wynik określonej operacji jest zerem).

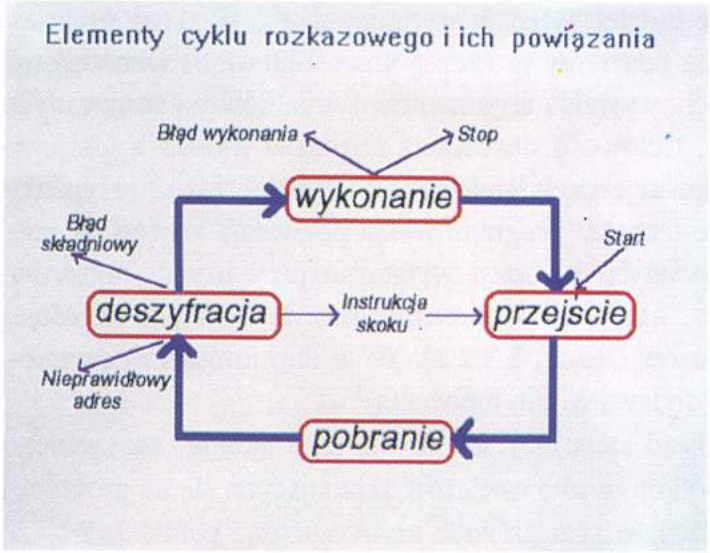
[3] Odpowiednie korzystanie z akumulatora do przechowywania wyników pośrednich przy bardziej skomplikowanych wyliczeniach stanowi jedną z podstawowych technik usprawniania obliczeń.

Należy podkreślić dwie własności układu sterującego, które w znacznej mierze kształtują obraz współczesnych komputerów.

Pierwsza z tych własności polega na stosowaniu (praktycznie bez wyjątków) zasady **sterowania w pętli zamkniętej.** Oznacza to, że układ sterujący wysyła polecenia do pozostałych podsystemów komputera i otrzymuje od nich zwrotnie potwierdzenia o wykonaniu rozkazu. W rezultacie można być pewnym, że wszystkie polecenia są poprawnie wykonane, a ewentualne błędy – powstające zazwyczaj z winy programisty, lecz możliwe także jako następstwo określonych niesprawności sprzętu komputerowego – zostaną wykryte i zasygnalizowane. Jest to bardzo ważna własność, decydująca o przydatności komputera w niektórych zastosowaniach (na przykład w sterowaniu procesów), gdzie nie można sobie pozwolić na ewentualność korzystania z komputera tolerującego błędy.

Druga ze wzmiankowanych zasad związana jest ze sposobem wykonywania rozkazów programu. Otóż układy sterujące mikrokomputerów "napędzane są" przez specjalne generatory kwarcowe, zwane w skrócie **zegarami.** Każdy kolejny krok cyklu rozkazowego (patrz niżej) jest wywoływany kolejnym impulsem zegara. Im szybciej działa zegar, tym szybciej liczy komputer, dlatego informatycy interesują się bardzo tym, czy ich komputer ma zegar o częstotliwości generowanych impulsów wynoszącej 10 MHz[1] czy 50 MHz – oznacza to bowiem ogromną różnicę w szybkości i wygodzie pracy. Częstotliwość zegara w wielu komputerach może być ustawiana przez użytkownika i jest wtedy widoczna na płycie czołowej w postaci cyfrowego wskaźnika.

2.4.1.2. Działanie cyklu rozkazowego



Układ sterujący działa w sposób cykliczny. Oto cztery kolejne **etapy tego cyklu**:

- pobranie rozkazu[2];
- deszyfracja rozkazu[3];
- wykonanie rozkazu[4];
- przejście do następnego rozkazu[5].

Wykonanie programu polega na kolejnym, cyklicznym wykonywaniu wymienionych wyżej czynności. Cykliczność polega na tym, że bezpośrednio po wykonaniu ostatniej z wymienionych operacji (to znaczy po ustaleniu adresu następnego rozkazu) następuje ponowne wykonanie pierwszej czynności, to znaczy pobranie tego

---

[1] Jednostka **MHz** (*megaherc*) oznacza milion impulsów w ciągu jednej sekundy.

[2] Przeniesienie kopii aktualnie wykonywanego rozkazu z pamięci do specjalnego **rejestru rozkazów** w strukturze układu sterującego (w pamięci rozkaz pozostaje nadal i może być powtórnie wykorzystany).

[3] Określenie rodzaju nakazanej do wykonania czynności na podstawie kodu operacji i ustalenie argumentu tej operacji na podstawie części adresowej rozkazu.

[4] Polega ono na sterowaniu przesyłaniem danych i pracą pozostałych podukładów jednostki centralnej zgodnie z treścią wykonywanego rozkazu.

[5] Ustalenie adresu w pamięci, pod którym znajduje się kolejny rozkaz. Przejście to najczęściej wykonuje się przez zwiększanie zawartości specjalnego **licznika rozkazów** w układzie sterującym.

rozkazu do rejestru rozkazów. Taki zamknięty cykl nie ma początku ani końca, zatem uruchamianie i zatrzymywanie programów wymaga specjalnych akcji.

Rozpoczęcie pracy programu polega na wprowadzeniu (zazwyczaj za pośrednictwem systemu operacyjnego) adresu pierwszego rozkazu do licznika rozkazów i zainicjowaniu w ten sposób całego cyklu (opcję tę można nazwać skrótowo START).

Zatrzymanie programu możliwe jest na kilka sposobów. Wśród rozkazów wykonywanych przez dowolny komputer znajduje się zawsze rozkaz przerwania obliczeń (nazwijmy go skrótowo STOP). Wykonanie takiego rozkazu przerywa obliczenia i przekazuje sterowanie ponownie do systemu operacyjnego. Jest to normalne zakończenie pracy programu. Możliwe są jednak także zakończenia nienormalne. Zakończenia takie związane są z określonymi **błędami**.

Błędów może być wiele rodzajów i przytoczone tu uwagi mają charakter pewnej ilustracji, nie zaś pełnego, wyczerpującego studium. Pierwszy rodzaj błędu, z którym się musimy liczyć, dotyczy **nieprawidłowej budowy użytego rozkazu**. Taki niepoprawnie zbudowany rozkaz nie daje się właściwie zinterpretować na etapie deszyfracji i maszyna wykrywa błąd. Dla odróżnienia od innych omawianych błędów nazwiemy go BŁĘDEM SKŁADNIOWYM (*syntax error*), chociaż termin ten bywa także używany w innych kontekstach. Nie jest to jedyny rodzaj możliwego błędu, gdyż poprawnie zbudowany rozkaz może nakazywać nieprawidłowe (z merytorycznego punktu widzenia) działanie. Przykładowo również na etapie deszyfracji możliwe jest pojawienie się NIEPRAWIDŁOWEGO ADRESU. Jeśli program usiłuje "sięgnąć" do nie istniejących (lub niedostępnych dla niego) obszarów pamięci – wówczas również zostaje przerwany.

Jeśli etap deszyfracji przebiegł bez zakłóceń, to nie oznacza, że rozkaz jest poprawny. Jego złe działanie może się bowiem ujawnić dopiero na etapie wykonania (na przykład poprawny rozkaz dzielenia wykorzystujący prawidłowe adresy może – na skutek podania błędnych danych – oznaczać dzielenie przez zero). Powstaje wówczas tak zwany BŁĄD WYKONANIA (*execution error*).

Omawiając różne wyjątkowe sytuacje w schemacie działania systemu sterowania komputera, niepodobna pominąć faktu, że niekiedy zamiast pełnego (wyżej omówionego) cyklu rozkazowego wykonywany jest "obieg skrócony". Dzieje się tak w przypadku wykonywania tak zwanych INSTRUKCJI SKOKU (*jump*). Normalny tryb działania komputera jest sekwencyjny, to znaczy instrukcje są wykonywane w takiej samej kolejności, w jakiej je napisano w programie i umieszczono w pamięci komputera. Niekiedy jednak ten regularny, sekwencyjny tok wykonywania instrukcji musi zostać zakłócony, gdyż z zastosowanego algorytmu wynika, że jako kolejna wykonywana instrukcja powinna wystąpić (w sposób warunkowy lub bezwarunkowy – por. rozdz. 5) jakaś inna instrukcja. Używa się wówczas rozkazu, którego działanie polega na wskazaniu tej następnej instrukcji do wykonania. Rozkaz taki nie ma w praktyce fazy wykonania, lecz po jego deszyfracji następuje bezpośrednio przejście do wskazanego rozkazu.

2.4.1.3. Mikroprocesory i komputery jednoukładowe

Aby z mikroprocesora zrobić komputer, trzeba jego strukturę uzupełnić przynajmniej o **pamięć** (**RAM** i zwykle także **ROM** – patrz porozdz. 2.4.2), **magistralę** i jej układ sterowania oraz przynajmniej podstawowe **układy wejścia – wyjścia.** Wprowadzone wyżej rozróżnienie pomiędzy jednostką centralną a mikroprocesorem pozwala między innymi dobrze określić, co jest czym w nowoczesnych, budowanych obecnie, maszynach wieloprocesorowych. W maszynach takich występuje nadal dająca się zidentyfikować jednostka centralna, która jednak zawiera pojedynczą (na ogół) pamięć i wiele oddzielnych, współpracujących ze sobą mikroprocesorów[1] (komunikujących się na

[1] Nowoczesne mikroprocesory bywają wyposażone w pamięć, na przykład ceniony mikroprocesor *Motorola 68040* ma wbudowane dwie pamięci "podręczne" (*cache*) o pojemności 4 kB każda. Nie jest to jednak pamięć "operacyjna", lecz obszerny bufor, którego zadaniem jest zmniejszenie liczby spowalniających pracę kontaktów z magistralą systemową. Dzięki temu mikroprocesor ten osiągał szybkość **20 MIPS** (milionów operacji na sekundę) i długo był w tej dziedzinie rekordzistą (*SPARC* osiągał 18 MIPS, a Intel 80486 – 15 MIPS). Obecnie rekordzistą jest procesor *Alpha*, a często stosowany jest bardzo szybki procesor *Pentium*.

przykład przez wspólną magistralę). Dzięki tym procesorom możliwe jest wykonywanie przez komputer kilku zadań równocześnie lub dzielenie jednego skomplikowanego problemu na kilka zadań wykonywanych równolegle dla zyskania na czasie.

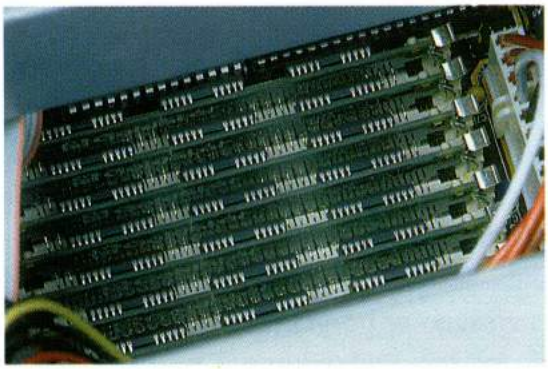
Warto także dodać, że obok mikroprocesorów, których struktura i zakres funkcji zostały omówione, budowane są układy scalone łączące w jednym monolitycznym "chipie" (układzie scalonym) wszystkie elementy kompletnej jednostki centralnej. Urządzenia takie, zwane **sterownikami** (ang. *microcontroller)* lub **komputerami jednoukładowymi**, znajdują jednak dość specyficzne zastosowania (na przykład przy budowie "inteligentnej" aparatury pomiarowej lub przy automatycznym sterowaniu różnych procesów – między innymi nowoczesnych robotów przemysłowych)[1].

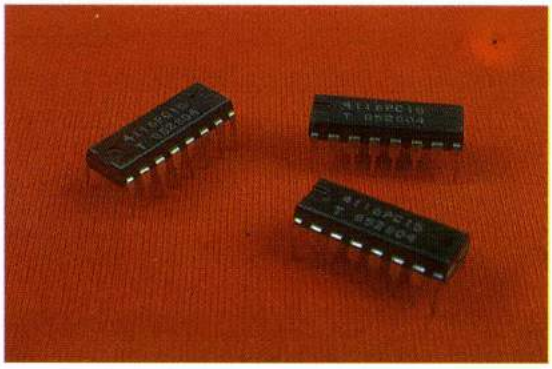
2.4.2. Pamięci wewnętrzne i zewnętrzne

Omawiając procesor komputera wspomnieliśmy kilkakrotnie o pamięci, jako że jest ona nieodzowną częścią jednostki centralnej. Ponieważ jednak pamięć stanowi szczególnie ważny (i szczególnie kosztowny, niestety) element każdego komputera, przeto powracamy do tego tematu i zajmiemy się nim nieco bardziej szczegółowo.

2.4.2.1. Pamięć operacyjna i jej właściwości

Pamięć zawarta w jednostce centralnej nazywana jest **pamięcią operacyjną**, gdyż w niej wykonywane są wszystkie operacje: w niej mieści się aktualnie wykonywany **program** i aktualnie przetwarzane dane. Ze względu na technologię wykonania[2] pamięć ta nazywana jest w skrócie **RAM** (*Random Acces Memory).* Musi być ona bardzo szybka. Szybkość tę określa się podając tak zwany **czas dostępu,** to znaczy odcinek czasu, jaki upływa od momentu, kiedy pewne informacje z pamięci są potrzebne, do momentu, kiedy informacje te zostaną odszukane, wydobyte i udostępnione (na przykład arytmometrowi). Obecnie standardowy czas dostępu wynosi poniżej jednej **mikrosekundy**, a w szczególnie szybkich maszynach osiągane są czasy rzędu kilkunastu **nanosekund**[3]. Jednak tak szybkie pamięci kosztują bardzo drogo. Szczególnie kosztowne są próby łączenia szybkiego działania z dużą pojemnością pamięci. Zatem konstruktorzy systemów komputerowych z reguły stosują pewne rozwiązanie mające charakter kompromisu: w jednostkę centralną wbudowuje się bardzo szybką, ale niezbyt pojemną **pamięć operacyjną**, natomiast z zewnątrz dołącza się do komputera bardzo pojemne **pamięci masowe**.





[1] Jako ciekawostkę można odnotować fakt, że komputer jednoukładowy wykorzystywany jest w maszynach standardu IBM PC do ... samej tylko obsługi klawiatury.

[2] Szybkie urządzenia pamięciowe, mogące pełnić rolę pamięci operacyjnej, budowano początkowo opierając się na różnych zasadach fizycznych. W pierwszym komputerze świata, maszynie Mark 1, **Terence Williams** zastosował w **1948** roku specjalne lampy pamięciowe *(capacitron).* W **1953** roku **James Forrester** wynalazł pamięć ferrytową w postaci pierścieniowych rdzeni magnetycznych z przeplecionymi przewodami. Pierwsze rdzenie miały średnicę 80 mm, ale już wkrótce standardem stała się pamięć na rdzeniach o średnicy 1,2 mm. Pamięć rdzeniowa dominowała w produkcji komputerów do końca lat siedemdziesiątych i pozostawiła wiele śladów w terminologii informatycznej (na przykład znane zabawy programistów pod nazwą "wojny rdzeniowe"). Obecnie stosowane pamięci są bez wyjątku budowane jako układy scalone **VLSI.**

[3] Warto przypomnieć sobie, że **nanosekunda** to jednostka czasu, w czasie której światło w próżni przebywa około **30 cm.**

2.4.2.2. *Pojemność pamięci i jej jednostki*

Mówiąc o pojemności pamięci powinniśmy posługiwać się jakąś określoną jednostką, gdyż inaczej określenia "duża pojemność" lub "mała pojemność" stają się eufemizmami i tracą konkretny sens. Obecnie niemal bez wyjątków komputery posiadają pamięć o organizacji **bajtowej**. Jest zatem rzeczą oczywistą i logiczną, że pojemność pamięci tych systemów wyraża się w **bajtach** (ang. *byte*). Jednak jeden bajt jest bardzo małą jednostką ilości informacji. Ileż bowiem treści może się zmieścić na 8 bitach? Łatwo policzyć: zaledwie $2^8 = 256$ rozróżnialnych stanów. Wystarcza to do zakodowania jednego znaku, lecz jest to o wiele za mało, by przedstawić jakąkolwiek interesującą liczbę.

Zatem jednostka pamięci musi być **wielokrotnością** bajtu, przy czym ze względów technicznych wygodnie jest, jeśli jest to całkowita potęga liczby **2**. Znaleziono taką jednostkę: jest nią tzw. **1 KB** (czytany – nieprecyzyjnie, jak się zaraz okaże – jako **kilobajt**). Dokładnie 1 KB = 2^{10} = **1024** bajty. Jest to nieco więcej niż tysiąc (stąd niedokładne jest odczytywanie tego jako *kilo...*), ale z zadowalającą w praktyce dokładnością można uważać, że maszyna ma tyle tysięcy bajtów, ile wynosi jej pojemność wyrażona w KB.

Mikrokomputery typu PC mają pamięci o pojemności przynajmniej **640 KB**, jednak zwykle dysponują pamięciami od **1 MB** do **16 MB**[1]. Buduje się już dziś komputery o pamięciach wewnętrznych wyrażanych w *gigabajtach*[2]. Jednak nasza krajowa codzienność to – ze względu na ceny – pojedyncze megabajty. Warto dodać, że nowoczesne oprogramowanie (zwłaszcza współpracujące z tak zwanym interfejsem graficznym) jest bardzo "pamięciożerne". Do niedawna trudno było znaleźć program, który czyniłby jakiś sensowny użytek z "nadwyżki" pamięci komputera IBM PC powyżej standardowych 640 KB (najczęściej obszar ten wykorzystywano jako tak zwany RAM–DISK, czyli zasób pamięci wykorzystywanej podobnie jak dyski twarde lub dyski elastyczne tylko bardzo szybkiej), ale teraz programy pracujące w systemie MS WINDOWS potrzebują przynajmniej 4 lub 8 MB pamięci. Oznacza to w sumie wzrost kosztów komputeryzacji, ponieważ ceny sprzętu elektronicznego maleją wolniej, niż wzrasta zapotrzebowanie na pamięć.

2.4.2.3. *Powody stosowania pamięci masowych*

Ustaliwszy, jaką pojemnością dysponują pamięci wewnętrzne dostępnych komputerów, możemy teraz przejść do (krótkiego z konieczności) omówienia **pamięci masowych**[3]. Zacząć trzeba od stwierdzenia, że pamięci tego typu **muszą** być stosowane, przy czym obok szeroko dyskutowanych przyczyn związanych z ograniczoną, często niewystarczającą pojemnością pamięci wewnętrznej, konieczność ich stosowania pojawia się również w związku z faktem, że **pamięć wewnętrzna komputera jest nietrwała.** Mówiąc dokładniej: w wykonaniach, jakie obecnie stosują producenci komputerów, pamięć wewnętrzna przechowuje informacje tylko tak długo, jak długo jest zasilana. Wystarczy zatem wyłączenie komputera z sieci – celowe, w związku z zakończoną pracą, lub incydentalne, związane z zanikiem zasilania w sieci energetycznej, a cała pracowicie gromadzona zawartość pamięci w ułamku sekundy znika. Tracone są w ten sposób wszystkie napisane programy, zebrane dane, wyliczone wyniki itp. Jest to cena, jaką przychodzi płacić za miniaturyzację pamięci, gdyż dawniej stosowane pamięci rdzeniowe były trwałe!

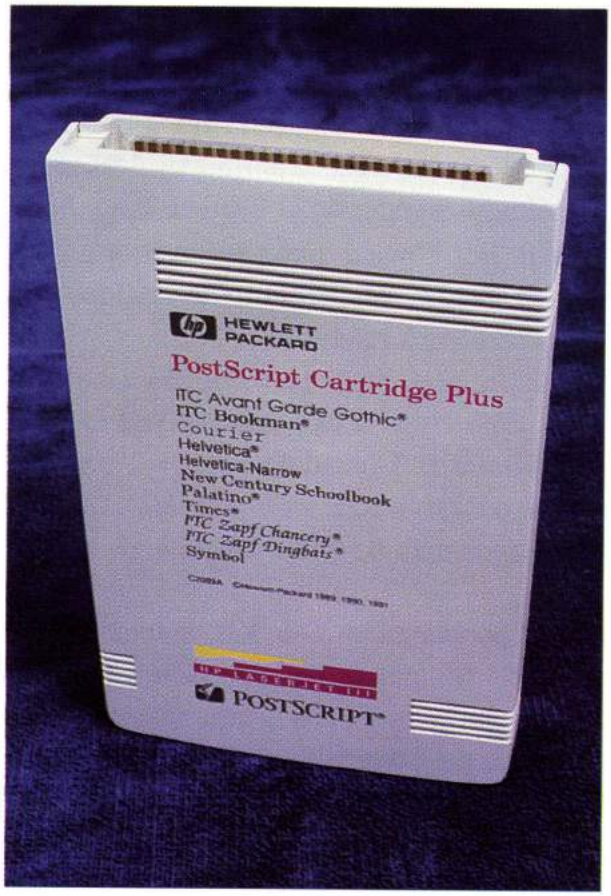
Naturalnie można zbudować półprzewodnikową pamięć nie tracącą swojej zawartości po każdym wyłączeniu zasilania. Takie pamięci buduje się i używa – lecz trwałość tych pamięci jest (w najtańszych wykonaniach) **zbyt duża**: można w nich raz jeden zapisać określone informacje i nigdy

---

1 Skrót **MB** odczytywany jest jako *megabajt*, co jest podobnym nadużyciem, jak *kilobajt*, bowiem 1MB = 2^{20} bajtów.

2 **1 GB** = 2^{30}, czyli ponad **miliard** bajtów!

3 Najwcześniej zastosowanym rodzajem pamięci masowej były karty dziurkowane. Karty te, wynalezione w **1804** i używane początkowo wyłącznie do zapamiętywania wzorów tkanin w tzw. krosnach żakardowskich, ulepszył i przystosował do potrzeb informatyki **Herman Hollerith.** Karty Holleritha zostały z powodzeniem użyte do opracowywania danych ze spisu powszechnego w USA w **1890** roku, a w **1928** roku firma **IBM** wypuściła na rynek pierwsze maszyny licząco–księgujące, wykorzystujące karty perforowane.



więcej nie można ich zmienić. Tego typu pamięci mają także zastosowanie przy budowie komputera jako tak zwane pamięci **ROM** (*Read Only Memory* – pamięć tylko do odczytywania). Na takich pamięciach wprowadza się do komputera stałe elementy oprogramowania (np. podstawowe części systemu operacyjnego, a także niektóre programy oferowane jako tzw. *cartridge* – patrz fotografia). Są wprawdzie odmiany pamięci ROM pozwalające w pewnym zakresie wymieniać zawarte w nich informacje: na przykład pamięci **PROM** "przepalane" dopiero u użytkownika na specjalnych programatorach, a także pamięci **EPROM**, których zawartość można skasować za pomocą naświetlania promieniowaniem **UV** (ultrafioletowym) oraz pamięci **EEPROM,** które mogą być kasowane i powtórnie zapisywane elektrycznie. Jednak z punktu widzenia typowego użytkownika są to nadal pamięci jedynie do odczytu, gdyż siedząc przy komputerze nie można zmieniać ich zawartości.

Tymczasem do wykonywania podstawowych w każdym komputerze czynności: przyjmowania nowych programów, akceptacji wejściowych danych, wyliczania wyników itp. – konieczne jest dysponowanie pamięcią umożliwiającą zarówno zapis, jak i odczyt informacji – a taką jest jedynie pamięć **RAM**, niestety nietrwała.

2.4.2.4. Budowa pamięci masowych

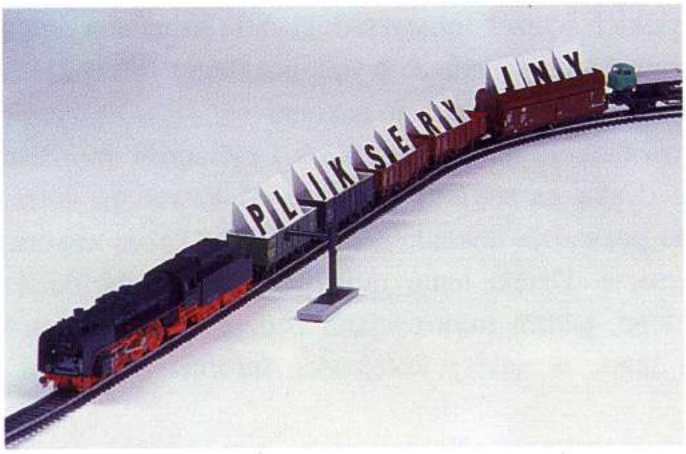
Tak więc pamięć masowa jest niezbędna w każdym komputerze. Istnieje wiele rodzajów pamięci masowych i wiele technologii ich wytwarzania. Jednak największe znaczenie mają obecnie pamięci na **nośnikach magnetycznych** i do nich ograniczymy dalszy opis. Zasada działania wszystkich pamięci na nośnikach magnetycznych jest identyczna z zasadą działania magnetofonu lub magnetowidu: Na odpowiednim nośniku magnetycznym (**taśmie, dyskietce** lub **dysku sztywnym**) nagrywane są informacje przesyłane przez komputer lub odczytywane są i przesyłane do komputera informacje uprzednio zapisane, przy czym fizyczna zasada zapisu opiera się na tych samych zasadach, co w fonografii lub technice video, a jedynie postać zapisywanych informacji stanowi pewną osobliwość – są to bowiem bity i bajty informacji cyfrowej, a nie dźwięki lub obrazy[1].

Pojemność, szybkość działania, wygoda użycia (ale i koszty) pamięci masowych zależą od rodzaju użytego **nośnika**, czyli materiału, na którym dokonywany jest zapis przechowywanych przez komputer informacji. Możliwe są tu różne rozwiązania, gdyż w historii informatyki znane są próby zapisu informacji niemal na wszystkim.

[1] Ta odmienność zresztą już niebawem należeć może do przeszłości, gdyż zalety zapisu cyfrowego spowodują, zapewne już wkrótce, zastąpienie analogowych (obecnie stosowanych) technik rejestracji obrazu i dźwięku – technikami cyfrowymi.

Najtańsze są niewątpliwie **taśmy magnetyczne**[1]. Taniość obejmuje w tym wypadku zarówno sam nośnik (kaseta taśmy jest znacznie tańsza od dysku), jak i koszt urządzeń zapisujących i odczytujących informację (tzw. *streamerów).* Niestety jednak, poza niskim kosztem taśmy nie prezentują w żadnym zakresie korzystnych właściwości. Szczególnie dotkliwa jest ich mała szybkość pracy[2]. Zapis lub odczyt informacji komputerowych na taśmie może trwać nawet kilkanaście minut – jest to czas szokująco długi w porównaniu z szybkością wszystkich innych działań komputera, których czas trwania wyraża się w ułamkach sekund i jest zwykle dla człowieka niezauważalny.

Gdyby wolny czas dostępu do zgromadzonych informacji był jedyną wadą taśm magnetycznych, być może ich niski koszt mógłby tę wadę na tyle skutecznie równoważyć, że ogółem użycie taśm byłoby znacznie powszechniejsze, niż to ma miejsce obecnie. Jednak taśmy wykazują niestety dalsze, znacznie bardziej dotkliwe wady. Otóż jedyną organizacją danych, jaką można utworzyć na taśmie magnetycznej, jest tzw. **plik seryjny.** Dane (lub programy) zapisane w ten sposób nie mają żadnego logicznego (związanego z ich treścią) uporządkowania, co powoduje, że można je odczytywać jedynie w takiej kolejności, w jakiej były uprzednio zapisane. Jest to bardzo istotne ograniczenie; jakiekolwiek próby wyszukiwania informacji potrzebnych i pomijania informacji zbytecznych są przy użyciu taśm magnetycznych bardzo uciążliwe i wymagają z reguły przeczytania *wszystkich* zapisów, aby skutecznie wybrać te nieliczne, potrzebne. W połączeniu z omawianą wyżej powolną pracą taśm prowadzi to do bardzo niewygodnej pracy z komputerem. Jeszcze gorzej przedstawiają się właściwości taśm magnetycznych, jeśli idzie o poprawianie zarejestrowanych informacji. Aby poprawić nawet jeden tylko bajt w liczącym kilka milionów bajtów pliku taśmowym, należy przekopiować całą taśmę – od początku do końca!

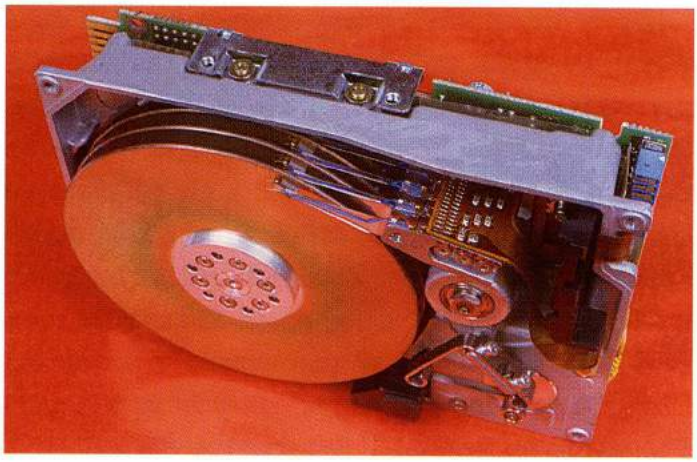


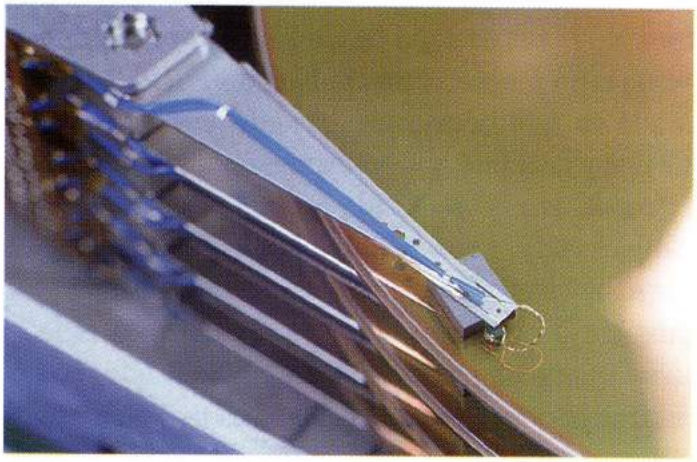
Wszystkie wymienione wady taśm spowodowały, że od dawna poszukiwano doskonalszych pamięci masowych. Obecnie takim najdoskonalszym urządzeniem pamięciowym jest **dysk magnetyczny**, znajdujący się wewnątrz komputera. Budowa dysku przypomina nieco płytę gramofonową,

---

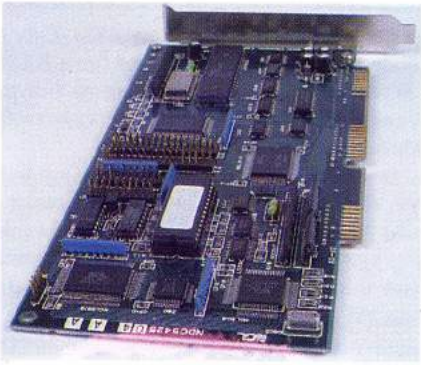
[1] Pierwsze taśmy magnetyczne pojawiły się w **1934** roku i znalazły zastosowanie w magnetofonach niemieckiej firmy **AEG.** Były to jednak oczywiście taśmy przeznaczone do zapisu dźwięku, czyli sygnału analogowego, a ich producentem była znana do dziś firma BASF. Natomiast technikę magnetycznego zapisu cyfrowego opracowała w **1945** roku firma **IBM.** Wynalazek ten wykorzystała firma *Univac*, która w **1951** roku wypuściła pierwszą pamięć cyfrową na taśmie magnetycznej. Urządzenie to, nazwane *Univac Universo,* wykorzystywało taśmę metalową o długości 37 km i stało się prototypem używanych do dziś stacji pamięci na taśmach magnetycznych.

[2] Pamięci na taśmach magnetycznych miały obok innych wad także i tę dodatkową, że zakładanie i zdejmowanie taśmy było czynnością uciążliwą i trudną. Z nostalgią wspominamy ośrodek obliczeniowy AGH, w którym obok rewelacyjnej (w tamtych czasach!) **Odry 1304** stało kilka dużych szaf z kręcącymi się szpulami taśmy, a ciągłe bieganie z nowymi rolkami, zakładanie, zdejmowanie, przekładanie itd. pozwalało informatykom na zachowanie doskonałej formy fizycznej i szczupłej sylwetki – nieosiągalnej dziś, kiedy przy pojemnych dyskach ten sam efekt osiąga się naciśnięciem jednego klawisza. Jednak współczesne taśmy magnetyczne też nie dają możliwości rozwijania kultury fizycznej wśród programistów: wprowadzona w **1971** roku przez firmę **3M** ćwierćcalowa kaseta z taśmą wraz z pochodzącym z **1956** patentem firmy *Ampex* na wirujące głowice dają możliwość wygodnego operowania taśmą o rozmiarach pocztówki i pojemności kilku gigabajtów.





organizacja zapisu na dysku



gdyż jest to płaska, okrągła, wirująca tarcza, na której informacje zapisywane są na górnej i dolnej powierzchni. Jednak na tej powierzchownej analogii podobieństwa między dyskiem i płytą gramofonową definitywnie się kończą. Zasada zapisu informacji na dysku jest bowiem identyczna, jak na taśmie, podobny skład ma także substancja magnetyczna powlekająca dysk i podobną budowę mają piszące lub czytające głowice.

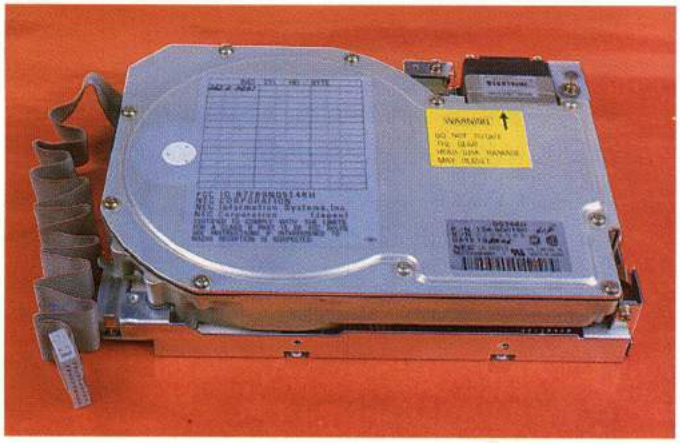
Zapis informacji na dysku magnetycznym jest dokonywany na ścieżkach mających formę koncentrycznych okręgów. Ścieżek tych na powierzchni dysku może być dużo – na przykład 80. Głowice zapisujące i odczytujące informacje przesuwane są od ścieżki do ścieżki przez precyzyjny silnik krokowy. Ponieważ niekiedy (dla zwiększenia pojemności) grupuje się kilka dysków na jednej osi (jeden nad drugim, z głowicami wchodzącymi w szczeliny pomiędzy wirującymi dyskami), przeto w organizacji zapisu na dysku wyróżnia się tak zwane **cylindry** złożone ze wszystkich ścieżek leżących dokładnie jedna nad drugą (a więc dostępnych przy jednakowym położeniu wszystkich głowic) na wszystkich dyskach całego pakietu.

Poza wyróżnieniem na dysku ścieżek i cylindrów możliwe jest podzielenie dysku na **sektory** podobne do kawałków tortu. Sektory i ścieżki pozwalają dokładnie zaadresować każdą zawartą na dysku informację. Dzięki temu możliwe jest jej odszukanie niezależnie od tego, jakimi informacjami będzie ona otoczona i niezależnie od tego, w jakiej kolejności informacje te były zapisywane.

Oczywiście lokalizacja informacji na powierzchni dysku, chociaż dzięki opisanej organizacji jest możliwa, jednak nie robi się sama. Wyszukiwanie potrzebnych danych, ustawianie głowic na poszczególne ścieżki, odliczanie kolejnych "przelatujących" pod głowicą sektorów, bajtów i bitów – wykonuje tzw. **sterownik dysku** (ang. *controller).* W użyciu są różne sterowniki twardych dysków o zróżnicowanych cenach i możliwościach. Najtańsze, ale i najgorsze, są sterowniki ST506 lub MFM. Nieco bardziej nowoczesne są sterowniki SCSI, przydatne przy tradycyjnym sposobie użytkowania komputera. Natomiast najefektywniejsze są sterowniki ESDI lub IDE, zwłaszcza że te ostatnie nie blokują miejsca na kartach rozszerzeń komputera, jako że cała część elektroniczna jest wbudowana w stację.

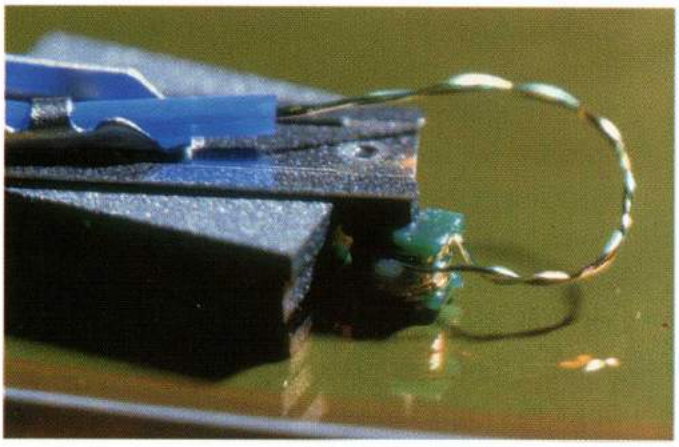
Obok wymienionych już zalet, dyski magnetyczne mają także inne atuty. Podstawowym jest oczywiście szybkość

działania. Tak zwane "twarde" dyski zapewniają zapis i odczyt informacji z szybkością kilkaset razy większą od tej, jaką oferują taśmy. Oznacza to, że operacja (na przykład wczytanie programu do komputera), która trwa przy użyciu taśmy kilkanaście minut – może być wykonana za pomocą dysku w kilka sekund. W ten sposób pamięć masowa na dyskach magnetycznych tworzy techniczne przesłanki dla podejmowania pewnych działań (na przykład budowania systemów informujących "na poczekaniu" o określonych sprawach lub zjawiskach), podczas gdy brak dysków te same operacje uniemożliwia. Warto uświadomić sobie co powoduje, że dyski twarde mają tak dużą szybkość działania. Po pierwsze, o czym już była mowa, z dysku czyta się tylko te informacje, które są potrzebne, bez jałowego przewijania zajmującego tak wiele czasu przy operacjach z taśmami. Po drugie dysk stale szybko wiruje i dlatego nie traci się czasu – jak przy taśmach – na jego rozpędzanie i zatrzymywanie. Ponadto obrotowy ruch dysku powoduje, że określone informacje co chwilę same "podjeżdżają" pod głowice, zatem wystarczy poczekać – średnio połowę czasu jednego obrotu dysku. A dysk wiruje z szybkością dziesięciu tysięcy obrotów na minutę! Wreszcie bardzo duża gęstość zapisu przyspiesza operacje zapisu i odczytu.



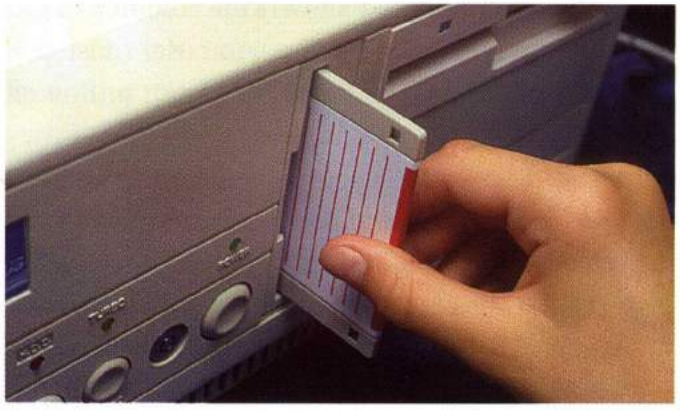
W dodatku do wymienionych już zalet dyski twarde mają bardzo małe rozmiary. Stosunkowo łatwo dostępny dysk o pojemności od **20** do **600 MB** (sześćset milionów bajtów!) ma średnicę około 10 cm. Warto przy tym uświadomić sobie, że nie są to bynajmniej największe pojemności (dostępne bywają dyski o podobnych rozmiarach i pojemności kilku gigabajtów, co odpowiada kilku milionom stronic standardowego maszynopisu).

Niestety, omówione wyżej dyski mają także swoje wady. Do najpoważniejszych wad tych urządzeń zaliczyć trzeba **cenę**: są to urządzenia tak drogie, że cena dobrego dysku może przewyższać cenę popularnego mikrokomputera! W dodatku niesłychanie gęsty zapis informacji na powierzchni dysku wymaga, by głowice zawieszone były nad jego powierzchnią na wysokości kilku mikronów. Warto uświadomić sobie, że tego samego rzędu wielkość mają ... bakterie, nie wspominając o znacznie większych drobinach kurzu. Z tego powodu zarówno produkcja, jak i eksploatacja twardych dysków wymagają szczególnej pieczołowitości. Jest to obecnie jedyna istotniejsza wada tych urządzeń, gdyż w związku z technologicznie trudną produkcją są one (i będą nadal) stosunkowo drogie, a ponadto mogą łatwo ulegać uszkodzeniom pod wpływem wstrząsów, zaś naprawa ich jest praktycznie niemożliwa.



Jak zawsze w przypadku, kiedy mamy do wyboru coś dobrego i drogiego (dyski twarde) lub coś taniego i kiepskiego (taśmy magnetyczne), wybieramy ... coś ze środka. Tym środkiem (nie wiadomo, czy "złotym"...) w przypadku pamięci masowych okazały się **dyskietki**. Każdy widział zapewne ten krążek elastycznej folii pokrytej masą magnetyczną, zamknięty w ochronnej plastykowej kopercie, nie misimy więc go opisywać.

Dyskietki mają bardzo małe rozmiary: można je bez trudu przechowywać, gromadzić, przewozić i przesyłać pocztą. Trzycalowa dyskietka mieści się bez trudu nawet w kieszonce koszuli, a ma



pojemność odpowiadającą kilkusetstronicowemu skoroszytowi! Używane są obecnie dyskietki o rozmiarze[1] 5,25" i 3,5".

Wadą dyskietek jest ich pojemność. Około jednego megabajta na jednej dyskietce to bardzo wiele, gdy porówna się z innymi tradycyjnymi metodami gromadzenia informacji, jednak w zestawieniu z pojemnościami dysków twardych okazuje się, że to niewiele. Do wad dyskietek można zaliczyć także wyraźnie wolniejszą (w stosunku do twardego dysku) pracę przy zapisie, odczycie i wyszukiwaniu informacji. Wada ta częściowo rekompensowana jest przez fakt, że dyskietki mogą być łatwo wymieniane: po zapełnieniu jednej z nich można jednym ruchem ręki wyjąć ją z komputera i na jej miejsce założyć nową. O pełnej ekwiwalentności dysków twardych i dyskietek trudno jednak mówić i każdy informatyk dąży do tego, by dysponować możliwie dużym twardym dyskiem.

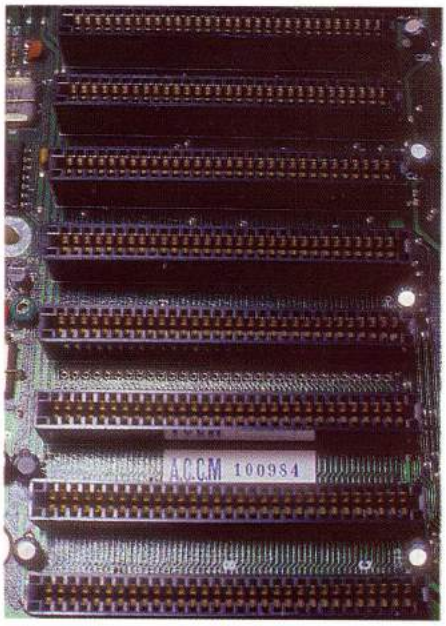
Taśmy i dyski magnetyczne (twarde i elastyczne) nie są jedynymi urządzeniami należącymi do kategorii pamięci masowych. Ostatnio coraz więcej uwagi poświęca się **laserowym dyskom optycznym,** których technikę pod nazwa *compact disk* upowszechniła pierwotnie fonografia. Dyski te mają ogromną pojemność i są dość szybkie w działaniu.

Chociaż dostępne są już także optyczne dyski i dyskietki zapisywalne przez użytkownika, jednak głównie oferowane są tak zwane moduły **CD – ROM**, nadające się wyłącznie do odczytu. Ich głównym zastosowaniem jest dystrybucja zbiorów danych, które są przydatne dla wielu użytkowników i dlatego warto je wydać w formie kupowanego przez wielu ludzi **CD–ROM.** Aktualnie "rekordzistką" jest biblia, której sprzedano ponad 3 mln egzemplarzy w formie dysków optycznych, na drugim miejscu jest encyklopedia.

2.4.3. Magistrala

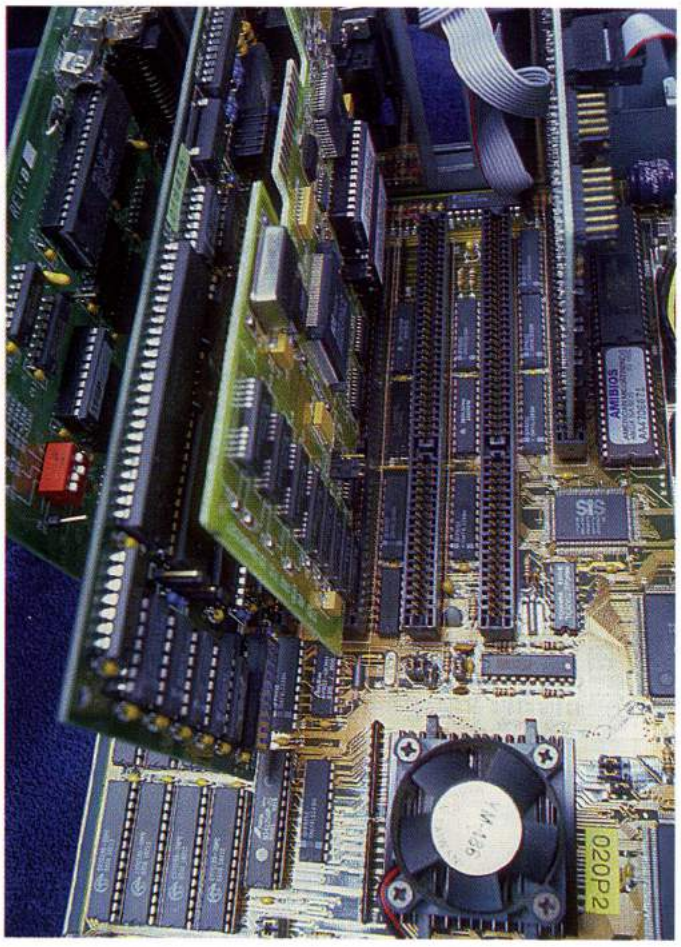
Wszystkie wymienione wyżej podukłady i systemy wchodzące w skład systemu mikrokomputerowego muszą być ze sobą połączone, a także muszą one komunikować się z urządzeniami peryferyjnymi (patrz dalej), zatem musi istnieć spinający je razem system przewodów.

[1] Mimo przyjęcia jako obowiązującego w całym świecie systemu metrycznego w informatyce ciągle operuje się calami. Wymiary dysków i dyskietek podawane są w calach (5,25" czyta się 5 i 1/4 cala).



Koncepcja **magistrali**[1] polega na tym, że *wszystkie układy komputera łączy się do jednej wspólnej linii łączności*, co daje możliwość swobodnej komunikacji typu "każdy z każdym", a także powoduje uporządkowanie i uczytelnienie architektury mikrokomputera, ale może równocześnie wywoływać konflikty w przypadku, kiedy kilka układów równocześnie musi przesyłać lub pobierać informacje. Konflikty te rozstrzygane są przez specjalny **sterownik magistrali,** (zwany także **arbitrem**)[2].

W strukturze magistrali wyróżnia się wiązki przewodów (lub – częściej – grupy ścieżek na drukowanym "platerze") tworzące tak zwane **szyny**. Wyróżnia się *szyny danych, szyny adresowe i szyny sterowania.* Zgodnie z nazwami: szyną danych przesyłane są dane (głównie **z** i **do** pamięci oraz **z** i **do** arytmometru), szyną adresową przesyła się adresy, wskazujące lokalizacje poszczególnych obiektów (na przykład danych i rozkazów) w pamięci, a szyny sterujące przesyłają sygnały sterujące i potwierdzenia wykonanych poleceń **od** i **do** układu sterującego.



Do magistrali dołączone są **układy wejścia/wyjścia,** za pomocą których procesor komunikuje się z **urządzeniami peryferyjnymi** – między innymi odbiera polecenia i dane od człowieka i wysyła na zewnątrz wyniki uzyskane w następstwie obliczeń.

W komputerach o tak zwanej "otwartej architekturze" do magistrali można dołączać (za pośrednictwem tak zwanych *slotów*) dodatkowe urządzenia wejścia/wyjścia, a także inne układy rozszerzające możliwości działania posiadanego komputera. Dzięki temu użytkownik może dobierać sobie taką konfigurację komputera, jaka najlepiej odpowiada jego potrzebom.

2.4.4. Urządzenia peryferyjne

2.4.4.1. Klasyfikacja urządzeń peryferyjnych

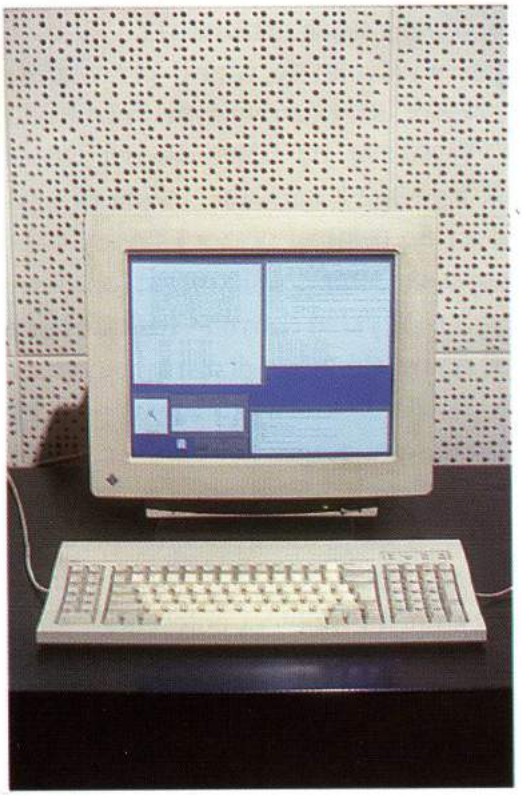
Jedna z kategorii urządzeń peryferyjnych została już wcześniej omówiona. Niewątpliwie bowiem **pamięci masowe** komputera stanowią rodzaj urządzeń peryferyjnych. Jednak obok tych pamięci w skład urządzeń peryferyjnych wchodzą liczne inne urządzenia, które teraz zostaną skrótowo omówione. Najliczniejsza grupa nie omówionych do tej pory urządzeń peryferyjnych związana jest z komunikacją komputera ze "światem zewnętrznym". Urządzenia te określić można jako **urządzenia wprowadzania danych** i **systemy wyprowadzania wyników obliczeń**, a współpracują z nimi

[1] Koncepcja ta wywodzi się z rozwiązania firmy *DEC*, stosowanego pod nazwą *UNIBUS* w ogromnie popularnych minikomputerach **PDP–11.**

[2] O zagadnieniach arbitrażu i problemach rywalizacji kilku urządzeń o dostęp do wspólnej magistrali mowa będzie dodatkowo w podrozdziale 2.5, ponieważ obok magistrali wewnętrznej, łączącej elementy wewnątrz jednostki centralnej, występują także magistrale zewnętrzne, spinające w jeden system kilka oddzielnych komputerów.

wzmiankowane wyżej układy wejścia/wyjścia jednostki centralnej komputera. Poza nimi do urządzeń peryferyjnych zaliczyć można **układy służące do połączenia komputera z procesem**, którym może sterować lub który może być przy użyciu komputera nadzorowany, a także **urządzenia sprzęgające komputery ze sobą** w celu budowy wielkich systemów w formie sieci komputerowych.

2.4.4.2. Konsola (klawiatura i monitor)



Najczęstszym źródłem danych dla komputera jest człowiek, on też jest odbiorcą wyliczonych przez komputer wyników, zatem wśród urządzeń peryferyjnych komputera nie może zabraknąć układów do konwersacyjnej współpracy z człowiekiem. Urządzeniami tymi są **konsole** obejmujące **klawiaturę** i ekran (dokładniej – **monitor**)[1]. W dużych komputerach do jednej jednostki centralnej przyłącza się kilka lub nawet kilkadziesiąt konsoli, pełniących rolę tak zwanych **końcówek** (*terminali*), za pomocą których z jednej maszyny może korzystać równocześnie wielu ludzi.

Konsola konsoli nierówna. Mogą się one różnić stosowanym monitorem (ekran o większej lub mniejszej rozdzielczości, monochromatyczny lub barwny, z możliwością prezentowania grafiki lub tylko samych tekstów itd.), używaną klawiaturą (liczba i sposób rozmieszczenia klawiszy[2], elektroniczne zabezpieczenie poprawnego działania klawiatury[3] itp.) oraz stopniem samodzielności (tzw. terminale proste i "inteligentne").

Klawiatura komputera zawiera zwykle około 100 klawiszy, których naciskanie powoduje generowanie impulsu elektrycznego przekazywanego do odpowiedniego wejścia komputera. Stosowany układ klawiszy przypomina klawiaturę zwykłej maszyny do pisania[4], jest jednak uzupełniony o dodatkowe klawisze.

Na typowej klawiaturze komputera obok klawiszy alfanumerycznych

---

[1] W mikrokomputerach często na takiej konsoli się kończy, gdyż właściciela nie stać na dalsze urządzenia.

[2] Aktualnie używana w komputerach klawiatura ma układ zaproponowany ponad sto lat temu przez **Christophera Scholesa** dla budowanych wówczas maszyn do pisania. Ponieważ jednak maszyny te zacinały się przy zbyt szybkim pisaniu – opracowana klawiatura ma tak rozmieszczone klawisze, aby ... uniemożliwić szybkie pisanie. W zastosowaniu do komputerów jest to jaskrawy anachronizm. Lepsza jest tu klawiatura zaproponowana przez **Augusta Dvoraka** (z *Uniwersytetu Washington*) – jednak tradycja jest silniejsza od rozsądku.

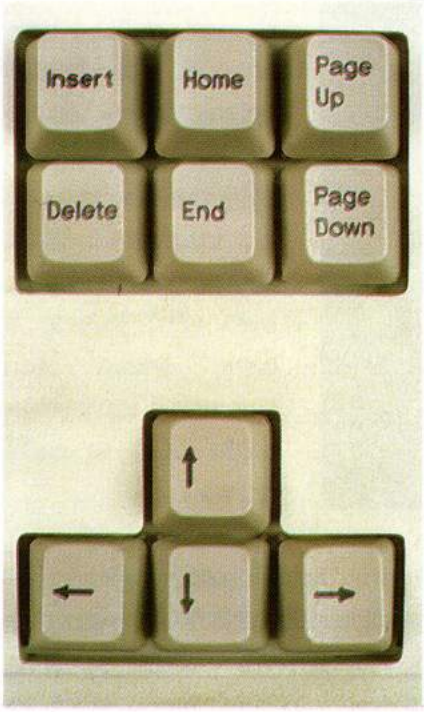
[3] W większych komputerach – na przykład IBM PC – klawiatura wyposażona jest we własny mikrokomputer sterujący i jest funkcjonalnie bardzo rozbudowanym systemem.

[4] W stosunku do polskich maszyn do pisania występują tu dwa odstępstwa, bardzo przeszkadzające na początku osobom biegle piszącym na maszynie: brak polskich znaków oraz przestawione (w stosunku do polskiej normy) pozycje klawiszy **Z** i **Y**.

wyróżnić można klawisze specjalne, między innymi: zmiany dużych liter ma małe i odwrotnie, oznaczone **Shift** i **CapsLock,** a także klawisze, pozwalające zmieniać znaczenie aktualnie naciskanych klawiszy[1] **(Ctrl** i **Alt)** oraz końca linii i potwierdzenia polecenia, oznaczony **Enter**.

Na uwagę zasługują też klawisze numeryczne (wydzielone w postaci osobnego bloku) oraz tzw. klawisze techniczne, np. **NumLock, ScrollLock, Ins, Delete, Pause, Break, Esc, PrtSc**. Z klawiszy tych w pierwszej kolejności odszukaj na swojej klawiaturze klawisz **Backspace** pozwalający usunąć ostatnio napisany znak oraz klawisz **Esc,** za pomocą którego porzuca się aktualnie wykonywaną czynność.

Osobną grupę stanowią klawisze funkcyjne (**F1, F2, ..., F10**), realizujące dodatkowe zadania[2].



Do tego dochodzą klawisze sterowania ruchem kursora[3] (**←→↑↓**, **Home, End, PgUp, PgDn**), pokazane obok. Klawisze oznaczone strzałkami przesuwają kursor o jedną pozycję we wskazanym kierunku (w lewo, w prawo, w górę lub w dół). Klawisz **Home** przemieszcza zwykle[4] kursor na początek, a klawisz **End** na koniec aktualnie pisanej linii tekstu. Wreszcie klawisz **PgUp** przemieszcza kursor do szczytu ekranu, a **PgDn** do dołu ekranu lub do końca aktualnie wypisanego tekstu. Ruchy kursora pod wpływem naciskania wymienionych klawiszy bywają modyfikowane równoczesnym naciśnięciem klawiszy **Ctrl, Shift** lub **Alt.**

Klawiatura umieszczona bywa niekiedy we wspólnej obudowie z mikrokomputerem, chociaż częściej stosowana jest oddzielna klawiatura, połączona z komputerem spiralnie zwiniętym kablem. Pozwala to na swobodne obsługiwanie klawiatury w dowolnym, wybranym przez użytkownika położeniu – na przykład ustawionej na kolanach lub trzymanej w rękach jak gitara (jest taka moda!).

Monitory i współpracujące z nimi **karty graficzne**[5] mogą być rozmaite. Ważniejsze rozwiązania są następujące.

---

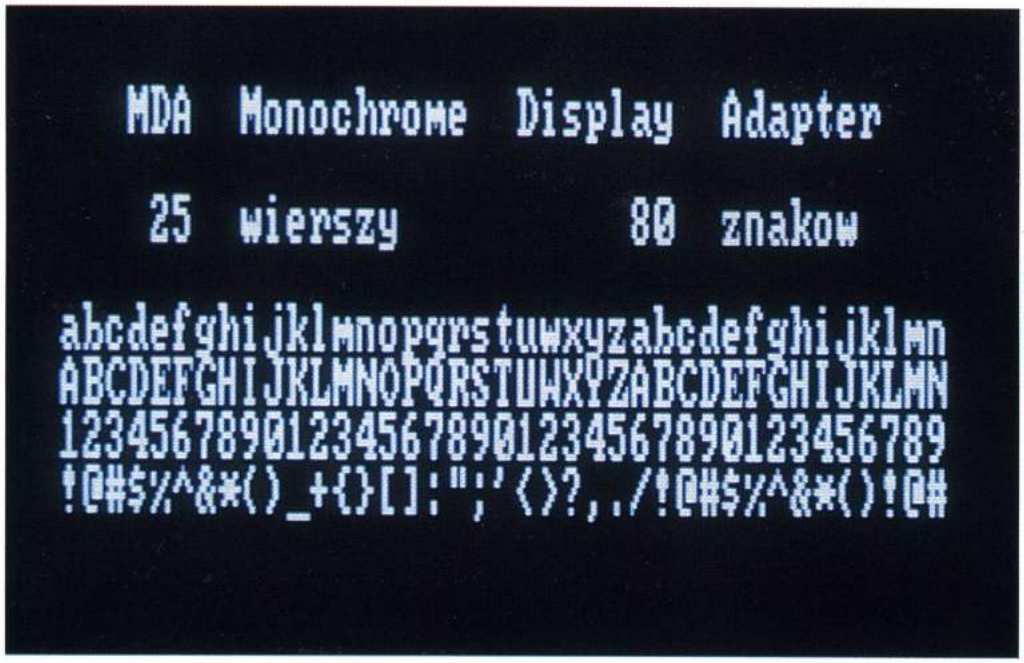
1 W ten sposób każdy klawisz ma przynajmniej cztery rozróżniane przez komputer role: jako "goły", naciskany z **Shift, Ctrl** lub **Alt**.

2 Użytkownik może sam określić ich znaczenie lub używany program wyznacza sposób ich interpretacji, na przykład w wielu systemach naciśnięcie klawisza **F1** przywołuje na ekran tekst wyjaśniający, co w tym momencie można zrobić lub jak osiągnąć zamierzony cel, (tak zwany *HELP).*

3 Kursor jest to znacznik na ekranie wskazujący, gdzie zostanie wpisany następny znak albo jaką należy wykonać czynność.

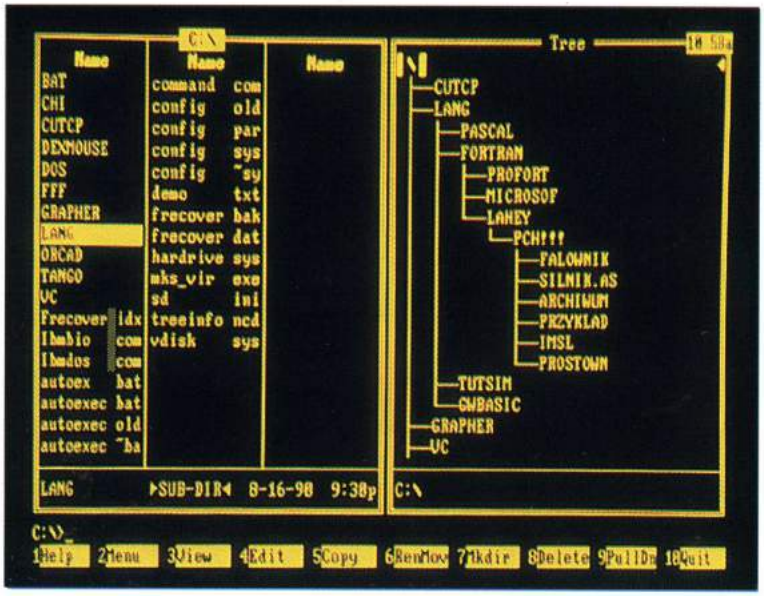
4 Ruchy kursora odpowiadające naciskaniu poszczególnych klawiszy zależne są od rodzaju wykorzystywanego programu.

5 Monitor jest jedynie urządzeniem wyświetlającym obraz, podobnie jak telewizor. Żeby jednak na monitorze można było oglądać teksty lub rysunki produkowane przez komputer – konieczne jest posiadanie układu elektronicznego (tzw. karty graficznej) określającego, które punkty na ekranie mają być jasne, a które ciemne, które kolorowe, a które po prostu czarne. Procesor ma sprawę uproszczoną – po prostu wysyła do odpowiedniego obszaru pamięci kody znaków, które trzeba wyświetlić, a potem wykonuje dalszą cześć programu. Natomiast sterowaniem monitora zajmuje się specjalny sterownik, czyli właśnie wspomniana karta graficzna.



Karta **MDA** *(Monochrome Display Adapter)* była najprymitywniejszą i najtańszą kartą sterującą pracą monitora. Karta ta wcale nie pozwalała tworzyć i wkorzystywać na ekranie komputera grafiki, a jedynie umożliwiała wyświetlanie (jednobarwne) tekstu: 25 wierszy po 80 znaków.

Karta **CGA** *(Color Graphic Adapter)* była znacznie udoskonalona – wprowadzała element grafiki i barwy, ale w bardzo ograniczonym zakresie. Na ekranie można było zmieścić rysunek o rozdzielczości 320 x 200 punktów i można było użyć jedynie 8 barw. Mimo tych ograniczeń karty standardu CGA są nadal używane.



Znacznie lepszą rozdzielczość i bogatszą gamę kolorów zapewnia karta **VGA** (patrz dalej). Nie wszystkich użytkowników jednak na nią stać i dlatego w powszechnym użyciu są też tanie monitory w standardzie ***Hercules*** (720 x 540 punktów), niestety dające obraz o jednej tylko barwie – najczęściej złocistej na czarnym tle (tzw. *amber*). Monitory te pozwalają na wygodną pracę z tekstem (a więc mogą zaspokoić większość potrzeb biurowych), a także z rysunkami – pod warunkiem, że nie muszą to być rysunki barwne (na przykład dobrze wychodzą rysunki techniczne).

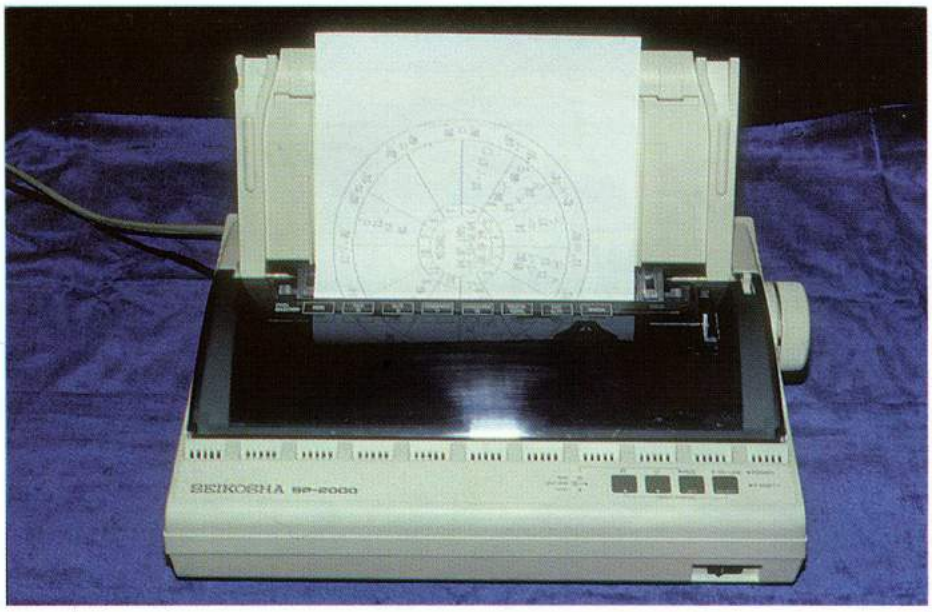
Wspomniana wyżej karta **VGA** wprowadziła nowość – monitor jest w tym standardzie sterowany analogowo, a nie cyfrowo jak przy wszystkich wcześniejszych standardach. Dzięki temu można uzyskać ogromną liczbę wyświetlanych na ekranie barw (256 spośród 262.114), a także możliwe stały się rozszerzenia rozdzielczości (800 x 600 punktów).

Przytoczona dyskusja nie wyczerpała wszystkich możliwości. Na przykład "po drodze" między **CGA i VGA** stosowana była tak zwana karta **EGA,** która zwiększała liczbę dostępnych kolorów (16), dawała większą rozdzielczość (640 x 350) i miała własną pamięć (256 KB), a nie była tak kosztowna jak **VGA**. Obecnie karty **EGA** zostały wyparte przez taniejący standard **VGA.** Z kolei na bardzo zamożnych i wymagających użytkowników czekają – tak zwane wersje **Super VGA** lub **XVGA** o rozdzielczościach 1024 x 768 i 1280 x 1024 i własnej pamięci przekraczającej 512 KB, których sterowniki są podporządkowane standardowi **VESA** (*Video Elctronics Standards Association*). O wszystkich możliwych monitorach nie sposób jednak napisać, gdyż jest ich aktualnie zbyt wiele.

2.4.4.3. Drukarki

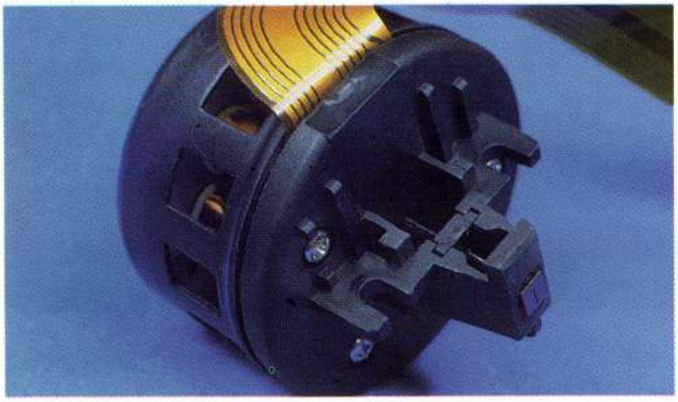
Najlepszy nawet monitor nie zaspokoi wszystkich potrzeb użytkownika komputera, gdyż obraz na ekranie takiego monitora jest nietrwały. Aby uzyskać trwałą kopię (ang. *hard copy*) tych informacji, które interesują nas dłużej, musimy posłużyć się jednym z licznych urządzeń peryferyjnych piszących lub rysujących na papierze. Wśród urządzeń piszących na uwagę zasługują rozmaite **drukarki**.

Wydruk tekstu opracowanego przez komputer lub wyliczonych wyników liczbowych może się odbywać za pomocą dowolnych drukarek, na przykład drukarek znakowo mozaikowych (ang. *dot matrix printer*). Zasada działania takiej drukarki polega na tym, że potrzebny znak tworzony jest przez wiele uderzeń specjalnych elektromagnetycznie napędzanych igieł, które uderzając w papier poprzez specjalną taśmę barwiącą (podobną jak w maszynie do pisania, lecz w drukarkach z reguły



zamkniętą w specjalnych – drogich niestety – kasetach) tworzą kropka po kropce potrzebny zarys wymaganego znaku.

Igły są małe i lekkie, w odróżnieniu od czcionek maszyny do pisania, w związku z czym drukarka mozaikowa może pisać szybciej (około 100 znaków na sekundę) i ciszej niż maszyna do pisania, a ponadto może być mniejsza, lżejsza i tańsza. Co więcej – litery składane z oddzielnych kropek mogą mieć różne rozmiary i kształty. Można na tej samej drukarce, a nawet w tym samym drukowanym dokumencie pisać litery proste i pochylone, poszerzone, wydłużone i pogrubione, lokowane u dołu (jako indeksy w algebrze) i u góry jako wykładniki lub uzupełnienia wzorów chemicznych. Niestety jednak – nic nie ma za darmo. Zalety drukarek mozaikowych okupione są kiepską jakością wydruku. Litery składane z oddzielnych kropek na ogół bardzo się nie podobają użytkownikom. Aby uzyskać lepszą jakość druku, stosuje się szereg zabiegów. Na przykład nakazuje się wielokrotne uderzenia igieł z minimalnym przemieszczeniem piszącej głowicy i papieru. Powoduje to polepszenie wydruku (kropki zlewają się i są niewidoczne), ale za cenę wielokrotnego spowolnienia procesu drukowania. Omawiany tryb pracy drukarki oznaczany jest zwykle skrótem **NLQ** (*near letter quality*) i dostępny jest na wszystkich lepszych drukarkach. Innym sposobem polepszenia jakości wydruku jest używanie większej liczby igieł (24 zamiast 9).



ilustracja przedstawia różne rodzaje fontów DRUKARKI 9 – igłowej

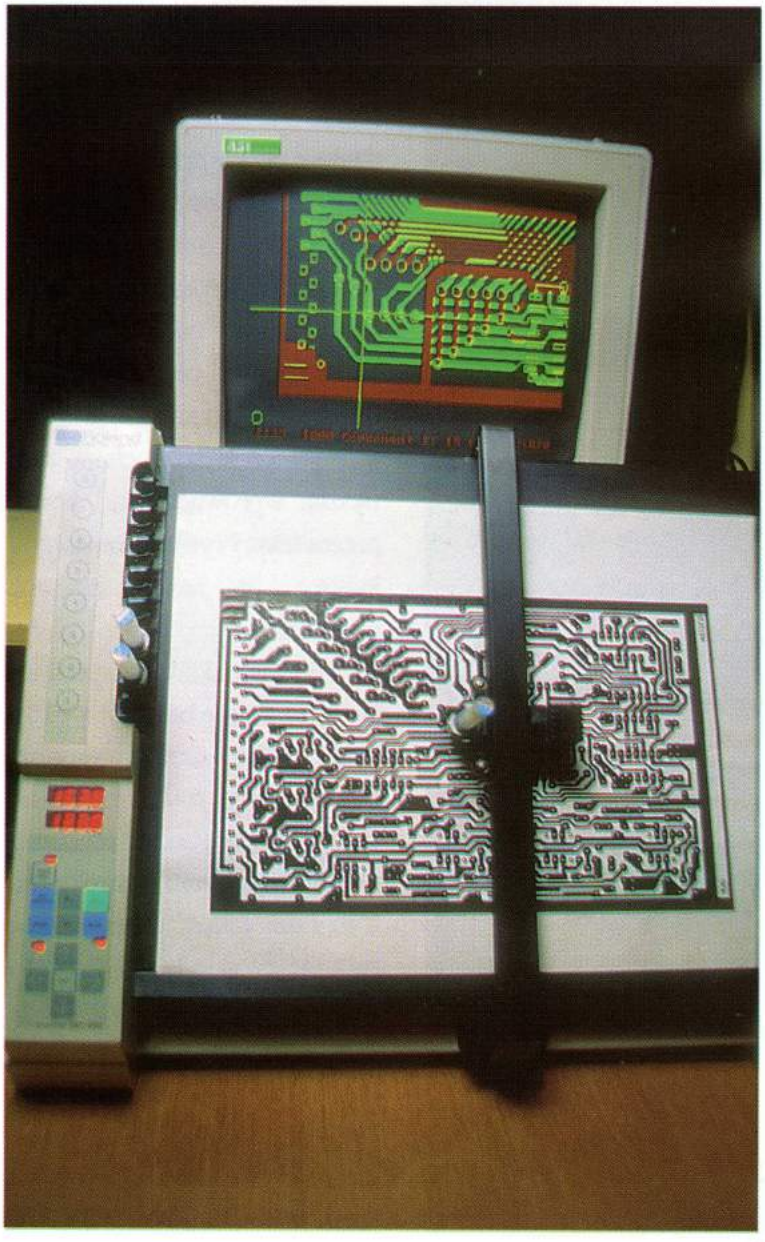
wydruk tekstu w trybie DRAFT na drukarce 24 – igłowej

wydruk tekstu w trybie NLQ na drukarce 24 – igłowej

Do poprawiania jakości wydruku używano także i innych metod. Swojego czasu największe nadzieje budziły drukarki działające na zasadzie szybkiego wyrzucania na papier maleńkich kropel atramentu przez dyszę specjalnej głowicy (tzw. drukarki typu *ink–jet*). Ich zaletą była szybka praca, dobra jakość druku i możliwość uzyskiwania barwnych wydruków o jakości porównywalnej z jakością osiąganą w profesjonalnej poligrafii.

Największym sukcesem rynkowym okazały się jednak **drukarki laserowe** (na fotografii obok). Działają one na zasadzie zbliżonej do popularnego kserografu. Sterowany z komputera laser zapisuje to co ma być wydrukowane na wirującym selenowym bębnie. Zapisane światłem lasera informacje zamieniają się na rozkład ładunków na powierzchni bębna. Naelektryzowane części bębna przyciągają drobinki barwnika (tak zwanego **tonera**) i przenoszą je na papier, gdzie zostają utrwalone przez odpowiednią obróbkę (chemiczną lub termiczną).

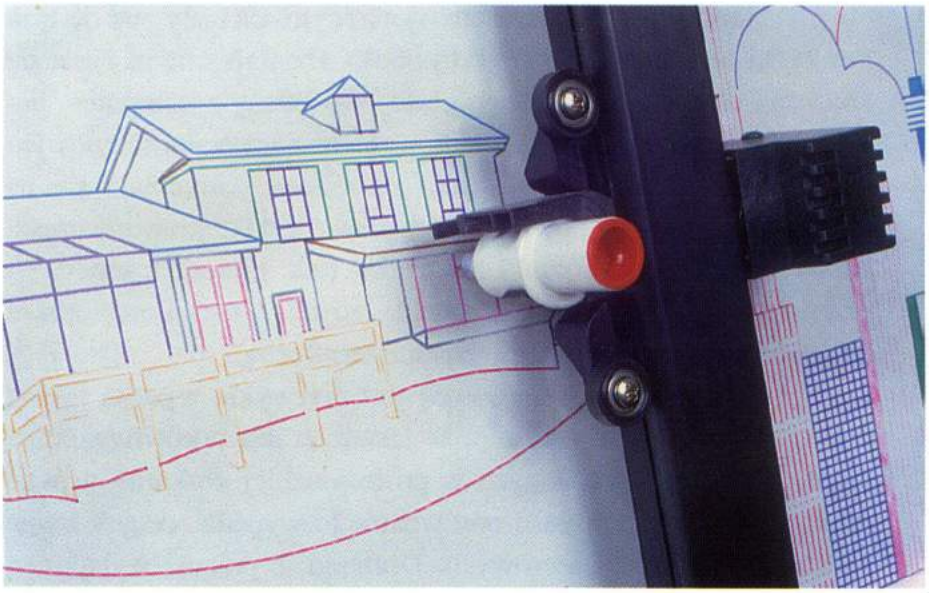
Zaletą drukarek laserowych jest niezwykle wysoka jakość druku. W reklamach pisze się, że jakość ta jest porównywalna z jakością druków wykonanych w profesjonalnej drukarni; w warunkach krajowej poligrafii porównanie to wypada zdecydowanie na korzyść drukarek laserowych. Drukarki te pracują wyjątkowo szybko, znane są modele zdolne do drukowania setek stronic tekstu na minutę. Jednak nie są one wolne od wad: główną jest bardzo wysoka cena. Trzeba ją zresztą płacić dwukrotnie: raz przy zakupie drukarki, której koszt przekracza typową cenę kompletnego komputera, a po raz drugi przy zakupie materiałów eksploatacyjnych: specjalnego papieru (odpornego na wysoką temperaturę i dostatecznie gładkiego), tonera, który się dość szybko zużywa, oraz po pewnym czasie selenowego bębna drukującego. Efekt jest jednak wart tej ceny, co możesz ocenić bezpośrednio w książce – tekst do reprodukcji wydrukowano za pomocą drukarki laserowej.



2.4.4.4. Urządzenia graficzne – plotery, skanery i digitizery

Omówione drukarki umożliwiają wyprowadzanie z komputera grafiki (nawet barwnej), jednak jakość tych rysunków nie jest na ogół najlepsza. Natomiast urządzenie rysujące, tak zwany **ploter,** umożliwia kreślenie dowolnych rysunków z ogromną precyzją[1], w kolorach, z użyciem różnych linii (grubych, cienkich, przerywanych, kropkowanych itd.), a także z uwzględnieniem dowolnych napisów kreślonych pod dowolnym kątem, z dowolną wysokością liter i przy swobodnie wybieranym kroju pisma. Nowoczesne plotery pozwalają

[1] Dokładność pozycjonowania piórka plotera jest porównywalna z grubością kreślonej przez niego kreski (około **0,1 mm**).



na automatyczne kreślenie wybranych fragmentów rysunku (prostokąty, wielokąty, elipsy, koła). Wyróżnia się **plotery płaskie** *(flat-bed plotter)* oraz **bębnowe** *(drum plotter)*. Pierwsze są łatwiejsze w użyciu, tańsze i mogą rysować na praktycznie dowolnym papierze. Jednak ich wadą jest stosunkowo powolna praca i fakt, że zajmują dużo miejsca. Drugie są bardziej kosztowne i wymagające (specjalny papier), ale są szybsze.

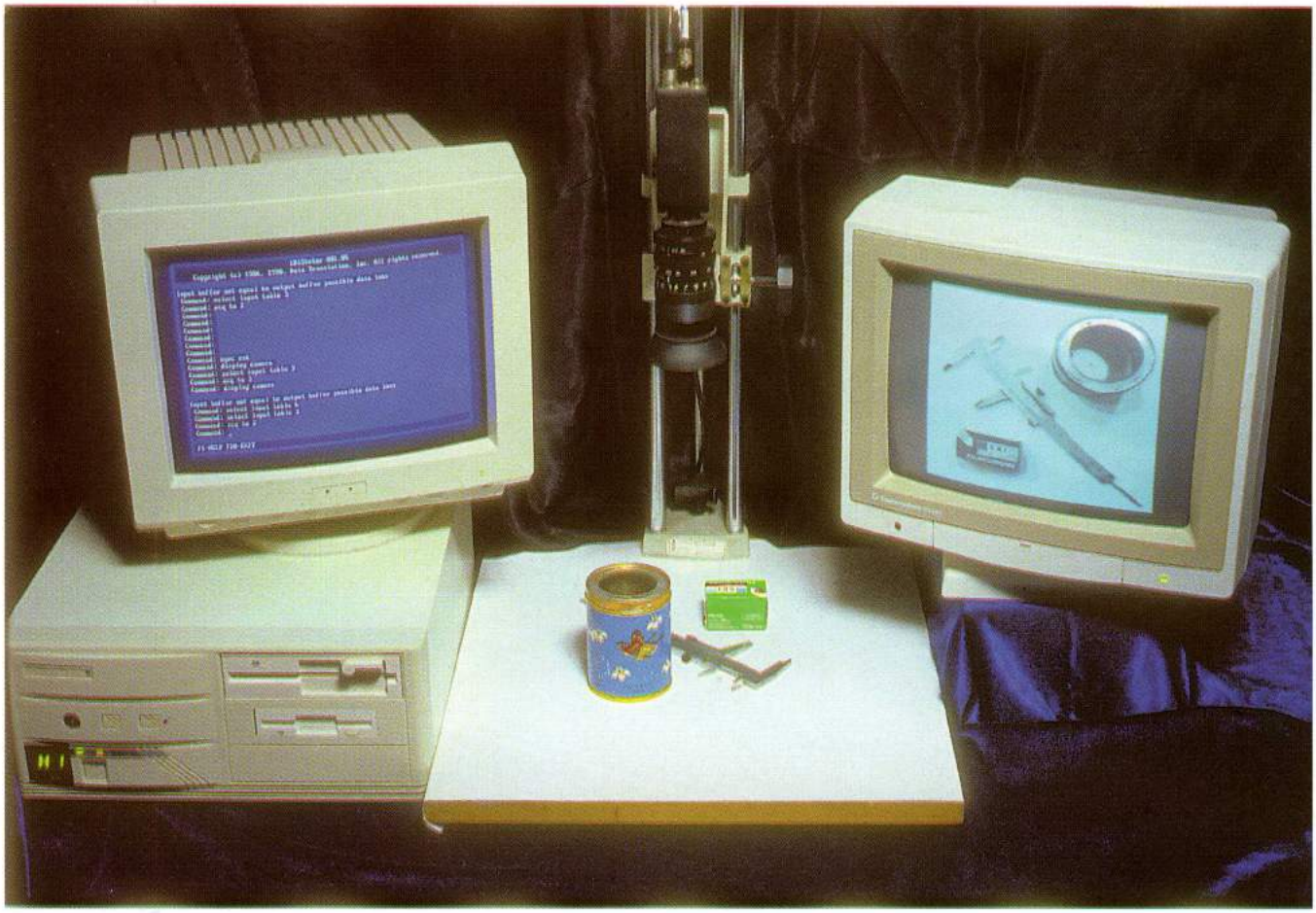
Odwrotną (w stosunku do ploterów) funkcję pełnią **czytniki rysunków.** Można tu wyróżnić urządzenia wprowadzające do komputera całe obrazy na zasadzie pobierania informacji o jasności i barwie wszystkich punktów obrazu (tzw. **skanery** od ang. *scaner).* Na fotografii obok pokazano działanie skanera, za pomocą którego prostym ruchem ręki można wprowadzić do komputera tekst i rysunki zawarte w książce czy nawet napisane odręcznie.

Bywają skanery wprowadzające obrazy barwne, a także takie, które same wykonują mechaniczny ruch "przemiatania" wczytywanego do komputera dokumentu, rysunku lub fotografii. Jakość skanera wyraża się zwykle podając informację o jego rozdzielczości w **DPI** *(dot per inch),* czyli w liczbie punktów na jeden cal). Dostępne skanery mają przynajmniej 300 DPI, a bywają i takie, które mają 1000 DPI!

Istnieją też urządzenia, w których człowiek sam wybiera i wskazuje komputerowi punkty, które należy zapamiętać (tzw. **digitizery** – patrz fotografia u góry stronicy). Za ich pomocą można dane zawarte na określonym obrazie (wykresie, fotografii, mapie itp.) wprowadzić do komputera i tam dalej "obrabiać". Zwykle praca człowieka przy digitizerze (patrz fotografie obok) jest ułatwiana przez obecność szeregu urządzeń pomocniczych – lupy "celownika", układów wyświetlających współrzędne wskazywanych punktów – oraz przez specjalne oprogramowanie, co nie zmienia faktu, że jest bardzo uciążliwa.

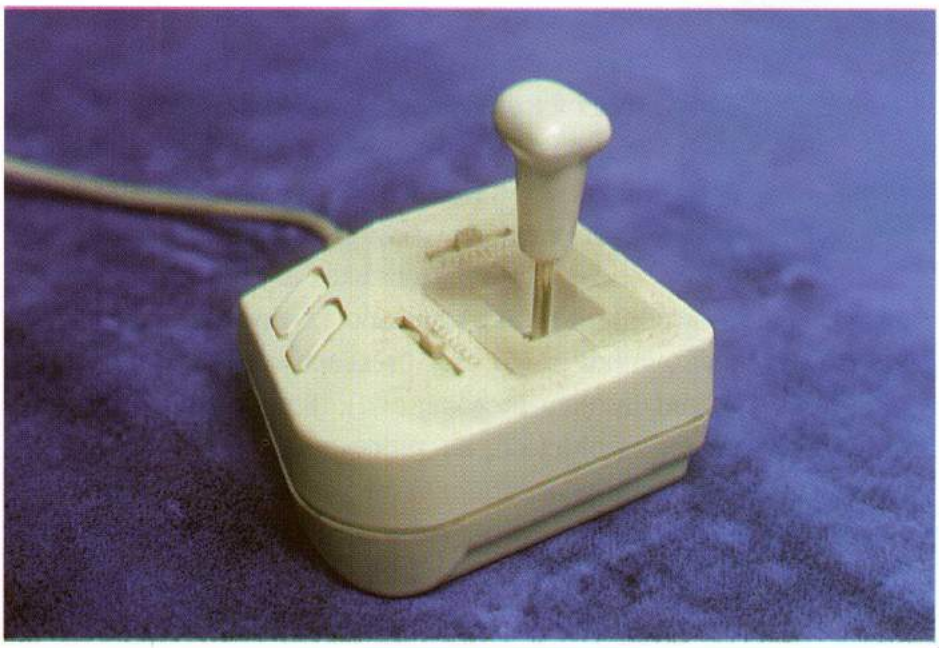
Skaner jest łatwiejszy w obsłudze, ale powoduje wprowadzenie do komputera informacji o bardzo wielu mało istotnych punktach obrazu, w wyniku czego cały "zeskanowany" obraz zajmuje bardzo dużo miejsca w pamięci (od kilku do kilkudziesięciu megabajtów!). Ten sam obraz opracowany za pomocą digitizera ma **kilkaset razy** mniejszą objętość, ale proces ręcznego wprowadzania kolejnych istotnych punktów obrazu trwa niekiedy wiele godzin.

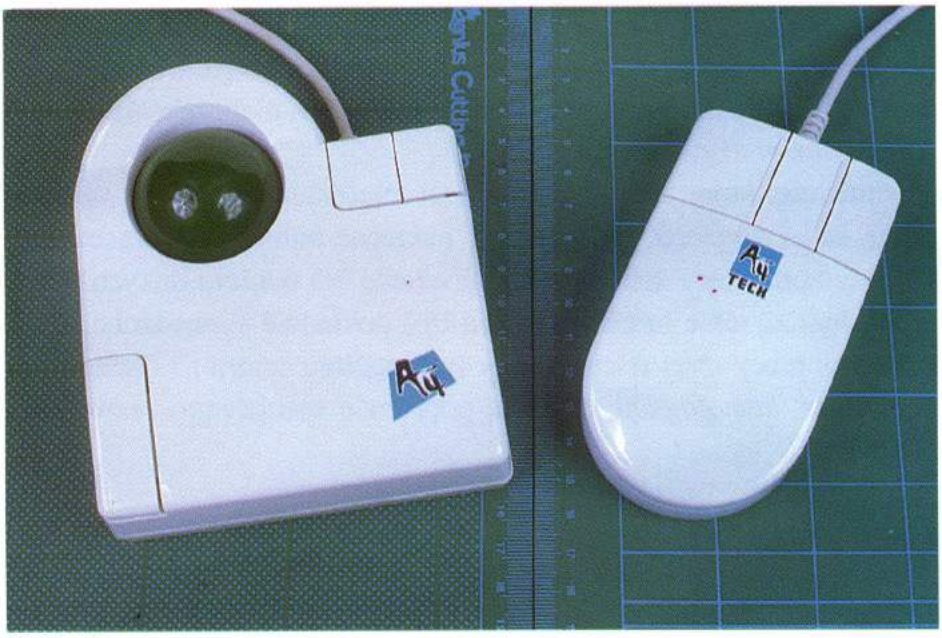
Obok wymienionych wyżej systemów do wprowadzania obrazów do komputerów służą także **cyfrowe przetworniki obrazu** (*frame graber* – patrz fotografia na następnej stronicy), pozwalające na operowanie obrazem pochodzącym bezpośrednio z kamery lub z taśmy video, a więc będącym wyobrażeniem **trójwymiarowego** fragmentu rzeczywistości. Komputer może taki obraz dowolnie przetworzyć lub rozpoznać, co daje możliwość bardzo ciekawych zastosowań wszędzie tam, gdzie trzeba zautomatyzować pracę, przy której człowiek posługuje się wzrokiem. Na tej zasadzie buduje się nowoczesne systemy sensoryczne dla robotów, komputery przeznaczone do strzeżenia różnych obiektów, a także nowoczesne systemy dla potrzeb diagnostyki medycznej.



2.4.4.5. Myszka i inne manipulatory

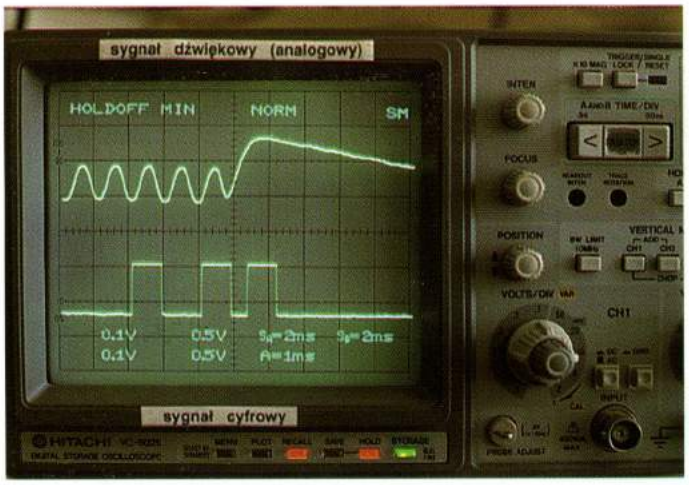
Nowoczesne korzystanie z komputerów polega na używaniu gotowych programów (por. podrozdz. 3.2). Sterowanie ich pracą opiera się najczęściej na wybieraniu jednej z proponowanych przez program możliwości. Wybór taki może być dokonywany przy użyciu klawiatury (na przykład przez wypisanie stosownego hasła lub odpowiedniej litery). Szybciej i wygodniej jest jednak odpowiedni wariant **wskazać**. Do wskazywania wymaganych obiektów na ekranie komputera można użyć tak zwanego **manipulatora**. W użyciu są znane z gier **manipulatory drążkowe** (*joystick* – patrz fotografia obok) albo też **manipulatory kulowe** (*trackball* – patrz fotografia na następnej stronicy). Jednak wszelkie rekordy powodzenia pobiła tak zwana "**myszka**" (*mouse* – też przedstawiona na fotografii na następnej stronicy) – małe pudełko połączone z komputerem za pomocą kabla, przesuwane po stole lub po specjalnej podkładce (*mouse pad*). Dzięki wygodnej pozycji ręki swobodnie spoczywającej na stole manipulowanie myszką jest wygodne i nie męczące, a precyzyjne oprogramowanie sterujące jej pracą pozwala bardzo dokładnie naprowadzać kursor na wymagany punkt ekranu – tak, że myszki można używać





nie tylko do wskazywania obiektów na ekranie, ale i do swobodnego rysowania. Myszka z reguły wyposażona jest dodatkowo w przyciski (od 1 do 3), których naciskanie może być (za pośrednictwem specjalnych programów) wykorzystane do manipulacji obiektami na ekranie komputera. Zwykle mówi się w takim przypadku, że "mysz tupnęła w tym miejscu" i definiuje się skutki takiego "tupnięcia".

2.4.4.6. Modemy



Zagadnienie komunikacji między komputerami będzie przedmiotem szczegółowych rozważań w podrozdziale 2.5, jednak dla kompletu informacji o urządzeniach wejścia/wyjścia nie można tu nie wspomnieć o urządzeniach służących do transmisji danych czyli do łączenia jednych komputerów z innymi[1].

Połączenie komputerów, jeśli ma być realizowane na dużą odległość, a jednocześnie skutecznie i tanio, musi odbywać się za pomocą typowych środków łączności (głównie linii telefonicznych). Jednak parametry linii telekomunikacyjnych dopasowane są do przesyłania głosu, a sygnały komputerowe mają zdecydowanie odmienny charakter (są to cyfrowe kody). Konieczna jest więc w nadajniku zamiana sygnału z postaci cyfrowej na sygnał dźwiękowy (modulacja) oraz na odbiorczym końcu linii zamiana sygnału z postaci śpiewnego "trelu", jakim był on przesyłany, z powrotem do postaci cyfrowej (demodulacja). Urządzeniem, które dokonuje tych czynności, jest tak zwany **modem** (nazwa pochodzi od słów **mo**dulator i **dem**odulator).

Rozróżnia się dwa typy modemów. Jedne ograniczają swoje działanie do omówionego wyżej transformowania sygnałów cyfrowych na "śpiew" przesyłanego linią dźwięku i na odwrót. Do samego wysyłania i odbierania

[1] Pierwsza udana próba transmisji danych miała miejsce **9 września 1940 roku**. Tego dnia **George R. Stibitz**, pracownik laboratorium *Bella* (firma *AT&T*), przesłał za pomocą dalekopisu dwie 8–cyfrowe liczby (dziesiętne) i sygnał nakazujący ich podzielenie. Odbiorcą i wykonawcą polecenia był elektromechaniczny (pracujący na przekaźnikach) kalkulator o nazwie *Complex Number Calculator*, zlokalizowany w innym laboratorium odległym o kilkaset kilometrów. Wynik otrzymany zwrotnie również za pomocą łącza dalekopisowego był prawidłowy, ale cała operacja trwała dość długo (ponad 30 sekund).

sygnałów służy w tym przypadku zwykły aparat telefoniczny, za pomocą którego wybiera się (kręcąc tarczą) odpowiednie połączenie, a potem kładzie się słuchawkę na odpowiedniej podstawce modemu i komputery mogą sobie "porozmawiać". Takie modemy, nazywane **modemami o sprzężeniu akustycznym,** są tańsze, ale bardziej kłopotliwe w użytkowaniu.

Istnieją także **modemy galwaniczne,** które włącza się do linii telefonicznej zamiast aparatu telefonicznego. Modemy takie mogą same wybierać połączenia ("nakręcać numery"), kontrolować czy "po drugiej stronie" zgłosił się inny komputer i dokonywać przesyłania lub odbioru dużych ilości informacji z bardzo dużą szybkością. Jednak takie modemy muszą być dokładnie sprawdzone, żeby ich użycie nie spowodowało zakłóceń w pracy sieci telefonicznej (szczególnie central) i dlatego ich używanie możliwe jest wyłącznie po tzw. *homologacji*, czyli na podstawie specjalnego zezwolenia stosownego urzędu telekomunikacyjnego.

2.4.4.7. Optyczne czytniki tekstów

Stosowanie komputerów w przetwarzaniu danych i redagowaniu tekstów stwarza rosnące zapotrzebowanie na metody automatycznego wprowadzania do komputera informacji z pisanych i drukowanych dokumentów. Proces ich czytania można zautomatyzować dzięki technice **OCR** (*optical character recognition),* czyli optycznym czytnikom tekstu. Technika ta polega na automatycznym rozpoznawaniu liter i czytaniu całych dokumentów.

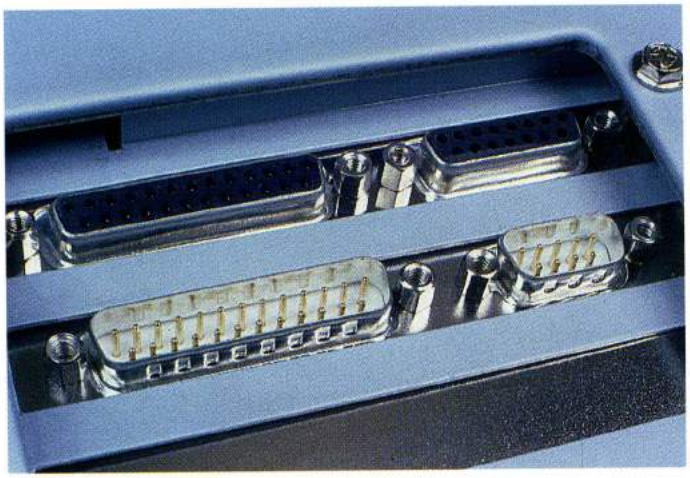
Systemy OCR spełniają różne zadania i mogą podlegać różnorodnej klasyfikacji. Zasadniczy podział, jaki można wprowadzić, dotyczy rodzaju rozpoznawanych znaków. I tak, można wyróżnić systemy rozpoznające znaki **drukowane** lub **pisane ręcznie**. Oczywiście trudność związana z rozpoznawaniem znaków pisanych ręcznie jest nieporównanie większa niż ta, która pojawia się w ramach rozpoznawania znaków drukowanych.

W obrębie wymienionych klas zagadnień możliwa jest oczywiście dalsza klasyfikacja. I tak, w obrębie rozpoznawania znaków drukowanych można oddzielnie rozpatrywać te systemy, które rozpoznają druk o jednej, ustalonej wielkości i formie czcionki. Jednak wyższy stopień doskonałości wiąże się z używaniem systemu rozpoznającego wiele krojów czcionki (tzw. "*multifont*")[1]. System dostosowany do jednego kroju i wielkości czcionki można zbudować szybko i łatwo, jednak jego przydatność jest bardzo ograniczona. Systemy akceptujące czcionki o różnej wielkości i o różnym kształcie (często bardzo wyrafinowane pod względem ozdobnej stylistyki) są trudniejsze do zbudowania, lecz tylko ich zastosowania są w pełni uniwersalne. Wśród systemów rozpoznających pismo ręczne jest jeszcze więcej możliwości podziałów. Można więc wyróżnić systemy rozpoznające **pojedyncze ręcznie pisane znaki** (na przykład kody pocztowe) lub **pismo ciągłe** (tworzone bez wyróżniania oddzielnych liter)[2]. Inne kryterium podziału pozwala wyróżnić systemy śledzące proces pisania (*on-line*) oraz systemy, które podejmują próbę odczytania tekstu już po jego napisaniu (*off-line*). Oczywiście te drugie są bardziej skomplikowane, gdyż mają trudniejsze zadanie – brak informacji o tym, w jakiej kolejności pisano poszczególne elementy wyrazu.

2.5. Sieci komputerowe

Połączenie kilku komputerów w system, zwany **siecią komputerową**, zwiększa możliwości każdego z użytkowników, gdyż potencjalnie każdy ma do dyspozycji wszystkie urządzenia sieci (np. wystarcza jeden duży dysk lub jedna drukarka dla wielu komputerów) oraz w razie potrzeby moc obliczeniową wszystkich komputerów sieci. Sieć ułatwia także wymianę programów, danych i komunikatów między użytkownikami sieci[3].

Mikrokomputery mają od razu wbudowane przez producenta pewne możliwości komunikacyjne (tzw. interfejs szeregowy **RS-232C**). Za jego pomocą komputer może współpracować z dowolną inną maszyną wyposażoną w taki sam interfejs. Procedury komunikacyjne wbudowane są w system operacyjny (moduł **BIOS**), przy czym komputer "widzi" interfejsy (typowo dwa) jako urządzenia **COM1** i **COM2**. Przy przesyłaniu informacji jeden z komunikujących się komputerów traktowany jest jako centralny (oznaczany **DCE**), zaś pozostałe jako tzw. terminale (jednostki podległe – **DTE**).



Szybkość transmisji, wyrażana w bitach na sekundę (tzw. **bodach**), wynosi 50 do 9600 przy stosowaniu standardowych programów BIOSu. Może ona jednak być zwiększona aż do 10 Mb/s w typowej sieci ***Ethernet*** lub nawet do 100 Mb/s w sieciach wykorzystujących łącza światłowodowe (**FDDI**).

Sieci **Ethernet** i systemy **FDDI** wymagają dołączenia do komputera urządzeń komunikacyjnych, tzw. **kart sieciowych** (np. karta *EtherCard Elite PLUS* firmy *Western Digital*). Karty sieciowe (patrz fotografia na następnej stronicy) są dość kosztowne, ale pozwalają sprzęgać komputery tak

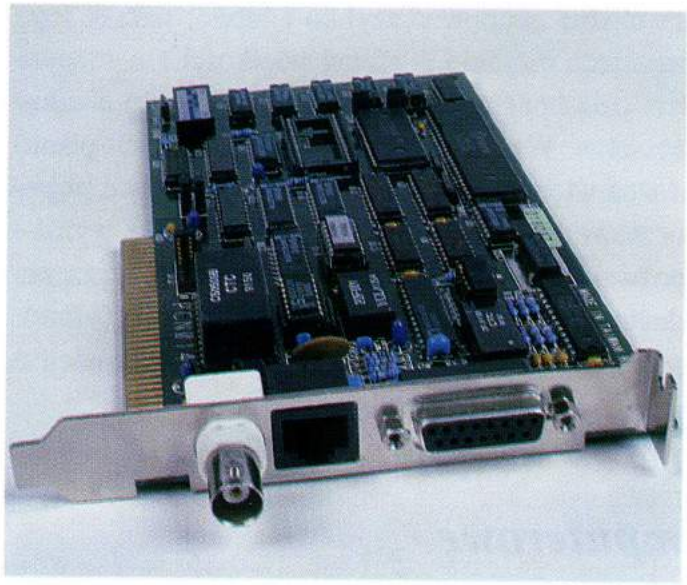
---

[1] Programem, który – jak się wydaje – ma dziś szansę na zdominowanie zastosowań typu **OCR** jest **Recognita Plus**. Może on współpracować z większością popularnych skanerów, akceptując bezpośrednio obrazy stronic tekstu z każdego z tych skanerów, a także obrazy stronic tekstu przygotowane uprzednio na dysku w jednym z popularnych standardów przechowywania obrazów (np. **PIC, TIF** i inne). Zaletą Recognity jest możliwość czytania tekstów składanych z wykorzystaniem kilkudziesięciu różnych krojów czcionki.

[2] W tym przypadku segmentacja wyrazu na oddzielne litery jest bardziej skomplikowana, niż samo rozpoznanie tych liter.

[3] Uważa się, że pierwszą organizacją opartą w całości na wykorzystaniu sieci komputerowych była **CPA** (*Computer Press Association*), skupiająca ponad 300 dziennikarzy z całego świata. Prezesem CPA jest **Hal Glatzer**, znany na całym świecie popularyzator zastosowań techniki komputerowej.

skutecznie, że użytkownik może wcale nie zauważyć faktu, że w pewnych momentach korzysta z zasobów innego komputera (np. z bazy danych na cudzym dysku). Podnosi to znacznie sprawność korzystania z sieci.



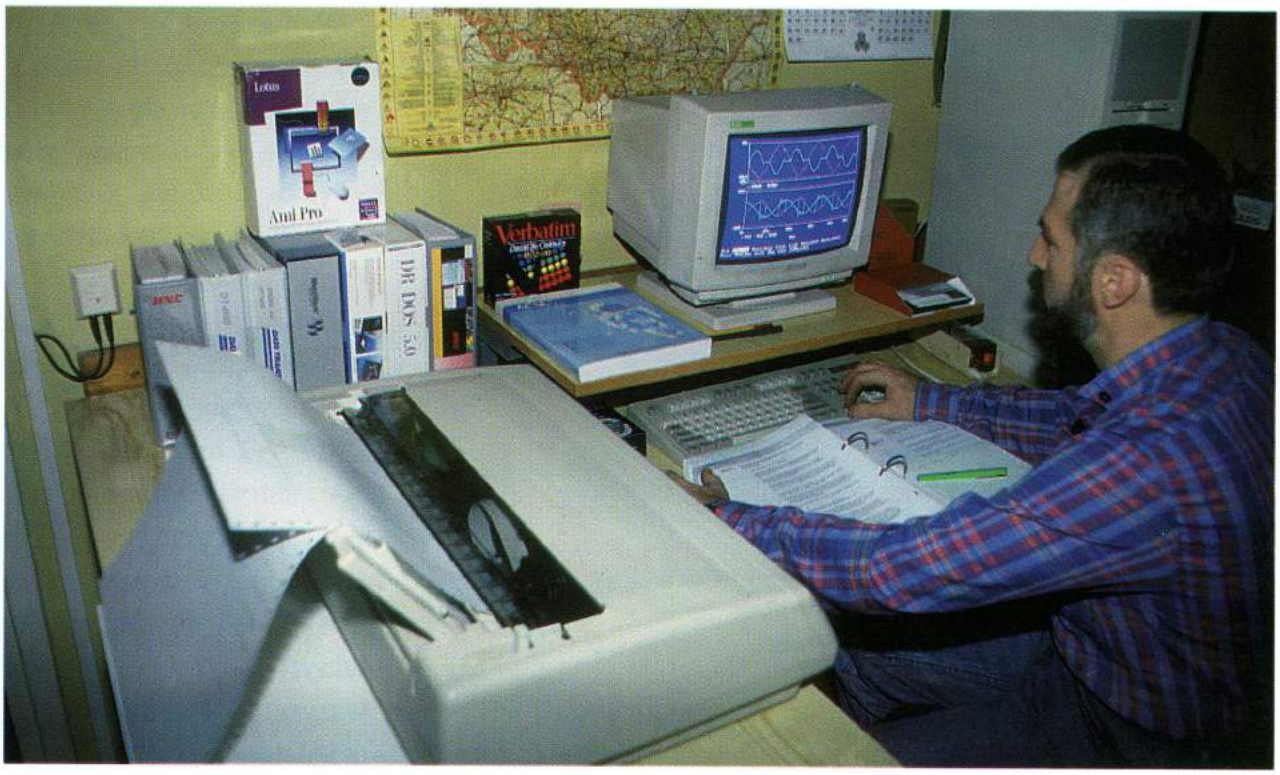
Przy większej liczbie komputerów wyróżnia się zwykle jeden z nich jako tzw. *master* (komputer nadrzędny), pozostałe zaś mają status *slave* (dosłownie: niewolnik, czyli komputer podporządkowany). Inny sposób klasyfikacji elementów sieci pozwala wydzielić komputer świadczący określone usługi, tzw. serwer *(server)* i komputery korzystające z tych usług, nazywane stacjami roboczymi *(work station).* Serwer jest zwykle dużym komputerem o bogatych zasobach (m.in. pamięci dyskowej), za to stacje robocze mogą mieć znacznie mniejsze rozmiary (i cenę!). Przykładem popularnej sieci tego typu może być instalacja sieci **Novell** opartej na komputerach PC.

Bogatszą i bardziej różnorodną strukturę miewają sieci lokalne nazywane **LAN** (*Local Area Network).* Jeszcze bogatsze możliwości ma sieć dalekiego zasięgu: metropolitalne (np. **Cyfronet**), ogólnokrajowe (np. **KASK**) czy międzynarodowe (np. **INTERNET).** Obszerniejsze omówienie tych sieci, ich zalet i wad znaleźć można w literaturze, której wykaz podano na końcu książki.

3. Oprogramowanie

3.1. Znaczenie i podział oprogramowania

O rzeczywistej wartości komputera decyduje jego oprogramowanie. **Komputer sam nie potrafi nic zrobić.** Do każdej, najbardziej nawet elementarnej czynności musi mieć odpowiedni program. Program taki można napisać samodzielnie i przez pierwsze dwadzieścia lat burzliwego rozwoju informatyki była to w gruncie rzeczy jedyna możliwość. Każdy użytkownik komputera musiał być programistą – co wymagało odpowiednich (bardzo wysokich!) kwalifikacji i opóźniało rozwój zastosowań informatyki, gdyż **proces pisania programu jest pracochłonny i wymaga wiele czasu**. Jeśli więc ktoś posiadał komputer i chciał go zastosować do rozwiązania określonego praktycznego problemu – to nie mógł tego zrobić natychmiast, lecz musiał czekać, aż zostanie napisany odpowiedni program. Niekiedy oznaczało to konieczność czekania przez całe lata...



Niezwykle ważne było zatem spostrzeżenie, że zadania powierzane komputerom mają w zdecydowanej większości przypadków charakter **typowy.** Jeśli więc ktoś kiedyś napisał program do tworzenia i użytkowania komputerowej bazy danych – to mija się z celem, by inni użytkownicy

pisali te same programy powtórnie! Zaczęto wymieniać programy, a potem – naturalną koleją rzeczy – zaczęto nimi handlować. Powstały wyspecjalizowane firmy, które zarabiały wyłącznie na pisaniu i sprzedaży oprogramowania, a większości użytkowników nieporównanie wygodniej było (i jest) kupić program, niż pisać go samemu. Spróbujemy wyjaśnić, dlaczego tak jest.

3.2. Dlaczego należy stosować programy firmowe?

Można wymienić trzy główne przyczyny sprawiające, że wybór programu gotowego (firmowego) zamiast programu samodzielnie pisanego jest wyborem racjonalnym.

1. Program firmowy **pozwala szybciej rozpocząć właściwe wykorzystanie komputera.** Zakup komputera – nadal, mimo malejących cen, bardzo kosztowny – uzasadniony jest zawsze faktem, że potrzebujemy konkretnych usług tej maszyny. Tymczasem programując ją samodzielnie odwlekamy – na długo, a czasem nawet na zawsze – moment, kiedy będzie ją można zastosować do rozwiązywania problemów, do których została zakupiona.
2. Program firmowy **cechuje się większą sprawnością**. Jeśli to samo zadanie na tej samej maszynie można wykonać za pomocą jednego programu w ciągu znacznie krótszego czasu niż za pomocą innego programu – wówczas powinieneś używać tego sprawniejszego programu. Badania wykazały, że program napisany przez dobrego fachowca wymaga **kilkadziesiąt** razy mniej czasu podczas rozwiązywania postawionego zadania, niż taki sam (w sensie celu działania) program napisany przez amatora[1].
3. Program firmowy **jest bezpieczniejszy w użyciu.** Podczas samodzielnego programowania zadania można popełnić błąd, który spowoduje, że uzyskiwane z komputera wyniki będą nieprawdziwe. Jeśli na takich nieprawdziwych wynikach oprzemy jakieś praktyczne działanie – wówczas konsekwencje mogą być opłakane. Tymczasem program firmowy jest **prawie na pewno** bezbłędny, ponieważ pisali go najlepsi fachowcy, a ponadto przed oddaniem go do użytku był on wszechstronnie sprawdzany.

Wymienione argumenty zdecydowanie przemawiają za tym, aby **starać się** – gdy tylko można – **rozwiązywać problemy za pomocą komputera wykorzystując gotowe programy firmowe.**

Trzeba tylko zwrócić uwagę na dwie sprawy. Po pierwsze programy nie są "własnością publiczną", lecz mają określonego autora, któremu należy się wynagrodzenie za jego pracę. Zatem – wbrew niestety powszechnej w naszym kraju praktyce – programy należy **kupować,** a nie pozyskiwać w formie "pirackich kopii". Po drugie wszystkie wymienione zalety odnoszą się do programów **dobrych firm komputerowych,** jako że w dziedzinie produkcji oprogramowania też jest co niemiara tandety. Wiedza na temat tego, jakie programy są dobre, a jakie złe,

[1] Warto podkreślić, co to oznacza: przykładowo, w profesjonalnym programie na odpowiedź komputera trzeba czekać kilkanaście sekund (co nie przeszkadza w pracy), a w rozwiązaniu amatorskim odpowiedź przychodzi po kilku minutach, kiedy użytkownik zdążył już się zdekoncentrować i prawie zapomniał, jakie postawił zadanie!

które firmy zasługują na zaufanie, a które nie – także stanowi elementarz dla zawodowego informatyka, ale przyda się też amatorom. Dlatego przytoczyliśmy dalej niektóre użyteczne wiadomości na ten temat.

3.3. Struktura oprogramowania firmowego

Jak wynikało z przytoczonych uwag wstępnych, o użyteczności komputera decyduje jego oprogramowanie wytwarzane przez wyspecjalizowane firmy *software'owe*. To właśnie oprogramowanie firmowe steruje pracą poszczególnych elementów sprzętu, kontroluje ich współdziałanie i wykorzystanie, potrafi także testować działanie komputera. Przede wszystkim jednak współpracuje z użytkownikiem i realizuje jego zadania. Oprogramowanie jest ważniejsze od sprzętu – **sukces rynkowy osiągnęły bowiem nie komputery lepsze i doskonalsze technicznie, ale te, które miały największą liczbę dobrych programów.**

W dziedzinie oprogramowania można wyróżnić uświęcony tradycją podział, przedstawiony obok. Będę je więc omawiał z zachowaniem tego podziału.

Oprogramowanie systemowe (na przykład **MS DOS**[1]) jest dla użytkownika praktycznie nieodróżnialne od sprzętu, wymienne też bywają ich funkcje. W rozdz. 3.4 omówię podstawowe funkcje oprogramowania systemowego na przykładzie dwóch najpopularniejszych dziś systemów operacyjnych.

Oprogramowanie narzędziowe służy do przygotowania i obsługi programów użytkowych, a wykorzystywane jest z reguły przez osoby umiejące programować.

Oprogramowanie użytkowe stanowi narzędzia do wykonywania konkretnych zadań. Najpopularniejsze i najbardziej uniwersalne zastosowania oprogramowania użytkowego to:

- pisanie i redagowanie tekstów (podrozdz. 3.5),
- gromadzenie, przechowywanie i wyszukiwanie informacji (podrozdz. 3.6),
- arkusze kalkulacyjne (podrozdz. 3.7),
- grafika komputerowa (podrozdz. 3.8),
- pakiety zintegrowane[2] (podrozdz. 3.9).

---

[1] Za twórcę systemu MS DOS uważa się **Billa Gatesa**, jednego z założycieli firmy *MICROSOFT*. Kariera tego niezwykłego człowieka zaczęła się od tego, że w 1975 roku, mając 19 lat, rozesłał do wszystkich producentów komputerów ofertę, że może sprzedać translator języka BASIC. Było to zuchwalstwo: Gates NIE MIAŁ translatora, a dopiero chciał go zbudować. Ponieważ chętnych do zakupu nie brakowało, Bill Gates wraz z przyjacielem **Paulem Allenem** pracowali dzień i noc przez sześć tygodni, po czym przedstawili propozycję BASIC-a, który stał się światowym standardem, a swoim twórcom przyniósł miliony dolarów czystego zysku. **MS DOS** powstał w podobnym tempie i był jeszcze większym sukcesem finansowym.

[2] To znaczy programy łączące w sobie możliwości wszystkich wymienionych wyżej grup oprogramowania użytkowego.

3.4. Systemy operacyjne i ich podstawowe funkcje

3.4.1. Zadania systemu operacyjnego

Zadaniem systemu operacyjnego jest zarządzanie pracą maszyny cyfrowej i podział zasobów (głównie czasu procesora i pamięci operacyjnej oraz urządzeń zewnętrznych) **pomiędzy użytkowników maszyny** – jeśli jest ich kilku. System operacyjny jest więc głównie administratorem komputera i od jego własności zależy w zasadniczym stopniu wygoda pracy użytkownika. Systemy operacyjne są zwykle bardzo skomplikowanymi, kosztownymi programami, dostosowanymi do indywidualnych wymagań właściciela komputera i do konfiguracji sprzętu pozostającego do dyspozycji.

Jeśli interesuje Cię wyłącznie stosowanie gotowych programów użytkowych, to znajomość systemu operacyjnego jest Ci potrzebna tylko w takim zakresie, jaki wymagany jest do uruchomienia programów użytkowych i ewentualnie do zrobienia rezerwowych kopii programów czy danych. Bardziej wymagający użytkownicy mogą potrzebować bardziej wyrafinowanych usług systemu operacyjnego, ale wówczas muszą poznać większą liczbę jego komend. Warto dodać, że własności systemu operacyjnego znacznie silniej wpływają na obraz komputera i jego możliwości niż rzeczywiste właściwości sprzętowe[1].

3.4.2. Pliki i ich identyfikacja

Jednym z podstawowych zadań każdego z obecnie używanych systemów operacyjnych jest obsługa pamięci dyskowych. Wyraża się to między innymi faktem, że nagminnie pojawia się w nazwach systemów określenie **DOS** od *Disk Operating System*. Dlatego w tym podrozdziale powinniśmy także przywiązać pewną uwagę do zagadnienia **plików** w pamięciach dyskowych i sposobów ich identyfikacji w systemach operacyjnych.

Informacje zapisane na dysku twardym lub na dyskietkach (mogą nimi być dane, wyniki, programy, odwzorowania tekstów czy rysunków itp.) noszą nazwę plików. **Plik** jest podstawowym obiektem podczas współpracy człowieka z maszyną cyfrową. Prawie wszystkie podawane podczas tej współpracy polecenia dotyczą operacji na plikach. Jeśli polecisz zapisać np. tekst programu w pliku o nazwie MOJPROG, to system operacyjny znajduje, wybiera i przydziela odpowiednie wolne fragmenty dysku, zapisuje na nich znaki tworzące program i zapamiętuje nazwę oraz rozmieszczenie elementów pliku w **katalogu** (spisie plików znajdujących się na dysku). Jeśli zechcesz teraz przepisać plik (na przykład na dyskietkę), to wystarczy wydać polecenie **skopiuj plik** MOJPROG nie interesując się zupełnie szczegółami technicznymi tej operacji. Bliższe informacje na temat używania plików w systemie **MS DOS** znajdziesz w kolejnym podrozdziale.

3.4.3. System operacyjny MS DOS

Niewątpliwie najpopularniejszym z aktualnie używanych jest system **MS DOS**[2] firmy *Micro-Soft*. Podane niżej informacje mają charakter podstawowy, bardziej szczegółowo możesz poznać ten system korzystając z dowolnej spośród licznych książek poświęconych obsłudze komputerów klasy

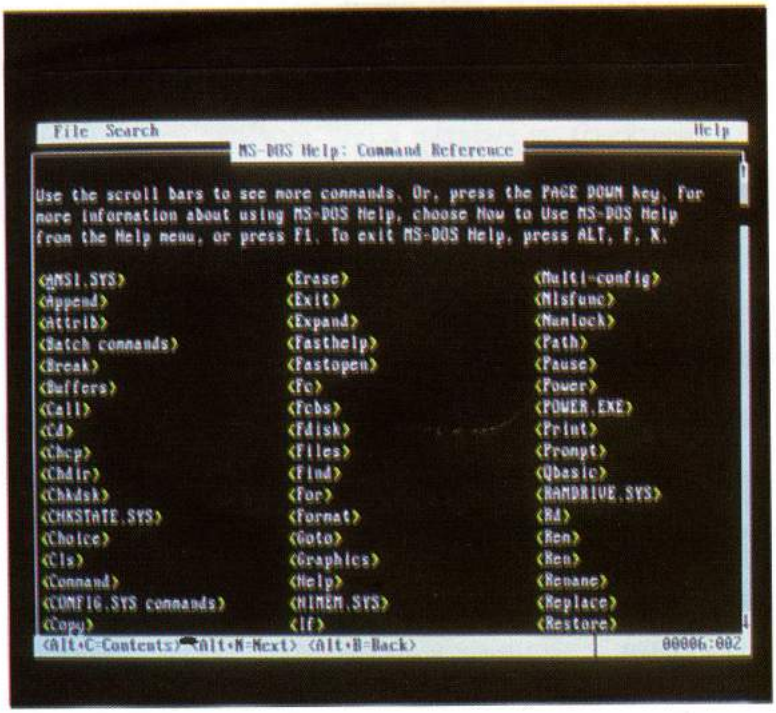
[1] Znanych jest obecnie bardzo wiele różnych systemów operacyjnych, przy czym często zdarza się, że jeden komputer może być eksploatowany za pomocą kilku różnych systemów operacyjnych (wówczas ta sama maszyna ma "pod" każdym systemem odmienne właściwości i możliwości). Z kolei bywa też tak, że różne maszyny mogą być eksploatowane z wykorzystaniem jednego, uniwersalnego systemu operacyjnego. Jest to sytuacja bardzo wygodna, gdyż dzięki standardowi systemu operacyjnego użytkownik może w identyczny sposób obsługiwać rozmaite komputery – bez konieczności poznawania każdego z nich z osobna. Takim standardowym systemem operacyjnym jest **MS DOS**.

[2] System **MS DOS** przechodził długą ewolucję zanim osiągnął stan, w jakim jest tu opisywany. Pierwszą wersję tego systemu opracował w 1979 roku **Tim Peterson**, pracownik firmy *Seattle Computer Products*. Głównym celem opracowania tego systemu było stworzenie możliwości uruchamiania programów napisanych pod **CP/M** na komputerach wyposażonych w procesor **Intel 8088/86**. System ten nabrał znaczenia, gdy zakupiła go (w **1981** roku) firma *Microsoft* i zastosowała do zyskujących popularność komputerów **IBM PC**.

IBM PC (niektóre z nich podałem na końcu książki). Jednak, nawet ograniczając swoje wiadomości o systemie DOS do przedstawionych tu podstawowych informacji, możesz stosunkowo sprawnie korzystać z komputera pracującego pod kontrolą tego systemu. Wiadomości te możesz też bardzo łatwo w każdej chwili uzupełnić: wystarczy napisać **HELP i nacisnąć Enter,** a system sam zacznie dostarczać potrzebnych informacji (niestety – po angielsku...).

3.4.3.1. Jak posługiwać się systemem MS DOS?

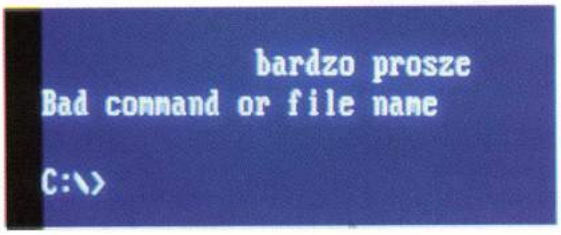
MS DOS służy do tego, żeby na polecenie użytkownika (czyli na **Twoje** polecenie ...) sterować pracą komputera i zmuszać go do wykonywania potrzebnych zadań. System zgłasza gotowość do przyjęcia polecenia wyświetlając znak zachęty (***prompt***). Na fotografii obok pokazałem, jak to najczęściej wygląda.



Z systemu DOS korzysta się, wydając mu **polecenia,** czyli pisząc na klawiaturze **nazwy czynności**, które ma wykonać. Przykładem takiego polecenia było opisane wyżej żądanie pomocy (HELP), za pomocą którego można (między innymi) dostać wykaz wszystkich znanych systemowi poleceń. Skutek wykonania takiego polecenia widzisz obok – ta długa lista to **niektóre** tylko polecenia, jakie możesz wydać systemowi MS DOS.

Po napisaniu każdego polecenia trzeba nacisnąć klawisz **Enter,** który oznacza **"zrób to!"**. Wcześniej można pisane polecenie zmieniać i poprawiać, kasując źle napisane litery klawiszem **Backspace.**

Jeśli wydane polecenie nie jest dla systemu zrozumiałe – wypisuje on ostrzeżenie, którego formę podałem niżej, i **nic nie robi**. Dlatego nie należy się obawiać wydawania systemowi poleceń, gdyż jest bardzo mało prawdopodobne, że się coś zrobi źle. Jeśli nie jesteś pewien, jaką formę ma potrzebne Ci polecenie, to i tak możesz śmiało zaryzykować jego napisanie. Najwyżej się pomylisz i wtedy **nic** nie zostanie zrobione. Jeśli natomiast uda Ci się i polecenie jest zrozumiałe – DOS gorliwie wykona żądane czynności, a potem znowu wyświetli *prompt* i będzie czekał na następne Twoje polecenia.



W następnych podrozdziałach pokażę Ci, jak łatwe i proste w użyciu są podstawowe polecenia DOS–u. W każdej chwili możesz jednak dowiedzieć się więcej za pomocą opisanego już polecenia **HELP.** Jeżeli nie wiesz, czego szukasz – napisz samo polecenie **HELP** i czytaj uważnie wszystko, co DOS sam o sobie napisze. Takie "szukanie w ciemno" może jednak długo trwać, dlatego jeśli wiesz **cokolwiek** możesz sobie znakomicie ułatwić sobie pracę. Na przykład, jeśli wiesz jak się nazywa polecenie, ale nie wiesz jak się go używa i co ono robi, możesz napisać

HELP ***polecenie***

gdzie oczywiście w miejscu słowa *polecenie* wpisujesz nazwę interesującego Cię polecenia. Dostaniesz na ekranie bardzo dokładny opis (z przykładami!), co i jak możesz zrobić.

DOS jest w Twoim komputerze najważniejszy – to on uruchamia wszystkie inne programy i on wkracza do akcji, jeśli te programy czegoś potrzebują. Warto więc od razu zapamiętać, że **polecenie uruchomienia dowolnego programu polega w DOS–ie na napisaniu nazwy tego programu.** Tylko tyle!

3.4.3.2. Rozpoczęcie pracy z systemem MS DOS

Podane w tym podrozdziale wiadomości **nie są niezbędne** do tego, żeby komputer wygodnie używać. Wszystko, o czym tu będzie mowa, wykonuje się automatycznie i nie wymaga od Ciebie żadnych działań. Dlatego, jeśli chcesz – możesz od razu zacząć czytać podrozdział 3.4.3.3. Jednak jeśli chcesz rozumieć te wszystkie "czary–mary", które dzieją się po włączeniu komputera – przeczytaj podany niżej tekst. Na pewno się opłaci!

Podstawowa część systemu operacyjnego **MS DOS** jest na stałe wbudowana do obsługiwanego komputera w postaci odpowiedniego modułu pamięci stałej (**ROM**). Część ta, zwana **BIOS**, służy do podstawowej obsługi sprzętu i dlatego musi być z nim na stałe związana. Pozostałe części systemu są **ładowane** (wczytywane z dyskietki lub z dysku twardego), w związku z tym możliwe jest używanie na tym samym komputerze różnych wersji systemu **MS DOS**[1]**.** Ładowana część systemu składa się z dwóch części: **jądra**[2] i zestawu programów realizujących tzw. **polecenia nierezydentne**[3]**.**

Ładowanie systemu operacyjnego odbywa się po włączeniu komputera lub po naciśnięciu przycisku RESET. W kolejności wykonywane są następujące czynności:

- **ROM–BIOS** testuje komputer, odszukuje program ładujący i uruchamia go;
- program ten ładuje z kolei do pamięci **jądro** i przekazuje mu sterowanie;
- system szuka pliku **CONFIG.SYS** i uwzględnia życzenia użytkownika tam zapisane;
- zostaje wczytany program **COMMAND.COM,** który od tej chwili przejmuje kontrolę;
- następuje wykonanie komend zawartych w pliku **AUTOEXEC.BAT**[4], a jeśli nie ma takowego, to system żąda podania aktualnej daty i czasu;
- wyświetlany jest znak zachęty[5] (np. **C:>**) i system czeka na polecenia użytkownika.

Reguła jest przy tym następująca: ROM–BIOS sprawdza, czy jest dyskietka w komorze **A**. Jeśli jest, to z niej usiłuje ładować programy IBMBIO.COM i IBMDOS.COM. Brak tych programów

---

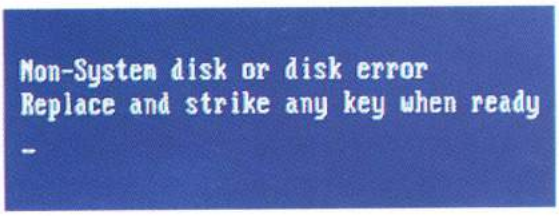
1 Niektóre programy "chodzą" tylko z konkretną wersją systemu, a z inną nie chcą współdziałać.

2 Jądrem DOS–u są trzy programy:
–IBMBIO.COM – zawiera procedury umożliwiające kontakt z urządzeniami wejścia/wyjścia;
–IBMDOS.COM – stanowi łącze programowe pomiędzy procedurami wyższego poziomu a programami DOS–u;
–COMMAND.COM – to program interpretujący polecenia użytkownika.

3 Są to te polecenia, których jądro systemu nie potrafi samo wykonać i musi ładować do pamięci specjalne dodatkowe programy.

4 Sposób budowy plików typu **BAT** opisany będzie w jednym z dalszych podrozdziałów.

5 Postać znaku zachęty (ang. *prompt*) bywa różna, ale zwykle podaje ona tzw. **aktualny napęd dyskowy** oraz (ewentualnie) **nazwę aktualnego katalogu.** O katalogach będzie mowa niżej, natomiast napęd dyskowy identyfikowany jest za pomocą litery: **A, B, C,** ..., przy czym na początku wyświetlany jest identyfikator tego napędu, z którego system ładował programy IBMBIO.COM i IBMDOS.COM.



wywołuje pojawienie się komunikatu ostrzegawczego i zawieszenie działania komputera. Natomiast brak dyskietki w komorze **A** powoduje ładowanie programów IBMBIO.COM i IBMDOS.COM z dysku **C** (jest to z reguły tzw. twardy dysk).

3.4.3.3. Napędy, katalogi i pliki w systemie MS DOS

Po załadowaniu systemu użytkownik może w każdej chwili wybrać dowolny napęd dyskowy jako aktualny, pisząc jego identyfikator z dwukropkiem i przesyłając ten komunikat do systemu. Na przykład, gdy system zgłosi się pisząc **C:\>**, użytkownik może przełączyć go na dysk **B** pisząc **B:**. Dialog wygląda wtedy tak, jak pokazano obok.

Ustaliwszy, z którego napędu korzystamy, możemy z niego wybrać potrzebny nam zbiór informacji, czyli tak zwany **plik** (*file*). Pliki można porównać do ułożonych na półce książek. Plik może zawierać dowolne informacje (na przykład dane albo teksty) lub program. Wyjątkowość programów polega jednak na tym, że są one ładowane do pamięci i wykonywane, a także mogą używać do swoich celów innych plików, na przykład plików z danymi i plików, w których program będzie zapisywać wyniki.

Każdy plik ma swoją **nazwę**, która pozwala go wskazać w celu wykonania (jeśli jest programem) lub wybrać spośród innych plików (jeśli zawiera dane). Nazwa pliku jest to ciąg **od 1 do 8 znaków**, po których może wystąpić **rozszerzenie nazwy** pliku poprzedzone znakiem "." (kropka). Rozszerzenie nazwy pliku jest to ciąg **od 1 do 3 znaków**[1].

Nazwy komend DOS–u nie powinny być używane jako nazwy plików. Niżej podano zarezerwowane nazwy urządzeń, które także nie powinny być używane jako nazwy plików.

- **CON** – konsola, tj. klawiatura i ekran;
- **AUX** lub **COM1** – pierwszy asynchroniczny adapter komunikacyjny[2];
- **COM2** – drugi asynchroniczny adapter komunikacyjny (jeśli jest);
- **LPT1** lub **PRN** – pierwsze łącze równoległe[3] (tu zwykle jest przyłączona drukarka);
- **LPT2** – drugie łącze równoległe (jeśli jest);
- **NUL** – urządzenie fikcyjne stosowane do testowania.

Trzyznakowe rozszerzenie nazwy pliku jest stosowane do rozróżniania **typów** plików[4].

---

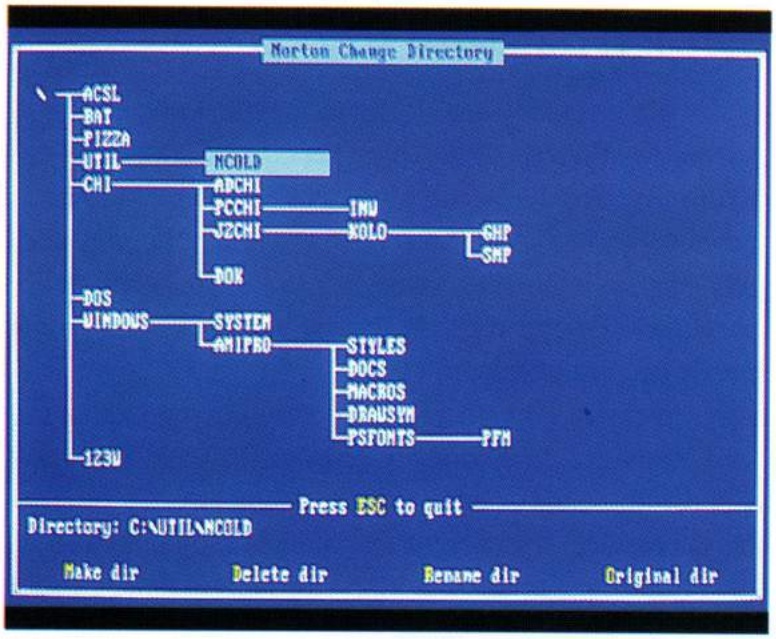
[1] Znakami dopuszczalnymi w nazwie pliku są: litery od **A** do **Z**, cyfry od **0** do **9** oraz znaki: **$ & # @ ! % ' ( ) - { } _ ~ .**

[2] Jest to po prostu gniazdko znajdujące z tyłu komputera, do którego można dołączyć różne urządzenia przesyłające informacje od albo do komputera. Adaptera tego używa się na przykład do dołączenia myszki albo do prostej komunikacji między dwoma komputerami.

[3] Słowo "równoległe" oznacza, że informacje przesyłane są w tym łączu równocześnie w "paczkach" po 8 bitów. W łączach **COM** informacje przesyłane są szeregowo, bit po bicie.

[4] Pewne rozszerzenia mają szczególne znaczenie dla systemu, na przykład:
–BAT – (ang. *batch*) – to plik składający się z komend dla DOS–u i wywołań programów do wykonania;
–COM – (ang. *command*) – oznacza program;
–EXE – (ang. *execute*) – to także "oznaczenie" programu gotowego do wykonania;
–SYS – (ang. *system*) – to plik zawierający informację dla systemu (np. CONFIG.SYS).
Oprócz wymienionych rozszerzeń, mających ustalone znaczenie, przyjęto na zasadzie umowy pewne dalsze rozszerzenia:
–BAK – (ang. *backup*) – to archiwalna wersja pliku który został zmieniony;
–DAT – (ang. *data*) – to plik z danymi;
–DOC – (ang. *document*) – to plik opisujący inny plik lub użytkowanie programu;
–HLP – (ang. *help*) – plik pomocniczy opisujący działanie programu;
–LIB – (ang. *library*) – plik z programami bibliotecznymi;

W celu ułatwienia zarządzania plikami wprowadzono pewne struktury – **katalogi** ***(directory)***, które są **plikami z nazwami plików**. Przy wyjaśnianiu roli katalogów znowu nasuwa się analogia z półką książek – plik jest jak gdyby konkretnym tomem, a katalog stanowi półkę książek zgromadzonych razem ze względu na podobieństwo tematyczne. Jeśli książek jest dużo, to niewątpliwie łatwiej odszukać potrzebny tom, gdy poda się obok jego własnej nazwy także nazwę półki, na której go można znaleźć[1]!



Analogia z półką na książki jest jednak ograniczona, ponieważ w systemie DOS elementami katalogów mogą być nazwy innych katalogów (**podkatalogi**). Tworzy się w ten sposób struktura drzewiasta, której "korzeniem", czyli początkiem wszelakich rozgałęzień jest katalog główny (*root* – ang. korzeń oznaczany znakiem \), a w której poszczególne katalogi stanowią albo "konary" dzielące się potem na drobniejsze gałęzie, albo elementy końcowe, tzw. "liście".

Podczas pracy z komputerem w każdej konkretnej chwili system "znajduje się" w jednym określonym miejscu tego drzewa katalogów, a zadaniem użytkownika jest orientowanie się i swobodne "wędrowanie" po całej strukturze. Wspomagają go w tym instrukcje systemowe, o których będzie mowa za chwilę, oraz specjalne programy, które także będą krótko omówione (NORTON COMMANDER). Także wspomniany już "znak zachęty" wskazuje zwykle na aktualne położenie w katalogu. Przykładowo:

- **C:\ >** oznacza, że użytkownik ma dostęp do głównego katalogu dysku C (nie ma wymienionych żadnych podkatalogów),
- **C:\CHIW\PODANIA >** wskazuje, że operator systemu najpierw przeszedł z katalogu głównego do podkatalogu CHIW, który następnie rozgałęział się dalej. Jednym z takich rozgałęzień jest podkatalog PODANIA, do którego w końcu zawędrował operator.

Zanim zostaną omówione podstawowe polecenia dla systemu operacyjnego MS DOS należy raz jeszcze przypomnieć, że większość operacji jest dokonywana na plikach lub urządzeniach zewnętrznych. W związku z tym przetwarzane pliki muszą zostać precyzyjnie zlokalizowane, tzn. musi zostać podana tzw. **ścieżka dostępu do pliku.** Ścieżka taka wymienia w kolejności drogę od korzenia wskazanego napędu dyskowego albo od aktualnie używanego katalogu do katalogu zawierającego potrzebny plik. Struktury katalogów zostały omówione wcześniej, tu natomiast należy dodać, że istnieją dwa symbole specjalne:

. – (kropka) oznacza katalog bieżący

.. – (dwie kropki) oznaczają katalog macierzysty bieżącego katalogu.

-TXT – (ang. *text*) – plik zawierający tekst;

oraz rozszerzenia, które będą często używane w związku z wykorzystywaniem konkretnych, opisanych dalej w książce programów:

-CHI – plik tworzony przez edytor tekstowy CHIWRITER (por. podrozdz. 3.5.);

-DBF – baza danych tworzona przez program DBASE (por. podrozdz. 3.6.);

-WKS, WK1, WK2 – dane z arkuszy kalkulacyjnych, utworzone programem *LOTUS* 123 (por. podrozdz. 3.7).

[1] Dodatkową zaletą takiej struktury jest fakt, że nazwy plików w różnych katalogach mogą się powtarzać bez powodowania niejednoznaczności. Na przykład jeśli kilku użytkowników korzysta z tego samego komputera, to każdy z nich – jeśli chce – może gromadzić swoje dane w pliku o nazwie DANE (bo tak najwygodniej!), ale jeśli każdy będzie dysponował oddzielnym katalogiem, to te dane nie pomieszają się ze sobą. Gdyby nie było udogodnienia, jakim są oddzielne katalogi, wymyślanie nazw dla plików mogłoby być bardzo uciążliwe i trudne!

Czasami zamiast nazwy można podawać jej wzorzec wykorzystując "pseudonimy":

* – zastępuje dowolny ciąg znaków do końca nazwy

? – zastępuje jeden dowolny znak.

Polecenie, które wykorzystuje tego rodzaju wzorce jest wykonywane dla wszystkich plików o nazwach zgodnych z wzorcem.

Dla każdego polecenia można także podać pewne parametry dodatkowe kierujące wyniki wykonywanych poleceń "w inną stronę" lub dokonujące na nich dodatkowych operacji. Oto postać tych parametrów:

- **> nazwa_pliku** – powoduje wpisanie wyników do pliku o nazwie **nazwa_pliku**;
- **>> nazwa_pliku** – działa jak wyżej, ale wyniki te zostaną dopisane na końcu pliku o nazwie **nazwa_pliku**;
- **< nazwa_pliku** – czytanie danych dla polecenia następuje z pliku zamiast z klawiatury;
- **program1 | program2** – złożenie programów w tak zwany potok, tzn. wyniki z jednego programu są danymi dla następnego.

3.4.4. Wybrane polecenia systemu MS DOS

3.4.4.1. Uwagi wstępne

Pełny wykaz poleceń systemu MS DOS (wraz z objaśnieniami) zajmuje w typowej dokumentacji komputera trzystu stronicową książkę. Nic dziwnego zatem, że podane niżej objaśnienia stanowią jedynie pewien wyciąg z tej bogatej dokumentacji. Przedstawiłem jedynie najczęściej używane polecenia[1], zestawiając je w sposób **tematyczny** (w większości podręczników opis jest ujęty alfabetycznie), a objaśnienia i przykłady ograniczyłem do niezbędnego minimum. Wszystko to sprawia, że opanowanie podstaw korzystania z systemu MS DOS jest – w ujęciu tej książki – szybkie i proste, ale powierzchowne i niekompletne, zatem wymagające w przyszłości uzupełniających studiów.

3.4.4.2. Najprostsze polecenia elementarnej obsługi komputera

W tym podrozdziale zebrane zostały niektóre najprostsze polecenia systemu MS DOS. Polecenia te są na ogół mało przydatne, gdyż z ich pomocą nie możesz zlecić komputerowi żadnych naprawdę istotnych czynności. Jednak zapoznanie z systemem warto rozpocząć od wydania mu tych właśnie poleceń i zaobserwowania jego reakcji, gdyż za pomocą tych operacji nie można niczego popsuć, ich użycie jest więc całkowicie bezpieczne. Efekt działania kilku z nich pokazuje fotografia obok.

- **help** – wyświetlenie na ekranie podręcznika używania systemu **DOS.** Podręcznik ten jest bardzo obszerny i oczywiście wyświetlany jest po angielsku, dlatego zwykle bardziej użyteczne jest następne polecenie.

---

[1] Poszczególne polecenia opisano w następujący sposób: podano ogólny wzór użycia określonego polecenia, pod nim zamieszczono bardzo skrótowe wyjaśnienie czynności wykonywanej przez komputer po podaniu omawianego polecenia i (w przypadku bardziej złożonych poleceń) przedstawiono jeden lub kilka przykładów z objaśnieniami.

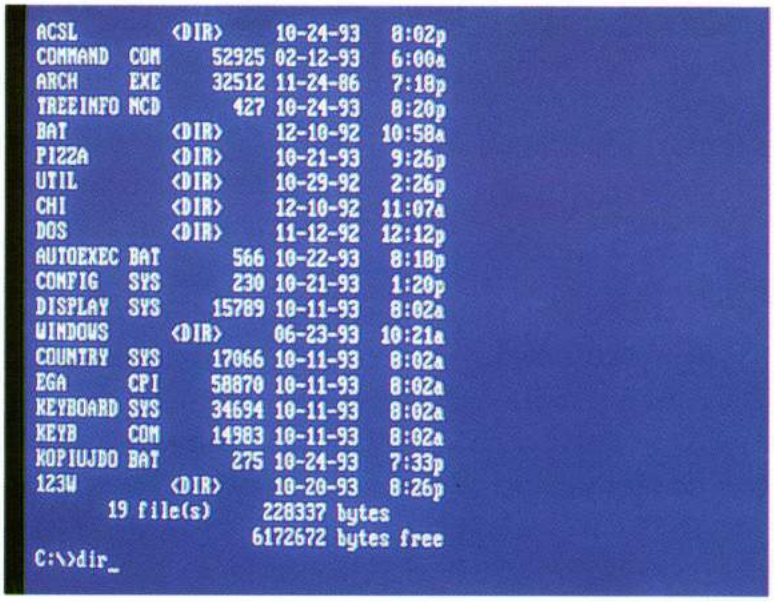
- **fasthelp** – wyświetlenie listy wszystkich poleceń systemu **DOS** z krótkimi objaśnieniami.
- **cls** – wyczyszczenie ekranu.
- **ver** – podanie numeru wersji systemu.
- **time** – wypisanie aktualnego czasu[1].
- **date** – Wypisanie aktualnej daty.
- **more** – skopiowanie strumienia danych na monitor strona po stronie

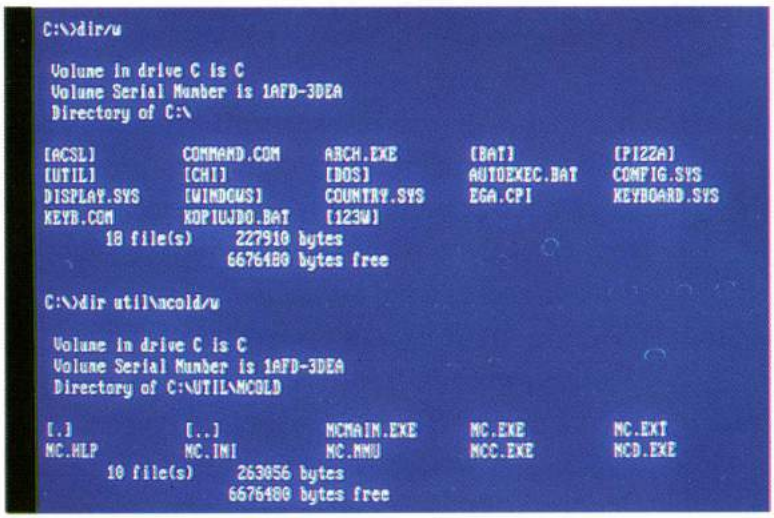
  Przykład:

  more < prog.txt

wypisuje na monitorze tekst pliku "prog.txt"

3.4.4.3. Operacje na całym dysku lub na całym katalogu





Podstawową czynnością przy pracy z dyskiem jest żądanie podania zawartości aktualnego katalogu **dir**. Inna często stosowana operacja to kopiowanie zawartości całej dyskietki **diskcopy**. Używając nowych dyskietek, trzeba na początku posłużyć się poleceniem[2] **format**, nakazującym przygotowanie dyskietki do wpisywania na niej plików.

A oto te polecenia:

- **dir {nazwa} {/p} {/w}** – wypisanie informacji o plikach. "Nazwa" określa grupę plików lub ścieżkę do katalogu. Jej brak oznacza wypisanie wszystkich plików z bieżącego katalogu.

  OPCJE:

  /p – wypisywanie strona po stronie

  /w – wypisanie w skróconej formie.
- **diskcopy x: y:** – skopiowanie całej zawartości dysku X: na dysk Y:[3].
- **format x: {/v:etykieta} {/s} {/b} {/q}** – sformatowanie[4] dysku X.

  OPCJE:

  /v – zapisanie etykiety na dysku (dla jego późniejszego rozpoznania);

  /s – zapisanie na dysku plików systemowych (dysk może służyć jako systemowy);

---

[1] Komputer prosi też o podanie czasu celem ewentualnego skorygowania swojego zegara. Jeśli nie chcesz wnosić korekty, wystarczy nacisnąć klawisz **Enter**.

[2] Omawiane tu polecenia mogą mieć **parametry**. Postać najczęściej używanych parametrów podano w objaśnieniach, przy czym używano nawiasów { i } dla oznaczenia tych części polecenia, które **mogą, ale nie muszą wystąpić**. Stosując polecenia w praktyce, można wpisać odpowiedni parametr (ale bez nawiasów {} !) lub można go pominąć. **Uwaga: każde polecenie systemu DOS można wywołać z parametrem /?. Uzyskuje się wtedy na ekranie komputera krótką "podpowiedź" jak tego polecenia używać.**

[3] Jeżeli X = Y, to program będzie żądał wymiany dyskietek w stacji X podczas kopiowania.

[4] Operacja formatowania we wczesnych wersjach systemu MS DOS oznaczała bezwarunkową utratę wszystkich plików. Dziś tak być nie musi, bo pliki można odzyskać za pomocą polecenia **unformat**, ale i tak trzeba to stosować ostrożnie.

/b – pozostawienie miejsca na pliki systemowe (można je potem dopisać);
/q – tzw. szybkie formatowanie (bez fizycznego usuwania plików).

3.4.4.4. Operacje na poszczególnych plikach

```
E:\LANG\PASCAL>copy dane.pas a:dane.bak
        1 file(s) copied

E:\LANG\PASCAL>dir a:

 Volume in drive A has no label
 Directory of A:\

DANE     BAK        539 10-16-93  11:43p
        1 file(s)        539 bytes
                     1212928 bytes free

E:\LANG\PASCAL>_
```

```
A:\>dir

 Volume in drive A has no label
 Directory of A:\

DANE     BAK        539 10-16-93  11:43p
        1 file(s)        539 bytes
                     1212928 bytes free

A:\>del dane.bak

A:\>dir

 Volume in drive A has no label
 Directory of A:\

File not found
```

Najczęściej potrzebne jest kopiowanie plików **copy** (pliki systemowe wymagają do tego polecenia **sys**) lub usuwanie zbędnych plików **del**. Możliwe jest także porównanie zawartości dwóch plików **comp**, a także zmiana nazwy pliku **ren.** Zawartość pliku może być wypisana na ekranie poleceniem **type.**

- **copy plik1 {plik2}** – kopiowanie **pliku1** na **plik2**.

  Przykład:

  copy a:dane

Kopiuje plik "dane" z dysku A do bieżącego katalogu (bo brak było parametru **plik2).**
Uwaga: jeśli w bieżącym katalogu był wcześniej plik o nazwie **dane,** to zostanie zmazany!

- **del plik** – usunięcie pliku.

  Przykłady:

  del dane

Usunięcie z bieżącego katalogu pliku **dane.**

  del *.*

Usunięcie wszystkich plików z bieżącego katalogu (**Uwaga:** bardzo niebezpieczne!).

- **comp plik1 plik2** – porównanie obu podanych plików ze sobą[1].

  Przykład:

  comp c:*.* a:

  Porównanie kolejno wszystkich par plików o tych samych nazwach z dysków **C** i **A**.

- **ren nazwa1 nazwa2** -- nadanie plikowi "**nazwa1**" nowej nazwy "**nazwa2**"

  Przykład:

  ren *.txt *.doc

  Zmiana nazw wszystkich plików o rozszerzeniu TXT na nazwy z rozszerzeniem DOC.

- **sys x:** – zapisanie zbiorów systemowych na dysku X[2].

- **type plik** – wypisanie zawartości pliku

  Przykład:

  type program.txt | more

  Powoduje wypisanie na monitor tekstu pliku "PROGRAM.TXT" strona po stronie.

Uwaga: omówione wcześniej polecenie systemowe **more** działa tu jak "filtr" i powoduje zatrzymanie procesu wypisywania po wypełnieniu całego ekranu. Naciśnięcie klawisza powoduje pokazanie na ekranie następnej porcji informacji.

- **sort < plik** – sortowanie zawartości podanego pliku.

---

1 Program wypisuje dziesięć pierwszych różnic. Jeśli pliki są identyczne, generowany jest komunikat: ***Files compare OK*** .
2 Dysk X powinien być sformatowany z opcją **/s** lub **/b,** plik COMMAND.COM trzeba skopiować oddzielnie poleceniem **copy.**

3.4.4.5. Wędrówka po drzewie katalogów

System pomaga w przenoszeniu się z jednego miejsca w drzewie katalogów w inne miejsce. Służy do tego polecenie **cd**. Ponadto system pozwala tworzyć nowe "gałęzie" (podkatalogi) za pomocą polecenia **md** (ilustruje to sekwencja fotografii z lewej) lub je "odcinać" za pomocą polecenia **rd**.

- **cd ścieżka** – ustalenie katalogu bieżącego (na roboczym lub podanym dysku) na katalog znajdujący się na końcu "ścieżki"

  Przykłady:

  cd \lotus

  Zmiana katalogu na **\lotus** bez względu na aktualne położenie.

  cd ..

  Zmiana katalogu bieżącego na katalog macierzysty (to znaczy bezpośrednio go poprzedzający).

- **md {ścieżka\} nazwa** – utworzenie podkatalogu o podanej nazwie w katalogu bieżącym lub na końcu podanej ścieżki.

  Przykłady:

  md podania

  Utworzenie podkatalogu "**podania**" w bieżącym katalogu.

  md c:\dokumenty listy

  Utworzenie podkatalogu **listy** w katalogu **dokumenty.**

- **rd ścieżka** – usunięcie katalogu znajdującego się na końcu "ścieżki".

Uwaga: katalog musi być pusty!!

3.4.5. *Przetwarzanie wsadowe w systemie MS DOS*

```
Edit: C:\kopiujdo.bat                    Line 1
echo off
echo w A: powinna byc dyskietka
arch a %1
copy %1.arc a:
del %1.arc
dir a:
rem ten bardzo krotki program przy pomocy
rem archiwizera arch pakuje wszystkie pliki
rem aktualnego katalogu do archiwum o podanej
rem jako parametr nazwie i zapisuje na dysku a:
```

Jedną z cech ułatwiających pracę w systemie DOS jest tzw. **przetwarzanie wsadowe**. Ułatwia ono pracę z komputerem wtedy, gdy trzeba wiele razy wykonywać te same czynności. Korzystanie z przetwarzania wsadowego polega na tym, że użytkownik zamiast wiele razy wydawać za pomocą klawiatury wszystkie kolejne polecenia systemowi MS DOS, może je zapisać w utworzonym przez siebie pliku jako zwykły **tekst**[1] (polecenia systemowe zawsze **są tylko napisami,** określającymi, co system ma dla Ciebie zrobić).

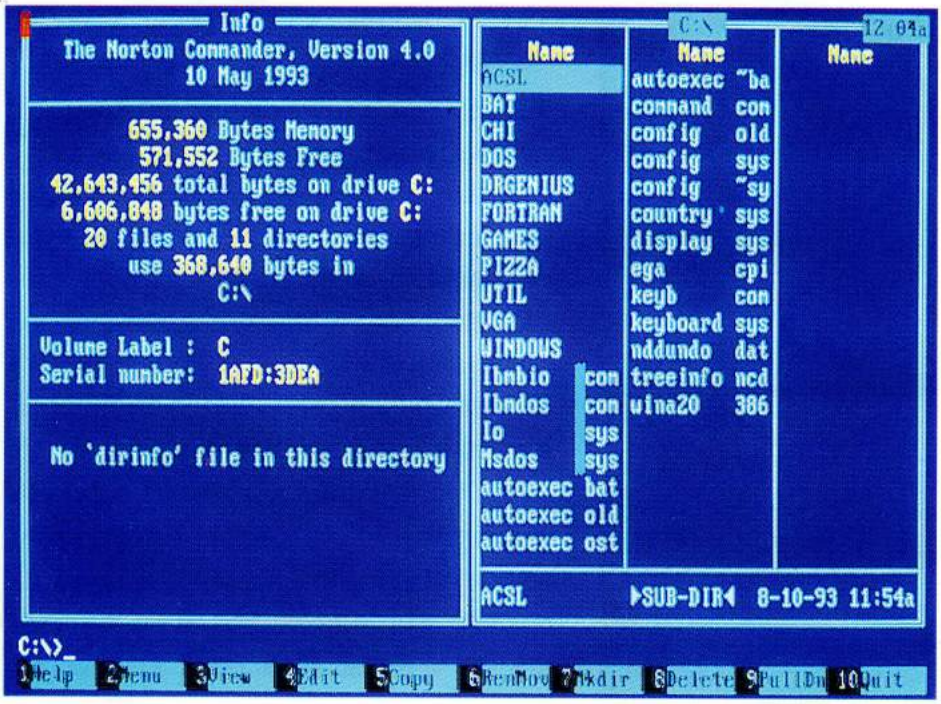
[1] Najwygodniej jest użyć do tego jakiegoś edytora – na przykład opisanego dalej edytora **NC.**

Jeśli następnie jako polecenie systemu DOS poda się nazwę tego pliku[1], zacznie się **automatyczne** wykonywanie całej sekwencji wcześniej zapisanych poleceń[2]. Po prostu DOS całkiem sam, bez żadnego Twojego udziału, odczytuje z podanego pliku kolejne komendy, a następnie je dokładnie realizuje. Przetwarzanie wsadowe wykonywane przez system może być zatrzymane przez naciśnięcie klawiszy **Ctrl+Break**. DOS może realizować tylko jedno przetwarzanie wsadowe, lecz ostatnią komendą pliku wsadowego może być **uruchomieniem następnego przetwarzania wsadowego**.

Wśród plików wsadowych wyróżniony jest plik **AUTOEXEC.BAT**, którego istnienie badane jest przez DOS podczas inicjowania pracy systemu. Jeśli ten plik istnieje, to DOS automatycznie uruchamia przetwarzanie wsadowe tego pliku.

3.4.6. Nakładki ułatwiające pracę z systemem MS DOS

Prace z systemem operacyjnym można znacznie ułatwić stosując tak zwane "nakładki" (ang. *shell*). Jest ich wiele (**Norton Commander, Xtree, Pro Gold, StupenDOS, File Manager, Tree-view, Directory Manager, Disk Director** i wiele innych). Ich wspólną cechą jest zmiana niewygodnego sposobu sterowania pracą systemu operacyjnego za pomocą wydawanych komend. Wykorzystując wspomniane nakładki, możesz wygodnie oglądać i porównywać zawartości poszczególnych plików i całych dysków, a typowe czynności (kopiowanie, przenoszenie i usuwanie plików) możesz wykonywać za pomocą uproszczonych poleceń (zwykle wystarcza naciśnięcie klawisza funkcyjnego lub wskazanie myszką).



Jako przykład często używanej nakładki omówimy tu działanie programu **Norton Commander**, którego nazwę (zgodnie z ogólnie przyjętym zwyczajem) będziemy dalej skracać do **NC**. Czynności wykonywane przez program NC, jak się zaraz przekonasz, odpowiadają w większości czynnościom wykonywanym przez sam system MS DOS, jednak podstawowa różnica polega na tym, że NC wykonuje te czynności łatwiej i przedstawia ich wyniki czytelniej.

Podczas pracy z programem **NC** na ekranie komputera stale widoczne są dwa panele[3], zawierające w formie tabelarycznej zawartości dwóch wybranych katalogów. W ten sposób bez uciekania się do polecenia **DIR** masz stale przed sobą wykaz plików, z którymi pracujesz, wykaz –

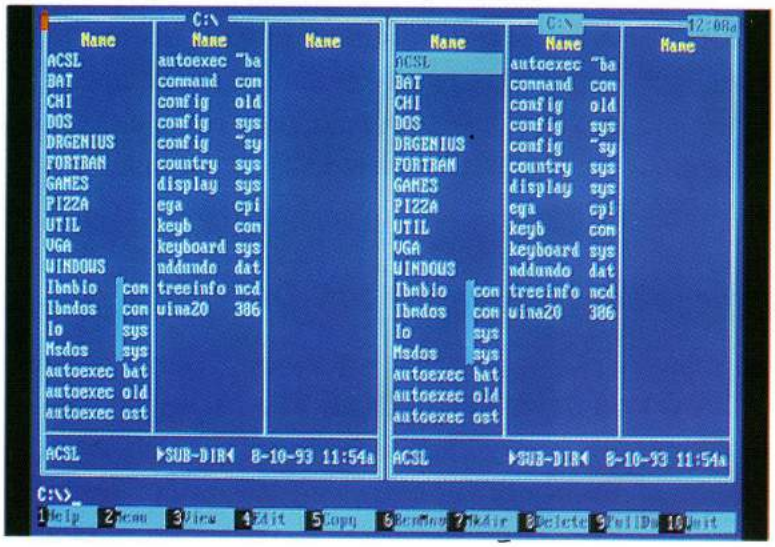
---

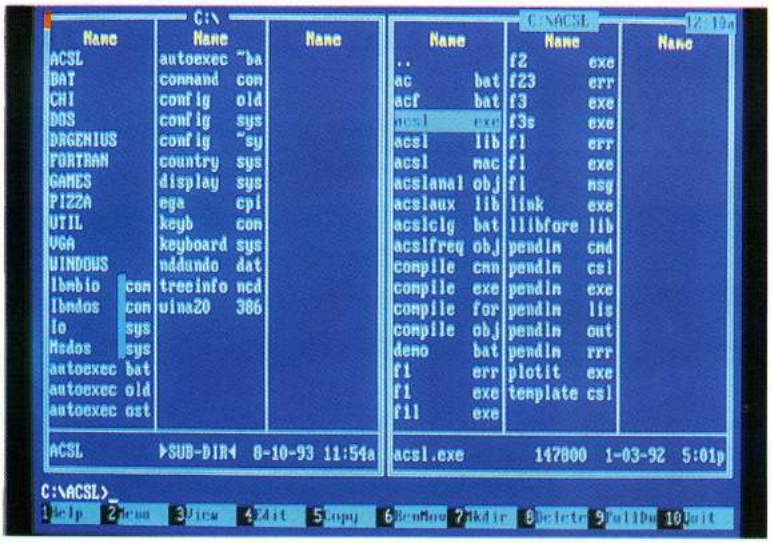
[1] Plik, w którym zawarte są komendy DOS–u przeznaczone do automatycznego wykonywania, nazywany jest plikiem wsadowym i musi mieć rozszerzenie "**.BAT**". DOS szuka takiego pliku, o podanej w formie polecenia nazwie, początkowo w bieżącym katalogu. Jeśli nie znajdzie, to będzie prowadził dalsze poszukiwania we wszystkich katalogach określonych w komendzie **PATH**.

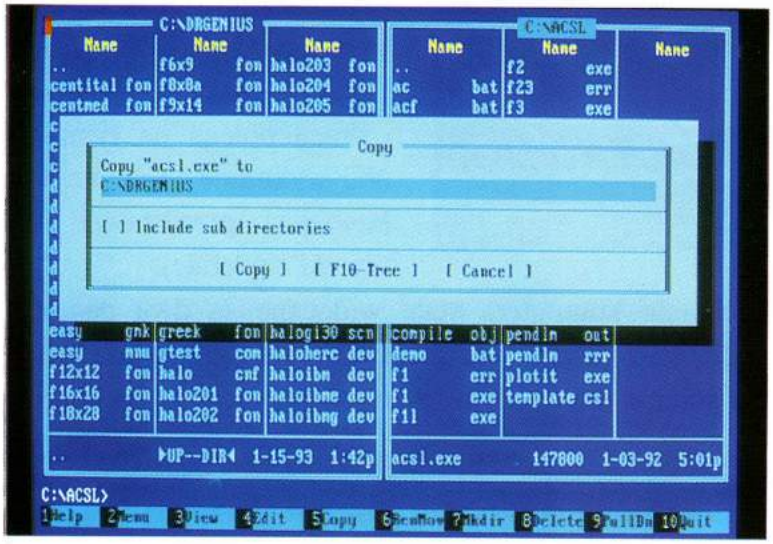
[2] Oprócz standardowych komend DOS–u plik wsadowy może zawierać następujące komendy stosowane głównie do sterowania przetwarzaniem wsadowym: **ECHO, FOR, GOTO, IF, PAUSE, REM i SHIFT.** Użycie tych komend jest stosunkowo łatwe, jednak ich szczegółowy opis wykracza wyraźnie poza zakres tej książki. Zainteresowany Czytelnik powinien w tym zakresie uzupełnić swoją wiedzę opierając się na literaturze. Na szczególne polecenie zasługuje książeczka ***Dana Gookina "DOS dla opornych"***.

[3] Przejście od jednego panelu do drugiego odbywa się po naciśnięciu klawisza **Tab**, natomiast wybór dysku, którego katalogi pokazywane są w panelach najłatwiej wykonać naciskając klawisze **Alt–F1** lub **Alt–F2** (odpowiednio dla lewego i prawego panelu) i wskazując potrzebny identyfikator dysku (**A, B, C** itd.) w wyświetlonym okienku.

dodajmy to – **posortowany** (najczęściej alfabetycznie, ale można inaczej, na przykład według czasu utworzenia poszczególnych plików) i jeśli potrzeba dodatkowo **przefiltrowany** (na przykład zawierający tylko pliki wykonywalne albo tylko pliki o wskazanym rozszerzeniu).







Już samo to jest godne uwagi, gdyż stała świadomość, jakie pliki są dostępne, wraz z wygodną metodą wskazania każdego z nich (kursorem) bardzo ułatwia pracę. Na tym jednak uroki systemu **NC** się nie kończą.

Operowanie systemem plików w programie NC odbywa się z użyciem klawiszy kursora i klawisza **ENTER**. Jeśli naprowadzimy kursor na określony element wchodzący w skład katalogu i naciśniemy klawisz ENTER, możemy uzyskać następujące działania:

- Jeśli wskazany składnik katalogu jest nazwą podkatalogu – nastąpi "wejście" do tego podkatalogu (podobnie jak po wykonaniu polecenia systemowego **CD**) i jego zawartość ukaże się w panelu (pokazują to sąsiednie fotografie);
- Jeśli wskazany składnik katalogu jest nazwą pliku wykonywalnego (z rozszerzeniem **EXE** albo **COM**) względnie nazwą pliku poleceń systemowych (z rozszerzeniem **BAT**) – nastąpi wykonanie tego pliku tak samo, jak po napisaniu nazwy tego pliku w DOS-ie;

W pozostałych przypadkach naciśnięcie klawisza **Enter** nie daje skutku[1].

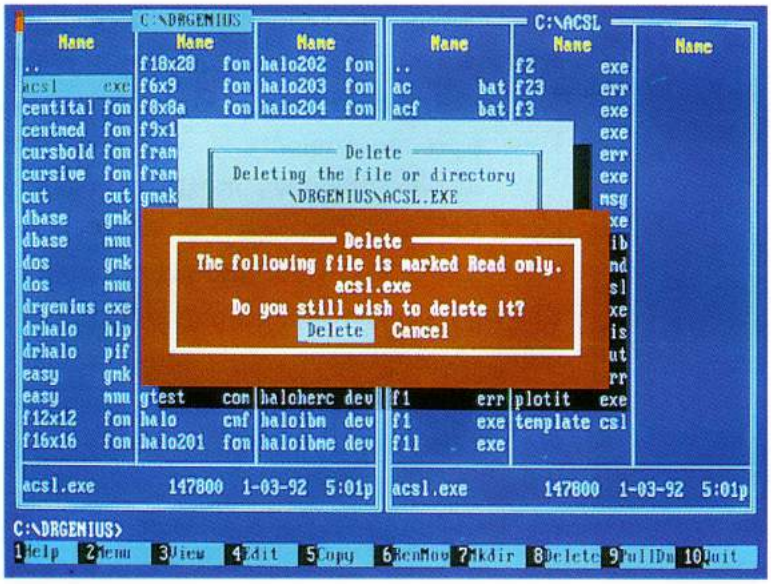
Obok łatwej wędrówki[2] po drzewie katalogów NC oferuje bardzo łatwe wykonywanie podstawowych operacji na plikach. Aby plik skopiować – wystarczy go wskazać i nacisnąć klawisz **F5** (kopiowanie następuje wtedy z katalogu otwartego w jednym panelu do katalogu otwartego w drugim panelu). Obraz kopiowania plików programem NC przedstawia sąsiednia fotografia. Zdążysz polubić ten widok!

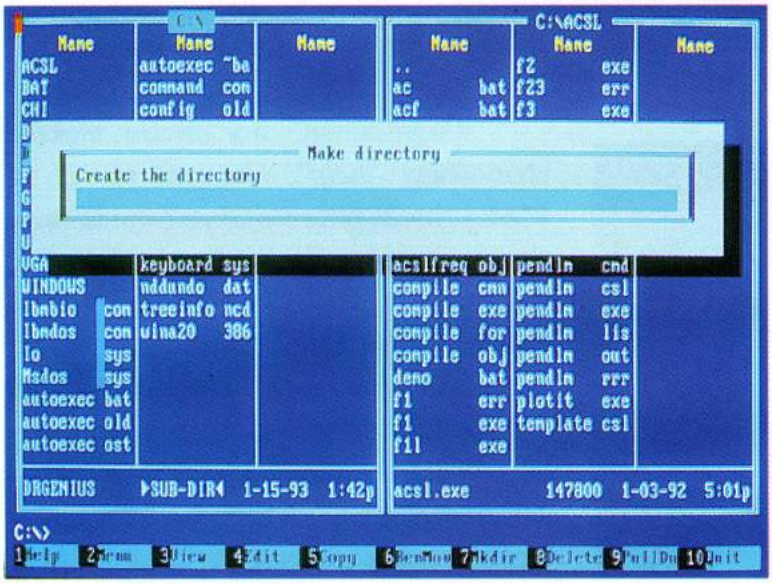
Aby wskazany plik usunąć, zamiast pisać **DEL** i nazwę pliku –

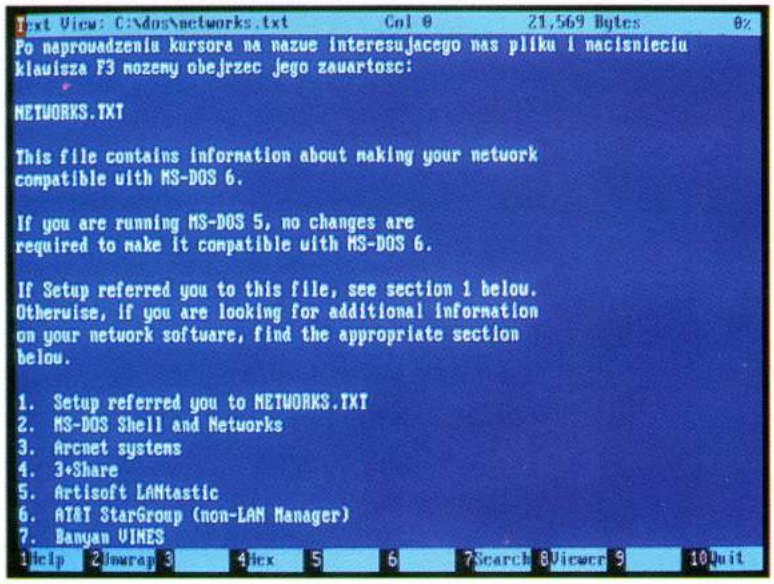
---

[1] Chyba, że wskazany plik ma rozszerzenie, dla którego użytkownik zdefiniował jakąś akcję (na przykład po wskazaniu pliku z rozszerzeniem **CHI** ma być wywołany edytor **ChiWriter**) – wówczas wywoływany jest potrzebny program – ale to już "wyższa szkoła".

[2] Warto dodać, że wędrówka taka może się odbywać zarówno "w głąb" (do kolejnych, zawartych jeden w drugim katalogów), jak i "na zewnątrz" (od katalogu pochodnego do katalogu macierzystego), ponieważ na początku wykazu składników każdego katalogu (z wyjątkiem *root*) podawany jest zwykle symbol .. oznaczający przejście do górnego katalogu.







naciska się klawisz **F8**. Ponieważ czynność kasowania może spowodować utratę potrzebnych informacji – program **NC** ostrzega użytkownika, że polecił skasować wskazany plik i prosi o potwierdzenie, dając do wyboru dwie możliwości – skasowania pliku lub cofnięcia polecenia (patrz fotografia obok).

Aby plik przemieścić (skopiować w inne miejsce i skasować w miejscu, gdzie był pierwotnie) – wystarczy nacisnąć klawisz **F6.** Ten sam klawisz **F6** służy do zmiany nazwy pliku (odpowiednik polecenia REN systemu DOS).

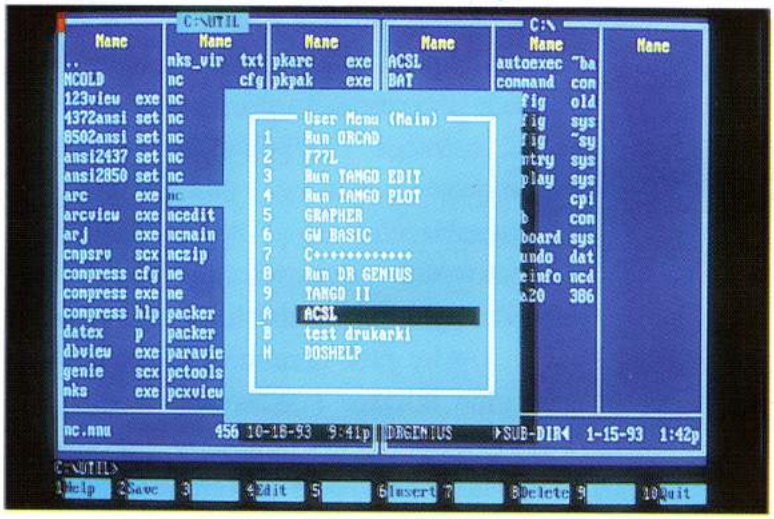
Wreszcie klawisz **F7** pozwala tworzyć nowe katalogi (identycznie jak polecenie **MD** systemu DOS). Fotografia obok pokazuje, jak łatwo i wygodnie można to zrobić.

Omówione czynności programu NC odpowiadały dokładnie funkcjom systemu operacyjnego; rola programu NC sprowadzała się w tym przypadku wyłącznie do ułatwienia Twojej pracy, abyś zamiast pisać określone komendy mógł osiągnąć zamierzony cel naciśnięciem jednego klawisza. Do tego jednak funkcje programu NC się nie ograniczają.

Za pomocą klawisza **F3** można obejrzeć zawartość dowolnego pliku. Jeśli jest to plik tekstowy – można go przeczytać, jak to widać obok. Jeśli jest to plik z programem – można go także analizować i to zarówno w postaci znakowej, jak i w formie kodów heksadecymalnych (szesnastkowo wyrażonych zawartości poszczególnych bajtów). Można też oglądać inne pliki (np. z baz danych).

Klawisz **F3** pozwala jedynie oglądać[1] zawartość wskazanych plików, natomiast klawisz **F4** aktywizuje bardzo zgrabny edytor (opisany szczegółowo nieco dalej), pozwalający wygodnie zmieniać zawartość dowolnych plików. Oczywiście taka zmiana może być łatwo i bezpiecznie dokonana w przypadku pliku, który zawiera tylko teksty. W przypadku pliku z

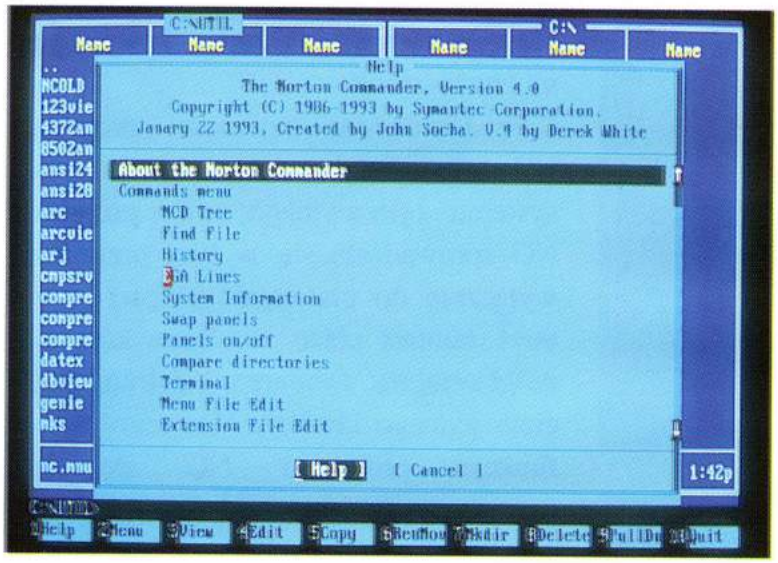
[1] Programy (bo jest ich kilka), które automatycznie uruchamia NC w momencie oglądania różnych plików, pozwalają na oglądanie w dogodnej formie dokumentów tworzonych przez różne edytory, rekordów baz danych, fragmentów arkusza kalkulacyjnego itp. Jest to nieocenione narzędzie przy wszelkich przeglądach plików na dyskach, porządkowaniu informacji, poszukiwaniu potrzebnego pliku itd.



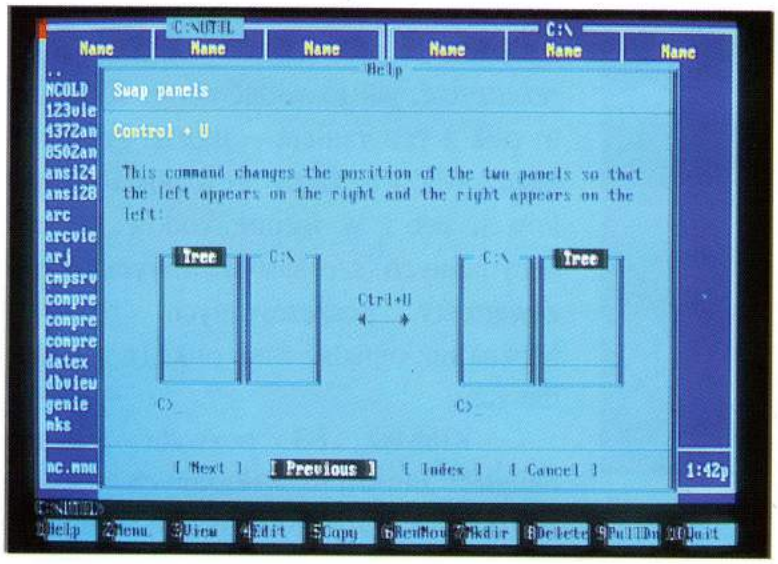
programem (np. typu **EXE**) użycie klawisza **F4** wymaga dużej wiedzy i może być niebezpieczne (program po poprawkach może nie działać!).

Bardzo ciekawa możliwość wiąże się z użyciem klawisza **F2**. Aktywizuje on tzw. **menu**, czyli wykaz często wykonywanych czynności, które (po odpowiednim zaprogramowaniu) program NC może wykonywać całkiem automatycznie. Rola użytkownika sprowadza się wtedy do wyboru z menu, przez wskazanie, potrzebnej czynności.

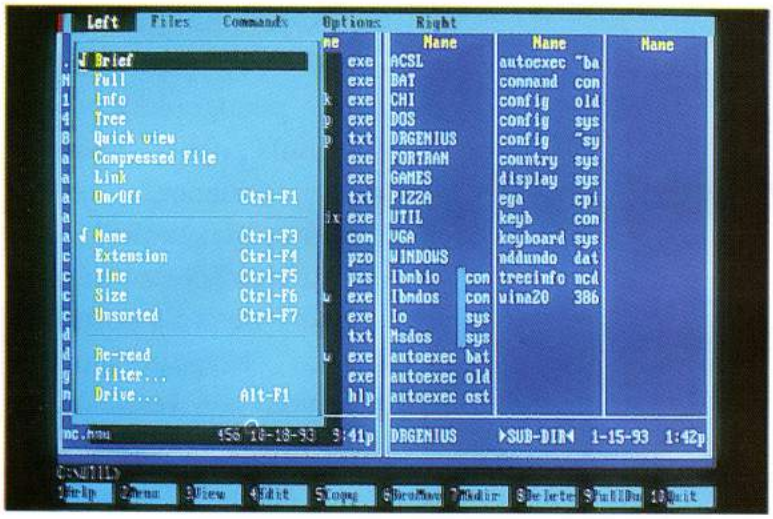
Opisane wyżej usługi programu NC należały do jego podstawowego i najprostszego "repertuaru". Ten bogaty i mądry program potrafi o wiele więcej, tylko szkoda miejsca na opisywanie tych bardzo licznych dodatkowych możliwości. Dlatego zamiast pełnego wykazu podam jeszcze tylko trzy użyteczne informacje.

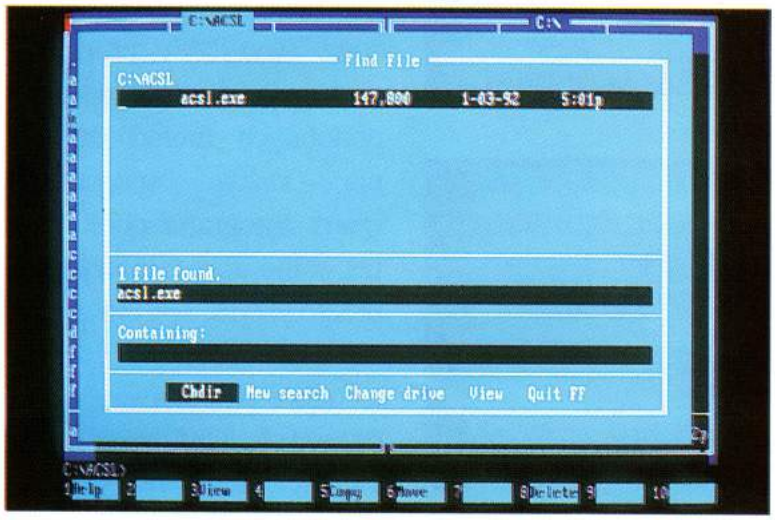


Program NC ma bardzo pięknie zbudowany "*help*". Oznacza to, że w każdej chwili podczas użytkowania programu można nacisnąć klawisz **F1**, otrzymując niezwłocznie podpowiedź na ekranie. Fragment "spisu treści" tej ściągawki pokazuje fotografia obok. Podpowiedź dotyczy zwykle poprawnego wykonania tej czynności, którą właśnie zacząłeś i nie wiesz, jak kontynuować. Można jednak w każdej chwili uzyskać informacje na dowolny temat, posługując się ekranem podpowiedzi jak książką – szukając potrzebnej informacji na kolejnych stronicach, które można łatwo "przewracać", a także można skorzystać z wyświetlonego na ekranie spisu treści, w którym wystarczy wskazać dowolne zagadnienie, aby natychmiast uzyskać szczegółową informację na zadany temat (patrz obok). Przy odrobinie cierpliwości można z tych "podpowiedzi" (niestety – podawanych wyłącznie po angielsku) uzyskać pełne informacje o wszystkich możliwościach programu NC. Cierpliwość jest jednak konieczna, bo wyświetlana na ekranie informacja liczy blisko 200 stronic!



Druga wiadomość, otwierająca drogę do licznych ciekawych przygód i eksperymentów z programem NC, wiąże się z użyciem klawisza **F9**. Za pomocą tego klawisza przechodzi się do szeregu rozwijalnych menu , znajdujących się u góry ekranu, ponad panelami pokazującymi





zawartości wybranych katalogów. Poruszając się wśród tych menu[1] za pomocą klawiszy kursora (patrz fotografia) i zatwierdzając wybór klawiszem **Enter,** możemy uzyskać od programu NC mnóstwo zaskakująco użytecznych usług.

Nie chcę Ci psuć przyjemności samodzielnego odkrywania wciąż nowych możliwości tego programu, więc nie podam tu żadnych szczegółów, powiem tylko, że odpowiednio "poproszony" program NC może bezpośrednio przesyłać pliki pomiędzy komputerami, może pokazywać drzewo katalogów w formie przejrzystego rysunku, może wyszukiwać zagubione pliki przeszukując wszystkie katalogi (pokazuje to fotografia obok) itd. Większość użytkowników nie ma wręcz pojęcia, jak bardzo wszechstronne i uniwersalne narzędzie stanowi program NC. Podczas "zwiedzania" menu po naciśnięciu klawisza **F9** wybitnie przydatny okazuje się klawisz **F1** i związane z nim podpowiedzi!

I na koniec ostatnia informacja. Najważniejszą umiejętnością przy używaniu dowolnego programu jest **umiejętność przerwania jego pracy**, gdy już nie jest nam potrzebny. W programie NC taki "hamulec bezpieczeństwa" związany jest z klawiszem **F10**. Jego naciśnięcie powoduje zakończenie pracy – chociaż NC prosi o potwierdzenie tej decyzji (klawiszem **Enter**), żeby uniknąć pomyłek.

3.4.7. MS Windows

Nową jakość w komunikacji użytkownika z komputerami (a w szczególności z ich systemami operacyjnymi) zapewnił program **MS Windows**[2], stanowiący środowisko graficzne[3], pozwalające użytkownikom komputerów na całkowite zapomnienie (jeśli chcą) o poleceniach systemowych i mozolnie wystukiwanych na klawiaturze skomplikowanych rozkazach. Nawet ułatwienia, jakie oferują nakładki w rodzaju opisanego wyżej programu Norton Commander, są mniej wygodne i mniej operatywne niż te, które oferuje nowe graficzne środowisko. Używanie Windows (czyli po prostu "okien") jest tak łatwe, że mimo istnienia wielu podręczników[4], większość osób korzystających z usług tego systemu rezygnuje z ich lektury i siada do komputera bez żadnego przygotowania, licząc

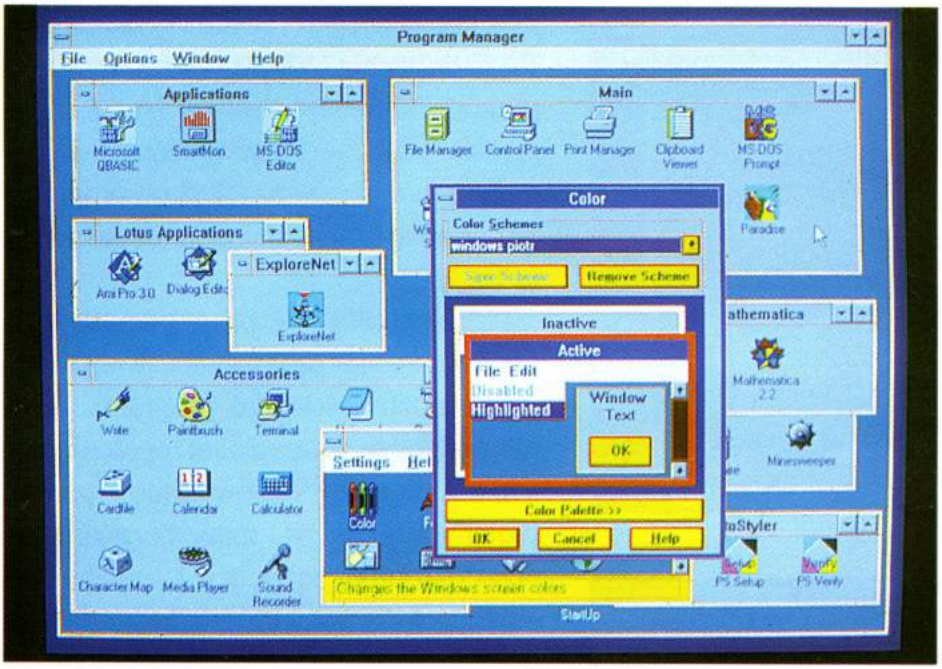
---

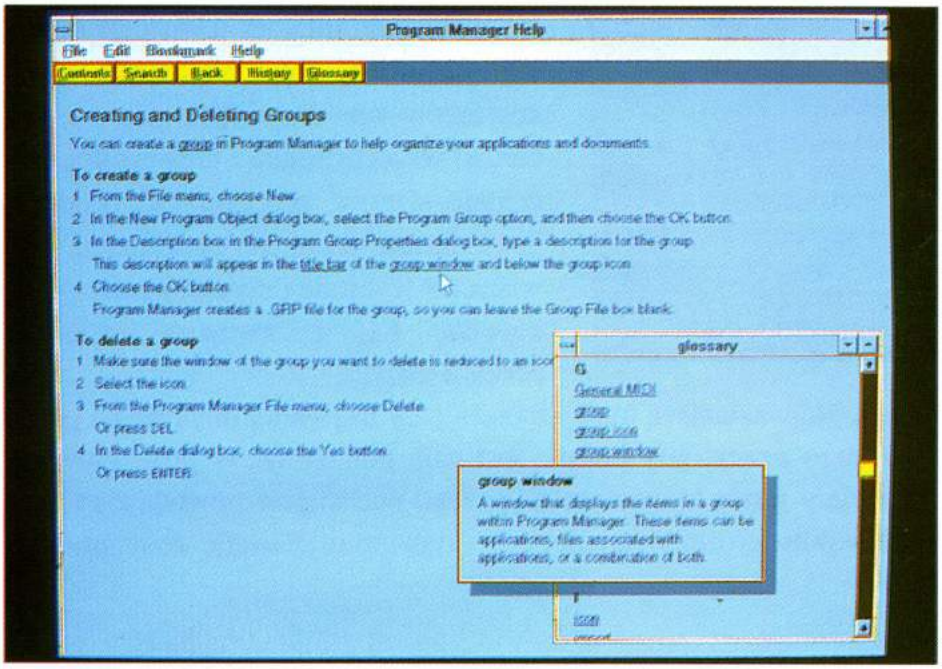
[1] Warto odnotować różnicę między menu **samego programu NC,** do którego można sięgnąć właśnie za pomocą klawisza **F9,** a menu **użytkownika,** które możesz sobie dowolnie zestawić i wywołać klawiszem **F2** w sposób, który opisałem wcześniej.

[2] Program **MS Windows** w wersji oznaczonej **3.0** został po raz pierwszy zaprezentowany **23 maja 1990** roku i ta data uznawana jest za punkt graniczny w historii rozwoju informatyki. Od tego dnia użytkownicy komputerów IBM PC przestali być uzależnieni od komend i komunikatów systemu MS DOS i dostali do swej dyspozycji nowe, wygodne środowisko graficzne, podobne do tego, jakie wcześniej mieli do dyspozycji użytkownicy **Macintosha.**

[3] Poprzednikiem MS Windows był program **GEM** firmy *Digital Research,* który się jednak nie przyjął.

[4] Klasyczną "biblią" użytkowników MS Windows jest podręcznik Briana Livingstona *"Windows 3 Secrets"*, wydana przez IDG Books Worldwide Inc., a w Polsce przez **Wibet** .





na to, że intuicyjna zrozumiałość używanych w systemie piktogramów (patrz fotografia obok) wraz ze stale dostępną, bardzo obszerną "ściągawką" (*help*) do każdego polecenia i programu – pozwolą poznawać system bezpośrednio w praktyce, bez konieczności studiowania teorii.

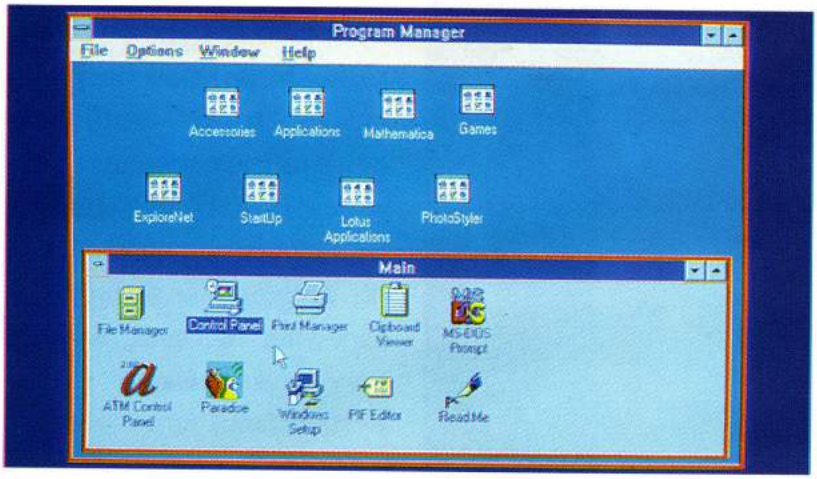
Ze względu na prostotę, czytelność i powtarzalność systemu Windows (wszystkie programy obsługuje się za pomocą podobnych metod działania, każda "ściągawka" (patrz fotografia obok) jest tak samo zorganizowana, z przejrzystym spisem treści na początku i łatwą techniką wyszukiwania potrzebnych informacji) – zwykle udaje się osiągać zamierzone cele tą metodą. Ponieważ jednak posiadanie pewnej wstępnej wiedzy o zasadach działania użytkownika w systemie Windows pozwala znacznie usprawnić działanie i oszczędza czas potrzebny na przeszukiwanie różnych "ściągawek" i skorowidzów (napisanych zresztą po angielsku...), dlatego w tej książce zawarłem podstawowe informacje i rady, które pomogą Ci zapewne we wstępnym poznaniu systemu Windows i w usuwaniu początkowych trudności, które mogą się pojawić.

Poznaj podstawowe własności tej nowej techniki działania. System Windows[1] pozwala realizować wszystkie Twoje potrzeby metodą wskazywania (myszką). Dlatego najważniejszym elementem, który trzeba odnaleźć na ekranie po rozpoczęciu pracy z systemem Windows jest wskaźnik myszki (*cursor*), pokazany na fotografii u góry następnej stronicy. Przesuwając myszką po blacie biurka, powodujesz identyczne ruchy wskaźnika na ekranie, co pozwala na wskazanie dowolnego obiektu[2]. Pamiętaj – myszka i jej kursor to Twoje nowe narzędzia!

---

[1] Nowsza i istotnie udoskonalona wersja programu, oznaczona numerem **3.1**, została przedstawiona **6 kwietnia 1992** roku, a pod koniec **1992** roku ukazał się program **Windows EE**, czyli **w teorii** wersja dla nas, Polaków, gdyż oznaczenie "EE" od dłuższego już czasu funkcjonuje w informatyce światowej jako skrót określenia *Eastern Europ* (Europa Wschodnia) i oznacza wersje programów pozwalające na swobodne używanie elementów alfabetów narodów słowiańskich (polskiego, czeskiego, słowackiego, serbskiego, ukraińskiego itd.). W rzeczywistości Windows EE są narzędziem znacznie mniej wygodnym, niż "klasyczne" okienka ze specjalnymi programami spolszczającymi (np. pakiet **Uni Win**).

[2] Na myszce znajdują się klawisze (dwa lub trzy); operując systemem Windows najczęściej używa się lewego klawisza. Krótkie jednorazowe naciśnięcie tego klawisza oznacza zwykle **wybranie** obiektu, dwukrotne szybkie naciśnięcie klawisza powoduje **wykonanie czynności związanej ze wskazywanym obiektem** (jeśli jakaś czynność jest przewidziana) a przemieszczanie myszki z naciśniętym kla-



Najczęściej wskazywanymi obiektami są opcje menu, ale dla części poleceń zastosowano w Windows jeszcze prostszą metodę działania: oto dla wybrania tych czynności wystarcza wskazanie myszką (i dwukrotne "tupnięcie") odpowiednich symboli graficznych (tak zwanych **ikon**[1]), będących synonimami programów albo określonych czynności. Ekran komputera przypomina blat biurka (*desktop*), na którym poukładane są obrazki symbolizujące programy albo ich grupy. Wskazując obrazek symbolizujący potrzebną w danej chwili grupę programów, uzyskujesz otwarcie na ekranie okna, w którym widoczne są z kolei ikony symbolizujące odpowiednie konkretne programy. Wskazanie ikony równoznaczne jest z rozkazem uruchomienia tego programu, który – na ogół – otwiera swoje własne okno na ekranie itd. Na serii fotografii obok możesz ten proces prześledzić na przykładzie wybrania grupy programów o nazwie **Main,** a z niej programu o nazwie **Control Panel** (jest to program sterujący pracą systemu Windows).

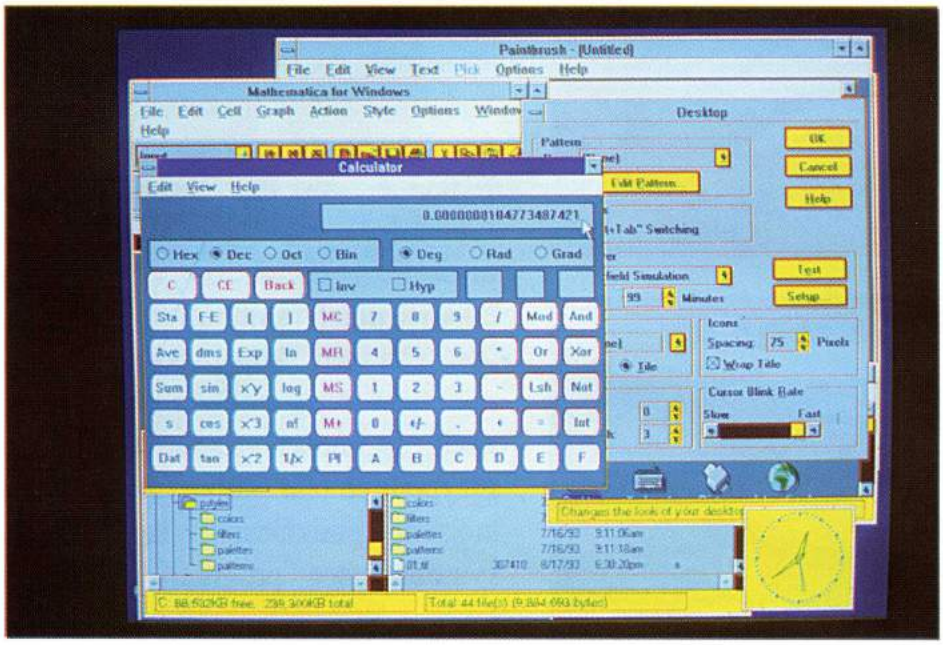
Okno jest prostokątnym obszarem na ekranie, wewnątrz którego zawarte są wszystkie informacje związane z jakimś określonym **procesem** – na przykład z wykonywaniem jakiegoś programu.

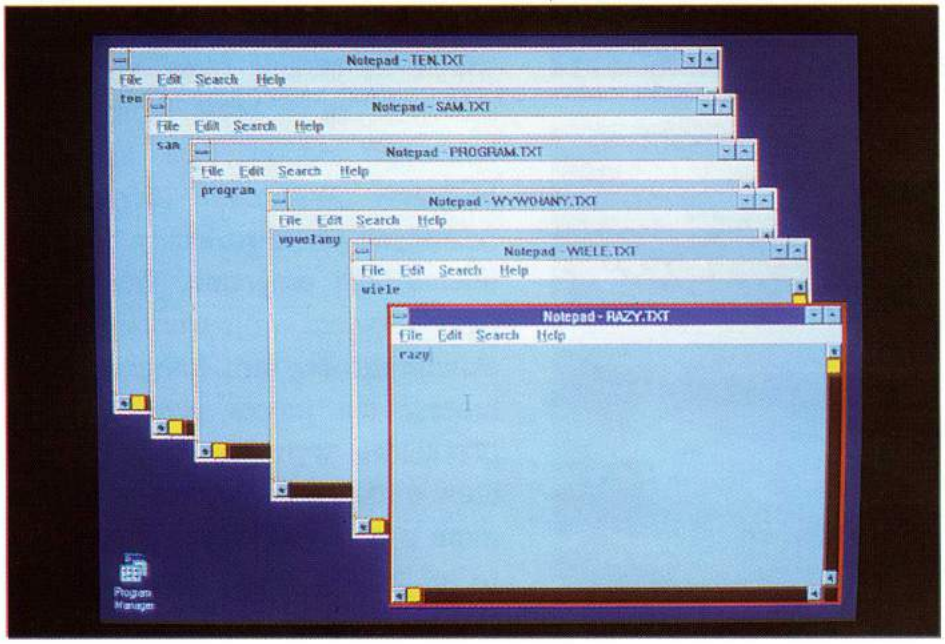
Największą zaletą systemu Windows jest fakt, że użytkownik może otworzyć dowolną liczbę okien, w których może równocześnie uruchomić dowolną liczbę programów. System Windows jest więc systemem **wieloprogramowym** i manipulując oknami można zmusić komputer, żeby równocześnie wykonywał kilka programów, pokazując wyniki każdego z nich w

---

wiszem powoduje "wleczenie" obiektu po ekranie.

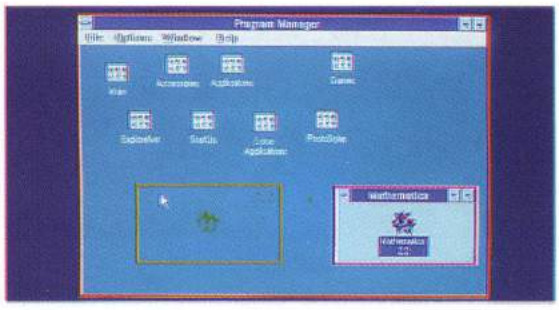
[1] Zyskująca popularność w polskiej terminologii informatycznej nazwa "**ikona**" jest w gruncie rzeczy brzydkim bezpośrednim spolszczeniem anielskiego terminu *icon*, mającego obecnie raczej znaczenie "symbolu" czy "piktogramu", a nie ikony w rozumieniu cerkiewnego malowidła. Nazywanie "ikoną" prymitywnego obrazka wyświetlanego przez komputer razi wielu językoznawców, ale – jak to zwykle bywa – powszechny *usus* językowy jest silniejszy od opinii specjalistów i trzeba się chyba będzie z tą nazwą pogodzić.

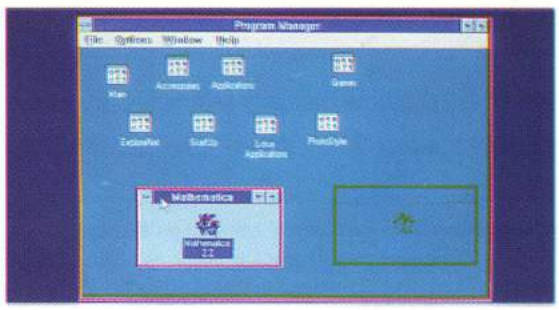


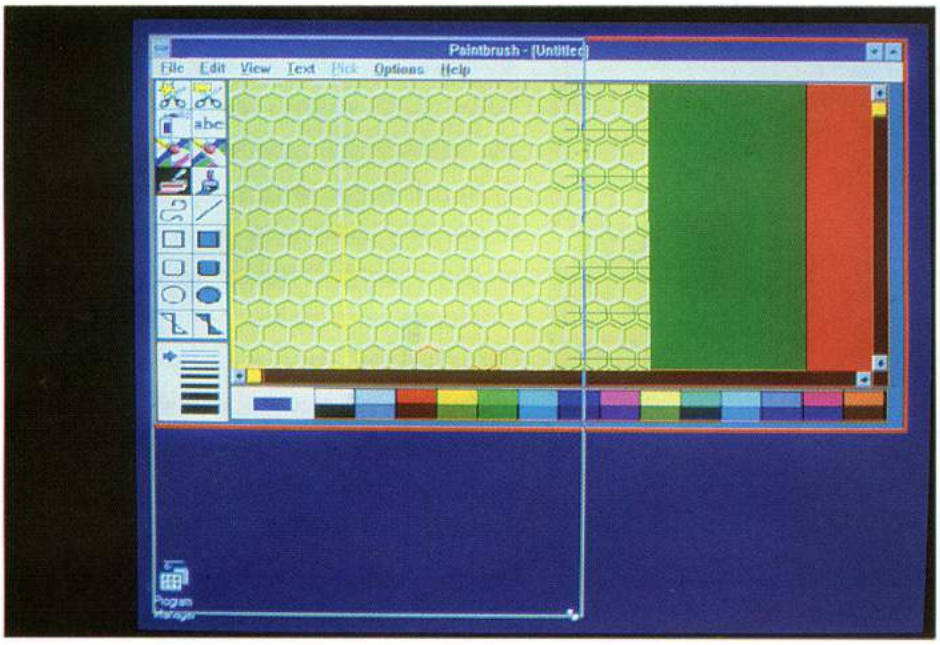


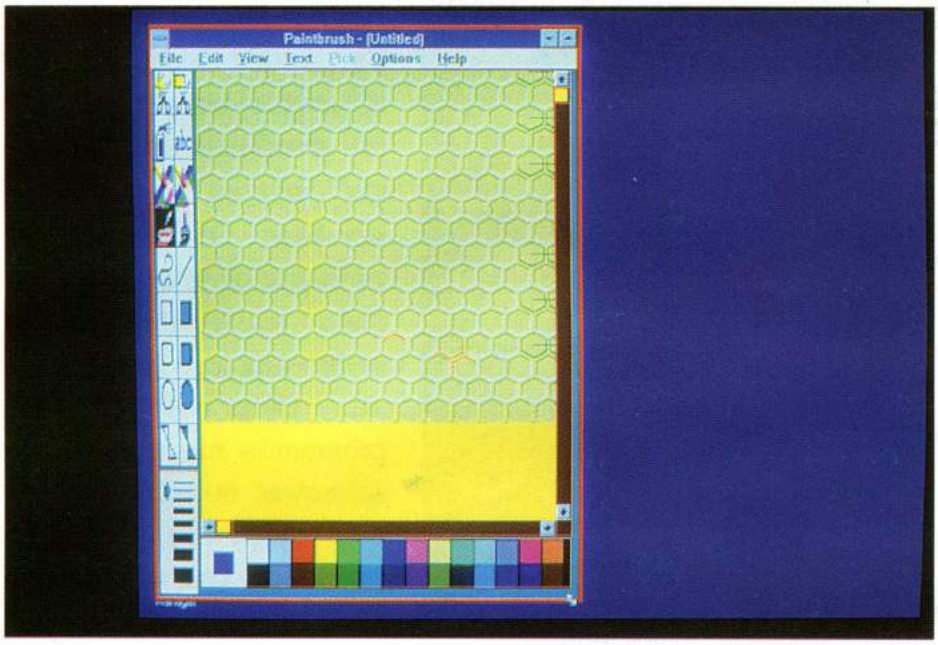
oddzielnym oknie – jak to pokazano na fotografii obok. Co więcej – można uruchomić równocześnie kilka kopii tego samego programu (na przykład edytora), każdą w oddzielnym oknie, i używać ich niezależnie od siebie! Ilustruje to dolna fotografia. Ten system można zresztą rozszerzyć dalej. Niektóre programy (na przykład bazy danych lub skomplikowane procesory tekstu), działając we własnym oknie, mogą tworzyć **okna potomne**, zawierające wyniki pracy programu dla oddzielnych zestawów danych albo te same wyniki w różnej formie (tabelka i wykres). Dzięki możliwości łatwego przenoszenia tekstów, rysunków i danych liczbowych z jednego okna do drugiego – technika ta zapewnia ogromną wygodę przy wykonywaniu złożonych zadań.

Użytkownik może dowolnie rozmieszczać i przemieszczać okna na ekranie. Wystarczy "uchwycić" (za pomocą myszki) okno w miejscu znajdującej się u góry "belki" (*title bar*) i przesuwając myszką (z naciśniętym przyciskiem) możesz przemieszczać okno z jednego miejsca w dowolne inne po całym ekranie. Taka technika, zwana "pochwyć i wlecz" *(catch and drag)*, może być stosowana także do ikon – możesz je na przykład w ten sposób przenosić lub kopiować z jednej grupy do drugiej (gdy reprezentują programy) albo z jednego katalogu do drugiego – jeśli reprezentują pliki. Zasada jest prosta – jeśli "wleczesz" ikonę bez żadnych dodatkowych zabiegów – następuje **przemieszczenie** obiektu (na przykład pliku), jeśli natomiast dodatkowo naciśniesz **Ctrl** – plik lub program zostanie **skopiowany.**





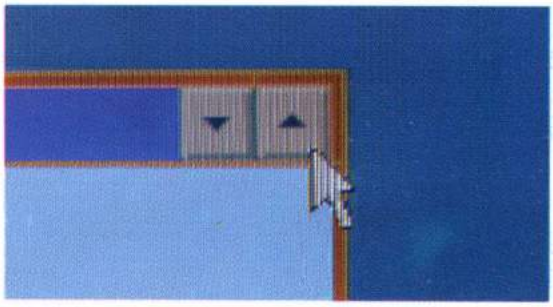


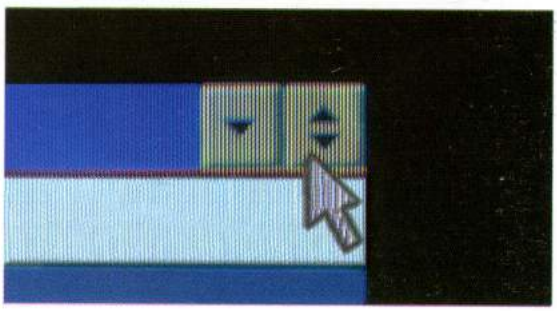


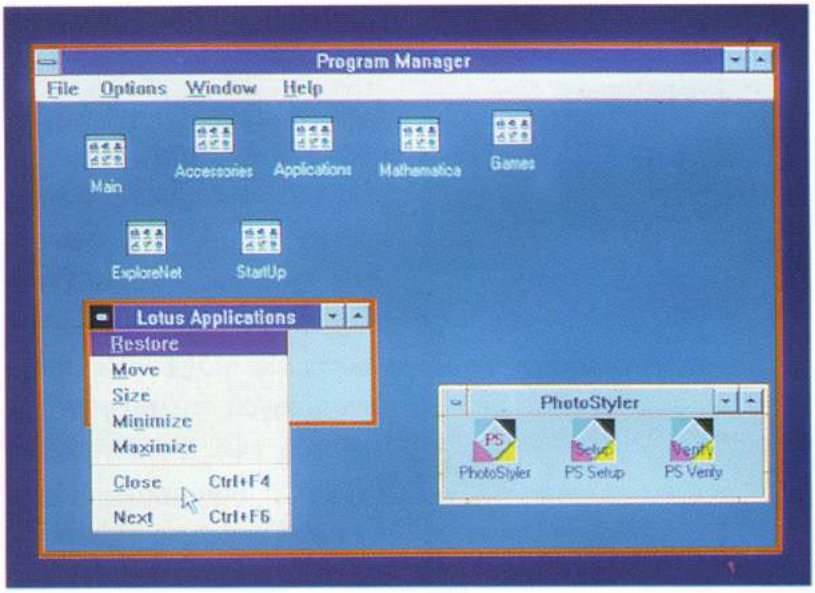
W podobny sposób, chwytając myszką za ramkę okna (*window border*), można je (na ogół) dowolnie powiększać lub zmniejszać. Napisałem "na ogół", ponieważ okna związane z niektórymi programami nie są skalowalne, nie mogą więc mieć zmienianych rozmiarów – stanowią one jednak raczej wyjątek. Fotografie obok pokazują przebieg skalowania okna. Najwygodniej jest uchwycić myszką prawy dolny narożnik okna, bo wtedy można je swobodnie rozciągać zarówno w pionie, jak i w poziomie. Pochwycenie innego punktu ramki okna pozwala na jego skalowanie albo tylko w pionie, albo tylko w poziomie. Skalowanie okien może powodować zmianę rozmiarów obiektów, które w nich występują, ale tak być nie musi – zależy to od rodzaju tych obiektów i programu, który jest uruchamiany w danym oknie.

W prawym górnym rogu okna widoczne są na ogół narysowane na ekranie "klawisze" ze strzałkami w górę i w dół. "Naciśnięcie" (przez najechanie na rysunek klawisza kursorem myszki, naciśnięcie przycisku na myszce) "klawisza" ^ powoduje powiększenie okna do maksymalnego możliwego rozmiaru (na ogół na cały ekran), natomiast naciśnięcie (za pomocą myszki!) klawisza v powoduje zmniejszenie okna do rozmiarów samej tylko ikony. Program związany z oknem zredukowanym do tej postaci nadal działa.

Jeśli okno jest w nietypowym wymiarze (powiększone lub zmniejszone), wówczas w jego narożniku pojawia się klawisz w postaci ◊, którego naciśnięcie przywraca typowy wymiar odpowiedniego okna, tj. taki wymiar, który wynika z jego skalowania.

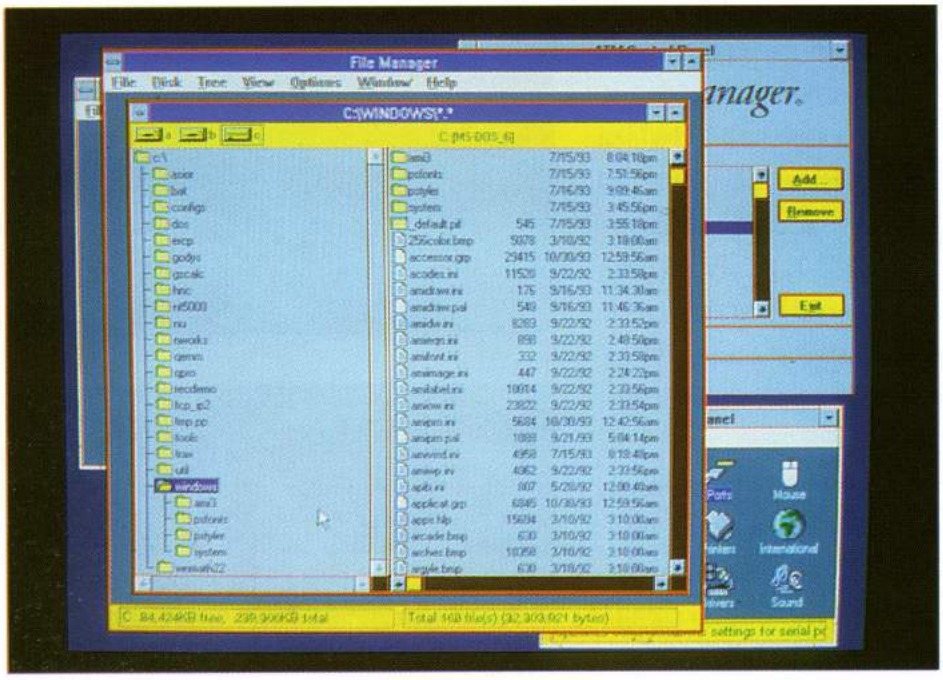


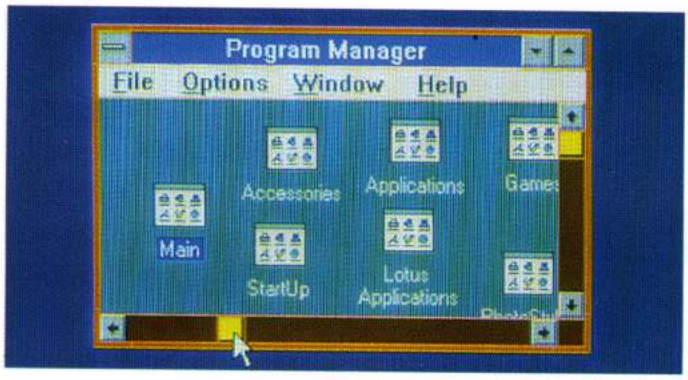




Te same czynności można także wykonywać w inny sposób. Na tytułowej belce okna, w jego lewym górnym rogu, mieści się zawsze przycisk **menu systemowego**, widoczny na fotografii obok. Naciśnięcie tego przycisku powoduje wyświetlenie małego menu, zawierającego polecenia powiększania (**Maximize**) pomniejszania (**Minimize**) lub odtwarzania w standardowej wielkości (**Restore**) danego okna. Dodatkowo menu to zawiera pozycję zamknięcia okna (**Close**), której wybranie powoduje **zakończenie działania odpowiedniego programu i usunięcie jego okna z ekranu w sposób ostateczny**. Ta ostatnia opcja jest najczęściej wybierana i dlatego po **dwukrotnym** "tupnięciu" myszką w klawisz - uzyskasz natychmiastowe zwinięcie okna bez konieczności rozwijania menu i wybierania jego elementów.

Okna poszczególnych programów mogą na ekranie zajmować dowolne położenie, mogą się nawzajem przesłaniać (jak to pokazano na fotografii obok), tworzyć stosy albo równomiernie ułożone talie. Możesz operować jedynie tym oknem, które jest w danej chwili **aktywne**. Jeśli jednak chociaż mały kawałek jakiegoś okna jest widoczny, to można je natychmiast uczynić oknem aktywnym przez wskazanie dowolnego jego skrawka i tupnięcie myszką. Aktywne okno zostaje momentalnie "wyciągnięte" na wierzch (patrz fotografia) i może być podstawą kolejnych działań. Programy związane z

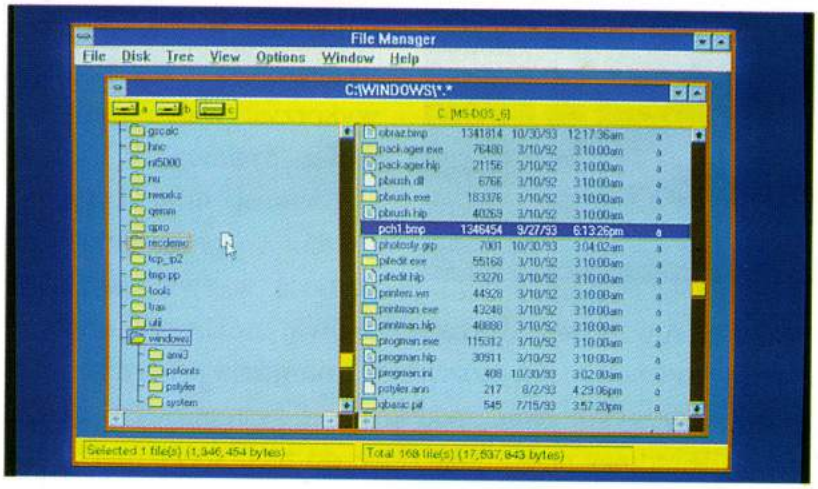


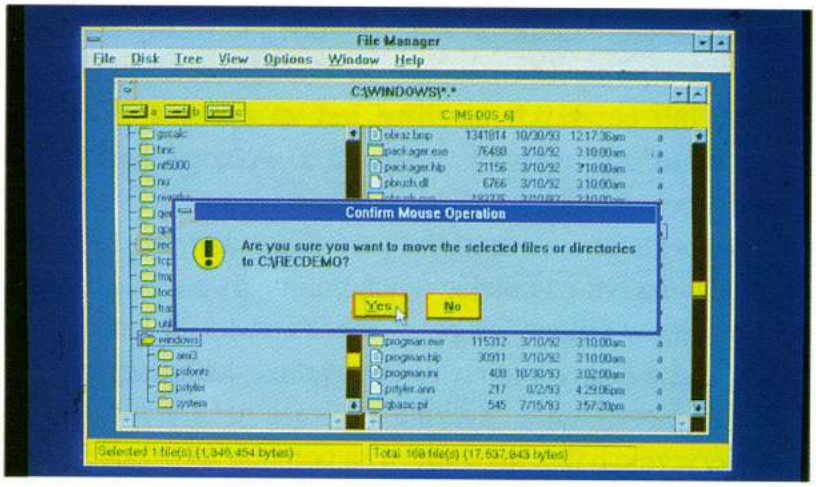


nieaktywnymi oknami działają nadal, ale nie mogą przyjmować danych.

Oczywiście wybrawszy mały rozmiar okna nie możemy widzieć całej jego zawartości na raz, gdyż na małym fragmencie ekranu nie da się sensownie ulokować długiego tekstu lub dużego rysunku. Dlatego okno zawiera zwykle "**suwaki**" (*scroll bars*), za pomocą których można przesuwać jego zawartość. Zwykle widoczny jest pionowy suwak po prawej stronie; z jego pomocą można "przewijać" (*scroll*) widoczny w oknie tekst w górę i w dół, czasem także dostępny jest poziomy suwak u dołu okna, służący do przesuwania tekstu w lewo i w prawo. Suwaki można przesuwać za pomocą myszki techniką "chwyć i wlecz" albo precyzyjniej – "tupiąc" myszką na klawiszach ze strzałkami na końcach suwaków.

Pod górną belką, zawierającą nazwę okna, znajduje się zwykle **belka menu**. Znajdują się na niej napisy w rodzaju ***File, Option, Help*** itp. Wskazanie takiego napisu i tupnięcie myszką powoduje rozwinięcie dodatkowego menu (patrz fotografia), z którego można wybrać żądaną czynność. Zwłaszcza korzystanie z bogatego menu **pomocy** (*Help*) jest bardzo wskazane.





Program MS Windows jest głównie nakładką na system operacyjny, pozwalającą na łatwe sterowanie pracą różnych programów lub manipulowanie plikami dyskowymi. Do sterowania pracą programów służy aplikacja **Program Manager**. Pozwala on wybrać i uruchomić (**Run**) dowolny program. Natomiast wszelkie operacje związane z plikami na dyskach można wykonać za pomocą wygodnego narzędzia o nazwie **File Manager**, wyłącznie za pomocą myszki. Na przykład kopiowanie (*copy*) lub przemieszczanie (*move*) plików polega na ich "wleczeniu" myszką z jednego katalogu do drugiego (patrz fotografie obok) w obrębie okna **File Manager**. Posługiwanie się tym programem jest bardzo łatwe, ponieważ **File Manager** pokazuje strukturę drzewa katalogów i zawartość wybranego katalogu w sposób dość podobny do

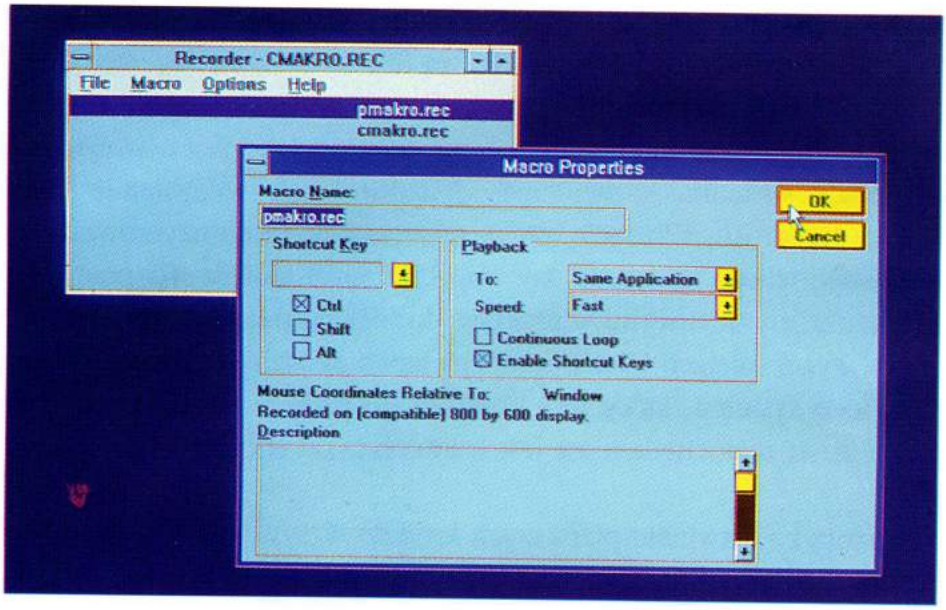
tego, jaki oferuje **Norton Commander.**

MS Windows oferuje użytkownikowi także szereg gotowych do wykorzystania pomysłowych programów. Są to między innymi: edytor, program rysujący, kalkulator, kalendarz z terminarzem, zegarek, notatnik, prosta kartotekowa baza danych. Ta lista może być łatwo uzupełniana o kolejne zadania i kolejne programy, włączane do systemu MS Windows za pomocą programu konfigurującego **Setup** albo pisane samodzielnie przez użytkownika.

Przy pomocy systemu **MS Windows** można uruchomić każdy program działający na komputerze IBM PC, jednak niektóre programy są pisane tak, aby wykorzystywały możliwości systemu Windows. Oznacza to m.in. komunikację z programem za pomocą wyświetlanych na ekranie i obsługiwanych myszką "przycisków", "suwaków" i "przełączników", możliwość uruchamiania pracy programu w trybie wielozadaniowym (*multitask*), możliwość lokowania programu i jego wyników w jednym z kilku okien itd. Programy wykonywane pod kontrolą systemu DOS muszą być przystosowane do współpracy z systemem Windows przez dołączenie do nich specjalnych plików **PIF**.

W związku z rosnącymi zasobami pamięciowymi komputerów klasy IBM PC ważny jest fakt, że MS Windows mogą bez ograniczeń korzystać z rozszerzeń pamięci. W dodatku, w komputerach zbudowanych na bazie procesora **Intel 80386** lub nowszych możliwe jest za pomocą Windows korzystanie z tzw. pamięci wirtualnej. Oznacza to możliwość korzystania z wolnego obszaru na twardym dysku w taki sam sposób, jak z pamięci operacyjnej.

Ważną cechą systemu Windows jest możliwość tzw. dynamicznej wymiany danych. Technika ta, zwana **DDE** (*Dynamic Data Exchange*), polega na tym, że mając uruchomione dwa równocześnie pracujące programy (w oddzielnych oknach) można w trakcie ich pracy przekazywać od jednego do drugiego potrzebne informacje. Na przykład, mając w jednym oknie otwarty arkusz kalkulacyjny (np. *Excel*), a w drugim edytor tekstu (np. *MS Word*), można automatycznie zmieniać dane w redagowanym dokumencie podczas eksperymentów numerycznych dokonywanych za pomocą arkusza.



Użyteczną cechą MS Windows jest możliwość stosowania w nim techniki makroinstrukcji, podobnych w zasadzie do plików **.BAT** w systemie DOS, ale znacznie wygodniejszych. Dzięki tej technice sekwencje rutynowo wykonywanych czynności, mogą zostać zapamiętane i wywoływane potem za pomocą pojedynczego hasła. Służy do tego narzędzie zwane **Recorder**.

Możliwości i udogodnienia systemu Windows można by jeszcze długo wymieniać i komentować. Jest to jednak niemożliwe: założona ograniczona objętość tej książki nie pozwala zamienić jej wyłącznie na opis jednego programu. Jednak – jak już powiedziałem na wstępie – Windows są systemem nastawionym na maksymalną czytelność i łatwą obsługę. Dlatego zamiast studiować podręcznik – lepiej poszukaj wolnego komputera i zapoznaj się z działaniem Windows w praktyce!

3.4.8. System UNIX

3.4.8.1. Podstawowe cechy systemu UNIX

Nie będzie chyba wielkiej przesady w twierdzeniu, że system UNIX[1] stanowić będzie w niedługim czasie największy problem dla wszystkich użytkowników komputerów. Problem ten polegać będzie na tym, że dla nowicjuszy skomplikowane i dość trudne w istocie zasady sterowania pracą komputera za pomocą systemu UNIX stanowić będą barierę znacznie wyższą i trudniejszą do sforsowania, niż obecnie konieczność opanowania współpracy z systemem MS DOS. Natomiast dla doświadczonych komputerowców szokiem będzie konieczność porzucenia wszystkich pracowicie nabytych i starannie pielęgnowanych przyzwyczajeń i nawyków. System UNIX jest bowiem nie tylko systemem trudniejszym od systemu DOS; system UNIX jest przede wszystkim systemem **innym.**

Na czym polega ta różnica?

Najogólniej mówiąc – na fakcie, że cała "filozofia" systemu DOS opiera się na założeniu, że jest to system przeznaczony **dla jednego użytkownika,** pracującego **z jednym komputerem** i wykonującego na nim **jedno zadanie**, natomiast w systemie UNIX z założenia przyjmuje się, że jest **wielu użytkowników** korzystających (być może nawet jednocześnie) z systemu, w którego skład wchodzi **wiele komputerów połączonych w sieć,** a każdy użytkownik może nakazać w tym systemie realizację **wielu zadań**[2].

---

[1] Kilka słów o historii tego – bez wątpienia wyjątkowo udanego – programu. UNIX powstał w **1969** w postaci asemblerowego programu dla minikomputera PDP–7. Wygoda pracy z tym systemem spowodowała spore zainteresowanie, w wyniku czego powstawały jego kolejne wersje, jednak dopiero wersja nr 7, napisana w **1978** roku dla PDP–11/70, zyskała szerokie uznanie. Sukces tej wersji związany był z jednej strony z ogromną popularnością minikomputerów PDP–11, a z drugiej warunkowany faktem, że UNIX 7 napisany był prawie w całości w języku **C**, w związku z czym zapewniona była jego łatwa przenośność na dowolne inne komputery. Warto zresztą dodać, że wystąpiło tu także charakterystyczne sprzężenie zwrotne: sukces UNIX–a wywołał falę zainteresowania językiem C, o którym obszerniej piszemy w rozdziale 6.7.8. Kolejna ważna wersja UNIX–a, oznaczana V/2, powstała w **1984** i stanowiła jednolity standard systemu nadający się dla różnych komputerów.

[2] Od **1973** roku UNIX wykorzystywany jest powszechnie przez środowiska akademickie, co z jednej strony spowodowało wzrost popularności systemu, a z drugiej przyczyniło się do jego wzbogacenia (np. tzw. *Berkeley Enhancements*).

Dlatego w systemie UNIX bardzo rozbudowane są mechanizmy (całkowicie nieobecne w DOS–ie) chroniące zasoby jednego użytkownika przed przypadkową lub zamierzoną ingerencją innych użytkowników. Dalej w systemie UNIX dostępne są – jako coś zupełnie naturalnego – mechanizmy komunikowania się z innymi komputerami i korzystania z zasobów sieci, co jest normalnym i podstawowo dostępnym trybem pracy systemu, a nie super dodatkiem jak w systemie DOS pracującym – przykładowo – w sieci NOVELL. Wreszcie UNIX zapewnia każdemu użytkownikowi możliwość uruchomienia i kontroli wielu zadań na raz, co daje zupełnie nowe możliwości pracy nad dużymi i skomplikowanymi problemami, niemożliwymi w praktyce do rozwiązania pod DOS–em.

Przy tak wielu wymienionych wyżej zaletach UNIX–a można łatwo zgodzić się na dodatkowy wysiłek konieczny do opanowania tego systemu, który **jest kluczem do dużych, profesjonalnych zastosowań informatyki.** W końcu jazda samochodem też jest trudniejsza od chodzenia pieszo, ale warto zdobyć prawo jazdy!

Podane niżej informacje pozwolą na stawianie pierwszych kroków w systemie UNIX, dzięki czemu nie będziesz całkiem bezradny, gdy zetkniesz się z nim jako użytkownik. Dla pełnego opanowania tego systemu konieczne będzie jednak przestudiowanie specjalistycznej literatury i sporo praktyki przy klawiaturze komputera.

3.4.8.2. Rozpoczynanie i kończenie pracy z systemem UNIX

Pierwszą czynnością, jaką musisz wykonać korzystając z systemu UNIX, jest zgłoszenie się do systemu (*login*). System wyświetla zachętę pisząc

login:

w odpowiedzi na co TY powinieneś podać swój identyfikator, uzgodniony uprzednio z administratorem systemu (*superuser*) i wprowadzony do pamięci systemu. Po sprawdzeniu, że użytkownik o podanym identyfikatorze istnieje, system pyta o podanie hasła

Password:

Podczas wpisywania hasła jego elementy nie pokazują się na ekranie, co zapobiega "podglądaniu" haseł przez osoby postronne i przeciwdziała ewentualnym nadużyciom. Hasło musisz sam sobie wymyślić i możesz je w dowolnej chwili zmienić; powinno być ono tajne, ponieważ po podaniu hasła komputer otwiera użytkownikowi dostęp do wszystkich Twoich zasobów (m.in. plików dyskowych), co może być przyczyną sporych kłopotów (inny użytkownik po skutecznym "włamaniu" może skopiować albo nawet zniszczyć cenne pliki).

Jeśli identyfikator i hasło zostało podane poprawnie – komputer podaje podstawowe informacje (np. datę i godzinę ostatniego "wejścia" do systemu, sygnalizuje, jeśli nadeszła przesyłka poczty elektronicznej, a także wyświetla niekiedy specjalny "biuletyn", zaprogramowany przez administratora systemu), a potem uruchamia dla Ciebie program konwersacyjny (*shell*), za pomocą którego będziesz wydawał dalsze polecenia.

Jeśli identyfikator lub nazwa są niepoprawne – komputer wypisuje:

Login incorrect.
Login:

i czeka na poprawne zgłoszenie. Po kilku niepowodzeniach UNIX zawiesza współpracę z "niesfornym" użytkownikiem, co ma zapobiegać łamaniu haseł metodą "prób i błędów".

Możesz zmienić swoje hasło. Zmiana hasła polega na napisaniu polecenia **passwd** i podaniu – najpierw starego hasła, a potem nowego (dwukrotnie, dla kontroli).

Zakończenie pracy w systemie UNIX następuje po podaniu polecenia **logout.**

3.4.8.3. Podstawy użytkowania systemu UNIX

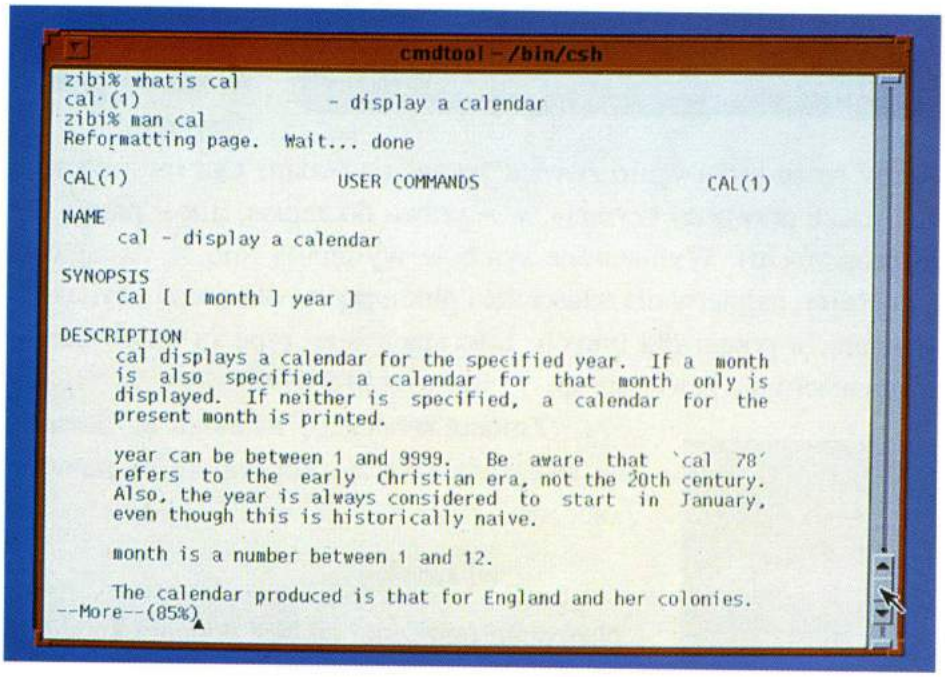
Polecenia w systemie UNIX są zwykle dwu– lub trzyliterowymi skrótami od słów angielskich, nie jest zatem łatwo wszystkie zapamiętać i poprawnie stosować. Na szczęście istnieje kilka sposobów, żeby zapytać i uzyskać **od systemu** informację o tym, jakie polecenie są do dyspozycji i co one robią. Jednym z prostszych sposobów uzyskania takiej **krótkiej** podpowiedzi jest użycie polecenia **whatis** (ang. co to jest), po którym trzeba podać skrót komendy, której znaczenie należy objaśnić. Na przykład polecenie

whatis cal

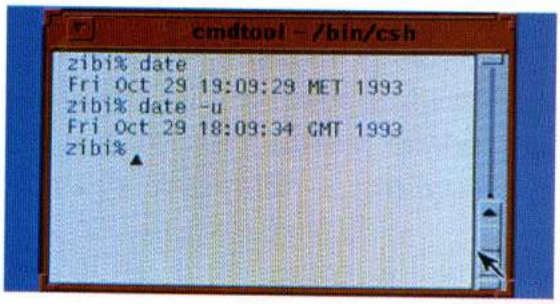
dostarczy informacji, że polecenie **cal** powoduje wyświetlenie kalendarza.

Dostępny jest też w systemie podręcznik, zawierający szczegółowe objaśnienia dla wszystkich elementów systemu. Wystarczy napisać **man** (od *manual* – podręcznik) i podać słowo, nazwę funkcji lub komendę, dla której chce się otrzymać bliższe informacje: co robi, jak ją stosować, jakich parametrów wymaga itd. Przykładowo pisząc

man cal



otrzymuje się **komplet** informacji o sposobie użycia polecenia **cal.** Trzeba jednak być ostrożnym: większość poleceń w systemie UNIX ma bardzo bogate, różnorodne możliwości. W efekcie podręcznikowe objaśnienia, dostarczane przez polecenie **man** obejmują zwykle kilka, a czasem nawet kilkanaście stronic tekstu. Przeczytanie takiego długiego objaśnienia na ekranie komputera wymaga nie lada cierpliwości!



Nie wszystkie polecenia w systemie UNIX są takie skomplikowane. Przykładowo, po napisaniu **date** otrzymuje się na ekranie dokładną informację na temat dnia i godziny. Normalnie podawany jest czas miejscowy, ale pisząc **date –u** możesz otrzymać czas **GMT** (*Greenwich Mean Time*), będący czasem uniwersalnym dla całego globu.

3.4.8.4. Katalogi i pliki w systemie UNIX

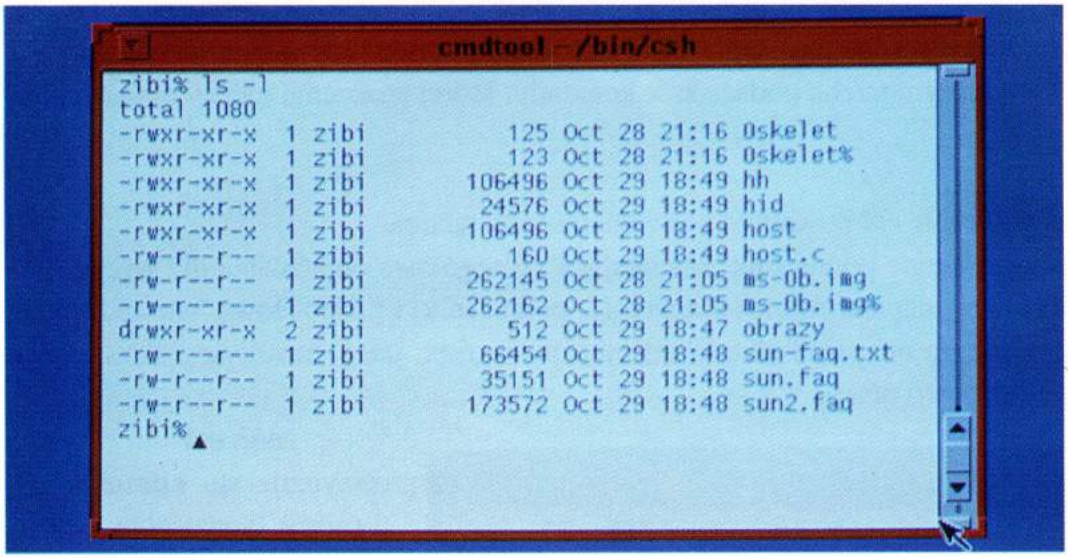
W momencie zgłaszania się do systemu otrzymujesz do dyspozycji pewien katalog, przydzielony Ci przez administratora systemu. Ten katalog nazwany jest Twoim **katalogiem bazowym** (*home directory*). Posługując się komendami systemu (zwłaszcza poleceniem **cd**), możesz przenieść się do dowolnego innego katalogu. W każdej chwili możesz się dowiedzieć, gdzie się aktualnie znajdujesz (to znaczy jaki jest Twój aktualny **roboczy katalog**). W tym celu wystarczy napisać polecenie

pwd

oznaczające *print working directory.*

Możesz sprawdzić, jakie pliki masz do dyspozycji w tym katalogu, po napisaniu polecenia **ls**. Inna forma pozwala na uzyskanie wykazu plików w wybranym katalogu:

ls *katalog*



Dalsze, często używane formy polecenia **ls** mają postać:

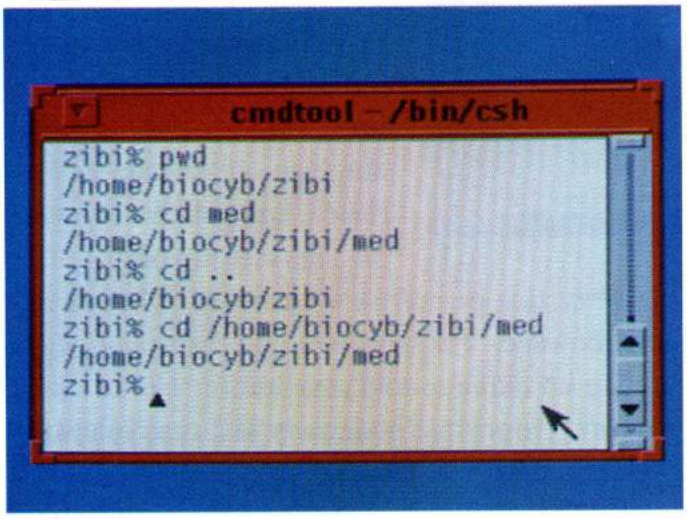
ls –l

informacje o plikach przedstawiane są w formie "długiej",

ls –a

podawane są także nazwy plików ukrytych[1].

Przy oglądaniu "długiej" formy opisu pliku warto zwrócić uwagę na podane tam informacje o prawach korzystania z pliku: **r** – oznacza prawo do czytania, **w** – prawo do zapisu, a **x** – prawo do wykonywania pliku (jeśli jest on programem). Wymienione symbole występują (lub są zastąpione znakiem –) w każdym opisie **trzykrotnie:** najpierw dla właściciela pliku, potem dla grupy użytkowników, której właściciel jest członkiem, a potem dla innych. Informacji tego typu (a także nazwy właściciela, wymienianej po uprawnieniach) nie zawierają spisy plików w **MS DOS.**



Zmiana roboczego katalogu na dowolnie wskazany katalog odbywa się za pomocą polecenia

cd *katalog*

Napisanie polecenia **cd** bez podania katalogu powoduje zawsze powrót do bazowego katalogu użytkownika (*home*).

Istnieje prosty sposób dotarcia do określonego katalogu, polegający na wypisaniu tak zwanej **ścieżki dostępu**[2]. Ścieżka taka ma budowę identyczną z budową w systemie DOS[3].

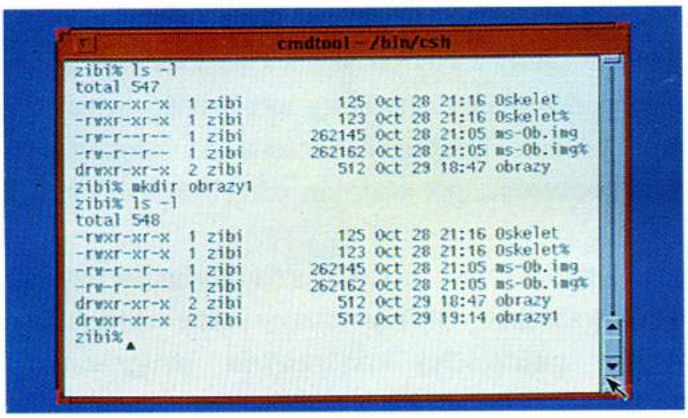
Gdy znajdziesz się już w potrzebnym katalogu i upewnisz się (za pomocą **ls**), że jest w nim program, którego chcesz użyć – wystarczy napisać nazwę programu, by go uruchomić. W ten sposób, uruchamiając programy, wykonuje się w systemie UNIX **wszelkie** zadania.

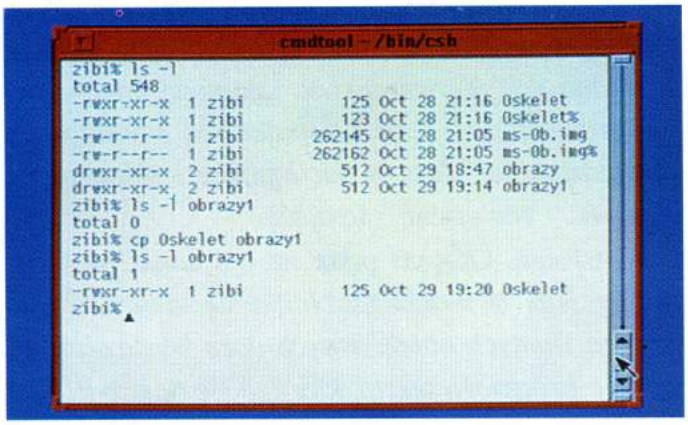
---

[1] Pliki ukryte w systemie UNIX charakteryzują się tym, że pierwszym znakiem ich nazwy jest kropka. Plików ukrytych nie należy kojarzyć z utajnianiem informacji (w końcu bardzo łatwo jest się do nich dostać), ale są to zwykle pliki o charakterze technicznym, tworzone automatycznie przez różne programy w czasie ich eksploatacji. Już po krótkim czasie eksploatacji różnych programów ma się w swoim katalogu sporą liczbę takich plików, których wykaz utrudniałby znalezienie w katalogu własnych plików użytkownika – dlatego normalnie są one ukryte.

[2] Punktem wyjścia dla takiej ścieżki może być korzeń drzewa katalogów (wtedy wpisuje się na początku ścieżki symbol */*), może być aktualny katalog roboczy (wtedy nie wpisuje się nic albo znak . (kropka), będący synonimem pełnej ścieżki od korzenia do aktualnego katalogu roboczego), a może być także katalog bazowy (*home*), którego ścieżka utożsamiana jest ze znakiem ~ (tylda). Często używanym skrótowym symbolem ścieżki jest znak .. znany z DOS–u.

[3] Separatorem, rozdzielającym kolejne składniki ścieżki katalogów w systemie UNIX, jest znak / (*slash*). Na fakt ten powinni zwrócić uwagę użytkownicy przyzwyczajeni do korzystania z systemu MS DOS – ponieważ w DOS–ie do podobnych celów służył znak \ (*backslash*) i łatwo tu o pomyłki.





Jeśli zajdzie potrzeba – możesz sobie zbudować nowy podkatalog w roboczym katalogu za pomocą polecenia

mkdir ***nazwa***

gdzie nazwę możesz sam wybrać[1], jak to pokazałem na fotografii obok. Jeśli chcesz – możesz także skopiować do tego katalogu potrzebne Ci pliki poleceniem

cp ***plik katalog***

Polecenie to nakazuje skopiowanie wskazanego pliku z aktualnego katalogu do katalogu wskazanego w poleceniu. Jego działanie pokazuje fotografia obok. Kopiowanie plików powoduje powstanie ich duplikatów, gdyż plik z miejsca, skąd był kopiowany, nie znika. Jeśli natomiast chcesz przenieść w inne miejsce określony plik, katalog lub cały fragment drzewa katalogów – możesz posłużyć się poleceniem **mv**. Polecenie to ma identyczną budowę z **cp.** Do usuwania plików używaj polecenia **rm.**

Jeśli chcesz zbudować nowy plik – możesz użyć polecenia **cat.** Ma ono wiele możliwych do użycia form i wiele zastosowań. Najprostsze zastosowanie polecenia **cat** polega na wpisywaniu prostych tekstów z klawiatury do wskazanego pliku. Plik w razie potrzeby zostanie utworzony. W tym celu należy:

1. Napisać polecenie **cat** > *plik*
2. Wpisać potrzebny tekst (może zawierać dowolną liczbę linii).
3. Nacisnąć **Ctrl–D**.

Polecenie **cat** pozwala także na szybkie obejrzenie zawartości niewielkich plików. W celu wyświetlenia na ekranie zawartości pewnego pliku należy napisać polecenie **cat** *plik.*

To wystarczy do rozpoczęcia pracy na UNIX–owym komputerze. Informacje na temat dalszych poleceń systemu UNIX znajdziesz łatwo w cytowanej na końcu książki literaturze.

3.5. Edytory tekstowe

3.5.1. Wprowadzenie

Redagowanie tekstów jest zajęciem, od którego najłatwiej i najwygodniej jest rozpocząć naukę korzystania z komputera, ponieważ jest to zadania łatwe i dla każdego zrozumiałe, a równocześnie stanowiące doskonałe ćwiczenie przed wszelkimi dalszymi zadaniami, jakie będziesz kiedykolwiek chciał wykonywać z użyciem tej mądrej maszyny. Na świecie miliony ludzi korzysta z usług komputera do redagowania różnych tekstów – na przykład sprawozdań, raportów, artykułów – a nawet dzieł literackich[2]. Dlaczego nie miałbyś i Ty z tego skorzystać?

---

[1] W systemie UNIX plikom można nadać **dowolną** nazwę – ani liczba znaków, ani ich rodzaj nie są w żaden sposób ograniczone.

[2] W literaturze spotyka się opinie, że pierwszym edytorem, który zyskał powszechne uznanie, był **Electric Pencil** przygotowany dla komputerów **Apple**. Jego sukces jako narzędzia używanego do pisania tekstów (a nie programów) był absolutnym zaskoczeniem dla jego twórców, wśród których na pierwszym miejscu wymienia się **Seymoura I. Rubinsteina**, autora późniejszego sukcesu programu **WordStar.**

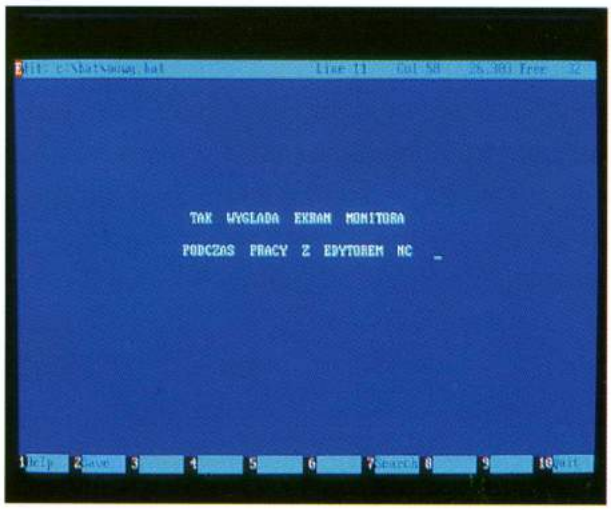
Wszystkie dostępne edytory pozwalają **pisać** dowolne teksty, **poprawiać je, uzupełniać** i dowolnie zmieniać ich formę, czyli **redagować**. Oczywiście każdy z edytorów ma swoje indywidualne zalety i sobie tylko właściwe wady. W dalszej części podrozdziału opiszemy możliwości edytorów tekstowych na przykładzie trzech programów. Zanim poznasz konkretne właściwości konkretnych edytorów spróbujmy jednak chociaż w kilku zdaniach podkreślić, jak ważnym narzędziem informatycznym stał się dziś edytor.

Świadomość wygody, jaką daje podczas pisania i redagowania dokumentów komputer wyposażony w dobry edytor tekstowy, jest już właściwie powszechna. Badania statystyczne wskazują w sposób przekonywujący, że **edytory tekstowe są najczęściej używanymi programami**. Przykładowo, w ubiegłorocznym lipcowym numerze **BYTE** przytoczono wyniki ankiety, w której na pytanie: *jakie jest najpopularniejsze zastosowanie używanego komputera?* aż **93%** (!) ankietowanych odpowiedziało, że przetwarzanie tekstów. Bardzo charakterystyczna jest przy tym cytowana wypowiedź jednego z respondentów: *"Choć w interesach trudno dać pierwszeństwo zakupowi komputera do przetwarzania tekstów, zastosowanie to może usprawiedliwić całą komputerową inwestycję."* **Justin Kaplan**, biograf (a więc nie informatyk!), powiedział, że ***"Jest to pociągające i zabawne, a tak różne od maszynopisania, jak latanie od psiego chodu."*** Nic dodać, nic ująć!

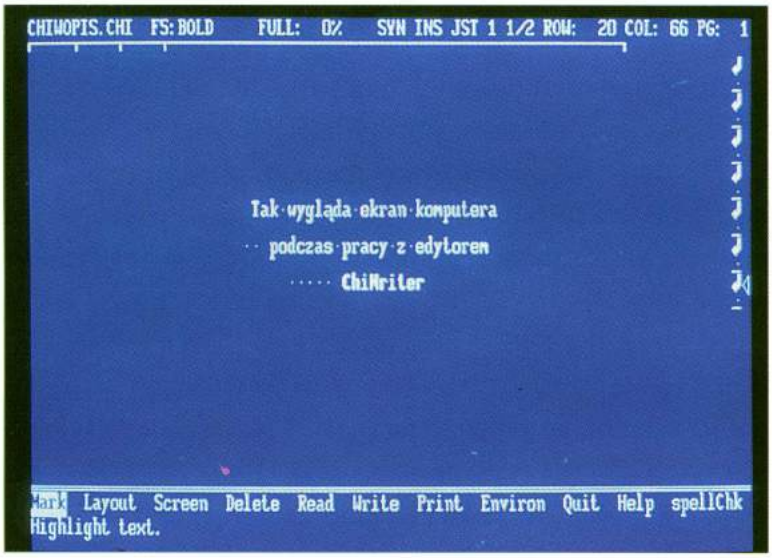
Wybór określonego edytora to decyzja ważna i istotna. Oto, co pisze na ten temat profesor **Stewart Brand:** *"Pisanie jest tak bardzo osobistym zajęciem, że ludzie zaczynają się identyfikować ze swoimi programami redagowania tekstów i nie znoszą żadnych obiektywnych ocen wystawianych tym programom. Ludzie w większości używają pierwszego procesora tekstu, którego się nauczyli. Jest on językiem ich palców, a wszystkie pliki muszą dochować przysięgi wierności wobec jego reguł".* Inny specjalista, **Steven Levy,** ujmuje to jeszcze bardziej dobitnie: *"Porównuję używanie procesorów tekstu do życia z drugą osobą. Zaczynasz z wielkim entuzjazmem i aprobatą, wszystko jest wtedy cudowne. Potem zauważasz, że inne (procesory tekstu) obiecują więcej. Większą różnorodność, przyjaźniejszy styl. Powstaje pytanie: czy warto zrywać związki, w które zainwestowałeś miesiące, dla jakiejś nowej sztuczki z przestawianiem słów, funkcji indeksującej, której użyjesz być może dwa razy, czy możliwości rozdzielania ekranu? Wybór procesora tekstu jest poważną życiową decyzją i nikt nie powinien (w kategoriach czasu, pieniędzy lub emocji) jej lekceważyć".* Tak więc, drogi Czytelniku, stoisz teraz przed decyzją niemal równie ważną jak małżeństwo. Na kolejnych stronach poznawać będziesz edytory, z których każdy jest wart tego, by go polubić i stale używać. Zastanów się, czy na pewno chcesz czytać dalej?

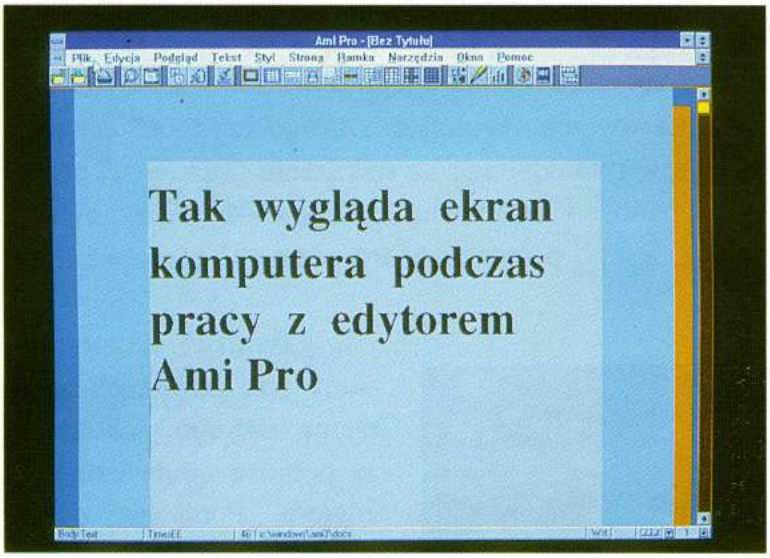
3.5.2. Przykładowe edytory

3.5.2.1. Uwagi wstępne



W książce zdecydowłem się na szczególnie obszerne zaprezentowanie aż **trzech** konkretnych, popularnych edytorów. Przedstawiłem więc z jednej strony najprostszy typ edytora tekstów (tzw. *plain ASCII),* spotykany między innymi w systemie **MS-DOS,** w programach organizujących pracę użytkownika komputera (na przykład omówiony niżej dokładniej edytor występuje w popularnym pakiecie **Norton Commander**), a także w *środowiskach zintegrowanych,* ułatwiających tworzenie programów (np. pakiety *Turbo C, Borland C++, Turbo Pascal* i wiele innych).





Z drugiej strony przedstawiłem bardzo chętnie używany w Polsce edytor **ChiWriter.** Edytor ten ma dość bogate możliwości (m.in. umożliwia łatwe wprowadzanie wzorów matematycznych czy tworzenie prostych schematów i tabel), a przy tym cechują go bardzo skromne wymagania – można go używać praktycznie na każdym, nawet najmniejszym komputerze z dowolną kartą graficzną i z dowolną drukarką. Jego zaletą jest możliwość samodzielnego definiowania znaków przez piszącego.

Na koniec zdecydowałem się na przedstawienie edytora wyższego rzędu, czyli tak zwanego **procesora tekstu,** przy czym ze względu na rosnącą popularność wybrałem **Ami Pro** firmy *Lotus.* Procesor ten jest w miarę prosty w obsłudze (zwłaszcza, że od kilku lat dostępna jest jego całkowicie spolszczona wersja), a równocześnie oferuje bardzo profesjonalne możliwości tworzenia, redagowania i formatowania tekstów, czego przykładem może być szata graficzna tej książki, w całości przygotowanej z wykorzystaniem **Ami Pro.**

3.5.2.2. Porównanie omawianych edytorów

Wymienione typy edytorów "obramowują" niejako problem komputerowego redagowania tekstów, z jednej bowiem strony pokazują najprostsze zasady korzystania z programów powalających łatwo pisać i poprawiać najprostsze teksty przy minimalnym wykorzystaniu możliwości zautomatyzowanego przetwarzania danych (programy **Norton Commander,** oznaczany dalej **NC,** oraz **ChiWriter**), z drugiej natomiast przedstawiają procesor tekstu nie tylko redagujący teksty, ale w pełni automatycznie je przetwarzający i formatujący. Nie bez znaczenia jest też fakt, że omawiane na wstępie dwa proste edytory funkcjonują w systemie **DOS** i oparte są na typowej dla tego systemu "filozofii" osiągania wymaganych celów za pomocą odpowiednio stosowanych kombinacji klawiszy[1], natomiast **Ami Pro** jest typowym programem wykorzystującym możliwości systemu **MS Windows,** zatem jego działanie opiera się na wskazywaniu myszką odpowiednich symboli graficznych.

Z kolei pomiędzy edytorem **NC** a **Chi Writerem** jest ta godna uwagi różnica, że pierwszy z nich działa w trybie *alfanumerycznym,* to znaczy zapamiętuje tylko kody pisanych liter, natomiast drugi pracuje w trybie *graficznym,* traktując cały pisany tekst jako obraz wymagający odtworzenia na ekranie i na drukarce, a każdą pisaną literę jako element tego obrazu. W rezultacie przestudiowawszy przedstawione tu opisy powinieneś sobie radzić z szeroką gamą różnych programów wspomagających pisanie tekstów.

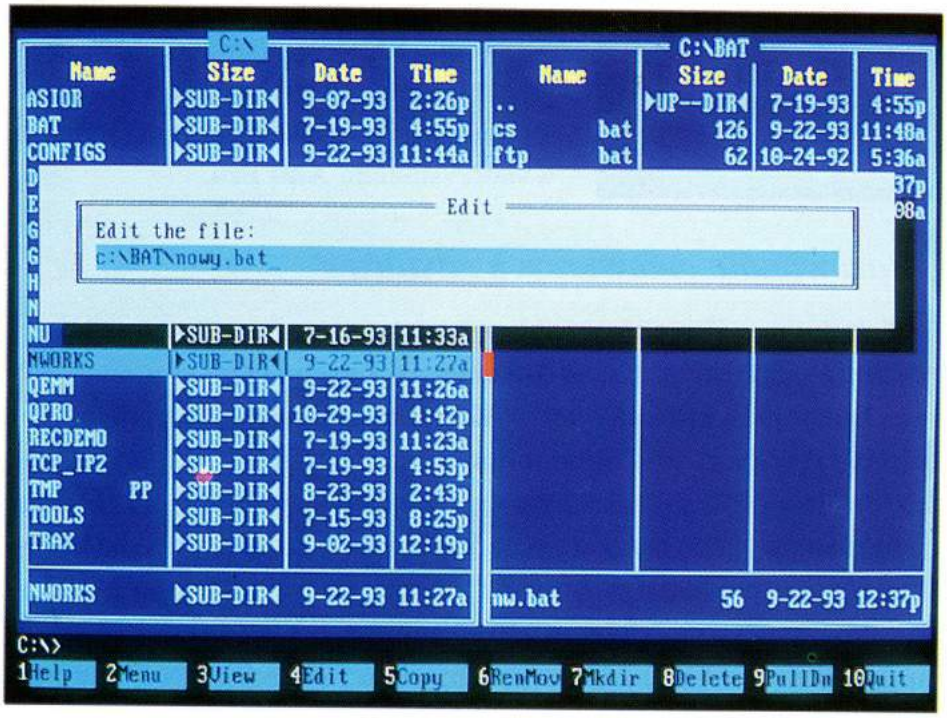
---

[1] W większości edytorów omawianego tu typu stosowane są kombinacje klawiszy, które stanowią powtórzenie zasad przyjętych na początku lat 80. dla historycznego edytora **WordStar**.

3.5.2.3. Edytor programu Norton Commander

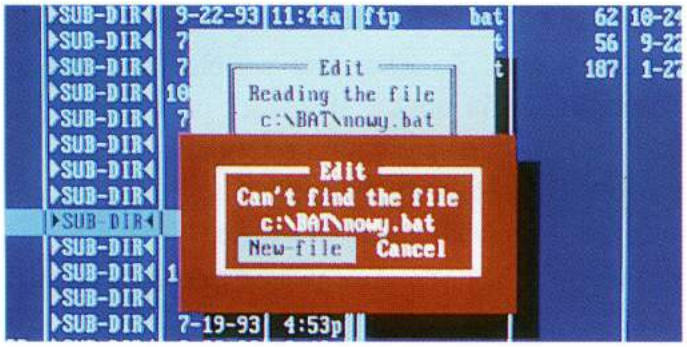
Program **Norton Commander,** opisywany w innym miejscu tej książki jako narzędzie służące do wygodnego sterowania pracą komputera podczas jego normalnej eksploatacji (związanej m.in. z przeglądaniem, kopiowaniem i kasowaniem plików), ma wbudowany bardzo prosty edytor, za pomocą którego można stosunkowo łatwo pisać krótkie i proste teksty (nie zawierające rysunków ani wzorów matematycznych). Edytor ten omawiam ze względu na jego prostą budowę, bardzo typowe zachowanie i dostępność. Równocześnie trzeba podkreślić, że edytor ten wyznacza jak gdyby *limes inferior* wśród omawianych tu programów. Praktycznie wszystkie inne edytory, łącznie z podstawowym edytorem wbudowanym do systemu **MS–DOS,** są od niego bogatsze w możliwości.

Praca nad tekstem przy użyciu każdego edytora może polegać na pisaniu nowego tekstu albo poprawianiu (redagowaniu) starego. Dlatego podstawową czynnością, jaką musi wykonać na początku użytkownik edytora, jest podanie nazwy pliku, w którym znajduje się (lub będzie się znajdował) redagowany dokument. Zwykle operacja ta nazywana jest **otwarciem *(OPEN)*** pliku. W bardziej rozbudowanych edytorach można w ramach tej operacji wskazać zarówno nazwę pliku zawierającego dokument, jak i nazwę katalogu (zwykle podając pełną ścieżkę dostępu), w którym się on znajduje. Edytor **NC** jest w tym zakresie nietypowy, co wynika z jego powiązania z programem, w którym wskazywanie plików jest jedną z podstawowych czynności. Dlatego czynność otwarcia pliku z tekstem podlegającym redagowaniu polega w tym edytorze na wskazaniu (kursorem) nazwy pliku w otwartym panelu programu Norton Commander i naciśnięciu klawisza **F4**. Wskazany plik zostaje przeniesiony w całości do bufora edytora[1], a jego początkowy fragment (23 pierwsze linie) wyświetlony zostaje na ekranie.

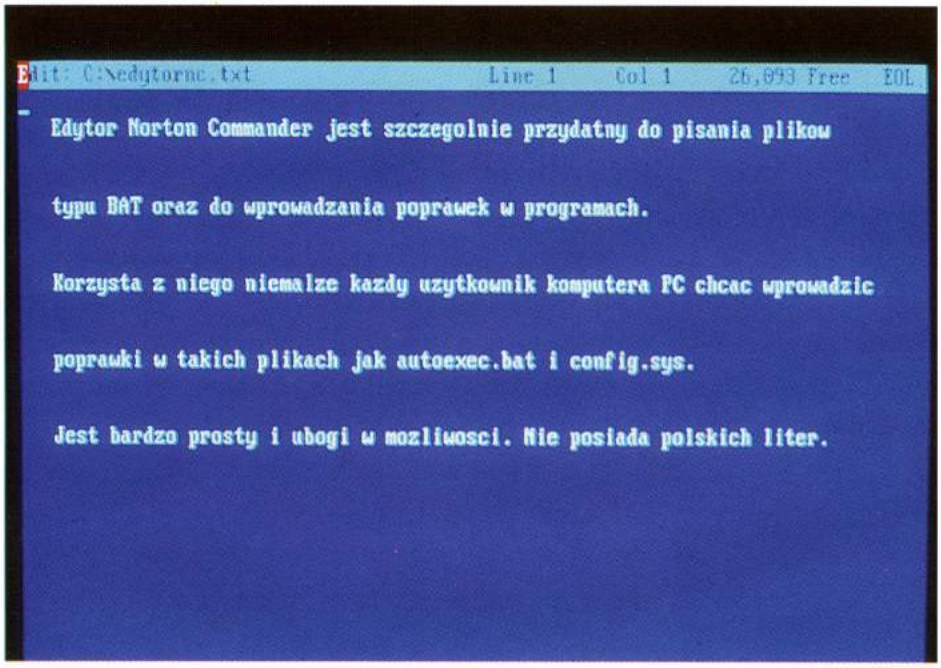


Bardziej skomplikowany jest przypadek tworzenia nowego dokumentu za pomocą omawianego tu edytora. Ponieważ dokument ten ma dopiero powstać, przeto nie ma jeszcze odpowiadającego mu pliku w żadnym z katalogów przeglądanych za pomocą programu Norton Commander, nie można zatem posłużyć się prostym zabiegiem wskazania pliku i naciśnięcia klawisza **F4**. Można natomiast skorzystać z możliwości zażądania edycji dowolnego pliku. Żądanie takie w programie Norton Commander wyraża się naciskając kombinację klawiszy **Shift–F4**. Powoduje to wyświetlenie okienka roboczego z miejscem, w którym powinna być wpisana nazwa edytowanego pliku. Edytor na ogół podaje własną propozycję nazwy pliku, który ma być redagowany. Jest to zwykle plik, który był poprzednio redagowany (niezależnie od tego, na jakim pliku ustawiony jest aktualnie kursor systemu **NC**). Użytkownik może jednak w tym okienku podać własną nazwę pliku i – jak zawsze w takich przypadkach – decyzja użytkownika jest dla komputera wiążąca.

[1] Buforem edytora jest wydzielony fragment pamięci o pojemności około 26,5 KB, w którym wykonywane są wszystkie czynności związane z redagowaniem tekstu. Ograniczony rozmiar bufora powoduje, że niektóre pliki tekstowe (za duże) nie mogą być redagowane z pomocą tego programu!



Jeśli plik o podanej nazwie jest dostępny w aktualnym katalogu – zostanie on otwarty i w opisany wyżej sposób przekazany do redagowania. Jeśli jednak pliku takiego nie ma – program wyświetla ostrzeżenie (pokazne na fotografii obok), które znaczy: *Nie mogę znaleźć wskazanego pliku* i daje użytkownikowi do wyboru dwie możliwości. Pierwsza oznacza, że użytkownik chce utworzyć nowy plik o wskazanej nazwie (na ogół o to właśnie nam chodzi i wtedy wystarczy nacisnąć klawisz **Enter**), druga daje Ci możliwość skasowania całej akcji po wybraniu opcji **Cancel**, którą trzeba wskazać (kursorem lub myszką) i zatwierdzić klawiszem **Enter**[1].



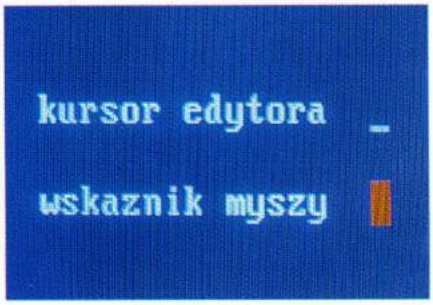
Po otwarciu pliku z tekstem można **w każdym edytorze** pisać nowe informacje lub redagować (poprawiać) stare, używając w najbardziej naturalny sposób klawiszy alfanumerycznych (do pisania nowego tekstu), klawiszy kursora wraz z klawiszami **Home, End, PgUp, PgDn** (do poruszania się wewnątrz tekstu i do znajdowania miejsca, gdzie chcemy coś dopisać lub usunąć), a także klawiszy **Backspace** oraz **Del** do usuwania zbędnych lub błędnie napisanych liter, słów, a nawet całych zdań. Warto przy tym zwrócić uwagę, że "wędrówka" po ekranie z użyciem klawiszy kursora ograniczona jest tylko do tej części widocznej płaszczyzny, którą zajmuje wpisany już do tej pory tekst; nie da się swobodnie "wkroczyć" kursorem w pustą, nie zapisaną przestrzeń i zacząć pisanie od jakiegoś dowolnego odległego punktu[2]. Chcąc wprowadzić do pisanego tekstu puste miejsca (na przykład w celu uczynienia tekstu bardziej czytelnym), trzeba "wbić" z klawiatury odpowiednią liczbę pustych miejsc (klawiszem odstępu czyli *spacji*), a potrzebne całkiem puste linie tekstu uzyskuje się wielokrotnym naciskaniem klawisza **Enter.**

Jeśli do komputera dołączona jest myszka, wówczas można bardzo wygodnie i łatwo ulokować kursor w dowolnym miejscu redagowanego tekstu. Wystarczy w tym celu naprowadzić wskaźnik myszki na dowolny znak w dowolnej linii i po naciśnięciu klawisza myszki (tzw. *click)* uzyskuje się ustawienie kursora edytora we wskazanym punkcie. Warto przy tym zwrócić uwagę, że wskaźnik

---

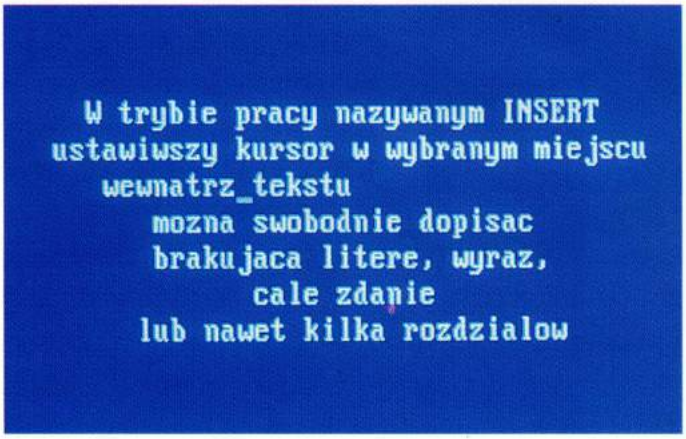
[1] Jeszcze raz podkreślam, że w tym jednym przypadku opisany wyżej sposób działania edytora programu NC jest nietypowy; większość edytorów podobnej klasy pozwala wprawdzie na podanie także nazwy redagowanego dokumentu już w momencie wywołania odpowiedniego programu. Dokument taki zostaje wówczas automatycznie otwarty i załadowany do edytora w chwili jego uruchomienia. Nie jest to jednak jedyna możliwość, gdyż w edytorach tych jest zawsze także możliwe wskazanie nazwy redagowanego dokumentu dopiero po otwarciu edytora – najczęściej po wybraniu komendy **Open** (dla istniejącego pliku) lub **New** (dla nowo tworzonego tekstu) z asortymentu oferowanych funkcji edytora, wymienianych w takim przypadku zwykle w menu **File**.

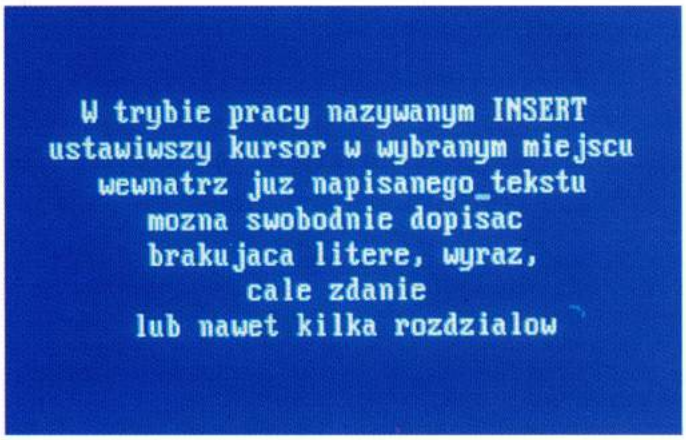
[2] Początkujący użytkownik miewa czasem kłopot z wyobrażeniem sobie, jak daleko na ekranie sięga obszar już zapisany, dlatego niektóre edytory (na przykład **Window Write**) stosują dodatkowy (poza ruchomym kursorem) marker końca tekstu. Jest to bardzo przydatne, szczególnie w przypadku "zapisania" pewnej partii tekstu pustymi liniami lub niewidocznymi na ekranie znakami odstępu. Omawiany tu edytor NC jest pozbawiony tego udogodnienia, zatem naciskając klawisze sterujące ruchem kursora musimy liczyć się czasem z "nieposłuszeństwem" komputera: mimo naciskania odpowiedniego klawisza kursora sam znacznik na ekranie nie porusza się, albo nieoczekiwanie przechodzi do nowej linii. Trzeba się z tym nieco oswoić.



myszki ma inny kształt (dużego prostokąta) niż kursor edytora (migający znak podkreślenia). To także jest wspólna cecha różnych edytorów (patrz fotofrafia obok). "Jeżdżąc" myszką musisz sie liczyć z tym, że osiągalne są wyłącznie miejsca wewnątrz już wcześniej zapisanego tekstu. Ponieważ wskaźnik myszki nie podlega temu ograniczeniu – możliwe jest wskazanie z jej pomocą takiego położenia kursora, które jest nielegalne z punktu widzenia zasad pracy edytora. Edytor zachowuje się wtedy inteligentnie i wytwornie: ustawia swój kursor w miejscu, które jest położone najbliżej wskaźnika myszy – ale jeszcze wewnątrz tekstu.

Edytor **NC**, podobnie jak wiele innych prostych edytorów, pozwala pisać wiersze tekstu o wiele dłuższe, niż pełna szerokość ekranu (80 znaków). Ten brak ograniczenia długości linii do rozmiarów ekranu jest w zasadzie zaletą, gdyż pozwala – przykładowo – tworzyć bardzo duże tabele, które potem przeniesione na drukarkę pozwolą wykorzystać pełną szerokość umieszczonego w niej papieru. Ma to jednak także i złe strony. Wynikają one z faktu, że po przekroczeniu podczas pisania liczby znaków mieszczącej się w jednym wierszu, następuje "przewijanie ekranu" – fragmenty tekstu odpowiadające początkowi wszystkich wierszy widocznych na ekranie wysuwają się w lewo i przestają być widoczne, natomiast w miarę pisania wyłaniają się kolejne fragmenty obszaru położonego na prawo od granicy, jaką stanowi ramka ekranu. Jest to dość niewygodne (wygląda, jakby się obserwowało duży arkusz papieru poprzez otwór o rozmiarach ekranu, suwając go z miejsca na miejsce), a ponadto dość powolne (na słabszych komputerach pojawia się przy tym zauważalne i bardzo denerwujące opóźnienie). Dlatego na ogół unika się przewijania ekranu, wymuszając (klawiszem **Enter**) przejście do nowej linii zanim tekst dojdzie do prawego marginesu[1].





Większość osób redagujących teksty z użyciem komputera preferuje tryb pracy nazywany *insert* (wstawianie), polegający na tym, że wprowadzając nowy tekst nie niszczy się tekstu wcześniej napisanego. W trybie tym ustawiwszy kursor w dowolnym miejscu wewnątrz już napisanego tekstu – możesz swobodnie dopisać brakującą literę, wyraz, całe zdanie lub nawet kilka rozdziałów. Istniejący wcześniej tekst rozsuwa się przy tym tworząc wolne miejsce dla nowego tekstu, bez utraty ani jednej litery. Obejrzyj dokładnie sąsiednie fotografie, a zobaczysz, na czym to polega. Edytor **NC** zawsze działa w taki właśnie sposób. Pracując z tym edytorem i ucząc się na jego przykładzie używania komputera do redagowania tekstów, doskonale poznasz i polubisz ten tryb pracy. Warto jednak wiedzieć, że inne edytory umożliwiają często piszącemu zrezygnowanie z wygody stosowania trybu *insert* i zastąpienie go trybem *replace* (zastępowanie), w którym pisany nowy tekst

[1] W doskonalszych edytorach piszący tekst jest uwolniony od konieczności śledzenia położenia tekstu w stosunku do prawego marginesu za pomocą mechanizmu **word wrap,** który dokładniej omówiony zostanie podczas prezentacji **ChiWritera**. Prosty edytor **NC** nie ma (niestety) mechanizmu **word wrap**.

zastępuje stary, wymazując go systematycznie znak po znaku. Tryb zastępowania rzadko jednak bywa naprawdę przydatny, a bywa kłopotliwy.

Podczas pisania i poprawiania tekstu warto mieć na uwadze następujące dodatkowe informacje, dotyczące opisywanego tu edytora **NC**, ale wysoce przydatne i zwykle identycznie działające także i w innych edytorach. Otóż, działanie klawiszy pozwalających na przemieszczanie się w obrębie tekstu można zmodyfikować naciskając je wraz z klawiszem **Ctrl.** I tak kombinacja **Ctrl–Home** powoduje przejście na sam początek redagowanego tekstu (sam klawisz **Home** powoduje przeniesienie kursora na początek redagowanej linii, a klawisz **PgUp** pozwala się przenieść na początek aktualnie redagowanej stronicy (czyli do górnego wiersza widocznego aktualnie na ekranie). Analogicznie kombinacja klawiszy **Ctrl–End** pozwala "jednym skokiem" dotrzeć na koniec redagowanego dokumentu, podczas gdy sam klawisz **End** kieruje nas na koniec redagowanego wiersza, a **PgDn** na koniec stronicy. Klawisz **Ctrl** modyfikuje też działanie zwykłych klawiszy kursora. Otóż klawisze te naciskane wraz z klawiszem **Ctrl** przemieszczają kursor **o całe słowa w lewo lub w prawo.** Jest to bardzo wygodne i przyspiesza pracę, gdy trzeba szybko "trafić" na początek pewnego słowa, które chcemy poprawić. To bardzo sprytne (słowa są różnej długości!) udogodnienie dostępne jest praktycznie we wszystkich edytorach, nawet tych najtańszych.

Interesujące możliwości udostępnia także zwykle kombinowanie klawisza **Ctrl** z klawiszami zawierającymi różne litery. Ta konwencja, wywodząc się z najwcześniejszych doświadczeń z edytorem **WordStar**[1], bywa często w popularnych edytorach "tajna", tzn. oficjalne opisy odpowiednich programów nie wspominają o tym ani słowem, ale biegłe palce doświadczonego programisty same układają się w znajome "akordy" – i, co ciekawe, wywołują oczekiwany skutek! Nie jest to właściwe miejsce, by przytaczać te wszystkie "sztuczki", ale w omawianym tu edytorze też funkcjonują tego rodzaju "chwyty". Na przykład naciśnięcie kombinacji klawiszy **Ctrl–T** powoduje skasowanie całego słowa (dokładniej – od aktualnej pozycji kursora do końca słowa, ale kursor z reguły ustawia się na początku słowa). Z kolei kombinacja klawiszy **Ctrl–Y** powoduje wykasowanie całej linii tekstu; trzeba jej używać ostrożnie, gdyż kilkakrotne nieuważne naciśnięcie **Ctrl–Y** może spowodować skasowanie (nieodwołalne w tym prymitywnym edytorze) sporej części pracowicie pisanego tekstu. Klawisze **Ctrl–T** oraz **Ctrl–Y** mają podobne działanie w większości szeroko używanych edytorów (na przykład w programie **Edit** wchodzącym w skład systemu **MS–DOS**), chociaż nie zawsze jest to opisane w instrukcji odpowiedniego programu. W edytorze **NC** są jednak dostępne także inne "akordy", których działanie rzadziej spotyka się w innych edytorach. Na przykład kombinacja **Ctrl–K** powoduje wykasowanie fragmentu tekstu od aktualnego miejsca ulokowania kursora aż do końca aktualnej linii, a kombinacja **Ctrl–W** powoduje skasowanie początkowej części słowa (do miejsca położenia kursora)[2].

Umiejętność pisania tekstu, kasowania błędnie napisanych fragmentów oraz poruszania się (za pomocą kursora) po całym tekście – to podstawowe umiejętności, potrzebne przy pracy z każdym edytorem, a wyjątkowo łatwe do nauczenia się przy stosowaniu prościutkiego edytora **NC**. Edytor ten może również posłużyć do tego, by nauczyć się (w uproszczonej postaci) niektórych dalszych właściwości, charakterystycznych dla większości edytorów.

Przyjrzyj się strukturze ekranu edytora podczas pisania tekstu (porównaj fotografię na następnej stronicy).

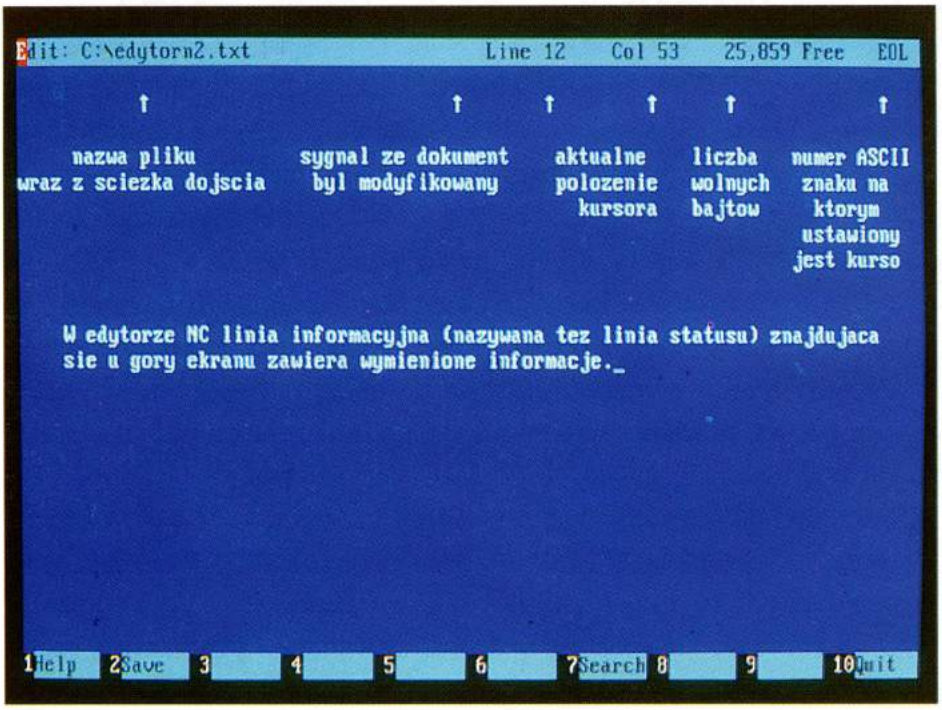
Ekran ten zawiera omawianą do tej pory część centralną, którą porównać można do arkusza papieru, na którym piszemy nasze teksty. Jednak u góry i u dołu ekranu widoczne są elementy pomocnicze, w tym prostym edytorze zredukowane do dwóch wyróżnionych linii, natomiast w

---

[1] Za twórcę programu **WordStar** uważa się **Seymoura I. Rubinsteina**, chociaż głównym programistą firmy *MicroPro* był **Bob Barnaby.**

[2] Przyzwyczajenie do używania kombinacji **Ctrl–K** może przeszkadzać przy "przesiadce" na inne edytory, ponieważ w wielu z nich sekwencja **Ctrl–K** zapoczątkowuje pewne bardziej złożone, niedostępne w edytorze **NC**, operacje na tzw. blokach.

doskonalszych edytorach mające często znacznie bardziej rozbudowaną formę. Jedna z tych linii pełni rolę informacyjną, a druga pomaga w sterowaniu pracą edytora. W edytorze **NC** linia informacyjna (nazywana też **linią statusu**) znajduje się u góry, a linia sterująca (nazywana też **linią menu**) u dołu ekranu[1].



Edytor **NC** w linii statusu podaje (kolejno, od lewej do prawej):

- nazwę pliku z redagowanym dokumentem (wraz z pełną ścieżką dojścia);
- sygnał, że dokument był już modyfikowany (obecność gwiazdki * w tym miejscu sygnalizuje dopiski w dokumencie i sugeruje celowość jego zapisu na dysk);
- sygnał " informujący, że następny wpisywany do dokumentu znak będzie traktowany dosłownie i musi być przez edytor wiernie wpisany do dokumentu, a nie ma być traktowany jako sekwencja sterująca edytorem[2] (uwaga: z opcji tej korzysta się bardzo rzadko i dlatego zwykle na ekranie brak znaku ");
- informacja o aktualnym położeniu kursora (w postaci napisu **Line** ***y*** **Col** ***x***, gdzie ***y*** oraz ***x*** są odpowiednio numerem wiersza i numerem kolumny; numeracja zaczyna się w lewym górnym rogu ekranu);
- informacja o liczbie wolnych bajtów w buforze tekstu (w postaci ***xxxx*** **Free**, gdzie ***xxxx*** jest stosowną liczbą);
- informacja o numerze ASCII znaku, na którym ustawiony jest w danej chwili kursor.

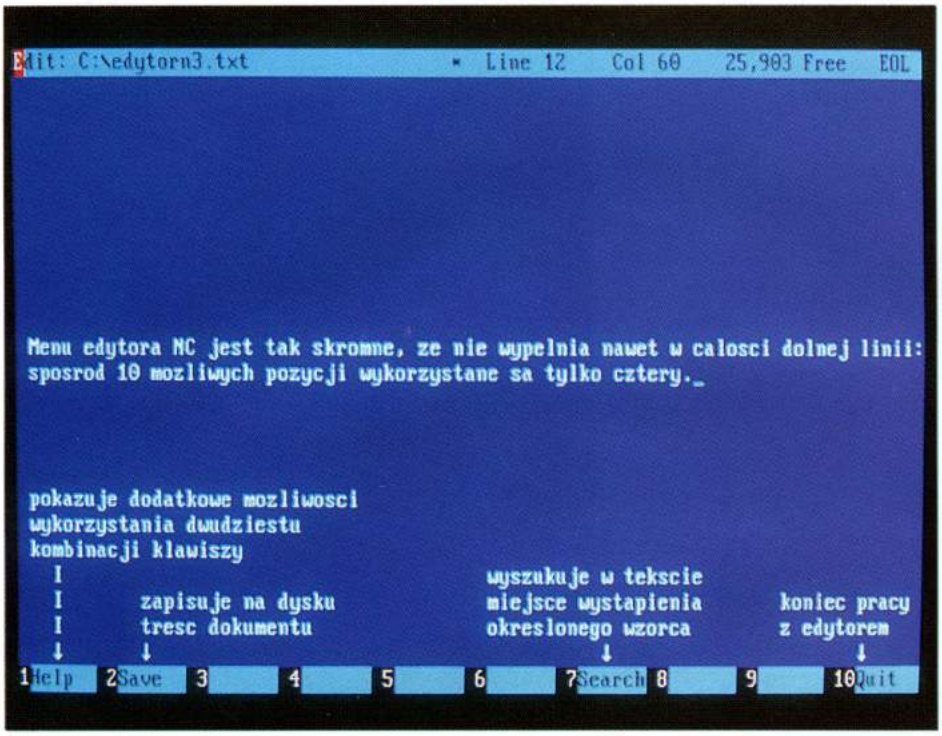
Szczególnie ostatnia z wymienionych informacji może być użyteczna dostarczając początkującym informatykom informacji o kodach wybranych znaków, co czasem bywa potrzebne przy analizie zawartości pamięci komputera albo przy pisaniu programów odwołujących się do wewnętrznej reprezentacji danych w komputerze.

Warto zawsze podczas korzystania z edytora obserwować opisane wskaźniki, w szczególności wypada śledzić zmniejszanie się liczby wolnych bajtów z każdym kolejnym napisanym znakiem, ponieważ brak takiej kontroli może spowodować, że niespodziewanie podczas pisania ważnego tekstu

---

[1] Nie należy do tego rozmieszczenia przywiązywać wagi, gdyż w różnych programach spotyka się często taki właśnie układ, ale widzi się także niekiedy układ odwrotny (menu u góry, a linia statusu u dołu) albo rozmaite inne kombinacje (na przykład wszystko u góry, wszystko u dołu, "okienka" w kątach ekranu itd.).

[2] Znaki specjalne, na przykład tworzące elementy rysunków (najczęściej różnych ramek), a także znaki formatujące wydruk, takie jak na przykład przejście do nowej stronicy na drukarce, pisane są na klawiaturze metodą równoczesnego naciskania kilku klawiszy (na przykład znak przejścia do nowej stronicy na drukarce uzyskuje się kombinacją klawiszy **Ctrl–L**). Jednak takie kombinowane układy klawiszy edytor normalnie "bierze do siebie", to znaczy interpretuje jako polecenia – na przykład skasowania całej linii tekstu. Jeśli jakaś kombinacja klawiszy nie daje się w ten sposób zinterpretować – edytor nic nie robi, tylko ostrzega cichym piśnięciem, że nie rozumie wydanego rozkazu. Żeby tego uniknąć trzeba przed naciśnięciem klawiszy generujących taki znak specjalny "zapowiedzieć" jego wprowadzenie do tekstu, używając przeznaczonej tylko do tego celu kombinacji klawiszy **CTRL–Q**. Właśnie po naciśnięciu klawiszy **CTRL–Q** w linii statusu pojawia się ostrzegawczy znak **"**. Jeśli teraz naciśnie się klawisze znaku specjalnego – w tekście pojawi się specjalny znak (na przykład dla kombinacji **CTRL–L** jest to kółko z krzyżykiem), a podczas drukowania dokumentu w tym miejscu wymuszone będzie specjalne działanie (na przykład przejście do nowej stronicy).



edytor odmówi posłuszeństwa z powodu wyczerpania się miejsca w buforze (który w omawianym edytorze jest nieco za mały do wielu zastosowań).

Dolna linia edytora podaje – bardzo skromne niestety – **menu** ponadpodstawowych możliwości tego edytora. Większe edytory miewają setki takich możliwości wybieralnych z menu, dlatego ich system menu jest z reguły **rozwijalny** – wskazanie jednej z możliwości wyświetlanych w linii menu powoduje otwarcie okienka, w którym program proponuje różne sposoby wykonania polecenia, jakie wydałeś. Czasem "dogadanie się" z bardziej rozbudowanym edytorem wymaga aktywacji całej kaskady okienek!

Menu edytora **NC** jest tak skromne, że nie wypełnia nawet w całości dolnej linii: spośród 10 możliwych pozycji wykorzystane są tylko cztery:

- **Help** – wskazanie tej możliwości oznacza żądanie podania "podpowiedzi";
- **Save** – nakazuje zapisać na dysku treść redagowanego dokumentu;
- **Search** – pozwala wyszukiwać w tekście miejsce wystąpienia określonego wzorca;
- **Quit** – nakazuje koniec pracy z edytorem i przejście do głównego panelu programu.

Opcje te możesz wybierać wskazując na nie myszką i naciskając jej klawisz, albo (jeśli nie używasz myszki) – naciskając klawisze funkcyjne od **F1** do **F10** o numerach, które są podane obok odpowiednich okienek z nazwami poleceń.

Przypatrzmy się teraz nieco bliżej czynnościom, jakie wykonuje edytor po wybraniu odpowiedniej pozycji z menu. Są one typowe w tym sensie, że wszystkie znane edytory **przynajmniej** te pozycje posiadają, często zresztą w znacznie bogatszej postaci.

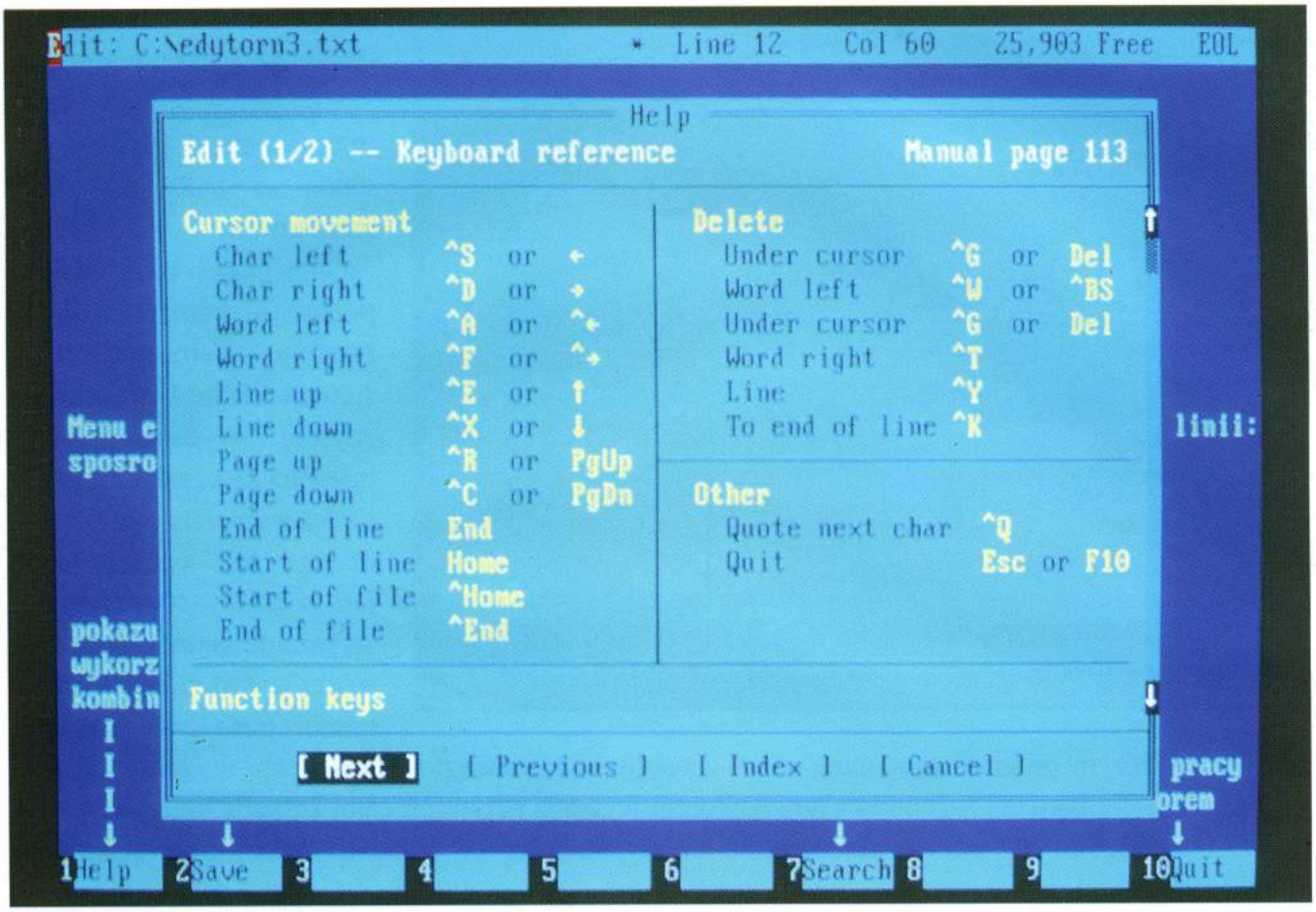
Klawisz **F1** przywołuje ekran podpowiedzi, informujący jak należy się posługiwać edytorem, jakie jest znaczenie poszczególnych grup klawiszy itp. W prostym edytorze **NC** jeden ekran może pomieścić wszystkie te "mądrości" (pokazałem to na następnej stronicy), natomiast w innych edytorach potrzeba czasem kilkuset stronic instrukcji[1].

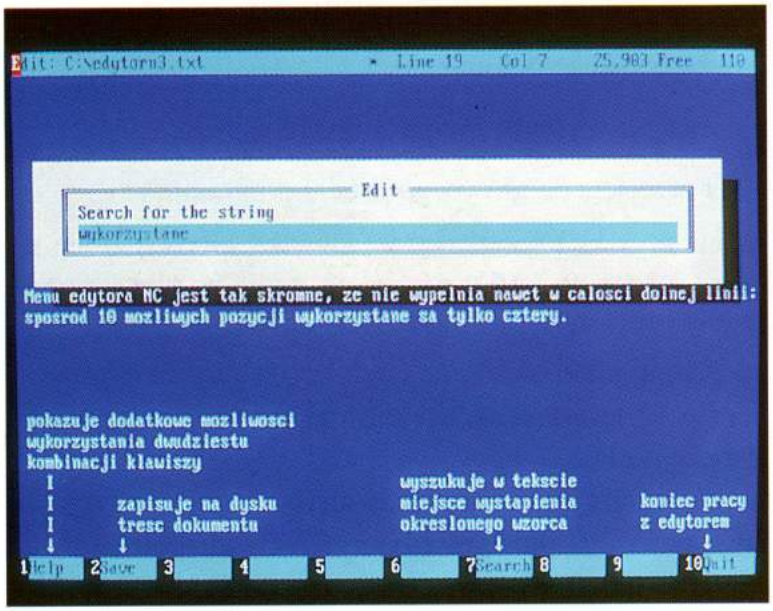
Klawisz **F2** przeznaczony jest do zapisywania przetwarzanego pliku na dysk, co w prostym edytorze **NC** w pełni wystarcza[2].

---

[1] W tych bardziej skomplikowanych menu sporą trudność sprawiać może odnalezienie wśród bardzo wielu innych – tej właśnie jedynej wiadomość, która jest w danej chwili potrzebna użytkownikowi. Jest to trudna sztuka i dlatego w bardziej rozbudowanych edytorach system podpowiedzi (*help*) obok zbioru samych wiadomości na temat możliwych działań programu i sposobów ich wymuszania – występuje także obszerny moduł sterujący procesem wyszukiwania potrzebnych informacji, ze skorowidzem, spisem treści, możliwością uzyskiwania pomocy kontekstowej itd.

[2] W bardziej rozbudowanych programach podobna opcja (nazywana zwykle **Save** i wiązana z pozycją menu o nazwie **File**) miewa liczne odmiany i warianty. Przykładowo można podczas rejestrowania tekstu na dysku zmienić nazwę pliku (opcja **Save as...**), wybrać dowolnie katalog, do którego plik będzie zapisany (opcja **Directory**), zapewnić automatyczny okresowy "zrzut" redagowanego tekstu na dysk w celu ustrzeżenia się przed ewentualną utratą tekstu w przypadku awarii zasilania komputera (opcja **Backup**), dostosować postać zapisu na dysku do różnych potrzeb (zapis samego tylko tekstu ASCII, zapis wraz z informacjami formatującymi, zapis w standardzie innych edytorów w celu ułatwienia wymiany plików itp.).



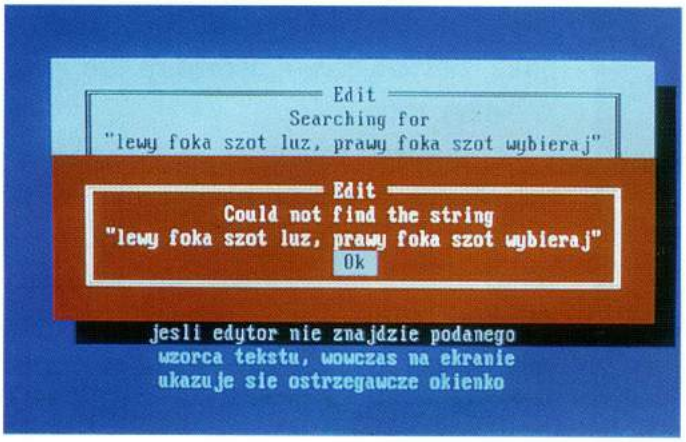


Klawisz **F7** nakazuje wyszukiwanie określonego wyrazu czy grupy wyrazów wewnątrz całego napisanego tekstu. Po naciśnięciu klawisza **F7** ukazuje się w centrum ekranu okienko, w którym można wpisać wyszukiwany wzorzec. Po wpisaniu szukanego wyrazu i naciśnięciu klawisza **Enter** edytor rozpoczyna poszukiwania i zatrzymuje się na tej linii, w której znaleziono szukany wzorzec tekstu, dzięki czemu można natychmiast wykonać czynność, którą mieliśmy na myśli nakazując poszukiwania – wykasować znaleziony wyraz, zmodyfikować go, dopisać itp[1].

Jeśli edytor nie znajdzie podanego wzorca tekstu, wówczas w centrum ekranu ukazuje się ostrzegawcze okienko (pokazane na fotografii na następnej stronicy). Komunikat ten trzeba potwierdzić (na ekranie widać podświetlony prostokąt z napisem **Ok,** na którym trzeba "tupnąć" myszką, albo wystarczy nacisnąć klawisz **Enter**). Po takim niepowodzeniu można zrezygnować z szukania albo można ponownie nacisnąć klawisz **F7** i tak zmodyfikować wzorzec wyszukiwanego tekstu, by jego znalezienie w przetwarzanym dokumencie było bardziej prawdopodobne.

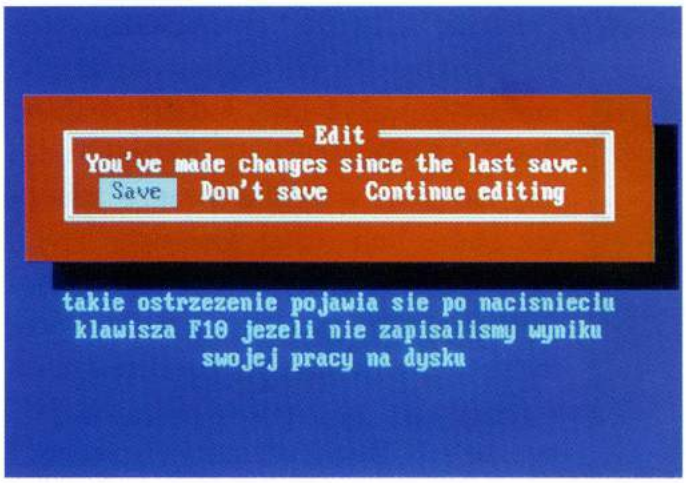
---

[1] Niestety, kursor edytora NC jest mało widoczny (cienka migająca kreska), czasem trzeba więc przez dłuższą chwilę wpatrywać się w ekran, żeby zauważyć, gdzie się znajduje wyszukany przez program tekst. Lepsze edytory potrafią tak "podświetlić" znaleziony wzorzec, że jest zauważalny od pierwszego razu. Sprawa nie jest – wbrew pozorom – tak całkiem banalna, gdyż proces szukania jest bardzo szybki i niemożliwy do wzrokowego śledzenia, a wyszukany wzorzec jest zwykle na innej stronicy niż ta,od której zaczynaliśmy wyszukiwanie, dlatego wykrycie w tym nowym kawałku tekstu, który nieoczekiwanie zjawia się na ekranie, kursora, a wraz z nim wyszukanego tekstu – wymaga pewnego wysiłku.



Ponowne naciśnięcie klawisza **F7** może być celowe również po zanotowaniu sukcesu (po znalezieniu wyszukiwanego wzorca). Edytor może wówczas ponownie szukać następnego wystąpienia tego samego wzorca (wystarczy wtedy samo potwierdzenie klawiszem **Enter**) albo można podać kolejny wzorzec tekstu, który ma być wyszukiwany[1].

Ostatnia z opcji wymienionych w linii menu programu, polecenie **Quit,** oznacza po prostu zakończenie pracy programu. Wskazanie tej pozycji menu lub naciśnięcie związanego z nią klawisza **F10** powoduje wyjście z procesu i powrót do paneli obrazujących zawartość wybranych katalogów na dysku. Ponieważ jednak **opuszczenie edytora oznacza zawsze utratę tekstu zawartego w jego buforze** – edytor ostrzega o możliwości utraty ważnych i potrzebnych danych. Dlatego jeśli dokonaliśmy jakichś poprawek w redagowanym tekście i nie zapisaliśmy wyniku swojej pracy na dysku (opcja **Save**) – wówczas po naciśnięciu klawisza **F10** pojawia się ostrzeżenie (pokazane niżej). Edytor daje przy tym trzy możliwości: **Save – NC** zapisze zmodyfikowany plik z tekstem na dysku w chwili kończenia pracy, **Don't save – NC** przerwie pracę i zniszczy tekst (na dysku pozostanie poprzednia wersja dokumentu), albo **Continue editing** – polecenie zakończenia pracy będzie zignorowane i proces edycji będzie kontynuowany.



Zwykle w opisanej sytuacji użytkownicy edytora wybierają opcję **Save** i dlatego jest ona przez edytor wstępnie podświetlona. Wystarczy nacisnąć **Enter,** żeby wybrać tę ewentualność. Pozostałe opcje muszą być wskazane myszką lub kursorem, aby mogły być wybrane.

Podsumowując ten rozdział, należy jeszcze raz podkreślić, że edytor programu **NC** jest bardzo prosty i ubogi w możliwości. W poważniejszych zastosowaniach jest to istotna wada i dlatego docelowo koniecznie trzeba poznać także inne edytory. Jednak do prostych zastosowań opisany edytor jest bardzo wygodny, a jego prostota ogromnie ułatwia naukę, dlatego warto zacząć od pełnego opanowania wszystkich jego możliwości zanim sięgnie się po coś większego – na przykład opisany niżej edytor ChiWriter albo procesor Ami Pro.

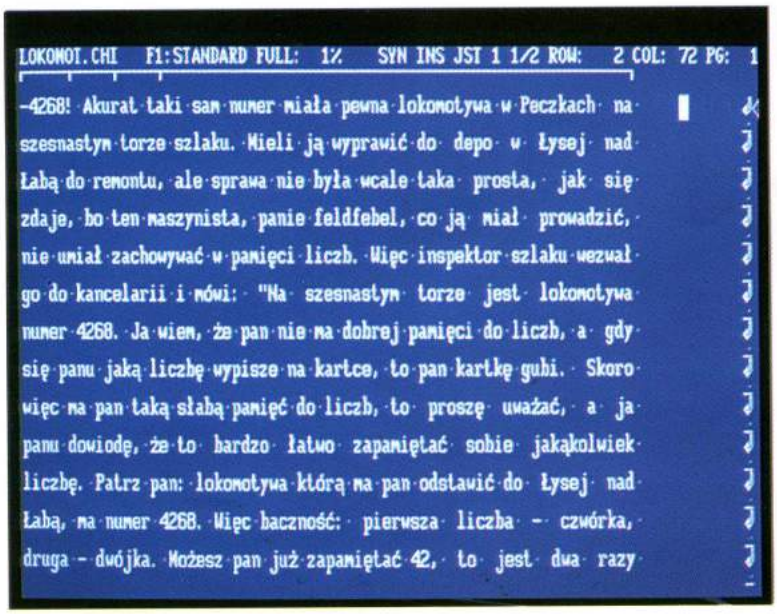
3.5.2.4. Edytor ChiWriter

Rozpoczęcie pracy z edytorem **ChiWriter** polega na wejściu do jego katalogu (najczęściej mającego nazwę **CHIW**) i wydaniu polecenia

CW *nazwa_dokumentu*

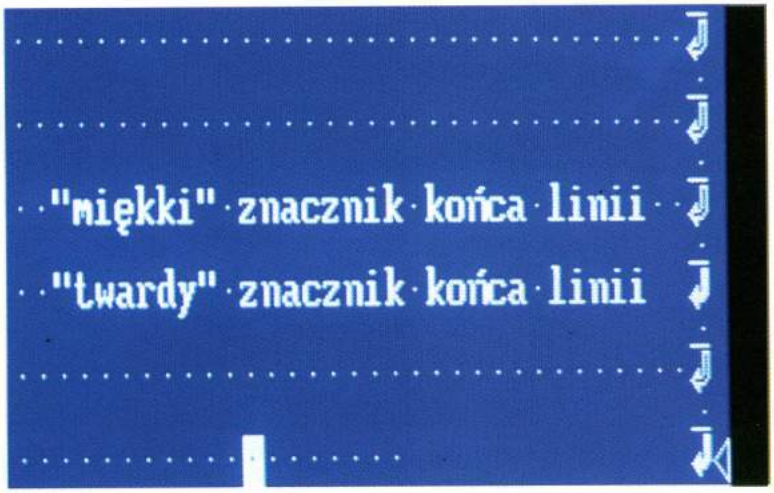
---

[1] W bardziej rozbudowanych edytorach operacja wyszukiwania wskazanego wzorca tekstu ma wiele dodatkowych udogodnień. Na przykład wyszukiwanie może przebiegać do przodu albo do tyłu w stosunku do aktualnej pozycji kursora w dokumencie, mogą być wyszukiwane tylko całe wyrazy zgodne ze wzorce albo także wystąpienia podanego wzorca jako **fragmentu** większego tekstu, podczas wyszukiwania można rozróżniać duże i małe litery albo je utożsamiać itp. Ponadto w większych edytorach dostępna jest opcja **Search and Replace** (wyszukaj i zamień) umożliwiająca znajdowanie podanych wzorców tekstu wraz z automatyczną ich zamianą na inny, nowy tekst. Jest to bardzo wygodne i uniwersalnie przydatne narzędzie, pozwalające (przy pomysłowym zastosowaniu) na dokonywanie w tekście bardzo pożytecznych zmian i poprawek przy minimalnym wkładzie pracy. Więcej szczegółów na ten temat zamieszczono w podrozdziale dotyczącym edytora **ChiWriter**, który oczywiście ma wspomnianą opcję i to w bardzo udoskonalonej formie.



gdzie ***nazwa_dokumentu*** jest nazwą pliku zawierającego tworzony lub poprawiany tekst.

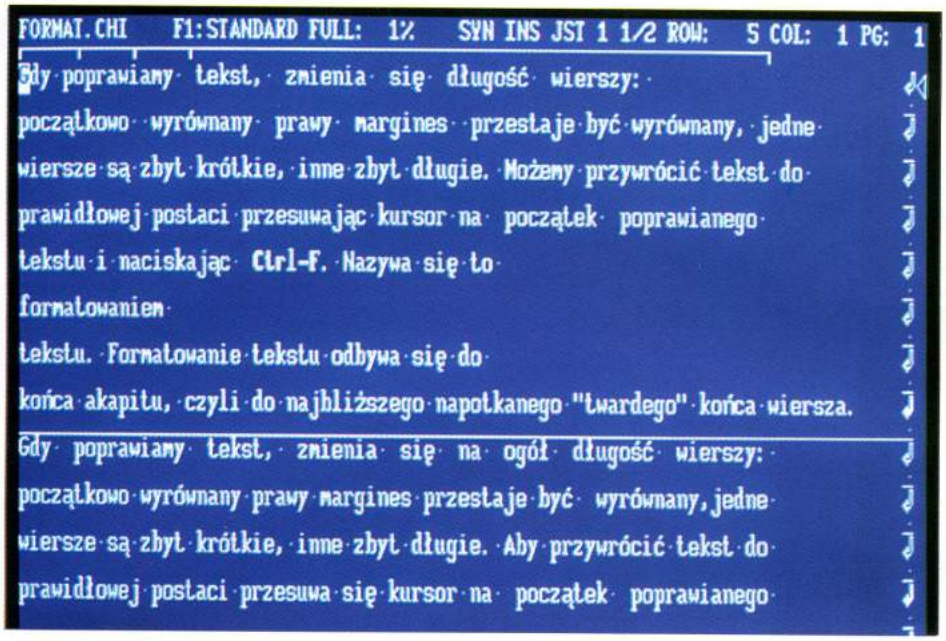
Pisanie tekstu w **ChiWriterze** odbywa się w sposób identyczny, jak w edytorze **NC**, jednak pisząc w **ChiWriterze** można nie martwić się o koniec wiersza: edytor automatycznie przenosi cały wyraz do nowego wiersza, jeżeli podczas pisania przejdzie się poza prawy margines. Następuje również wyrównanie prawego marginesu w zakończonym właśnie wierszu poprzez wstawienie dodatkowych odstępów (spacji) pomiędzy wyrazami. Spacje wstawiane przez edytor dla wyrównania marginesu (tzw. "miękkie" spacje) ukazują się na ekranie jako puste miejsca, w odróżnieniu od spacji wprowadzanych przez użytkownika naciskaniem klawisza odstępu (tzw. "twarde" spacje). "Miękkie" spacje mogą być usunięte przez edytor, np. przy zmianie szerokości marginesów, natomiast "twarde" spacje stanowią część tekstu i mogą być usunięte tylko przez użytkownika poprawiającego tekst. Podobnie edytor rozróżnia także "miękkie" i "twarde" przejścia do nowego wiersza. Po prawej stronie ekranu można zauważyć dwa rodzaje znaków: Pusty w środku "miękki" znacznik końca linii lub pełny w środku "twardy" znacznik końca linii (patrz fotografia obok). "Miękkie" przejścia do nowego wiersza występują tam, gdzie edytor automatycznie przeniósł do następnego wiersza tekst wykraczający poza prawy margines. Podobnie jak "miękkie" spacje, również i "miękkie" przejścia do nowego wiersza mogą być później zmienione przez edytor, np. przy zmianie marginesów. "Twarde" przejścia do nowego wiersza uzyskuje się przez wciśnięcie klawisza **Enter** i podobnie jak "twarde" spacje są one częścią tekstu, zatem mogą być usunięte tylko przez użytkownika[1]. Wprowadzanie "twardych" przejść do nowego wiersza ma sens tylko na końcu akapitu.

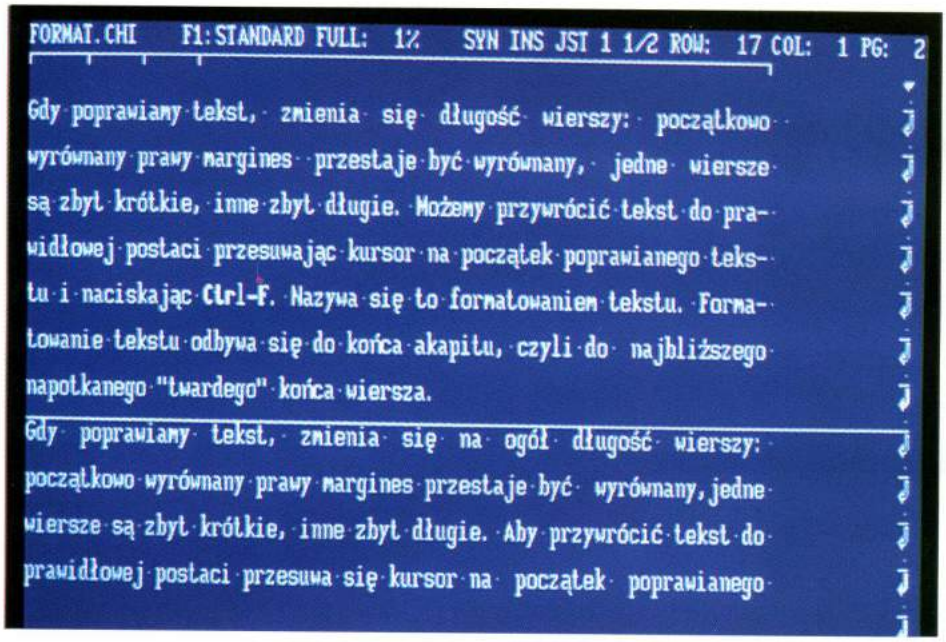


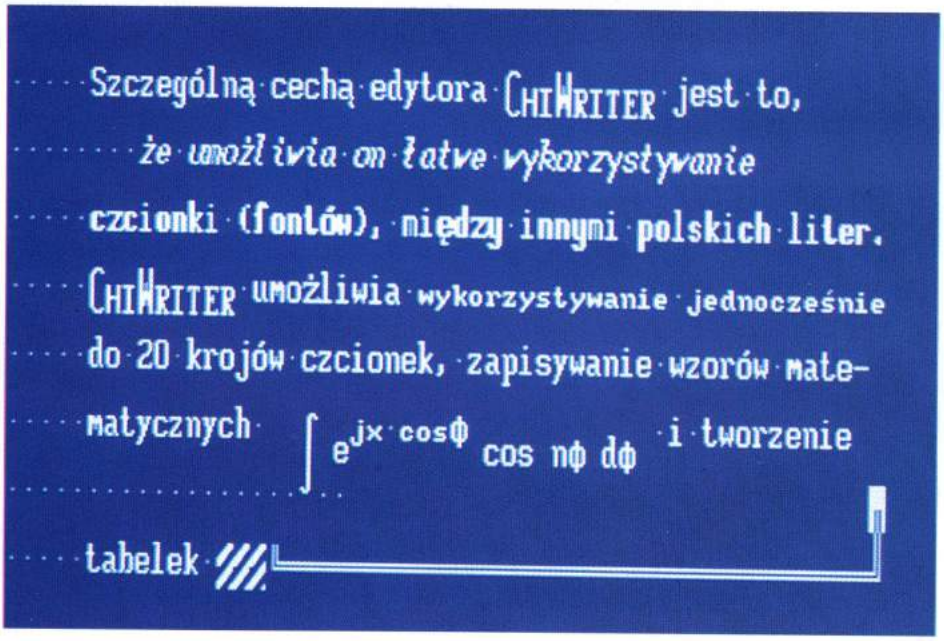
Gdy poprawiamy tekst, zmiania się na ogół długość wierszy: początkowo wyrównany prawy margines przestaje być wyrównany, jedne wiersze są zbyt krótkie, inne zbyt długie. Aby przywrócić tekst do prawidłowej postaci, przesuwa się kursor na początek poprawianego tekstu i wciska **Ctrl–F**. Efekt takiego zabiegu przedstawia seria dwóch fotografii na następnej stronicy. Czynność ta nazywa się formatowaniem tekstu. Formatowanie tekstu odbywa się do końca akapitu, czyli do najbliższego napotkanego "twardego" końca wiersza. Wyjaśnia to, dlaczego nie należy nadużywać klawisza **Enter** podczas pisania tekstu. Tam, gdzie wciśnięto **Enter**, zawsze będzie przejście do nowego wiersza.

ChiWriter automatycznie oblicza i zaznacza w tekście miejsca, gdzie podczas drukowania nastąpi zmiana strony. Miejsca takie oznaczone są liniami kropkowanymi przez całą szerokość ekranu, Numer aktualnej strony można też odczytać z linii statusu – po napisie PAG: jest pokazywany

[1] Aby usunąć niepotrzebne "twarde" przejście do nowej linii, czyli połączyć dwie następujące po sobie linie w jedną, przesuwa się kursor do początku drugiej linii i kasuje się znacznik końca linii używając klawisza **Backspace.**







numer strony, na której aktualnie znajduje się kursor. Dodatkowo podawana jest także informacja o dokładnej lokalizacji kursora w ramach stronicy.

Do przeniesienia się na konkretną stronę używa się zestawu klawiszy

CTRL–G.

W odpowiedzi na to program wyświetla komunikat "**Goto page:**", a Ty wpisujesz numer tej strony i wciskasz **Enter.**

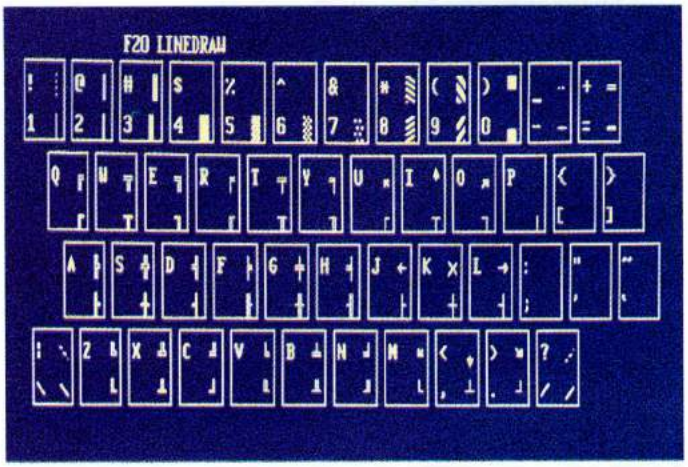
Szczególną cechą edytora **ChiWriter** jest to, że umożliwia on łatwe wykorzystywanie wielu krojów czcionki (*fontów*), między innymi polskich liter[1]. **ChiWriter** umożliwia wykorzystywanie jednocześnie do 20 krojów czcionek. Popatrz na tekst pokazany na dolnej fotografii – tak właśnie można pisać w tym edytorze.

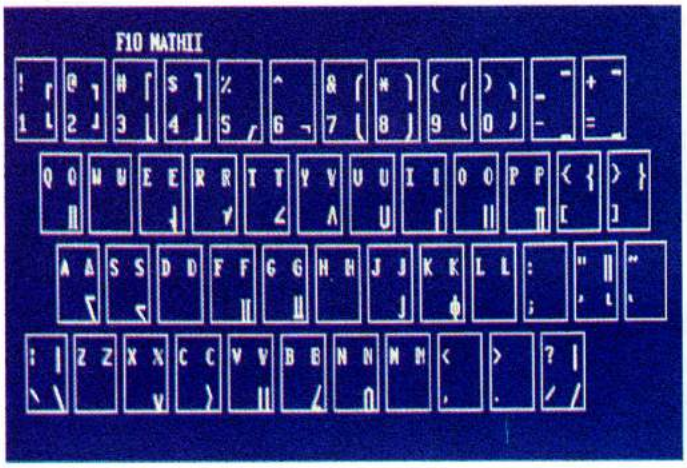
Do przełączania krojów czcionek służą klawisze funkcyjne (**F1 – F10**). Pierwsze 10 fontów uzyskuje się przez wciskanie samych klawiszy funkcyjnych, pozostałe 10 – przez wciskanie klawiszy funkcyjnych wraz z **Shift.** Przyporządkowanie fontów klawiszom funkcyjnym może być różne w różnych instalacjach programu **ChiWriter**, gdyż jest definiowane przez użytkownika.

Aby zmienić font wciska się dwukrotnie odpowiedni klawisz funkcyjny,

[1] Problem polskich liter omówiony będzie przy prezentacji edytora **Ami Pro,** ponieważ w **ChiWriterze** bywa rozwiązywany różnie przez różnych użytkowników, co początkującym może sprawiać spory kłopot.

odpowiadający wybranemu fontowi. W linii statusu jest napis, który pokazuje numer i nazwę aktualnego fontu (zwykle widnieje tam napis F1:STANDARD oznaczający, że w użyciu jest standardowy font dla zwykłych liter).

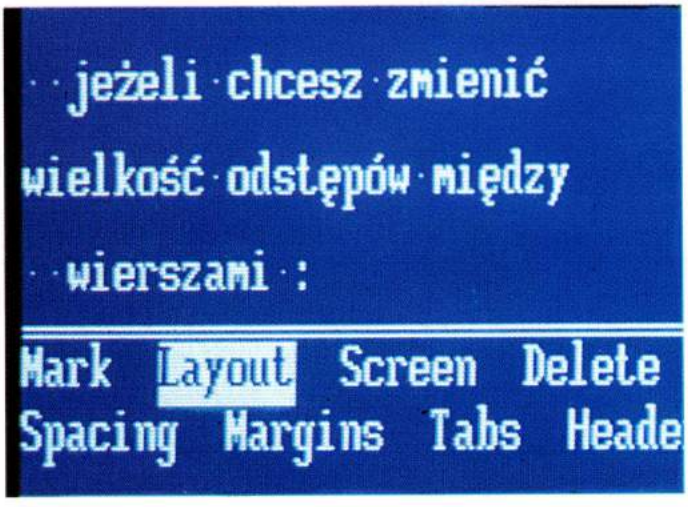




W niektórych fontach (np. zawierających symbole matematyczne lub elementy ramek) trudno się domyślić, który klawisz odpowiada której literze (lub znakowi). Wciska się wtedy klawisz odpowiadający potrzebnemu fontowi, a następnie **Alt–H**. Pokaże się wtedy na ekranie aktualne przyporządkowanie klawiszy do znaków danego fontu (patrz fotografia obok). Można wtedy przeglądać inne fonty, wciskając odpowiednie klawisze funkcyjne (patrz fotografia niżej) albo natychmiast wstawić dany znak do tekstu (wciskając klawisz odpowiadający żądanemu znakowi). Żeby ustalić na dłużej font, którym będziemy pisać więcej niż jedną literę – naciskaj odpowiedni klawisz funkcyjny **dwukrotnie.**

Pisząc tekst, możesz stosować różną wielkość odstępów między wierszami. Po uruchomieniu edytora przyjmowany jest początkowo odstęp **1**, ale wielkość odstępu możesz swobodnie wybierać z **menu** programu, widocznego u dołu ekranu. Objaśnię teraz jak korzystać z menu, ponieważ jest to podstawowy sposób sterowania edytorem.

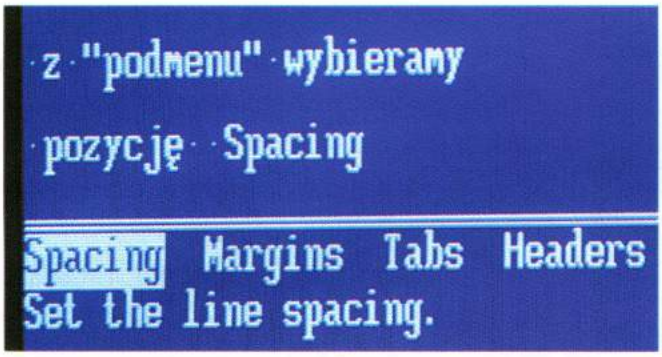
Wciśnij **Esc**. Pojawi się wtedy rozjaśniony prostokąt w linii menu. Możesz go przesuwać klawiszami kursora. Aby wybrać komendę lub wejść do podmenu, trzeba przesunąć prostokąt na nią i wcisnąć **Enter.** Większość komend z menu zawiera "podmenu", w związku z czym można wybierać podkomendy tą samą metodą. Gdy przesuwa się strzałkami rozjaśniony prostokąt na poszczególne komendy menu, w linii poniżej menu pojawia się albo podmenu danej komendy, albo – jeżeli ta komenda nie zawiera już żadnego podmenu – jej krótki opis (niestety po angielsku)[1]. Klawisz **Esc** pozwala powrócić do edycji tekstu bez wybierania żadnej komendy.

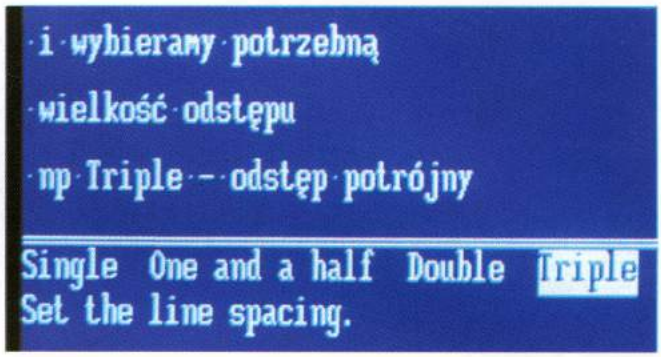


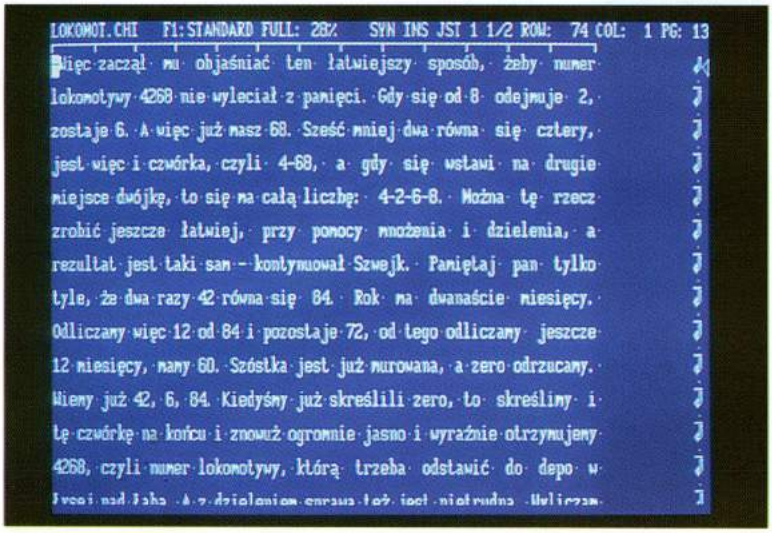
Obejrzymy to teraz na konkretnym przykładzie. Aby zmienić wielkość odstępów między wierszami, musimy poszukać komendy **Layout** (rozmieszczenie). Po wybraniu tej komendy z menu "schodzimy" do podmenu i wybieramy z niego pozycję **Spacing** (odstępy), a następnie wybieramy potrzebną wielkość odstępu (na przykład **Triple** – odstęp potrójny). Przebieg potrzebnych działań pokazałem na fotografiach obok i na następnej stronicy u góry.

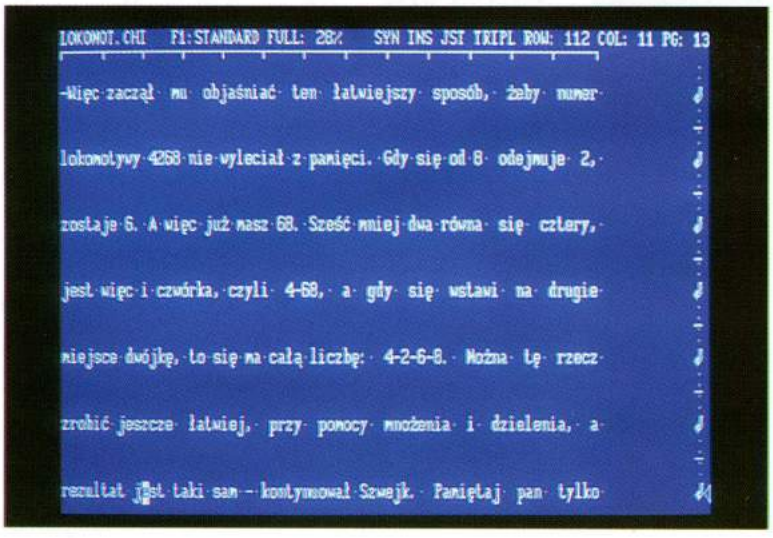
Jeżeli zmienisz odstęp między wierszami, a następnie użyjesz operacji formatowania

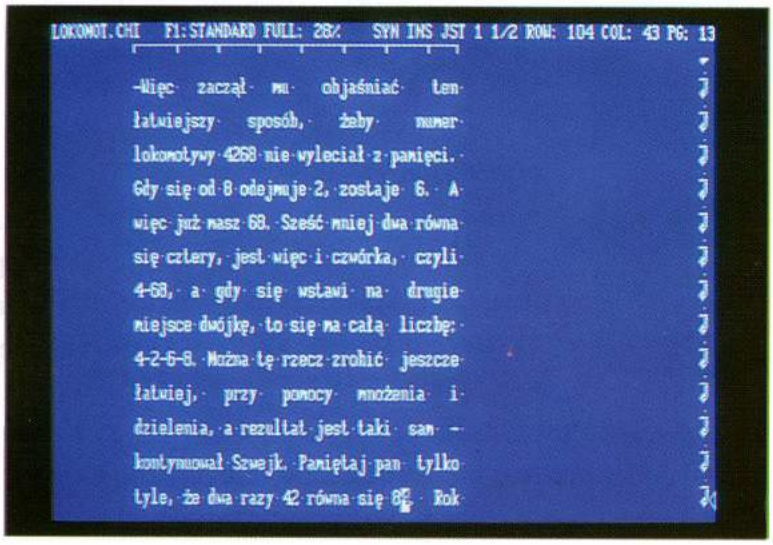
[1] Dłuższy opis aktualnie podświetlonej komendy można uzyskać wciskając **Alt–H.**









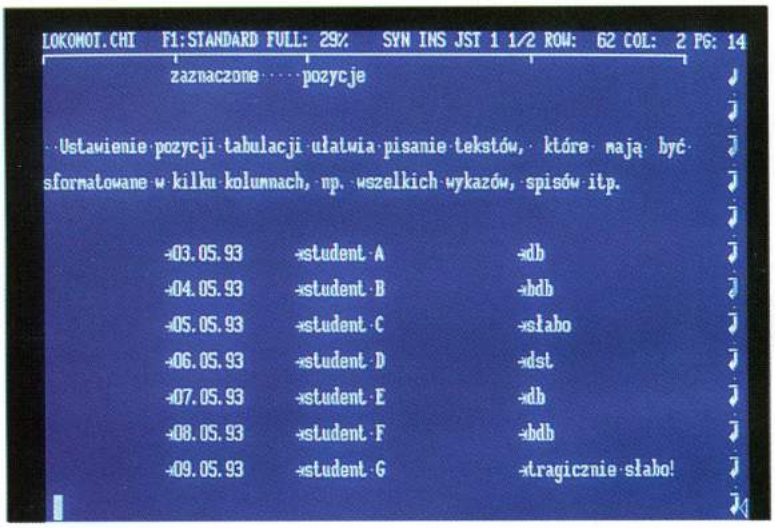


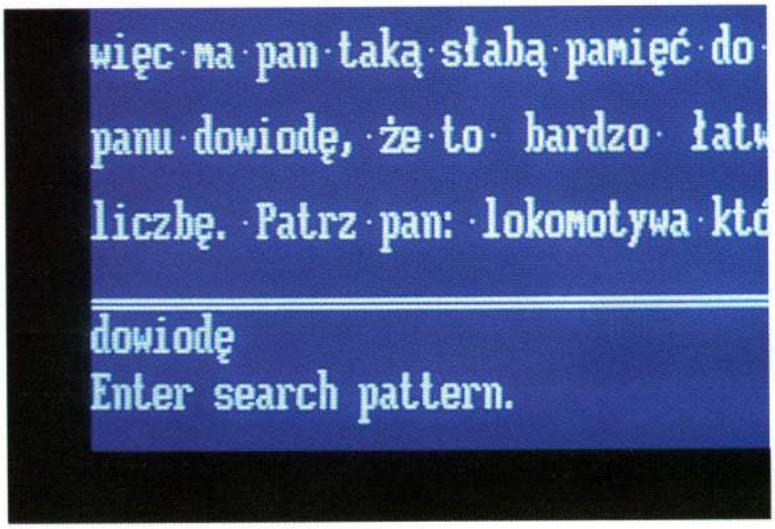
tekstu, wiersze sformatowanego akapitu będą miały nowo ustawione odstępy. Fotografie obok pokazują, jak zmienia się obraz tekstu po przeformatowaniu na inną wartość odstępu.

Aby zmienić położenie lewego albo prawego marginesu dla pisanego tekstu, przesuwa się kursor do odpowiedniego miejsca i wciska **Ctrl–[** dla lewego marginesu albo **Ctrl–]** dla prawego. Warto przy tym zwrócić uwagę na wskaźnik pozycji tabulacji oraz marginesów, który znajduje się pod linią statusu – będzie on pokazywał nowe położenie marginesów. Jeżeli wykonasz formatowanie tekstu przy zmienionych marginesach, sformatowany akapit zostanie dostosowany do nowych położeń marginesu. Tym sposobem możesz łatwo i szybko przeformatować tekst na różną liczbę znaków w wierszu. Efekty takiego formatowania można prześledzić na fotografii obok.

Funkcje formatowania pojedynczych wierszy realizują klawisze **Ctrl–C** i **Ctrl–M. Ctrl–C** centruje linię, w której znajduje się kursor, **Ctrl–M** natomiast dosuwa ją do prawego marginesu.

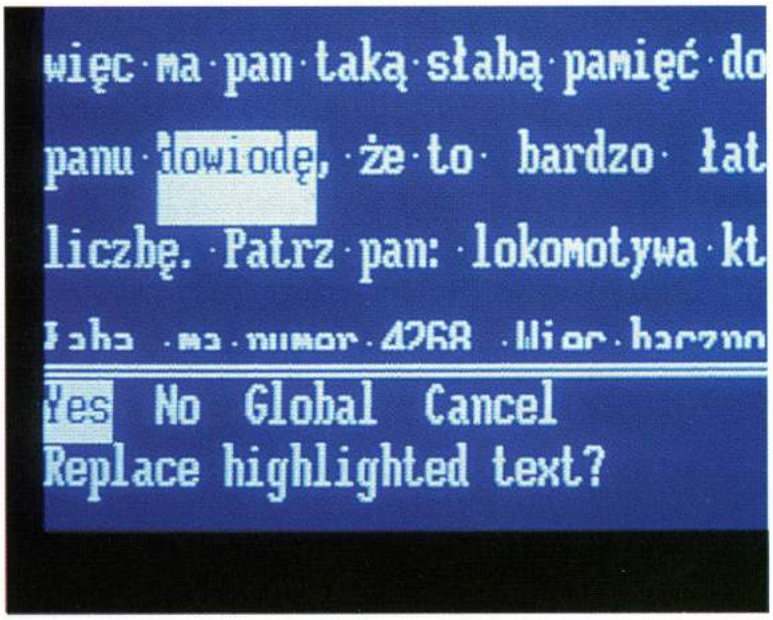
Wciśnięcie klawisza **Tab** z lewej strony klawiatury przenosi kursor do następnej pozycji tabulacji – jest to użyteczne przy pisaniu tekstów, które mają być sformatowane w kilku kolumnach, np. wszelkich wykazów, spisów itp. Pozycje tabulacji oznaczone są na wskaźniku u góry ekranu. Możesz je łatwo dostosować do swoich potrzeb. Wciśnięcie klawiszy **Ctrl–T** wstawia pozycję tabulacji, jeżeli jej w danej kolumnie nie





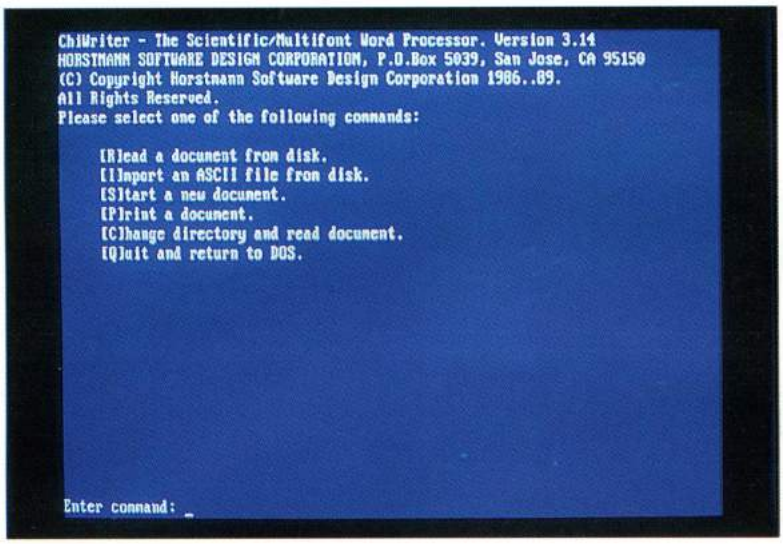
było, usuwa natomiast, jeżeli była. Użycie klawisza tabulacji umieszcza na ekranie specjalny znacznik, którego nie będzie jednak widać w tekście wydrukowanym na drukarce. Efekt korzystania z tabulacji pokazuje fotografia.

Edytor **ChiWriter** ma bogatsze (w stosunku do **NC**) możliwości wyszukiwania i zamiany fragmentów tekstu. Aby odnaleźć fragment tekstu (frazę) w tekście, wciskasz **Ctrl–S**, po czym wpisujesz poszukiwaną frazę (patrz fotografia) i naciskasz **Enter**. Tekst jest przeszukiwany od pozycji kursora do końca tekstu. Po znalezieniu frazy kursor jest ustawiany na jej początku. Jeżeli chce się znaleźć kolejne wystąpienie tej samej frazy, **Ctrl–L** powtarza ostatnie przeszukiwanie. Można nie tylko wyszukiwać daną frazę, ale też automatycznie zamieniać ją na inną. Aby zastąpić daną frazę inną, należy zamiast **Ctrl–S** użyć **Ctrl–R**, wprowadzić tekst, który ma być wyszukany, a następnie tekst, który ma być wstawiony zamiast niego. Jeżeli komputer znajdzie dany tekst, zapyta się, czy wymieniać go, czy nie (patrz fotografia). Jeśli chcesz go wymienić wciskasz **Enter**, jeśli nie, przesuwasz podświetlony prostokąt (tak samo, jak w menu) na opcję **No** i wciskasz **Enter.** Jeśli nie chcesz, by komputer pytał Cię o każdy znaleziony tekst (zamienić, czy nie?), lecz zamienił go automatycznie w każdym miejscu, wybierz opcję **Global**. Natomiast jeżeli chcesz przerwać dalsze poszukiwanie, wybierz **Cancel.**



Dokument przechowywany jest w pamięci operacyjnej komputera, jeśli więc wyłączysz komputer – dokument zginie. Dlatego bardzo ważny jest zestaw komend pozwalających zapisywać dokumenty na dysk oraz poleceń pozwalających dokumenty wczytywać z dysku.

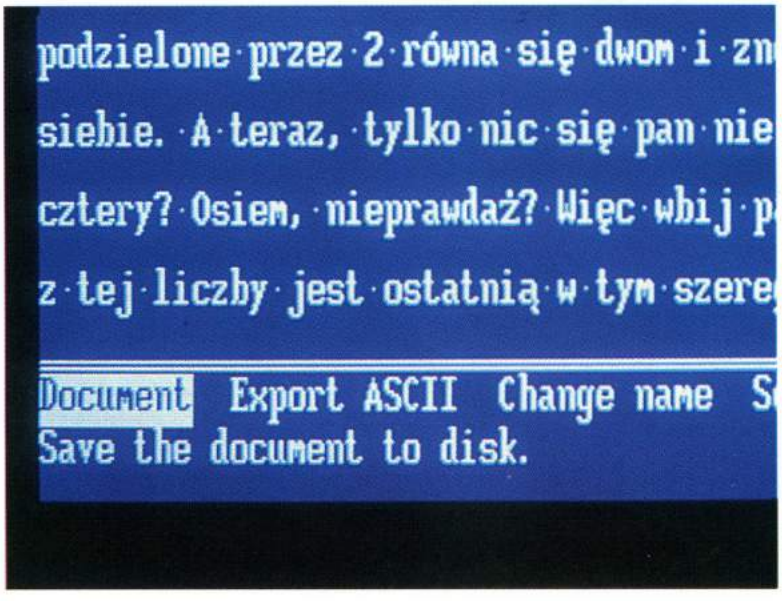
Jak wspomniałem na początku rozdziału, rozpoczynając pracę z **ChiWriterem** możesz od razu podać nazwę tekstu znajdującego się już na dysku, który chcesz poddawać dalszej obróbce, i wówczas edytor uruchamia się od razu z wczytanym wskazanym tekstem. Podobnie, jeśli w wywołaniu **CW** podasz nazwę nie istniejącego dokumentu – edytor sam utworzy potrzebny nowy plik. Jeśli jednak nie podasz nazwy pliku, po uruchomieniu edytora



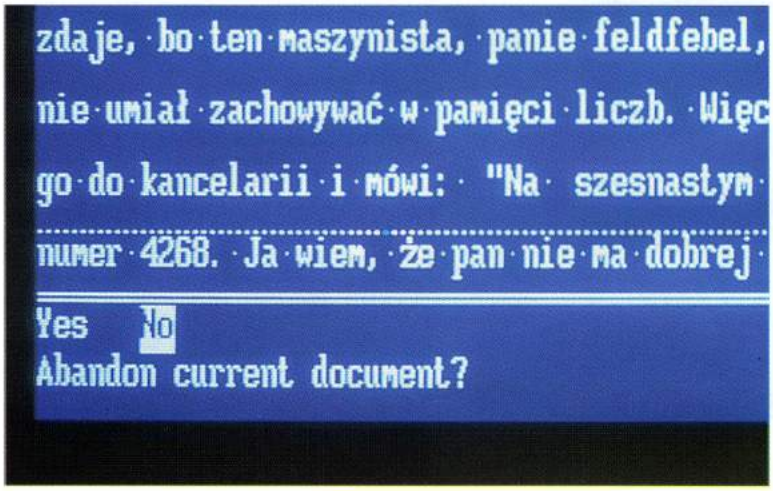
pojawi się **początkowe menu** (patrz fotografia). Wówczas jeśli chcesz wczytać tekst, który wcześniej był już przetwarzany za pomocą tego edytora – wciśnij klawisz **R**. Na ekranie pojawi się lista nazw plików znajdujących się na dysku, które zawierają teksty ChiWritera (patrz fotografia niżej). Pierwsza nazwa będzie rozjaśniona. Klawisze kursora przesuwają rozjaśnienie w żądanym kierunku, **Enter** wybiera podświetloną nazwę tekstu do wczytania. Jeśli potrzebnej nazwy tekstu nie ma wśród wyświetlonych nazw (np. znajduje się on na innym dysku lub w innym katalogu), możesz wpisać ją ręcznie z klawiatury.

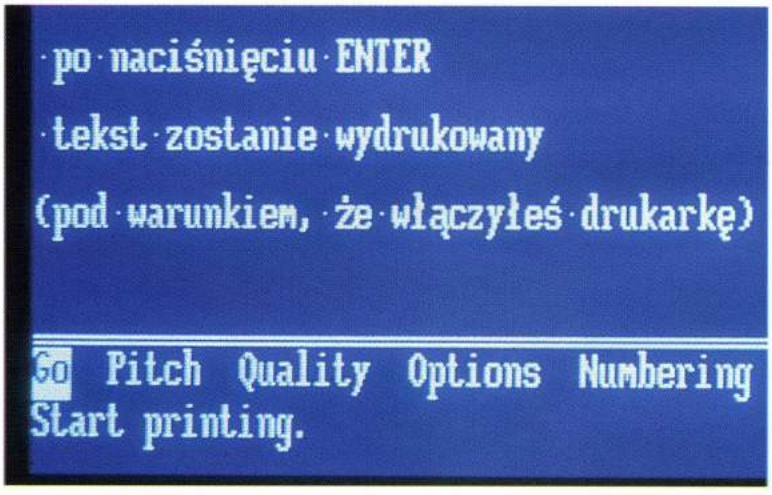
Oczywiście wczytywać tekst do edytora lub rozpoczynać pracę z nowym tekstem można nie tylko na samym początku, lecz także i później w dowolnym momencie pracy. Zanim jednak zdecydujesz się na wczytanie do naszego edytora innego tekstu lub rozpoczęcie zupełnie nowego powinieneś zapisać na dysku tekst aktualnie znajdujący się w pamięci, aby nie przepadł. W tym celu wskazujesz z menu opcję **Write/Document.** Widok ekranu w czasie korzystania z tej opcji przedstawia fotografia niżej.

Dobrze jest także zapisywać tekst na dysku co jakiś czas podczas pracy, gdyż w każdej chwili może się zdarzyć, że ktoś wyłączy Ci niechcący (albo złośliwie) prąd lub że przez pomyłkę dokonasz w edytorze jakiejś operacji powodującej katastrofalne spustoszenie naszego tekstu (np. każesz zamienić wszystkie litery "a" na "e"). Posiadanie w miarę aktualnej kopii tekstu na dysku zabezpieczy Cię przed zmarnowaniem wielu godzin pracy. Na wypadek, gdybyś nie pamiętał o zapisywaniu tekstu na dysk, edytor robi to za Ciebie automatycznie, zapisując co jakiś czas tekst w pliku o nazwie BACKUP.CHI, o czym informuje wyświetlając na ekranie specjalny komunikat (pokazany na fotografii na następnej stronicy). Musisz tylko ustawić w opcji **Write** parametr **Backup frequency,** określający, co ile minut ma być wykonywana kopia bezpieczeństwa. Pamiętaj: lepiej częściej niż za rzadko!



Możesz też wczytać z dysku inny tekst i rozpocząć nad nim pracę. Aby wczytać tekst, używaj opcji menu **Read/New document.** Jeżeli dotychczasowy tekst nie był zapisany na dysk (lub został zmieniony po ostatnim zapisie na dysk), pojawi się pytanie, czy na pewno chcesz go usunąć z





pamięci bez zapisywania (patrz fotografia niżej). Odpowiedź "**No**" odwołuje wczytywanie i powraca do dotychczasowego tekstu,dając Ci możliwość zapisania go na dysku. Jeżeli odpowiesz "**Yes**" (lub tekst był uprzednio zapisany na dysku) pojawi się lista plików, z której możesz w znany już sposób klawiszami strzałek i **Enter** wybrać plik do wczytania. Jeszcze w tym momencie klawisz **Esc** umożliwia odwołanie wczytywania i powrót do uprzednio opracowywanego tekstu.

Opcja menu **Read/Merge document** umożliwia włączenie tekstu znajdującego się na dysku do tekstu aktualnie opracowywanego w miejscu, gdzie znajduje się kursor. Tu również wybiera się nazwę pliku, który ma być wczytany i dołączony.

Aby wydrukować tekst, wybierasz z menu opcję "**Print**", włączasz drukarkę i wybierasz podopcję "**Go**". Przez chwilę na ekranie będą pojawiać się informacje o fontach, które program ładuje do pamięci, po czym rozpocznie się drukowanie (fotografia obok).

Dla zakończenia pracy z programem **ChiWriter** należy wcisnąć **Alt–Q**.

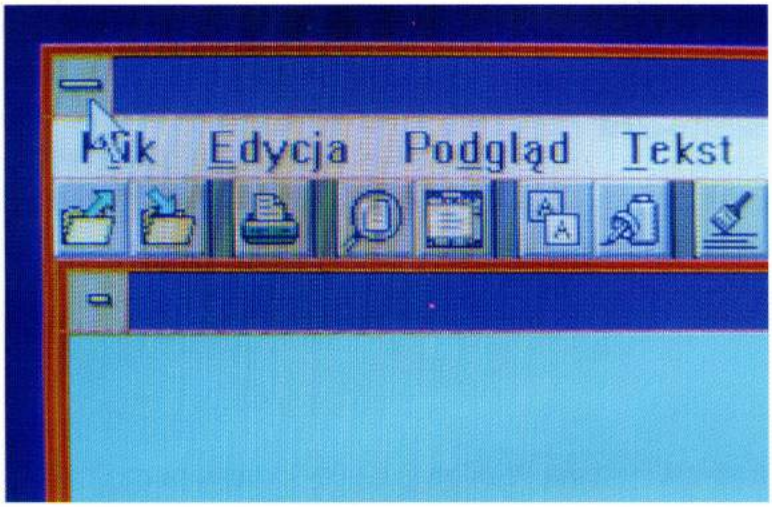
3.5.2.5. Zasady używania procesora tekstów Ami Pro

Procesor tekstów **Ami Pro** jest programem bardzo rozbudowanym, dlatego podczas jego prezentacji w książce ograniczę się do podania jedynie jego **podstawowych elementów**[1]. **Ami Pro** ma jednak dość dobrze zredagowaną, dostępną w każdej chwili podczas jego użytkowania, napisaną po polsku "ściągawkę", zatem możesz zacząć pracę z tym edytorem **prawie** bez żadnej wiedzy początkowej i – w miarę potrzeb – możesz się sam douczać, jak korzystać z kolejnych udogodnień tego programu, przywołując (klawiszem **F1**) okienko podręcznika.

Najważniejsze wiadomości, potrzebne Ci na początku, dotyczą oczywiście tego, jak zacząć pracę z **Ami Pro** i jak ją zakończyć. Generalnie można powiedzieć, że Ami jest programem działającym w systemie Windows i odnoszą się do niego wszystkie te uwagi, jakie zwykle wiążą się z korzystaniem z aplikacji Windows. Przypomnij sobie: **Rozpoczęcie pracy** następuje po wskazaniu

[1] Warto podkreślić, że oryginalna dokumentacja programu, dostarczana przez firmę *Lotus*, obejmuje trzy tomy, z których główny (*"AmiPro 3.0 Edycja Polska – Instrukcja Użytkowania*) ma 628 stronic tekstu (nie licząc indeksu i spisu treści!). W dodatku dokumentacja ta nie **objaśnia** zasad używania edytora, tylko stanowi **spis jego możliwości.** Jest oczywiste, że w tej sytuacji jeden podrozdział książki nie może zawrzeć w sobie **całej** wiedzy na temat tak rozbudowanego programu.

ikony programu i dwukrotnym szybkim tupnięciu myszką. Jedyna trudność może polegać na znalezieniu tej ikony i dlatego warto zapamiętać, że Ami wchodzi zwykle w skład grupy o nazwie ***Lotus Applications*** (patrz fotografia niżej).



Zakończenie pracy z Ami może nastąpić po dwukrotnym szybkim tupnięciu myszki[1] w lewym górnym rogu okna edytora (na "klawiszu" – jak to pokazuje fotografia obok). Jest jeszcze kilka innych sposobów zakończenia pracy z edytorem, ale bardziej opłaca się znać i biegle stosować jedną z nich niż pamiętać wszystkie.

Wszystko, co wiesz o pisaniu w innych edytorach da się natychmiast zastosować w Ami: tak samo kieruje się ruchami kursora[2], te same klawisze służą do pisania liter, cyfr, znaków itp., identyczne są zasady usuwania błędnie napisanych słów czy znaków. Jedyna nowość, z którą warto się od razu zapoznać, polega na możliwości swobodnego **stosowania w tekstach polskich liter.**

Polskie litery stanowią generalnie problem przy posługiwaniu się komputerem. Nie ma ich na klawiaturze, nie są możliwe do wyświetlenia na ekranie bez stosowania specjalnych zabiegów[3], są trudności z uzyskaniem ich na drukarce[4]. **Ami Pro** rozwiązuje ten problem w sposób generalny – wprowadza do asortymentu systemu Windows swoje znaki w taki sposób, że nie tylko sam z nich korzysta, ale i wszystkie inne aplikacje (na przykład programy rysujące czy bazy danych) mogą także tych znaków używać. To duża wygoda. Wygoda ta okupiona jest jednak pewną ceną: uzyskanie polskiego znaku w tekście wymaga kolejnego naciśnięcia **dwóch** klawiszy[5]: klawisza zawierającego znak tyldy "~" (lub odwróconego apostrofu "`") oraz klawisza litery, którą należy "spolszczyć". Tak więc **Ą** otrzymuje się naciskając ~A, **ń** to ~n

---

[1] **Wszystkie** czynności edytora mogą być alternatywnie sterowane myszką albo za pomocą klawiatury (na przykład zakończenie pracy można też uzyskać naciskając klawisze **Alt–F4).** Jednak w książce konsekwentnie podajemy tylko sposób działania za pomocą myszki.

[2] Kursor wskazujący na miejsce wpisywania lub poprawiania tekstu w **AmiPro** ma formę grubej pionowej kreski, a jego rozmiar wskazuje na wielkość używanej czcionki. Poza tym kursorem na ekranie widoczny jest drugi kursor, sterowany bezpośrednio ruchami myszki. Wygląda on jak duża litera **I,**gdy znajduje się w polu pisania tekstu względnie jak gruba strzałka, gdy wyprowadzi się go w obszar *menu* lub *ikon* sterujących pracą programu. W obszarze pisania tekstu "tupnięcie" myszki przywołuje kursor tekstowy do miejsca, w którym znajduje się aktualnie kursor myszy. Jest to bardzo szybki i bardzo skuteczny sposób sterowania kursorem przy pisaniu tekstów.

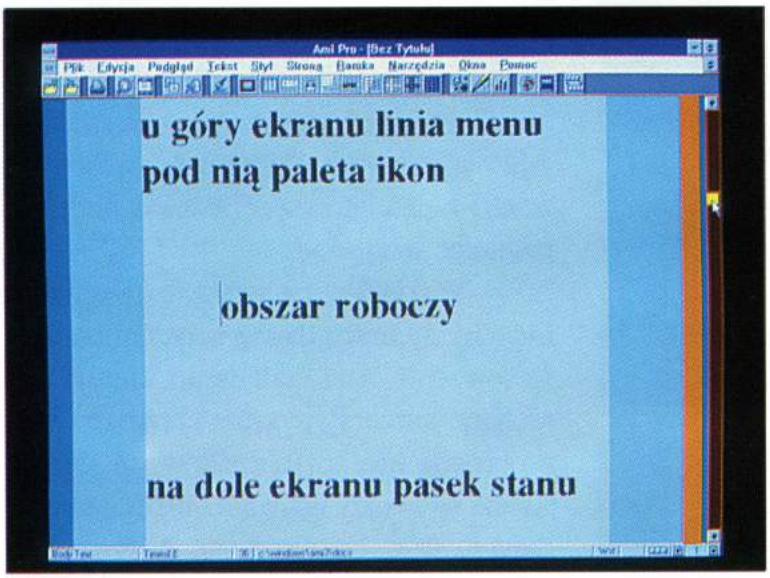
[3] Odpowiednia wstawka programowa pozwala wprawdzie uzyskać polskie litery na monitorach typu **EGA** lub **VGA**, ale już w przypadku popularnego w Polsce monitora **Hercules** konieczna jest do tego elektroniczna przeróbka karty graficznej w komputerze, co jest kosztowne i kłopotliwe.

[4] Na drukarkach laserowych i niektórych igłowych można uzyskać polskie litery ładując ich wzorce (tzw. *fonty*) do wewnętrznej pamięci drukarki przed rozpoczęciem drukowania. Taka technika (znana pod nazwą *download)* jest łatwiejsza niż elektroniczne przerabianie pamięci EPROM sterującej pracą drukarki, ale wymaga zwykle ponawiania procesu ładowania po każdym wyłączeniu drukarki, co jest nieco kłopotliwe i o czym łatwo zapomnieć – a wtedy kilkanaście stronic tekstu może wydrukować się z użyciem różnych "robali" zamiast polskich znaków i całą pracę trzeba będzie powtarzać.

[5] Sposób używania polskich liter może być w Ami Pro zdefiniowany także odmiennie, na przykład skorzystanie z polskiej litery jest możliwe,jeśli naciśnie się równocześnie z odpowiednią literą (np. **A**) klawisz **Alt.** Zależy to od sposobu zadeklarowania klawiatury podczas instalacji programu Ami Pro. Opisany wyżej sposób korzystania z polskich liter jest najczęściej spotykany, wnosi najmniej ograniczeń i znany jest szeroko pod nazwą "klawiatury programisty".

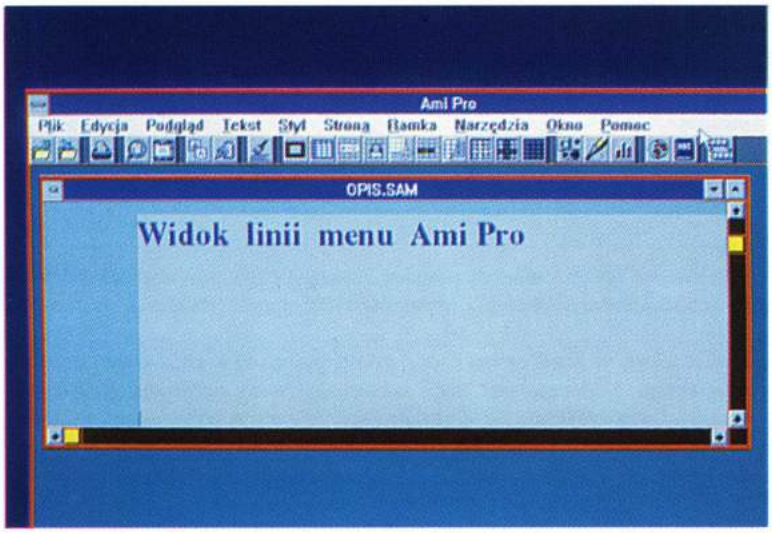
więc **Ą** otrzymuje się naciskając ~A, **ń** to ~n itd. Jedyny problem polega na rozróżnieniu liter **ż** i **ź** – w Ami przyjęto zasadę, że **ż** otrzymuje się z litery **z**, a **ź** z sąsiedniej litery **x**[1].

Ekran edytora **Ami Pro** w ogólnym zarysie przypomina ekran edytora **NC**, jest jednak od niego znacznie bogaciej wyposażony. Prawie cały ekran wypełnia **obszar roboczy**, czyli powierzchnia, na której ukazywać się będzie pisany przez użytkownika tekst. Jest to oczywiście korzystne – zasadnicza funkcja edytora polega na pisaniu tekstu i im mniej rzeczy w tym przeszkadza – tym lepiej. Doświadczony "wyjadacz" potrafi tak ustawić parametry edytora, że na ekranie **naprawdę** może nie być nic, poza redagowanym tekstem. Taki zaawansowany użytkownik zna na pamięć wszystkie "chwyty" pozwalające za pomocą równoczesnego naciskania wielu klawiszy nakazywać **Ami Pro** wszystkie wymagane czynności – dlatego pusty ekran jest dla niego najbardziej naturalnym sposobem szybkiej i wydajnej pracy. Jednak dla wygody mniej wprawnych (na przykład początkujących) użytkowników są na standardowym ekranie **Ami Pro** dodatkowe elementy, ułatwiając sterowanie edytorem. Tymi elementami pomocniczymi są:



- **linia menu**,
- **paleta ikon**,
- **pasek stanu**.

Linia menu znajduje się (zgodnie z tradycją systemu Windows) u góry ekranu, bezpośrednio pod tytułową "belką", na której widnieje nazwa edytora – sama lub wraz z tytułem aktualnie przetwarzanego dokumentu (zależy to od tego, czy dokument zajmuje całe okno edytora, czy mieści się w nim w postaci zmniejszonej – być może wraz z kilkoma innymi dokumentami). Linia menu, służąca do sterowania pracą edytora, zawiera bardzo wiele pozycji, a każda z nich ma możliwość "rozwijania się" w kolejne szczegółowe menu. Z kolei ich pozycje otwierają dostęp do specjalnych okien itd. W sumie za pomocą linii menu można wydać edytorowi **kilka tysięcy** różnych poleceń i dyspozycji, których samo tylko wyszczególnienie zajmuje sporą książkę. Żebyś jednak nie pozostał bez pomocy przy stawianiu pierwszych kroków z edytorem **Ami Pro** – opiszemy i objaśnimy Ci teraz w skrócie zawartość linii głównego menu. Znajdują się w niej kolejno następujące opcje:

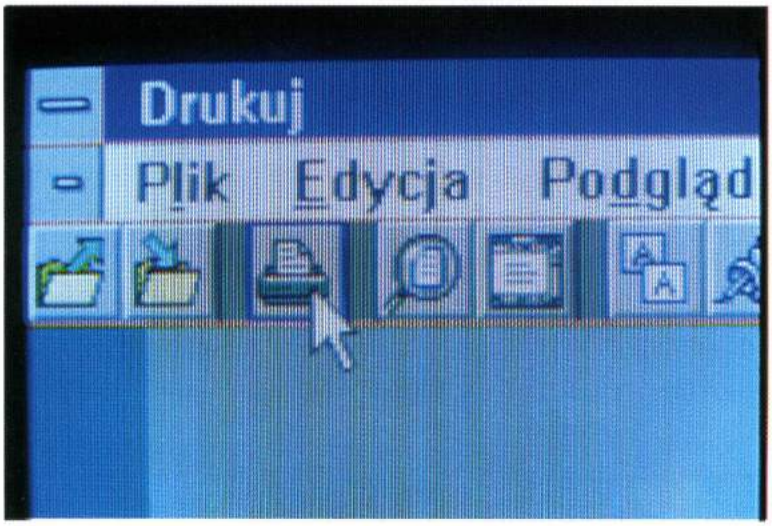


[1] Do systemu tego można dość łatwo przywyknąć i stosuje się go w miarę wygodnie, chociaż napisanie – przykładowo – słowa "**żółw**" wymaga aż siedmiu uderzeń w klawiaturę, zamiast czterech.

- **Plik** – operowanie plikami, otwieranie i zamykanie kolejnych dokumentów, a także polecenia związane z drukowaniem dokumentów;
- **Edycja** – zaznaczanie, usuwanie, kopiowanie i przemieszczanie fragmentów tekstu, a także operacje wyszukiwania i zamiany oraz powiązań z różnymi programami;
- **Podgląd** – określa parametry wyświetlania tekstu na ekranie, pozwala zobaczyć redagowaną stronicę w powiększeniu i w pomniejszeniu, uzupełnić ją przymiarami itd.;
- **Tekst** – służy do wybierania wielkości i kroju czcionki (normalna, pogrubiona, pochylona, podkreślona itp.), a także odstępów, sposobu wyrównania i wcięć w tekście;
- **Styl** – pozwala wybrać, zmodyfikować lub zdefiniować styl akapitu (patrz dalej);
- **Strona** – umożliwia określenie wyglądu całej strony, ustawienie numeracji itp.;
- **Ramka** – służy do operowania ramkami, pozwalającymi między innymi wprowadzać do tekstu ilustracje (rysunki);
- **Narzędzia** – jest to obszerne menu pozwalające na tworzenie i przetwarzanie grafiki, operowanie tabelami i wzorami, ustawianie parametrów edytora i wyglądu ekranu itp.;
- **Okno** – pozwala wybierać aktywne okna i rozmieszczać je na ekranie;
- **Pomoc** – umożliwia w każdej chwili uzyskanie informacji pomocniczych (jest to bardzo poręczna i bogata w treści "ściągawka").

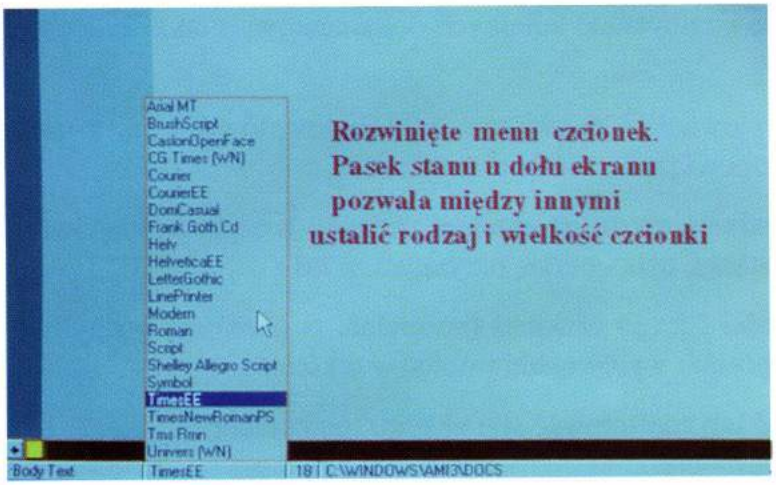
Poprzez odwołania do linii menu i korzystanie z rozwijających się kolejno pozycji "podmenu" można wydawać edytorowi dowolne polecenia i można uzyskiwać praktycznie wszystkie jego usługi. Technika odwołania do linii menu jest elementarnie prosta: wystarczy wskazać odpowiednią pozycję myszką i "tupnąć".

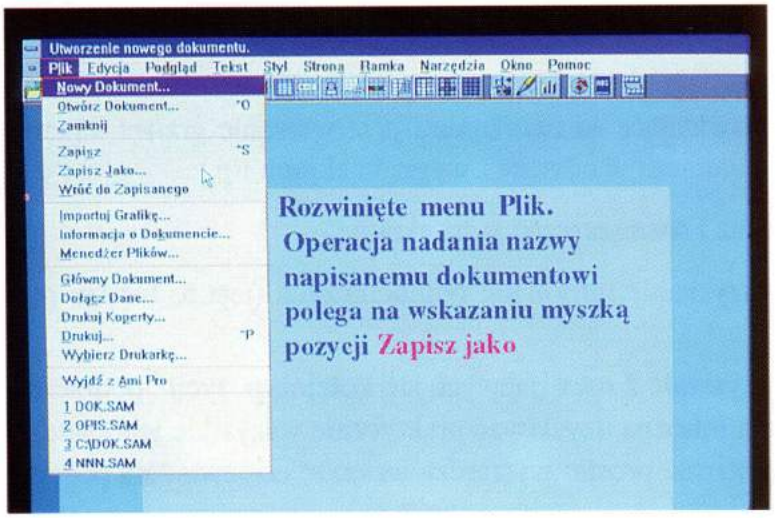
Paleta ikon[1] dostarcza alternatywnej (w stosunku do linii menu), jeszcze łatwiejszej metody sterowania pracą edytora Ami Pro. Dla wykonywanych przez edytor operacji przewidziano sugestywne w kształcie obrazki (ikony), które mogą być wykorzystane do szybkiego i wygodnego wywoływania wymaganych czynności. W celu nakazania edytorowi określonego działania wskazuje się myszką potrzebną ikonę. Naciśnięcie lewego klawisza myszki spowoduje natychmiastowe wykonanie wymaganej czynności, natomiast naciśnięcie prawego klawisza pozwala dowiedzieć się, co właściwie dana ikona znaczy – na górej belce okna edytora ukazują się wtedy odpowiednie teksty objaśniające – jak to pokazano na fotografii obok.

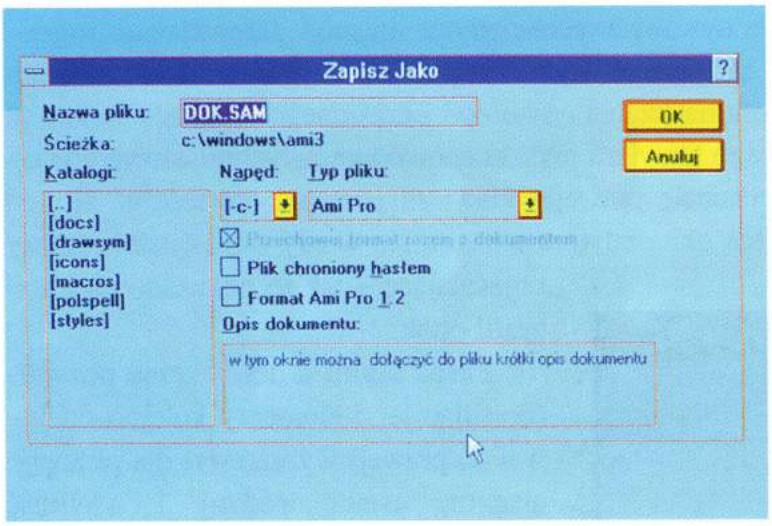


Pasek stanu u dołu ekranu pozwala (opisując jego elementy kolejno od lewej do prawej) wybrać **styl** dla pisanego akapitu, ustalić **rodzaj i wielkość czcionki**, określić położenie kursora, przełączyć tryb pracy edytora z opcji wstawiania tekstu na opcję zastępowania starego tekstu nowym, wreszcie pozwala wybrać jedną z wielu palet ikon oraz przejść do dowolnej innej stronicy w obrębie redagowanego dokumentu.

[1] Paleta ikon może zostać umieszczona w dowolnym punkcie ekranu, jednak zwykle widzi się ją u góry ekranu, bezpośrednio pod linią menu. Producenci Ami Pro przewidzieli kilka wymiennych palet, dostosowanych do typowych zadań wykonywanych przez użytkownika, a ponadto bardziej doświadczony użytkownik może sam sobie skompletować taką paletę ikon, z jaką mu się będzie najwygodniej pracowało.







Widok paska stanu na fotografii obok pomoże w orientacji, jakie są jego możliwości. Wszystkie te możliwości uzyskuje się po "tupnięciu" myszką w odpowiednim fragmencie paska stanu.

Napisany dokument, jeśli był tylko zwykłą "wprawką", można porzucić w momencie kończenia pracy z edytorem, ale można mu także nadać nazwę[1] i zapisać z tą nazwą w postaci pliku na dysku. Operacja nadania nazwy polega na wskazaniu myszką pozycji menu **Plik** (patrz fotografia obok), a następnie na wybraniu z tego menu pozycji **Zapisz jako**. Po wskazaniu tej opcji otwiera się na ekranie okno dialogowe (pokazane na fotografii niżej), w którym użytkownik **musi** wpisać wybraną **Nazwę pliku**, ale **może** także wskazać (myszką) katalog (a także napęd dyskowy), w którym chce ten plik umieścić, atrybuty pliku (takie, jak jego typ i format), a także może dołączyć do pliku krótki **opis dokumentu**, który bardzo ułatwi późniejsze jego wyszukanie[2].

Zatwierdzenie wszystkich dokonanych wpisów w oknie nadania plikowi nazwy poprzez naciśnięcie (myszką!) "klawisza" **Ok** powoduje zapisanie pliku pod wskazaną nazwą na dysku oraz przemianowanie odpowiedniego okna w samym edytorze. Wybranie klawisza **Anuluj** pozwala zrezygnować z nadania dokumentowi nazwy.

Jeśli dokument ma już nadaną nazwę – możesz nakazać jego zapisanie na dysku znacznie szybciej i wygodniej, mianowicie naciskając kombinację klawiszy **Ctrl–S**. Możesz także wybrać pozycję menu **Plik** i w niej pozycję **Zapisz**, albo – jeszcze prościej – zgodzić się na sugestię systemu (naciskając klawisz **Enter** lub wskazując myszką klawisz **Yes**[3]), który w momencie kończenia pracy z edytorem informuje, że plik z tekstem został zmieniony, w związku z czym musisz odpowiedzieć na pytanie, czy dokonane zmiany należy zapisać na dysku,

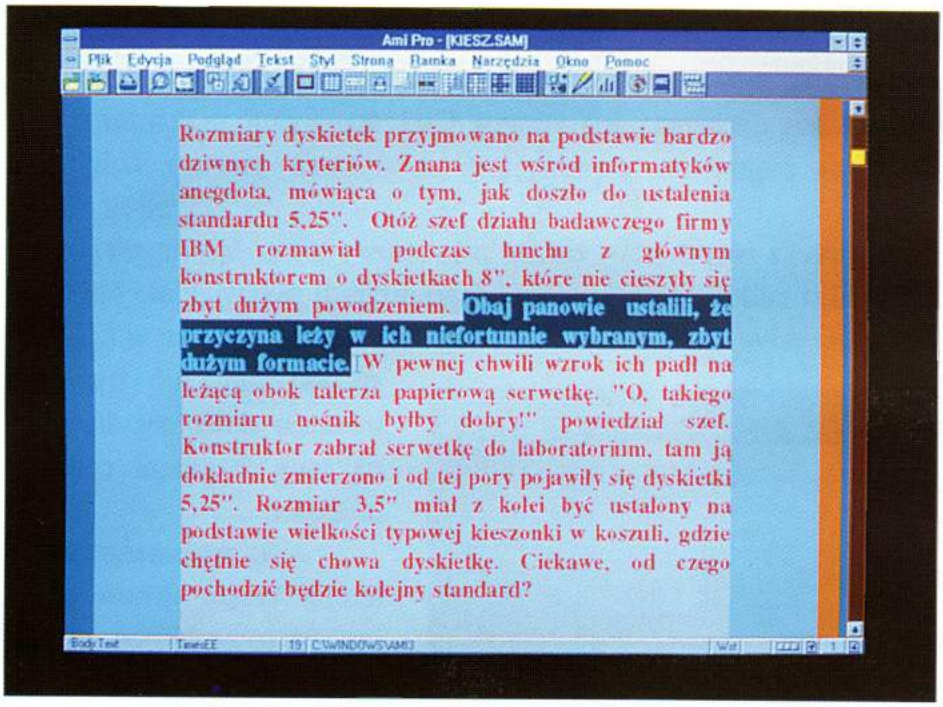
---

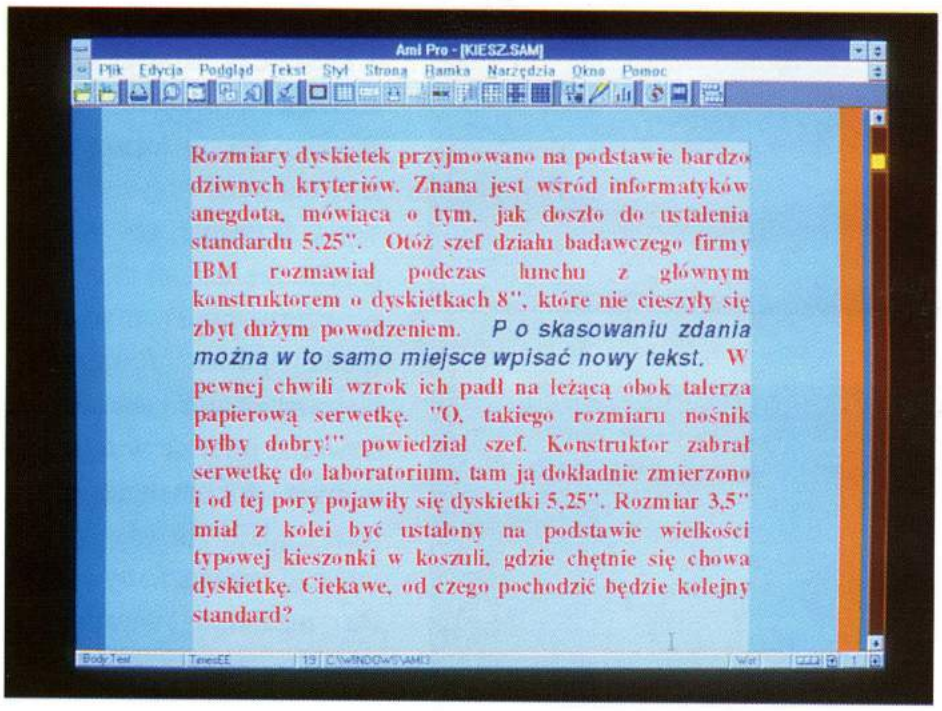
[1] W momencie rozpoczęcia pisania nowego tekstu Ami udostępnia użytkownikowi do redagowania tego dokumentu nowe okno o nazwie **Bez tytułu.** Jeśli takich nowych dokumentów redaguje się kilka – są one nazywane odpowiednio **Bez tytułu 1, Bez tytułu 2** itd. Jednak w tej postaci dokument nie może znaleźć się na dysku, dlatego operacja zapisu dokumentu musi być poprzedzona operacją nadania mu nazwy.

[2] Doświadczony użytkownik może zresztą uzyskać znacznie więcej, na przykład może zabezpieczyć dostęp do dokumentu specjalnie wybranym hasłem, nie będziemy jednak tych wszystkich dodatkowych możliwości wymieniali.

[3] Mimo polonizacji programu Ami Pro niektóre okienka dialogowe pokazują słowa angielskie, gdyż są obsługiwane przez system MS Windows, a ten w standardowej wersji prowadzi dialog z użytkownikiem w języku angielskim.

czy skazać na zapomnienie. Możesz też w tym momencie wycofać się z decyzji zakończenia pracy i powrócić do edycji pliku. Opcje te są podobne do opisanych w edytorze **NC.**





Bardzo wygodną usługą, jaką może oddawać każdy bardziej rozbudowany edytor, jest możliwość wykonywania pewnych operacji na całych (dowolnie obszernych) **zaznaczonych** fragmentach tekstu (patrz fotografia obok). Zaznaczywszy określony fragment tekstu można go następnie w całości **przenieść** w inne miejsce, **skopiować** albo **zastąpić** innym tekstem (porównaj fotografię u dołu). W zaznaczonym tekście można także zmienić kształt i wielkość używanej czcionki albo rozmieszczenie tekstu. O tym będzie jednak mowa niżej, w tej chwili trzeba natomiast przedstawić metody służące do zaznaczania wybranych fragmentów tekstu.

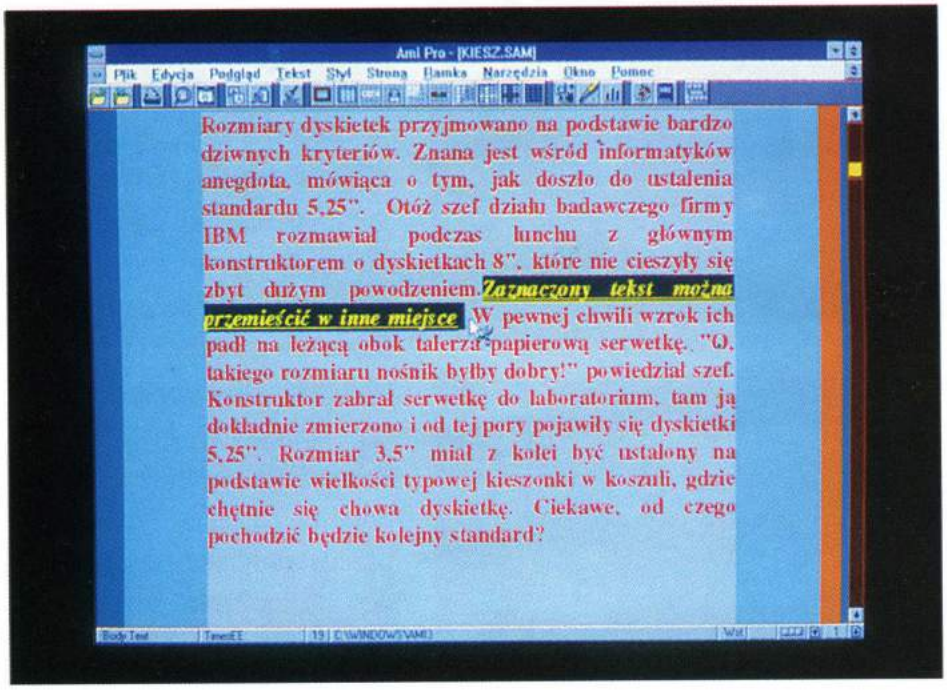
Najprostszą metodą zaznaczenia wybranego fragmentu tekstu jest "przejechanie" przez ten tekst myszką z przyciśniętym lewym klawiszem. Po prostu ustawia się wskaźnik myszy na początku zaznaczanego fragmentu tekstu, naciska się klawisz i ciągnie się mysz do chwili, aż zaznaczany fragment tekstu (podświetlany na ekranie w trakcie zaznaczania) nie obejmie całego zaznaczanego fragmentu. Wtedy zwalnia się przycisk myszy i tekst pozostaje zaznaczony – gotowy do wykonania na nim w całości jakiejś operacji.

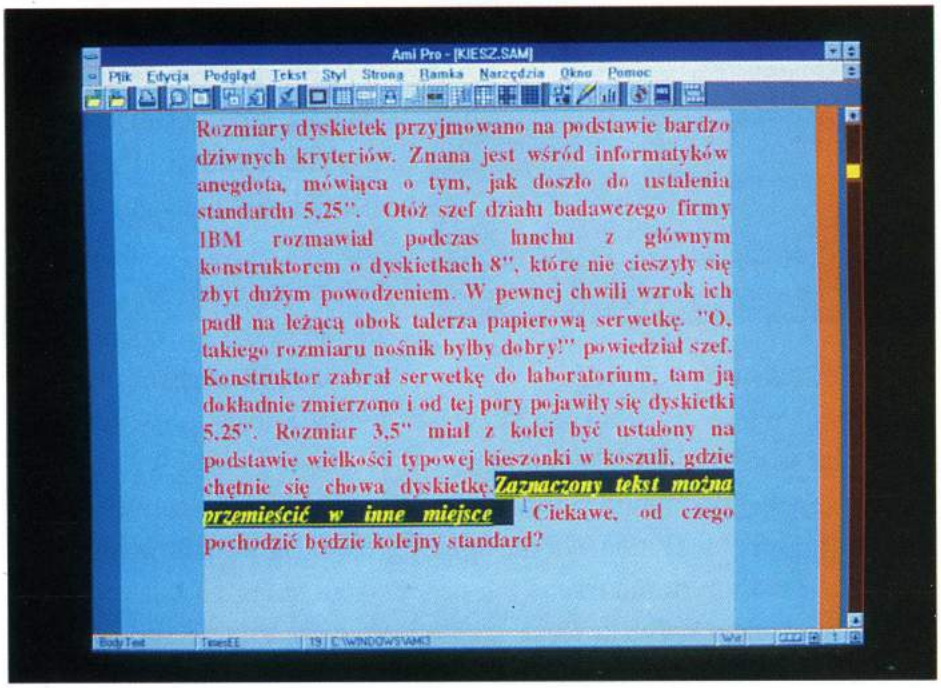
Opisany wyżej sposób zaznaczania tekstu wymaga pewnej wprawy, zwłaszcza jeśli trzeba precyzyjnie ustawić koniec zaznaczanego fragmentu. Dlatego obok opisanej techniki zaznaczania tekstu można korzystać z pewnych ułatwień, dzięki którym nie trzeba tak precyzyjnie manipulować myszką, a granica zaznaczanego tekstu będzie zawsze pokrywała się z końcem wyrazu, zdania lub akapitu. Wystarczy w tym celu skorzystać z następującej konwencji: Po wskazaniu myszką jakiegokolwiek wyrazu **dwukrotne** naciśnięcie klawisza myszki powoduje zaznaczenie **całego wyrazu** niezależnie od tego, jakie położenie wewnątrz wyrazu zajmował kursor myszki. Po takim dwukrotnym

naciśnięciu klawisza dalsze ciągnięcie myszki powoduje zaznaczanie (podświetlanie) tekstu, przez który wędruje myszka **całymi wyrazami**, a nie pojedynczymi literami.

Jeszcze "grubsze" zaznaczanie tekstu uzyskać można za pomocą klawisza **Ctrl.** Pojedyncze tupnięcie myszką wykonane w czasie, kiedy na klawiaturze naciśnięty jest klawisz **Ctrl** spowoduje zaznaczenie całego **zdania** (ciągnąc myszkę możesz w ten sposób zaznaczać kolejne całe zdania), zaś **dwukrotne** tupnięcie myszką w czasie, gdy naciśnięty jest klawisz **Ctrl,** spowoduje zaznaczenie całego **akapitu** (tu także możliwe jest ciągnięcie, powodujące zaznaczanie szeregu kolejnych akapitów)[1].

Po zaznaczeniu starego tekstu możesz natychmiast zacząć pisać nowy tekst. Już pierwszy naciśnięty klawisz spowoduje, że stary tekst zniknie z ekranu, a na jego miejscu pojawiać się będzie – w miarę pisania – nowy tekst. Ponieważ edytor automatycznie formatuje tekst – możesz w opisany sposób zamienić fragment tekstu o dowolnej długości na inny fragment o innej długości.





Jeśli zaznaczony tekst trzeba tylko **skasować** – możesz to zrobić naciskając klawisz **Del.** Jeśli natomiast chcesz zaznaczony tekst **przemieścić** w inne miejsce – wystarczy ustawić gdziekolwiek w obrębie zaznaczonego tekstu kursor myszy, nacisnąć przycisk myszy **i trzymać go**. Kursor przybierze kształt nożyczek z czerwonym markerem (patrz górna fotografia). Naprowadzając marker na dowolne miejsce w redagowanym dokumencie ustalasz, gdzie należy przemieścić zaznaczony tekst. Po ustaleniu właściwego miejsca **puszczasz przycisk myszy.** Zaznaczony tekst zniknie z miejsca, gdzie był pierwotnie i pojawi się w punkcie, w którym był "nożyczkowy" kursor w momencie zwolnienia klawisza myszy (ilustruje to dolna fotografia). Jeśli opisane czynności

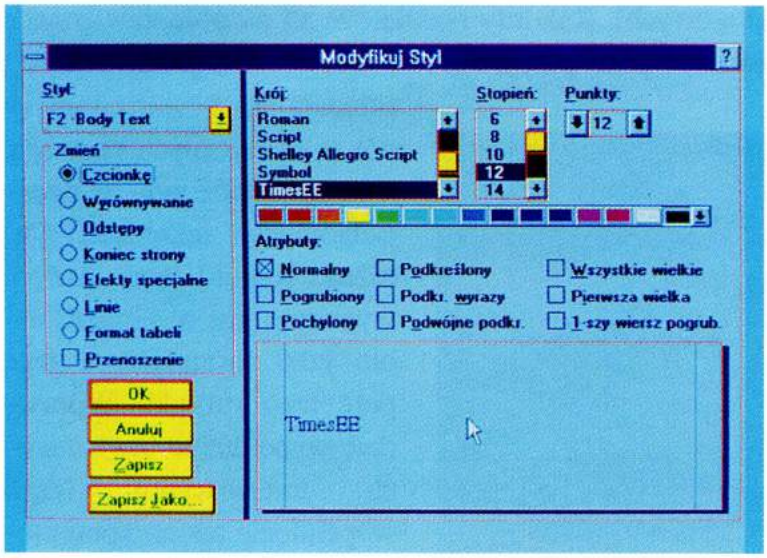
[1] Opisane wyżej sposoby zaznaczania tekstu nie wyczerpują wszystkich możliwości. Często wygodniej jest zaznaczyć wymagany fragment tekstu przesuwając po nim kursor z równoczesnym naciśnięciem klawisza **Shift** (ta technika nie wymaga odrywania rąk od klawiatury w poszukiwaniu myszki), są zresztą także i inne sposoby. Jednak znacznie ważniejsze, niż poznawanie kolejnych technik **zaznaczania** tekstu, jest omówienie tego, **co z zaznaczonym tekstem można zrobić.**

wykonasz z naciśniętym klawiszem **Ctrl** – zaznaczony tekst zostanie **skopiowany,** to znaczy pojawi się w nowym miejscu nie znikając z miejsca, z którego go "pobrano".

Operując na całych blokach zaznaczonego tekstu, narażasz się, że jeden nieostrożny ruch może spowodować skasowanie dużej partii pracowicie napisanego tekstu lub może pomieszać fragmenty tekstu w sposób trudny do późniejszego "odkręcenia". Bez obaw! Każdy dobry edytor daje możliwość **cofnięcia** omyłkowo wydanego polecenia (lub nawet kilku poleceń). W Ami Pro wystarczy nacisnąć klawisze **Ctrl–Z** i edytor natychmiast odtwarza poprzednią (sprzed nieopatrznie wykonanego "cięcia") postać tekstu.

W odróżnieniu od omówionych wyżej prostych edytorów, umożliwiających tylko pisanie i poprawianie **treści** tekstu, procesor Ami Pro pozwala w bardzo wyrafinowany sposób wpływać także na **formę** tekstu. Służy do tego szereg mechanizmów, z których najważniejszym jest pojęcie **stylu.**

Styl obejmuje wiele aspektów opisujących formę tekstu, takich jak **ustawienie marginesów, krój i wielkość czcionki, sposób wyrównywania tekstu, odstępy** i wiele innych. Z każdym dokumentem pisanym w Ami Pro związana jest cała **lista stylów**, określająca, jakie style są w tym dokumencie używane i co one oznaczają. Możesz wybrać jedną z dostarczonych przez producentów list stylów, ale możesz także wprowadzić własną listę stylów, dostosowaną do swoich upodobań i przyzwyczajeń. Zanim podam więcej szczegółów wyjaśnię, dlaczego operuje się całą **listą** stylów, a nie jednym stylem. Wynika to z faktu, że podczas pisania dokumentu zwykle w różnych jego częściach potrzebujemy różnych stylów dla zaakcentowania roli tych części[1]. Dlatego każda lista stylów obejmuje kilka różnych stylów, które można dowolnie przypisywać poszczególnym **akapitom**[2] dokumentu. Najprościej jest skorzystać z "przycisku" zmiany stylu, znajdującego się w lewym dolnym rogu okienka Ami Pro. Przycisk ten opisany jest aktualnie wybraną nazwą stylu (najczęściej jest to styl **Tekst** używany w Ami Pro do pisania większości akapitów normalnego tekstu). Jednak naciśnięcie tego przycisku (myszką) powoduje rozwinięcie menu, którego pozycjami są wszystkie aktualnie dostępne style. Przykładowo w liście stylów **dokument.sty** poza stylem **Tekst** mogą znajdować się style: **Lista, Tytuł, Nagłówek, Przypis,** za pomocą których można budować różne wyliczenia i numerowane listy, tworzyć tytuły i nagłówki oraz formatować przypisy[3].



Aktualnie używany styl może być modyfikowany. W tym celu należy nacisnąć klawisze **Ctrl–A.** Ukaże się wtedy pokazane na fotografii okno dialogowe, w którym za pomocą licznych suwaków, przycisków i paneli sterujących można ustalić praktycznie wszystkie ważniejsze parametry stylu akapitu. Wybrawszy w tym oknie odpowiadające nam wielkości czcionki, sposoby wyrównywania tekstu, odstępy i inne elementy stylu – możemy nakazać stosowanie zmodyfikowanego stylu w obecnie pisanym dokumencie naciskając klawisz

---

[1] Na przykład w typowym dokumencie występują różne nagłówki, tytuły, stopki, przypisy, tablice, trzeba w innym formacie wpisać datę, w innym adres, a w jeszcze innym wyliczane w formie oddzielnych punktów pozycje inwentarza. Każdy z tych elementów wymaga innego stylu: tytuły muszą być pisane większą czcionką i ustawione na środku stronicy, data powinna być dosunięta do prawego marginesu, wyliczane pozycje pisze się z wcięciem z lewej strony i kolejno numeruje...

[2] Akapit tworzy porcja tekstu rozciągająca się od początku akapitu do najbliższego miejsca, w którym **wymuszono** przejście do nowej linii naciśnięciem klawisza **Enter.** Początek akapitu dla pierwszego akapitu pokrywa się z początkiem dokumentu, natomiast dla wszystkich następnych przypada w miejscu zakończenia poprzedzającego akapitu.

[3] Wybór stylu dla aktualnie redagowanego akapitu może być także wykonany za pomocą klawiszy funkcyjnych, których numery towarzyszą nazwom odpowiednich stylów w menu. Przykładowo styl **Tekst** może być uzyskany poprzez naciśnięcie klawisza **F2,** a styl **Tytuł** uzyskać można za pomocą klawisza **F8.**

Ok. Edytor zmieni wtedy formę wszystkich fragmentów tekstu, które były pisane rozważanym stylem – także i tych, które napisano wcześniej, przed modyfikacją stylu!

Jeśli zmodyfikowanego stylu chcemy użyć w przyszłości – trzeba nakazać zapisanie zmodyfikowanej listy stylów na dysku. Wykonuje się to za pomocą wyboru z menu opcji **Styl** i pozycji **Zapisz nową listę stylów.**

Procesor tekstu Ami Pro ma liczne dalsze możliwości, jednak nie będziemy ich tu dyskutować, gdyż nie można zmienić tej książki wyłącznie w podręcznik używania edytorów. W skład oprogramowania komputera wchodzą bowiem liczne dalsze, użyteczne programy, którym teraz z kolei poświęcimy uwagę.

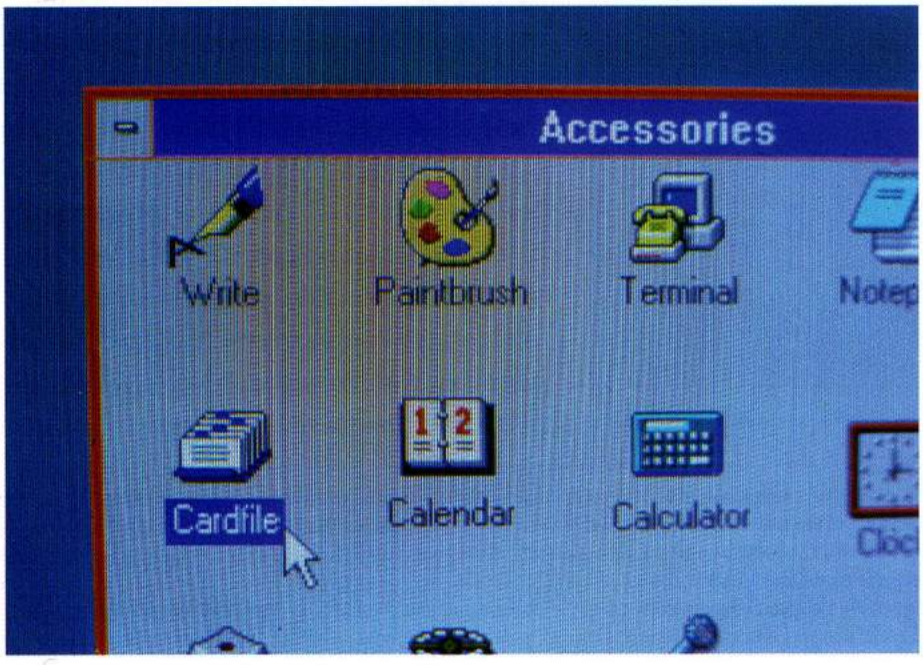
3.6. Systemy zarządzania bazą danych

3.6.1. Uwagi ogólne

Gromadzenie i wyszukiwanie danych jest bardzo pracochłonne i zabiera wiele czasu. Nic więc dziwnego, że do tego celu zastosowano komputery wykorzystując ich możliwości pamiętania dużych ilości informacji i szybkiego ich przetwarzania. Podstawą zastosowań komputerów w wielu dziedzinach są właśnie **programy zarządzania bazami danych**[1] (ang. **DBMS** – *Data Base Menagement Systems*). Programy te umożliwiają *wprowadzanie* danych, *przechowywanie* ich (np. na dysku), *uaktualnianie*, *sortowanie* i *wyszukiwanie*.

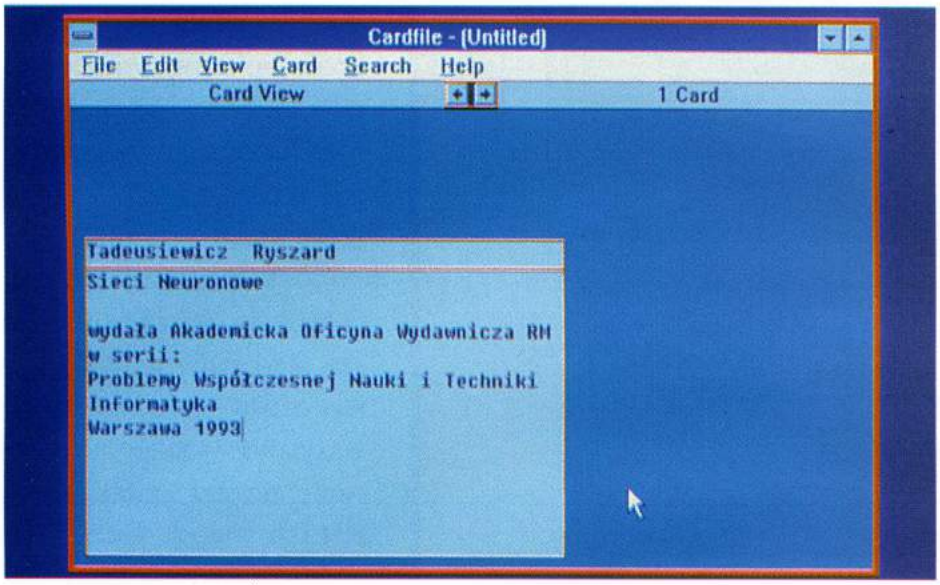
3.6.2. Kartotekowe bazy danych

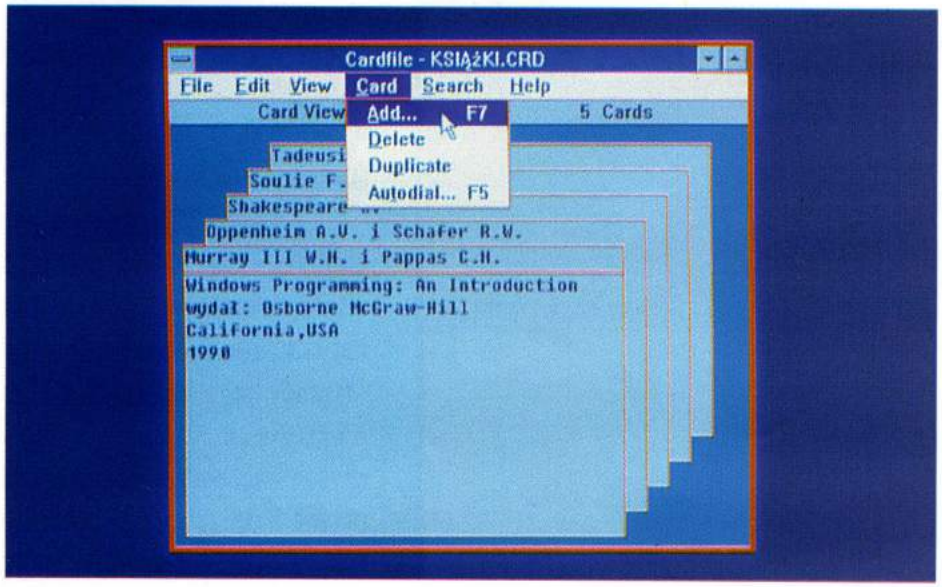
Najprostszy rodzaj bazy danych to baza danych **typu kartotekowego.** Składa się ona zawsze z **programu zarządzającego** i ze **zbioru danych**. Zbiór danych jest przechowywany na dysku jako **plik**. Plik ten składa się z odrębnych dokumentów (**rekordów**) o identycznej strukturze, lecz o różnej zawartości (np. zbiór książkowych kart katalogowych). Rekordy podzielone są na **pola** zawierające **tekst, liczby** lub inne dane (np. **daty**). Każdemu polu przypisuje się **nazwę**. Wszystkie rekordy składają się z takich samych pól o identycznych nazwach, ale **zawartość pól** w poszczególnych rekordach jest różna. Rekord bazy danych wygodnie jest wyobrazić sobie jako formularz, a jego pola jako rubryki. Wprawdzie wypełnienie bazy danych informacjami (czyli wypełnienie takich komputerowych formularzy) jest pracochłonne, lecz ich dalsze przetwarzanie jest już proste i szybkie.

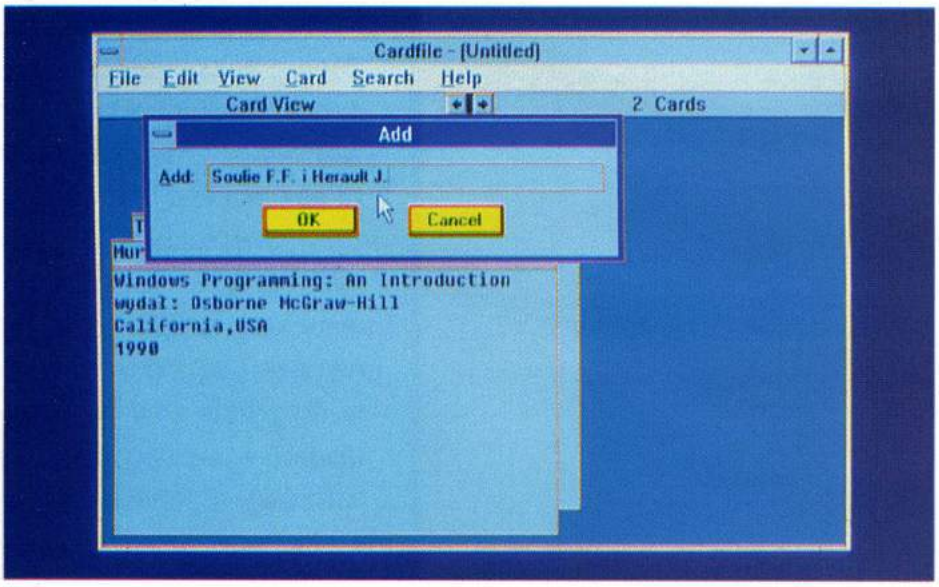


Zaprogramowanie i instalacja najprostszej bazy

[1] Uważa się, że początek nowoczesnych systemów zarządzania bazami danych dała publikacja **Edgara F. Codda** z lipca **1970** roku, zaś wszystkie węzłowe etapy rozwoju tej idei także wiążą się z nazwiskiem tego badacza.



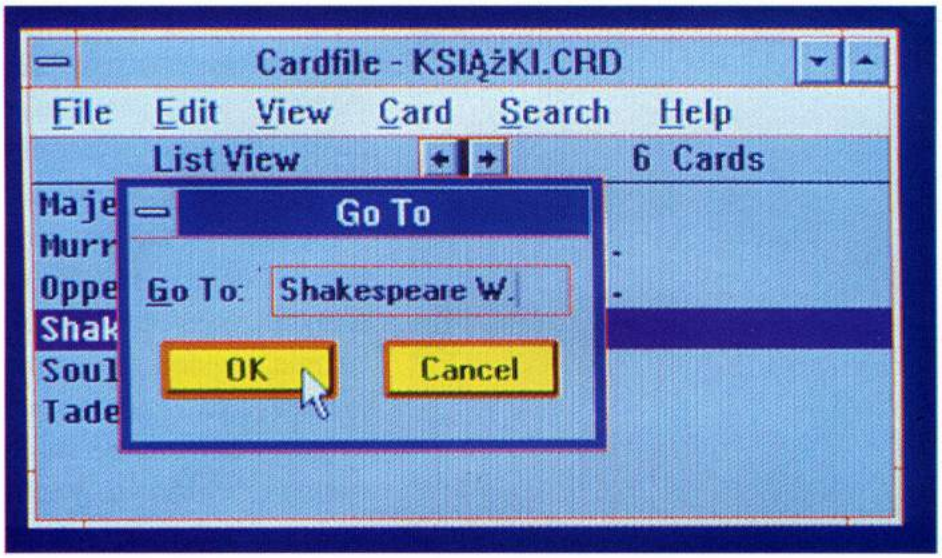


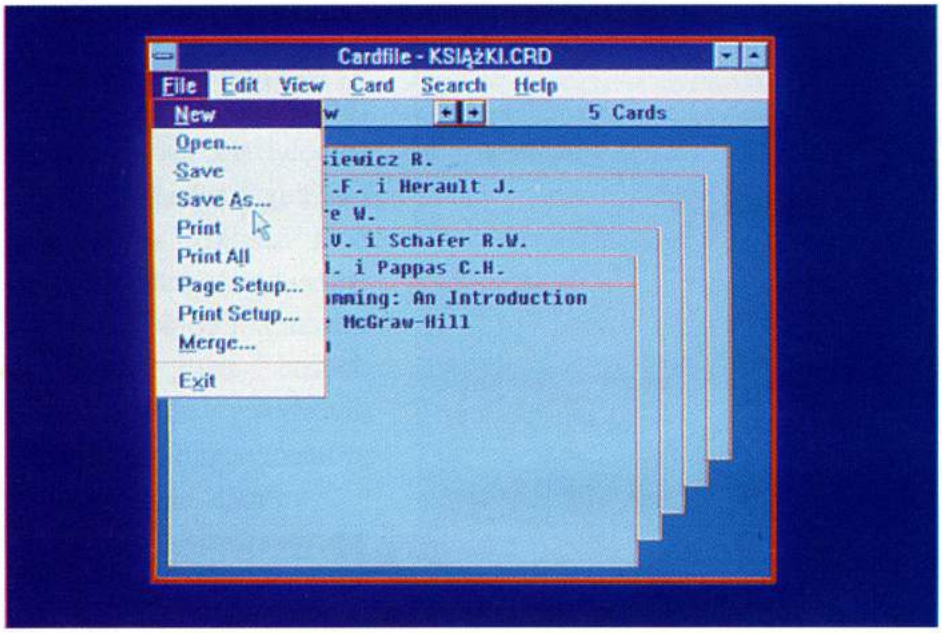


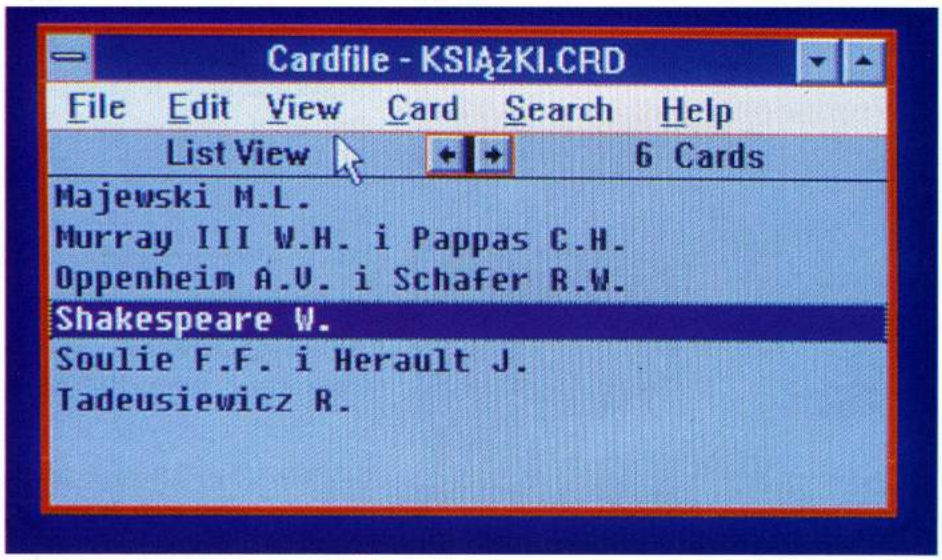
danych jest wykonalne nawet dla zupełnego laika. Poznamy to na przykładzie bazy danych **CardFile** "wbudowanej" w system Windows. Wywołanie tej bazy następuje po dwukrotnym "tupnięciu" na jej "ikonie", widocznej na fotografii na poprzedniej stronicy. Otwiera się wtedy okienko, w którym widoczny jest pojedynczy pusty rekord (patrz obok). Rekord ten składa się tylko z dwóch pól. Pierwsze pole (górny pasek na obrazku) stanowi tak zwany **indeks** (według zawartości tego pola rekordy są automatycznie sortowane i wyszukiwane). Drugie pole, widoczne niżej, służy do przechowywania treści gromadzonych informacji.

Praca zaczyna się od **wprowadzanie danych do bazy**. Przykładowo, wykorzystując bazę do zrobienia katalogu posiadanych książek można wpisywać nazwiska autorów w pasku indeksu, a tytuły książek i ewentualne uwagi na ich temat – w polu treści rekordu. Pierwszą kartę można natychmiast wypełniać – wystarczy wskazać myszką (dwa razy tupnąć!), gdzie się chce wpisywać tekst i już można zanotować pierwszego autora i pierwszy tytuł.

Dalsze rekordy tworzy się równie łatwo korzystając z pozycji menu **Card** (patrz fotografia wyżej), w którym są pozycje umożliwiające dodanie nowego rekordu (**Add**), usuwanie zbędnych rekordów (**Delete**) lub ich powielania (**Duplicate**). Korzystając z tych pozycji, można uzupełniać i aktualizować bazę danych. Zwróć uwagę, że podczas dodawania rekordu program najpierw pyta o **indeks** (nazwisko autora książki), a dopiero potem pozwala wypełnić kartę (widok ekranu podczas





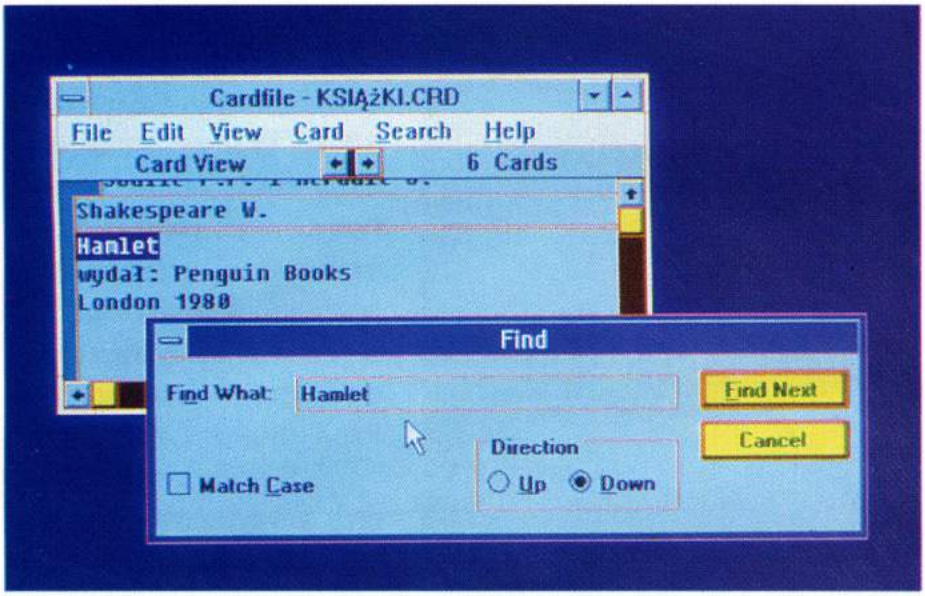


wprowadzania nowego rekordu pokazałem na fotografii na poprzedniej stronicy). Natomiast chcąc usunąć rekord z bazy – trzeba go **wskazać**.

Wskazanie rekordu polega na "tupnięciu" myszą jego indeksu. Jeśli kart jest mało – widoczne są wszystkie indeksy i wskazanie nie jest trudne. Kiedy jest ich więcej, można korzystać z widocznych u góry strzałek i "przewijać" kartotekę względnie można skorzystać z menu **Search,** a w nim z opcji **Go To** (patrz górna fotografia).

Gdy już w bazie znajdzie się wystarczająco wiele rekordów – należy ją zapisać na dysku, żeby nie utracić jej zawartości po wyłączeniu maszyny. Korzysta się wtedy z pozycji menu **File** (patrz środkowa fotografia) i pozycji **Save as** (jeśli baza jest nowa i trzeba jej nadać nazwę) lub **Save.** Taką zapisaną na dysku bazę można potem wprowadzić do programu za pomocą opcji **Open.**

Wyszukiwanie danych w bazie jest znacznie ułatwione dzięki temu, że rekordy są posortowane według indeksów (w przykładzie – alfabetycznie wg autorów), a ponadto można na życzenie zobaczyć (menu **View**) ich zbiorczą **listę** (patrz fotografia powyżej), na której też można wskazać potrzebny rekord. Co więcej, dostępne są dwa sposoby automatycznego wyszukiwania potrzebnych danych (menu **Search**) – według indeksów (**Go To** – pokazne na górnej fotografii) i według treści (**Find** – patrz fotografia na następnej stronicy). Możesz więc wyszukać dane o książce, podając nazwisko autora (lub kilka pierwszych liter



nazwiska) względnie podając dowolną daną, jaka wystąpiła w treści opisu (np. jedno słowo tytułu lub rok wydania).

Baza **Cardfile** ma jeszcze liczne dalsze udogodnienia (np. możliwość włączania rysunków), nie będę ich jednak tu omawiał.

Opisana baza danych miała z góry narzuconą strukturę rekordów, dzięki czemu była prosta w użyciu, ale uboga. Większe systemy zarządzania bazami danych są zwykle programami ogólnego przeznaczenia i dopiero użytkownik przed uruchomieniem "dopasowuje" program do swoich potrzeb. Między innymi oznacza to, że przed wprowadzaniem danych należy ustalić **strukturę rekordu.** Podaje się wtedy, z ilu i z jakich pól będzie się składał rekord oraz nadaje się nazwy tym polom. Konieczne jest też zdefiniowanie **indeksów**, według których informacje będą wyszukiwane i porządkowane, a także **raportów**, czyli sposobów wyświetlania i drukowania informacji.

3.6.3. Relacyjne bazy danych i program dBase

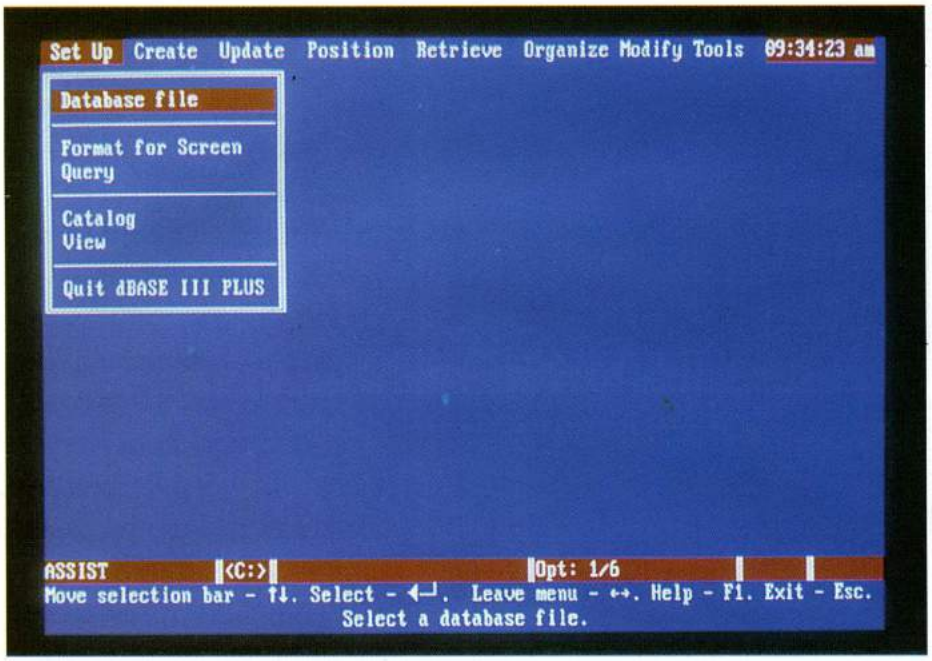
W profesjonalnych zastosowaniach z reguły stosowane są tzw. **relacyjne bazy danych.** Są one z założenia wieloplikowe, a każdy z plików ma inną strukturę rekordów, w których występują jednak pola wspólne. Pomiędzy rekordami w poszczególnych plikach jest określona pewna relacja, dzięki czemu komputer może wyszukać w różnych plikach dane dotyczące tego samego zagadnienia. Bazę relacyjną można sobie wyobrazić w uproszczeniu jako zespół tablic z danymi, powiązanych między sobą przez kolumny, w których występują wspólne dane. Odszukanie określonego rekordu (wiersza) w jednej tablicy pozwala, poprzez kolumny zawierające wspólne dane, wybrać odpowiedni rekord w innej tablicy. Na przykład odnajdując dane określonego pracownika w pliku zawierającym dane kadrowe, można automatycznie (poprzez numer ewidencyjny pracownika) odnaleźć jego dane na liście płac w pliku należącym do księgowości.

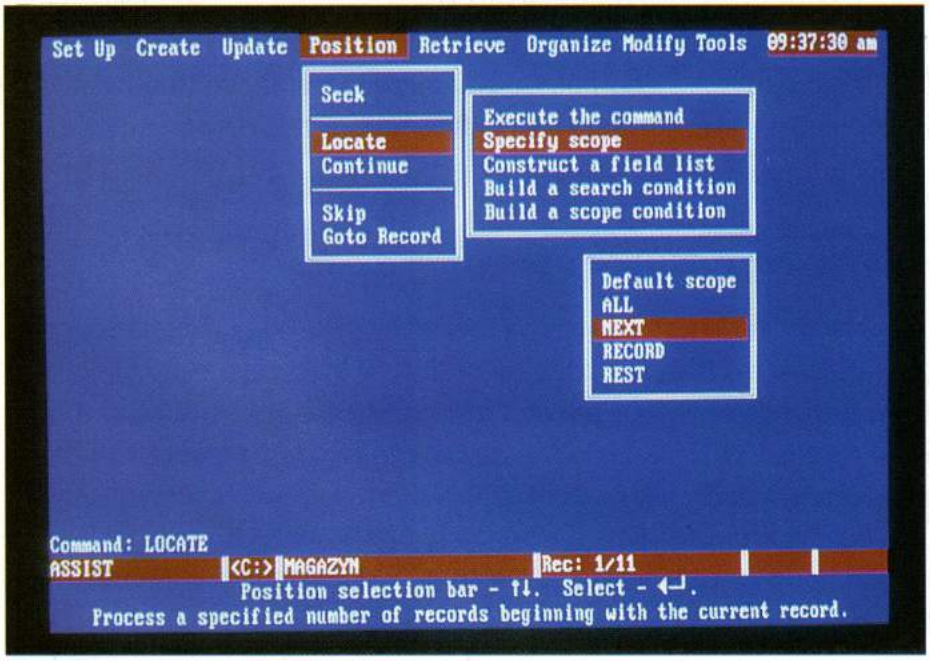
Zaletą relacyjnych baz danych jest uniezależnienie formy, w jakiej wprowadza się informacje od sposobu ich dalszego użytkowania. Bazy takie pozwalają nawet na zmiany struktury rekordu podczas wyświetlania informacji w stosunku do tej struktury, jaką komputer przechowuje w swojej pamięci dyskowej. Oznacza to, że każdy użytkownik może korzystać z tej samej bazy danych otrzymując z niej dane indywidualnie dla niego wyszukane i indywidualnie dla niego uporządkowane w najwygodniejszej formie. Można więc przyjąć, że baza jest jedna, ale każdy użytkownik inaczej ją widzi i inaczej wykorzystuje.

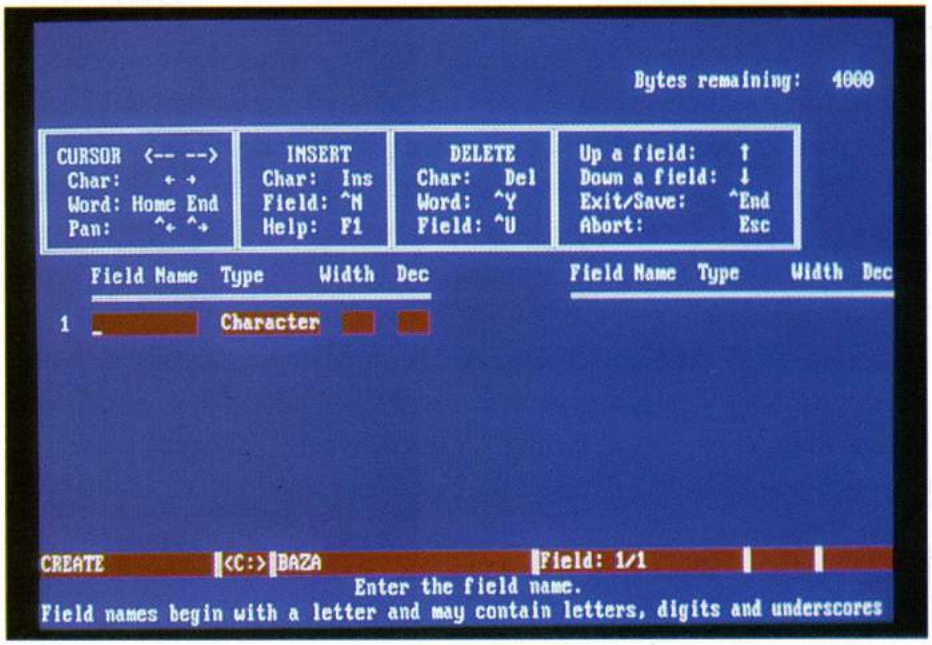
Najbardziej znany i najchętniej używany programy tworzący i obsługujący relacyjne bazy danych to **dBase**[1]. Program ten wytworzyła amerykańska firma *Ashton–Tate* – początkowo dla komputerów *Jet Propulsion Laboratory* w ramach programu kosmicznego **Mariner**. "Kariera" programu **dBase**[2] zaczęła się w momencie opracowania wersji **dBase III** dla mikrokomputerów pracujących

[1] Rozmiar sukcesu programu **dBase** najlepiej ocenić na podstawie danych dotyczących sprzedaży: otóż w samym tylko **1990** roku firma ***Ashton–Tate*** zanotowała 82 miliony dolarów czystego przychodu za ten program! Wkrótce potem została jednak wykupiona przez firmę ***Borland*** i od tej chwili dalsze produkty ukazują się pod tą właśnie nazwą.

[2] Za twórcę programów **dBase** uważa się **Wayne Ratliffa,** który zadebiutował w **1979** roku programem o nazwie VULCAN, w którym zawarte były wszystkie najważniejsze koncepcje, leżące dziś u podstaw nowoczesnych systemów baz danych.





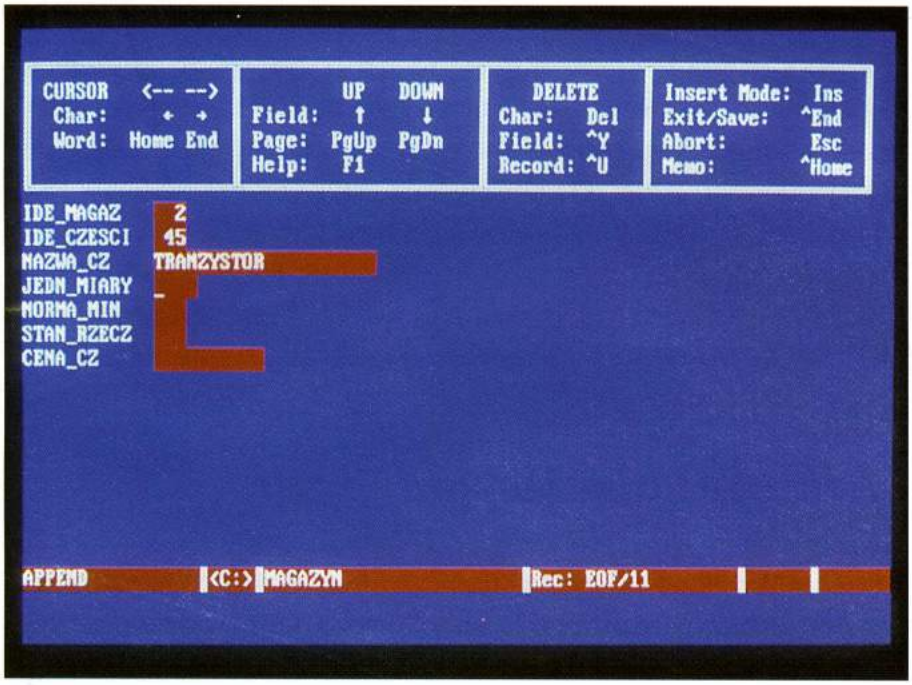


pod kontrolą systemu **MS-DOS**. Potem powstawały kolejne wersje (**dBase III+, dBase IV**), zyskując licznych zwolenników na całym świecie, chociaż oczywiście są w użyciu i inne programy (np. **Sybase** lub sygnowany sympatyczną lisią mordką program **Fox Pro**).

Pierwszą czynnością przy korzystaniu z programu **dBase** jest wybór pliku zawierającego bazę danych (patrz fotografia obok). Dalsza komunikacja pomiędzy operatorem a programem **dBase** odbywa się przez wybieranie poleceń z usłużnie podsuwanych menu (patrz fotografia obok) lub za pomocą komend wprowadzanych z klawiatury, które są natychmiast wykonywane.

System **dBase** ma także wbudowany własny język programowania umożliwiający wykonywanie na zawartości plików całych programów. Nie jest to jednak na ogół potrzebne, bo nawet korzystając tylko z podstawowych możliwości programu **dBase**, możesz swobodnie zdefiniować sobie strukturę rekordu (patrz fotografia obok), a także ustalić podstawowe własności tworzonej bazy danych. Podpowiedzi usłużnie podsuwane przez program u góry i u dołu ekranu bardzo ułatwiają tę pracę.

Po zdefiniowaniu struktury bazy i struktury rekordu możesz zacząć



wypełniać ją konkretnymi wiadomościami. Wprowadzasz wówczas dane do bazy korzystając po prostu z ekranowego formularza i przeprowadzasz potrzebne operacje wybierając odpowiednią opcję z gotowego menu. Także i tu towarzyszą Ci podpowiedzi systemu (patrz fotografia obok).

Program **dBase** jest bardzo wygodny i pożyteczny, prawdziwy luksus stanowią jednak dopiero specjalne własne programy użytkowe napisane specjalnie dla Ciebie za pomocą języka **dBase** (albo pasującego do tych samych zastosowań języka **Clipper).** Programy takie mogą być lepiej dostosowane do Twoich potrzeb, na przykład mogą zawierać własne menu, z których będzie Ci wygodniej korzystać. Możesz zatem nie znać rozkazów programu **dBase** i mimo to używać takiego programu, ograniczając się wyłącznie do korzystania z gotowego menu, utworzonego przez programistę tworzącego "pod" **dBase** bazę dla konkretnego Twojego zastosowania. Jest to wygodne, tylko zwykle najtrudniej jest tak opisać swoje potrzeby, żeby informatyk naprawdę zbudował bazę **idealnie** pasującą do potrzeb. To jest jednak całkiem osobny temat, którym zajmiemy się w innym miejscu książki.

3.6.4. Wielodostępne banki danych i systemy ekspertowe

Wśród innych istniejących typów systemów dostarczających informacje warto wymienić **bazy wielodostępne** i **systemy ekspertowe.** Wykraczają ona poza zakreślone wyżej ramy typowych baz danych i są chwilowo (w Polsce) raczej egzotycznym dodatkiem do tematyki lokalnych i pozbawionych "inteligencji" baz danych, znacznie częściej spotykanych w naszej codziennej rzeczywistości – ale i znacznie mniej ciekawych.

Wielodostępne banki danych to duże bazy, które mogą być jednocześnie przeglądane lub modyfikowane przez wielu użytkowników pracujących z różnych terminali lub komputerów połączonych w sieć[1]. W wielodostępnych bankach danych musi być zastosowany mechanizm blokowania rekordów, polegający na tym, że gdy jeden użytkownik zmienia rekord, to rekord ten musi być zablokowany dla innych użytkowników. Stosowane są także zabezpieczenia w postaci haseł oraz udostępniania konkretnym użytkownikom tylko określonych pól.

Specjalnym rodzajem nowoczesnej bazy danych jest tak zwany **system ekspertowy.** Jest to na ogół nowoczesny program[2], oparty na osiągnięciach tak zwanej sztucznej inteligencji, który dzięki zgromadzonej **bazie wiedzy** oraz korzystając z mechanizmów **automatycznego wnioskowania** umożliwia uzyskiwanie komputerowych rad i opinii – podobnie jak w przypadku pytania eksperta

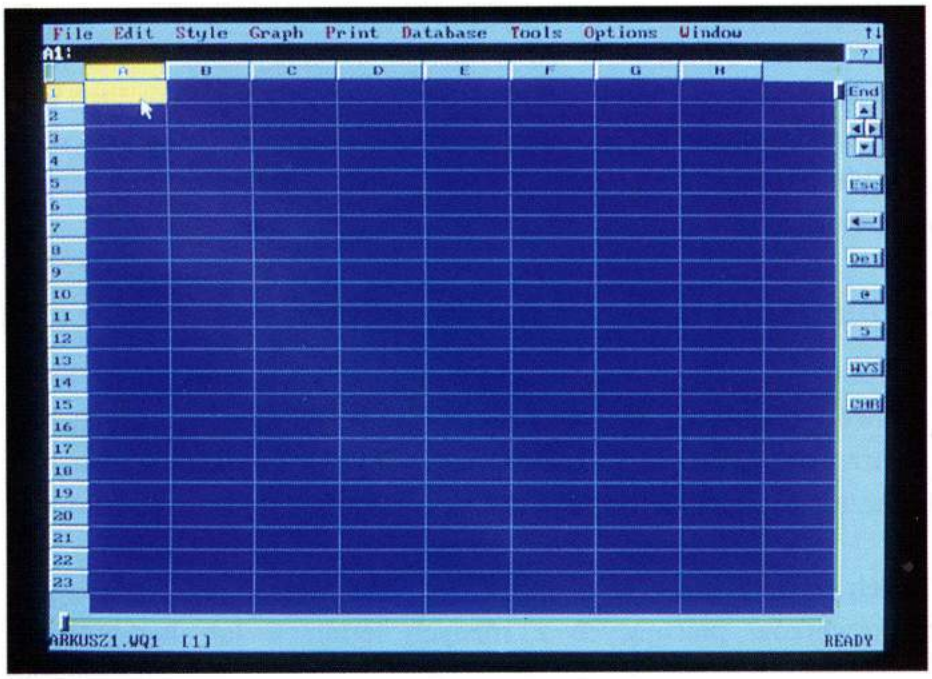
[1] W krajach o wysokim stopniu rozwoju teleinformatyki (np. Francja) dostępne są usługi, polegające na łączeniu się (nawet z domowego komputera lub specjalnego urządzenia zwanego MINITEL) za pośrednictwem sieci telefonicznej z dużym komputerem dysponującym jakąś konkretną bazą danych (np. na temat rozkładu jazdy i cen biletów kolejowych). Użytkownik może wtedy uzyskać potrzebną mu wiadomość (np. o potrzebnym połączeniu kolejowym) oraz wprowadzić własną informację (np. dokonując rezerwacji biletu).

[2] Jednym z bardziej znanych systemów omawianego tu typu był klasyczny system o nazwie **Mycin**, służący amerykańskim lekarzom jako system doradczy w zakresie stosowania antybiotykoterapii. Inne znane systemy ekspertowe budowano dla zastosowań w ekonomii: doradzają one, jak grać na giełdzie, obsługują handel zbożem, stanowią dla wielu "businessmanów" personalnego doradcę w sprawach księgowości małego przedsiębiorstwa, podatków lub okoliczności prawnych związanych z określoną sferą działalności gospodarczej.

(stąd nazwa). Systemy tego typu (nazywane niekiedy także systemami doradczymi) zbudowano już dla bardzo wielu dziedzin zastosowań, a zainteresowanie nimi wciąż rośnie[1], przeto obok programów do konkretnych celów budowane są programy, które stanowią "otoczkę" dla dowolnego zbioru informacji. Informacje, dzięki umieszczeniu w tej otoczce, tworzyć mogą specjalizowany system ekspertowy dla określonego konkretnego zastosowania. Amerykanie nazywają narzędzia programowe tego typu *shell* (dosłownie "muszla"). Z bardziej znanych programów tego typu wymienić można: **EXPERT-BASE** firmy *Intelligent Terminals, Inc.*; **EXSYS** firmy *Exsys, Inc.*; **1-st CLASS** firmy *Programs in Motion, Inc.*; **INSIGHT2+** firmy *Level 5 Research, Inc.*; **Micro Expert** firmy *McGraw-Hill, Inc.*; **PERSONAL CONSULTANT** firmy *Texas Instruments, Inc.* i wiele innych.

3.7. Arkusze kalkulacyjne

Arkusz kalkulacyjny[2], nazywany niekiedy arkuszem elektronicznym (*spreadsheet*), jest programem, pozwalającym szybko i wygodnie przeprowadzać obliczenia i dowolne kalkulacje typu opracowywania planów finansowych, kalkulacji inwestycji, budżetu, podatków i ogólnie mówiąc – dowolnej księgowości. Pierwszym szeroko znanym programem tego typu był **VisiCalc** dla komputerów *Apple*[3], na którym wzorowano wiele innych, na przykład *Contex MBA, SynCalc, Tiny Troll, OmniCalc, Multiplan, SuperCalc, Symphony, VP-Planer*. Programem, który zrobił w tym zakresie nie kwestionowaną karierę był słynny, będący dziś "klasykiem", program **1-2-3** firmy *Lotus*[4], sprzedany[5] w ilości ponad 10 milionów egzemplarzy. Innymi popularnymi programami tego typu są **Quatro Pro** firmy *Borland*[6] (który będzie używany w przykładach) albo skojarzony z systemem MS Windows znakomity arkusz o nazwie *Excel*.



---

[1] Dziedzina ta (jak wszystkie inne) nie jest też wolna od pewnych dziwactw, na przykład wielkim powodzeniem cieszy się w USA system ekspertowy, doradzający, jak ... napisać testament.

[2] Uważa się, że koncepcja arkusza elektronicznego pochodzi od **Dana Bricklina**, który w **1978** (jeszcze jako student *Harvard Business School*) skonstruował pierwszą wersję programu **VisiCalc**. Rozwój tej koncepcji z pomocą **Boba Frankstona** doprowadził do ogromnego sukcesu rynkowego, a obydwu twórców uczynił ludźmi bardzo zamożnymi.

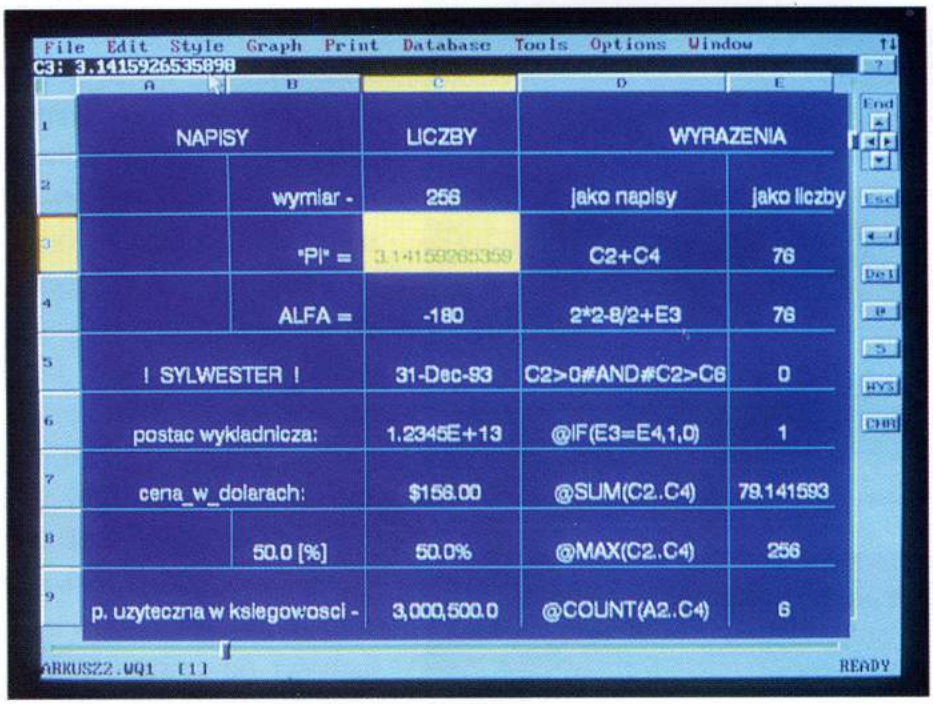
[3] Program arkusza kalkulacyjnego zależny jest oczywiście od komputera, dla którego jest opracowywany, natomiast sposób jego używania jest zawsze taki sam. Jest to jeden z powodów dużej popularności tego podejścia, wygodnego dla niewprawnego użytkownika komputera.

[4] Twórcą potęgi firmy *Lotus* był **Mitch Kapor**. W **1977** napisał on mały program o nazwie *Tiny Troll*. Programem tym zainteresował się **Bob Franston** (współtwórca sukcesu programu *VisiCalc*) i zachęcił Kapora do stworzenia programu bogatszego w możliwości i szybszego. **1-2-3** był odpowiedzią na to wyzwanie. Został zaprezentowany po raz pierwszy w styczniu **1983** roku i do dziś jest jednym z najlepiej sprzedawanych programów na świecie. Szacuje się, że przyniósł on około **500 milionów** dolarów dochodu.

[5] Arkusze kalkulacyjne należą do programów stosunkowo drogich – cena typowego pakietu wynosi od kilkuset do tysiąca dolarów. Jednak nie zawsze trzeba taki program kupować, żeby używać arkusza, dostępne są bowiem programy bezpłatne lub rozprowadzane za symboliczną opłatą (tzw. *shareware*), które mają wszystkie podstawowe cechy arkusza kalkulacyjnego. Są to między innymi **AsEasyAs, ASEasy, InstaCalc, ExpressCalc, FreeCalc, Goalseeker, AtLast, Wordplan, PowerSheet, PC Calc Plus, EZ Spread, TurboCalc.**

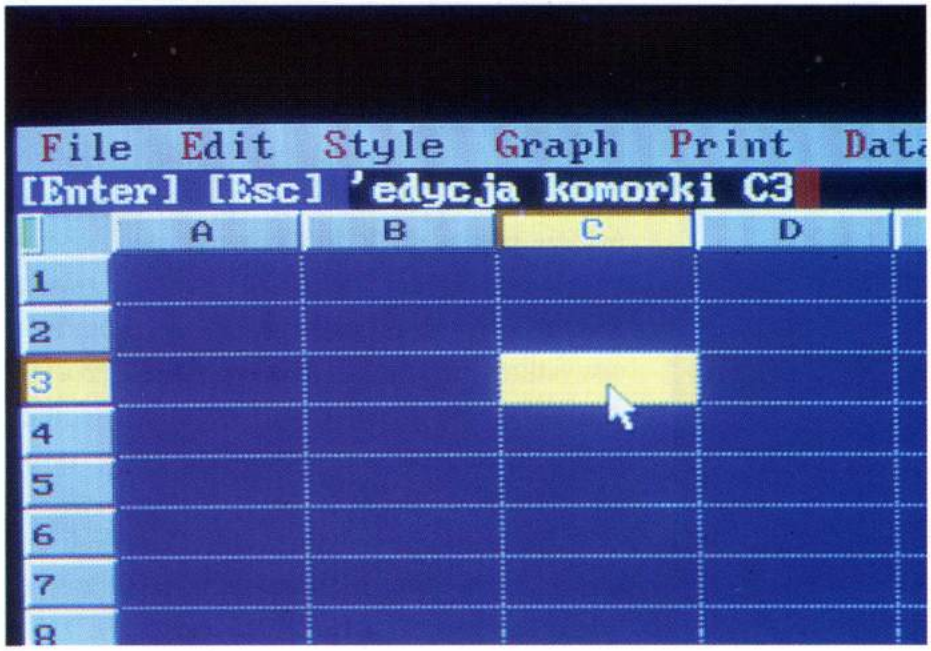
[6] Obecnie firma *Borland International Inc.* dzieli się na trzy działy: arkuszy kalkulacyjnych (ich ostatnim produktem jest właśnie wspomniany arkusz **Quatro Pro**), którym zarządza bezpośrednio szef i założyciel firmy **Steven Kahn,** dział baz danych kierowany przez **Boba Dickersona** i dział języków programowania, którym zarządza **Eugene Wang.**

Podstawowym elementem programu kalkulacyjnego jest duży **arkusz** podzielony na **wiersze** i **kolumny.** Prostokąty utworzone na przecięciu linii pionowych i poziomych nazywa się **komórkami.** Komórki te są identyfikowane za pomocą współrzędnych – zwykle kolumny oznacza się **literami** (jedną lub dwiema), wiersze zaś – **liczbami** (podobnie jak na szachownicy). Identyfikator złożony z litery (kolumny) i cyfry (wiersza) nazywać będziemy dalej krótko **numerem** komórki. Każda komórka może zawierać informacje: **liczby, napisy,** a także **wyrażenia** – jak pokazałem na fotografii obok.



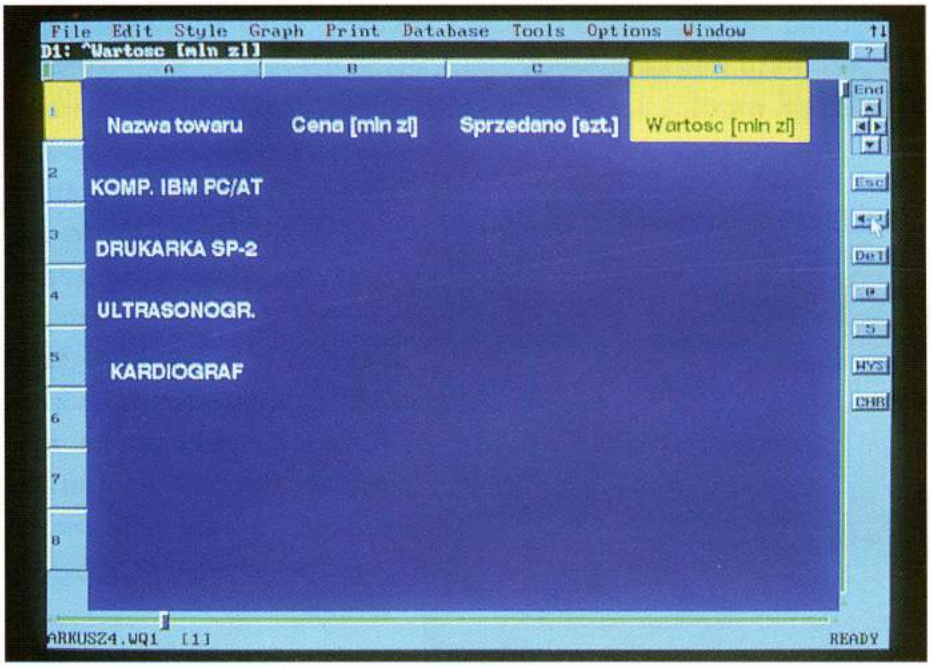
Arkusz ten jest odpowiednio zakodowany w pamięci komputera. Na ekranie nie można na raz wyświetlić całego arkusza, gdyż jest zbyt wielki (przykładowo 256 kolumn na 2048 wierszy). Ekran stanowi więc okienko, przez które oglądasz fragment arkusza. Fragment ten wskazuje się za pomocą polecenia zmiany miejsca wyświetlania lub zmieniając położenie kursora[1].

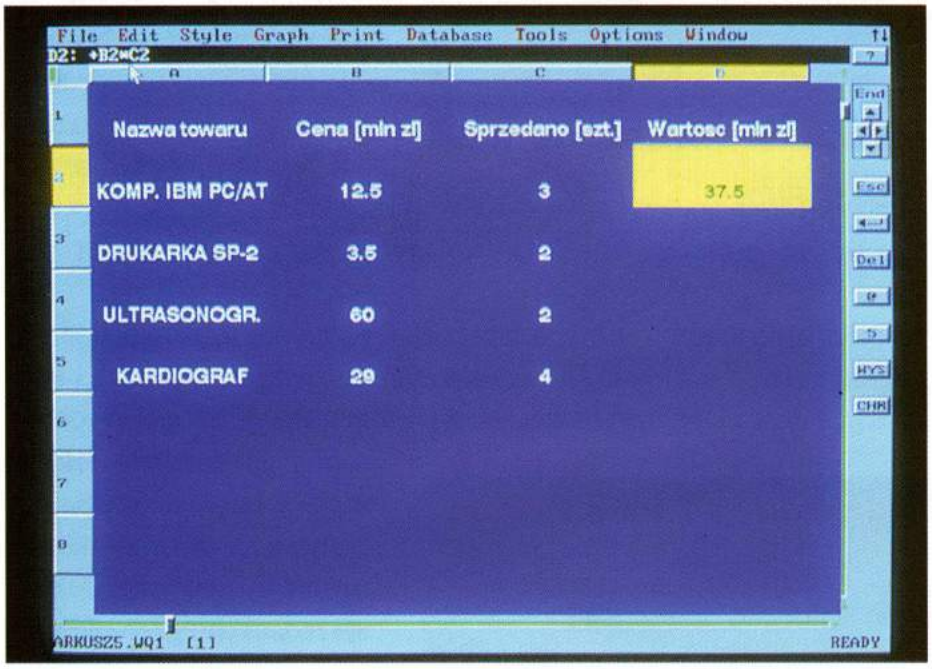
Zazwyczaj arkusza używa się do sporządzania różnych tabel, dlatego na początku wpisuje się (sposób wpisywania pokazuje fotografia niżej)[2] nagłówki (napisy określające znaczenie wierszy i kolumn), a następnie wypełnia się arkusz danymi wpisując liczby lub wyrażenia. Przykładowo, wyobraź sobie, że budujesz arkusz mający stanowić rachunek: zestawiasz w nim nazwy towarów, ich ceny, ilość sprzedanych egzemplarzy i wartość. Wpisujesz więc – posługując się okienkiem edycyjnym w lewym górnym rogu ekranu (patrz fotografia obok) – dane, które mają się znaleźć w odpowiednich komórkach. Dla pierwszego wiersza arkusza są to stosowne nagłówki, a w pierwszej kolumnie – nazwy towarów (patrz fotografia



[1] Warto tu zauważyć analogię do tekstu przetwarzanego przez edytor tekstowy: tam również ogląda się przez "okienko ekranu" jedynie pewien fragment całości.

[2] Podczas wpisywania nowych informacji do komórki trzeba najpierw ustawić w tej komórce duży, prostokątny kursor, a następnie można już wpisywać dowolne informacje za pomocą klawiatury. Wpisywane znaki pojawiają się najpierw w okienku edycyjnym (ponad górną krawędzią arkusza). Tworzone napisy można tam do woli zmieniać i poprawiać jak w edytorze. Dopiero po naciśnięciu klawisza **Enter** zapisana treść kopiowana jest do komórki arkusza i jest w niej od tej chwili stale widoczna.





obok). Następnie w odpowiednich polach[1] arkusza wpisujesz kolejno ceny i liczbę sztuk sprzedanych wyrobów (patrz fotografia poniżej).

Do tej pory arkusz zachowuje się właściwie identycznie, jak edytor – z wyjątkiem tego, że sam program pilnuje porządku w tabeli i formatowania komórek[2]. Natomiast jego prawdziwą użyteczność można poznać dopiero po zastosowaniu **wyrażeń.**

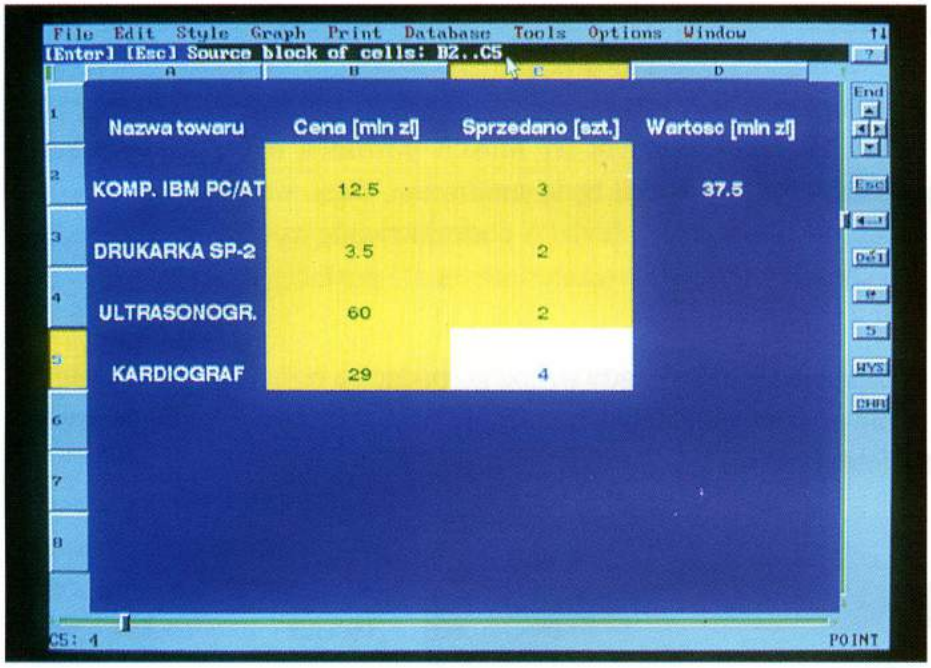
Wyrażenie wpisane do danej komórki określa wartość tej komórki w zależności od zawartości innych, wskazanych komórek oraz wartości pewnych stałych. Przykładowo komórka **D2,** w której uwidoczniona będzie **wartość,** może być wyliczona z formuły postaci **+B2*C2,** gdzie **B2** jest numerem komórki, w której pamiętana jest **cena,** a **C2** jest numerem komórki, w której zapamiętano **ilość**[3]. Od tej chwili arkusz zaczyna mądrze pomagać Ci przy prowadzeniu potrzebnych rozważań i kalkulacji, ilekroć bowiem zmieni się zawartość jednej z komórek, wszystkie związane z nią wyrażenia zostają automatycznie przeliczone i w odpowiednich komórkach arkusza pojawiają się nowe wartości.

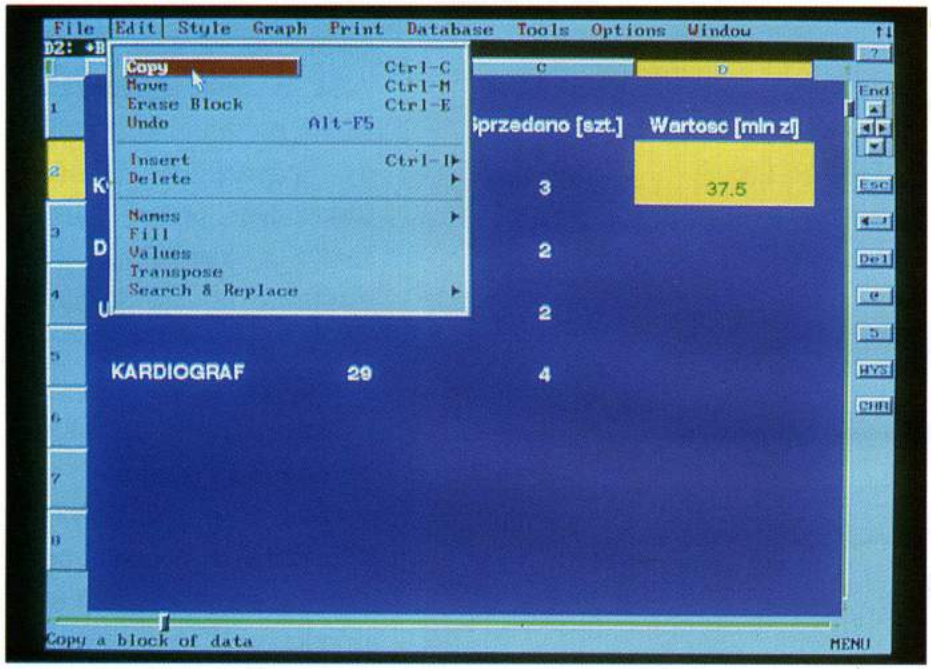
Dowolny prostokątny zbiór komórek arkusza może być zdefiniowany jako **zakres** (patrz fotografia na następnej stronicy), który może być **kopiowany, kasowany,**

[1] Przechodzenie od jednej komórki do drugiej następuje w bardzo prosty sposób po naciśnięciu klawisza kursora.

[2] Dostępne są opcje określające format znajdujących się w komórkach liczb, tzn. liczbę miejsc po przecinku, położenie w komórce – centralne lub z wyrównaniem lewo– i prawostronnym do granicy komórki, a także zapis z podziałem wielocyfrowej liczby na grupy po 3 cyfry (tzw. opcja *currency* stosowana przy zapisie wartości interpretowanych jako kwoty pieniędzy).

[3] Wyrażenie zwykle rozpoczyna się od symbolu +, by program arkusza wiedział, że ma do czynienia z wyrażeniem, a nie z tekstem (co byłoby przyjęte, gdyby pierwszym wpisanym symbolem była litera) lub z liczbą (gdyby pierwsza była cyfra).





przenoszony w inne miejsce arkusza lub **zapamiętywany** w pliku (na dysku). Przy kopiowaniu lub przenoszeniu komórek i zakresów w wyrażeniach następuje automatyczne przeadresowanie argumentów względem nowego położenia. Oznacza to, że w nowym miejscu **wyrażenie odnosi się do komórek o tym samym położeniu względem komórki, w której jest zapisane.**

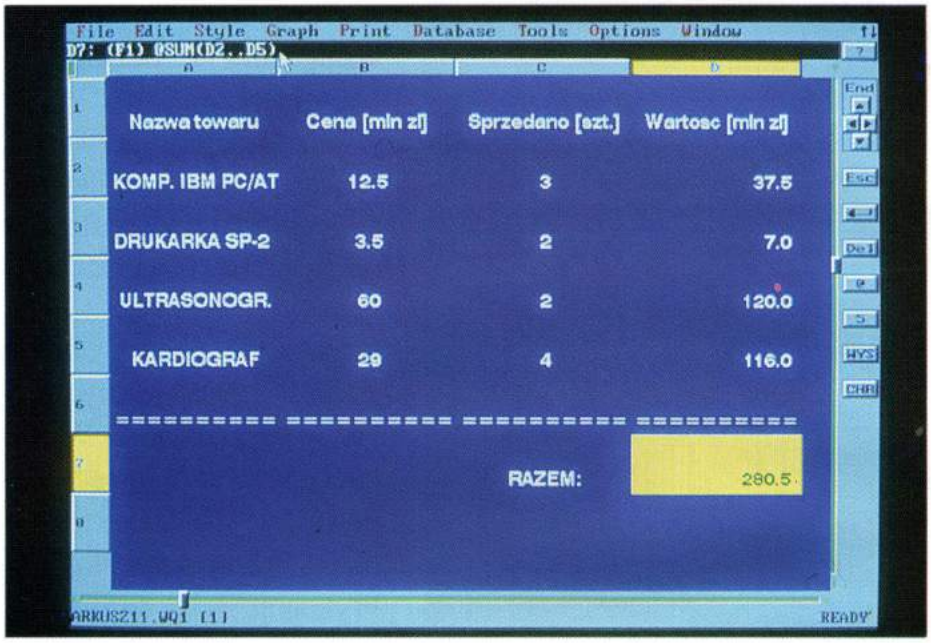
Poznajmy tę ważną własność arkusza kalkulacyjnego na przykładzie. W opisywanym rachunku zasada wyznaczania pola **wartość** w wierszu nr **3** powinna być identyczna, jak w wierszu nr **2.** Wystarczy więc **skopiować** odpowiednie wyrażenie z komórki **D2** do komórki **D3.** Wykonuje się to wybierając z **menu**[1] programu opcję **copy** (patrz fotografia) i wskazując (kursorem w obszarze arkusza) skąd należy kopiować informacje i gdzie je umieścić. Po takim zabiegu do komórki **D3** zostanie skopiowane odpowiednie wyrażenie, ale odwołujące się do komórek **B3** i **C3,** a nie – jak w oryginalnej komórce – **B2** i **C2** (patrz fotografia obok). W podobny sposób można skopiować potrzebne wyrażenie do komórek **D4** i **D5.** W tym celu wygodnie jest skorzystać z faktu, że na pytanie programu o miejsce, do którego chcemy kopiować dane, można wskazać **dowolny zakres,** ustawiając kursor w lewym górnym rogu prostokątnego obszaru, który ma być zakresem, naciskając znak . (kropka) i ciągnąc kursor do prawego dolnego rogu potrzebnego zakresu (zaznaczony obszar zmienia barwę na ekranie). Operacja kopiowania (będącego w tym przypadku w praktyce powielaniem) wykonana zostanie do wszystkich komórek wskazanego zakresu.

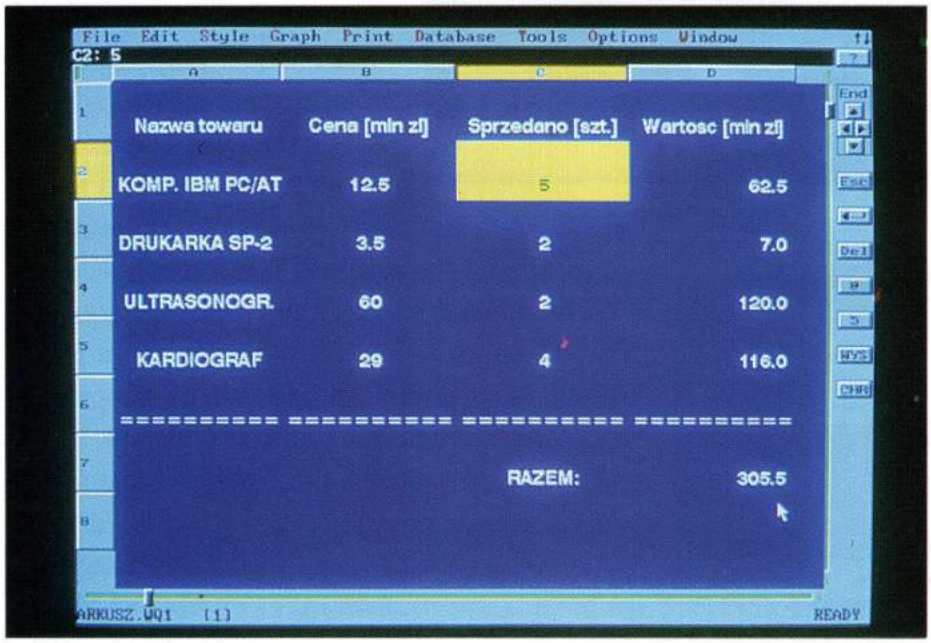
[1] Przejście do linii **menu** (z reguły widocznej u góry arkusza) następuje z reguły po naciśnięciu klawisza /, a dalsze etapy wyboru z menu dokonują się zgodnie z regułami, które opisano wyżej dla edytorów (przesuwanie kursora i zatwierdzanie **Enter**).

Pojęcie **zakresu** jest też przydatne, jeśli jakieś działanie w arkuszu chcemy wykonać na wielu komórkach. Przykładowo załóżmy, że chcemy podsumować nasz rachunek. W tym celu najpierw w całym wierszu nr **6** wpiszemy teksty tworzące poziomą kreskę[1], a następnie w komórce **D7** wpiszemy wyrażenie zawierające funkcję[2] **@SUM.** Jako argument tej funkcji powinien być podany **zakres.** Wszystkie liczby znajdujące się wewnątrz tego zakresu będą sumowane, dając wartość, która będzie uwidoczniona w rozważanej komórce. W przykładzie chodzi o podsumowanie całej kolumny **D,** dlatego wyrażenie, które wpiszemy do komórki **D7,** będzie ostatecznie miało postać

@SUM(D2 .. D5)

(właśnie tak, z dwiema kropkami, zapisuje się zakres, gdy trzeba go podać za pośrednictwem klawiatury). Tabela zostanie automatycznie podsumowana, a odpowiednia liczba, będąca wynikiem, zostanie wpisana do komórki **D7** (patrz fotografia niżej).





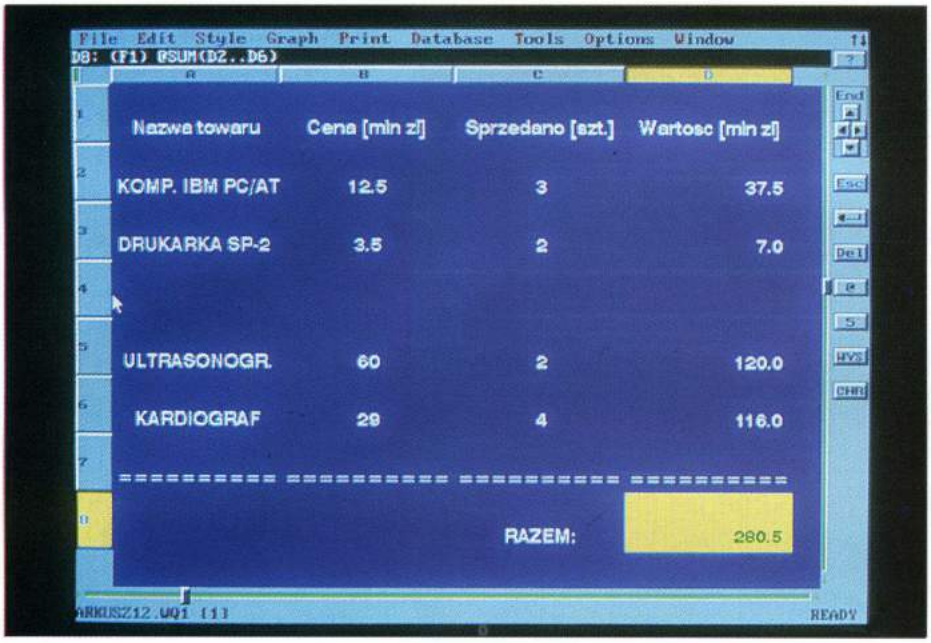
Od tej pory cokolwiek **zmienimy** w arkuszu – wszystkie pozycje łącznie z globalną sumą będą automatycznie przeliczane i natychmiast widoczne na ekranie. Fotografia u dołu przedstawia tabelę po dokonaniu zmiany jedynie w komórce wskazanej przez kursor. Łatwo zauważyć, że związane z nią komórki tabeli zmienione zostały przez program automatycznie. Takie działanie programu pozwala na sprawdzanie, jaki wpływ na elementy arkusza będą miały zmiany pewnych komórek, czyli na prowadzenie prognozowania. Polega to na zmianie[3] niektórych danych, przeliczaniu wartości z arkusza i obserwacji zmian wyników (porównaj fotografię obok). Możliwe jest także prowadzenie tą metodą analiz typu "co by było gdyby" i szukanie rozwiązań optymalnych taką metodą "symulacji".

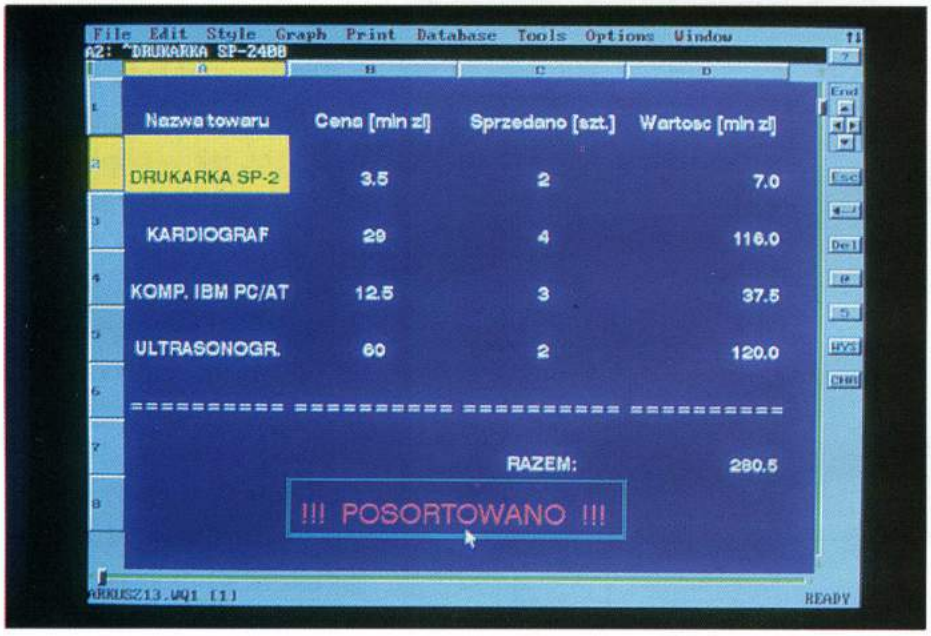
---

1 Wypełnienie komórki powtórzeniami jednakowego znaku osiąga się pisząc znak \ i znak do powtarzania, na przykład \=.

2 W wyrażeniach oprócz czterech podstawowych działań arytmetycznych można też używać **funkcji matematycznych** (logarytm i jego odwrotność, pierwiastek kwadratowy, funkcje trygonometryczne, wartość modulo, część całkowita, generator liczb losowych), **funkcji statystycznych** (obliczanie na zbiorach liczb sumy, wartości średniej, wariancji i odchylenia standardowego, znajdowanie wartości minimalnej i maksymalnej itp.) oraz **funkcji logicznych i finansowych** (na przykład oprocentowanie kredytu lub okres spłaty długu).

3 Zmiana zawartości komórki nazywa się **edycją.** W tym celu trzeba wskazać potrzebną komórkę, a potem nacisnąć klawisz **F2.**

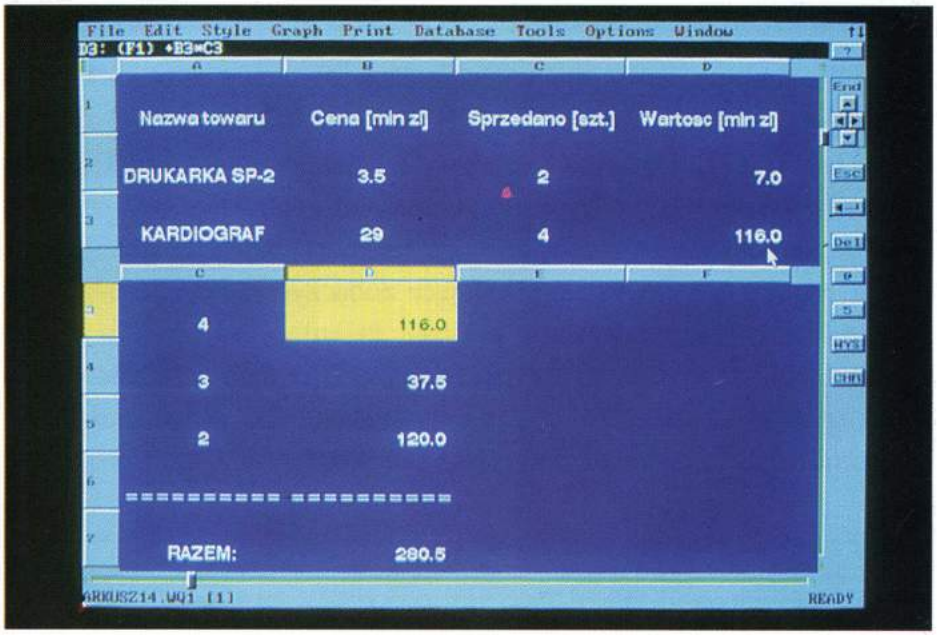


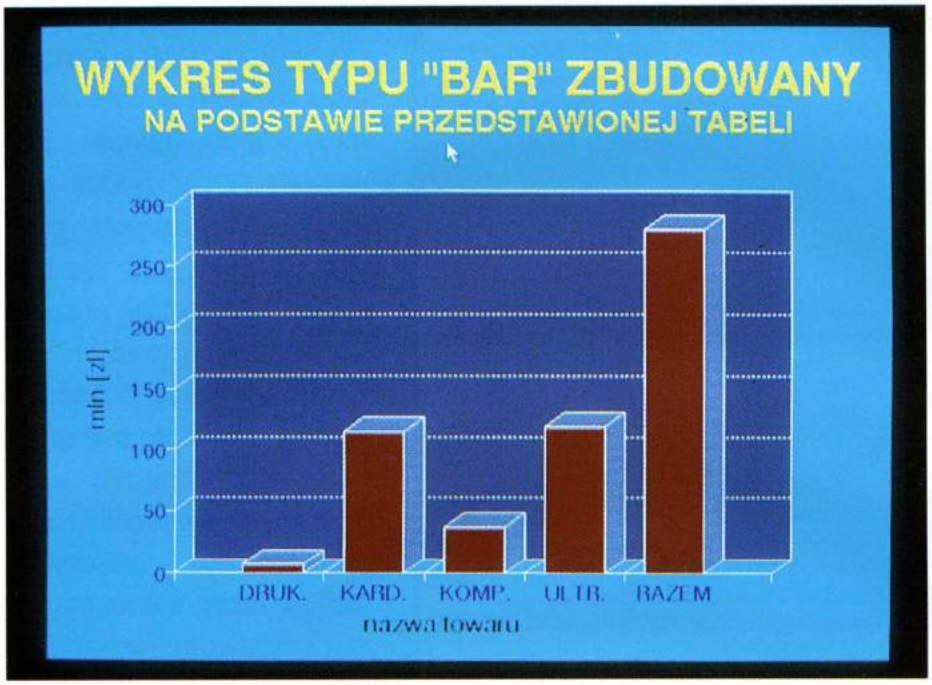


W czasie wypełniania arkusza można korzystać z różnych dodatkowych możliwości, takich jak: wstawienie w środek arkusza dodatkowych pustych wierszy lub kolumn (patrz fotografia), porządkowanie (sortowanie) komórek i całych wierszy według dowolnie podanych reguł (na fotografii niżej budowany rachunek został posortowany alfabetycznie), a także automatyczne wyszukiwanie w dużym arkuszu określonych informacji według zadanego kryterium. Można też tak "zwinąć" arkusz na ekranie, żeby w kilku równocześnie widzieć kilka odległych fragmentów dużego arkusza, na przykład lewy górny i prawy dolny jego róg.

Na fotografii niżej przedstawiłem ten sposób prezentacji fragmentów naszej przykładowej tabeli. Ta tabela była na tyle mała, że mieściła się na ekranie także w formie nie zwiniętej, gdyby jednak była większa – byłby to jedyny sposób równoczesnego obserwowania przyczyn (zmian w polach tabeli) i skutków (zmian globalnej sumy) w trakcie prowadzenia obliczeń.

Na tym możliwości arkusza wcale się nie kończą. Dowolne szeregi liczb z arkusza mogą być wykorzystane do zbudowania





różnego typu **wykresów**, które wyświetlane są na ekranie na życzenie użytkownika. Przykład takiego wykresu, zbudowanego na podstawie danych z rozważanej cały czas tabeli, pokazałem na fotografii obok.

Dowolna zmiana w danych liczbowych wprowadzona z klawiatury powoduje przeliczenie wielkości zależnych i zmianę przebiegu wykresu. W ten sposób możesz badać modele[1] dowolnych systemów i obserwować ich reakcje na zmianę danych, dowiadując się, jak w odpowiednich warunkach zachowa się modelowany system (patrz przykładowa fotografia obok), a w razie nonsensownych odpowiedzi możesz łatwo skorygować postać modelu. Podczas modelowania można łatwo tworzyć potrzebną dokumentację, ponieważ żądany fragment arkusza oraz dowolne badane wykresy mogą być w każdej chwili wydrukowane.

Programy kalkulacyjne zapisują dane z arkuszy na plikach dyskowych stosując własne, charakterystyczne sposoby zapisu. Na przykład pakiet *Lotus* **1–2–3** zapisuje swoje pliki (z rozszerzeniem **WKS** lub **WK1**), a **Quatro Pro** swoje (z rozszerzeniem **WQ1**) w taki sposób, że ich odczytanie w formie znakowej jest praktycznie niemożliwe. Z tego powodu programy służące do oglądania zawartych w pamięci komputera informacji (na przykład **NC**) stosują specjalne nakładki (programy pomocnicze, np. **123VIEW**), pozwalające zobaczyć zawartość takich plików w sposób zrozumiały dla użytkownika. Natomiast same arkusze kalkulacyjne potrafią zwykle odczytywać i odpowiednio przekształcać dane zapisywane przez najbardziej rozpowszechnione systemy obsługi baz danych (na przykład pliki z rozszerzeniem **DBF** budowane przez program **dBase**). Także rozbudowane programy graficzne potrafią zwykle odczytywać i interpretować dane z programów kalkulacyjnych.

---

[1] Bardziej rozbudowane programy umożliwiają definiowanie procedur wykonujących skomplikowane obliczenia na elementach arkusza, np. obliczanie regresji liniowej. Dlatego programy kalkulacyjne mogą być wykorzystywane do obliczeń inżynierskich i naukowych, a odpowiednie zaprojektowanie reguł wiążących poszczególne wiersze arkusza pozwala na prowadzenie prostego modelowania procesów (na przykład gospodarczych) i może być używane do prognozowania przyszłości.

Z arkuszami kalkulacyjnymi skojarzone są często dodatkowe programy, wspomagające korzystanie z arkusza i pozwalające uzyskać dodatkowe usługi[1]. Najnowsze tendencje w rozwoju programów kalkulacyjnych to wyposażanie ich w coraz bogatsze możliwości graficzne oraz tworzenie arkuszy wielowymiarowych i umożliwienie automatycznej wymiany informacji między pamięcią komputera a dyskiem, co pozwala budować arkusze o objętości większej niż pamięć operacyjna komputera.

3.8. Grafika komputerowa

Większość komputerów ma "wbudowane na stałe" podstawowe możliwości graficzne. Przykładowo, najpopularniejsze na mikrokomputerach języki programowania (**Pascal, C, Basic**, a zwłaszcza **LOGO**) posiadają z reguły wbudowane mechanizmy umożliwiające kreślenie prostych rysunków. Również tworzenie "trwałych kopii" rysunków jest "zaszyte" w systemie operacyjnym maszyny. Przykładem może tu być klawisz **Prt Sc**, występujący na klawiaturze mikrokomputerów typu IBM PC. Klawisz ten pozwala na wykonanie w dowolnym momencie kopii ekranu na drukarce – jeśli więc na ekranie znajdował się jakiś rysunek – to będzie on odwzorowany na drukarce[2].

Jeśli nie wystarczają Ci te podstawowe możliwości – możesz użyć jednego z wielu programów tworzących obrazy na ekranie i jednego z programów kopiujących ekran na drukarkę w sposób udoskonalony. Przykładem takiego programu dla IBM PC może być popularna nakładka (to znaczy program stale rezydujący w pamięci niezależnie od innych programów) o nazwie **PIZAZZ** (patrz fotografia). Program ten, opracowany w firmie *Application Techniques* umożliwia drukowanie obrazu powiększonego, pomniejszonego, obróconego, wyciętego z fragmentu ekranu itd.

Wśród olbrzymiej liczby istniejących programów graficznych dość wyraźnie wyróżniają się następujące obszary zastosowań:

– rysowanie wykresów,

– przygotowywanie różnych wydawnictw i prezentacji ekranowych,

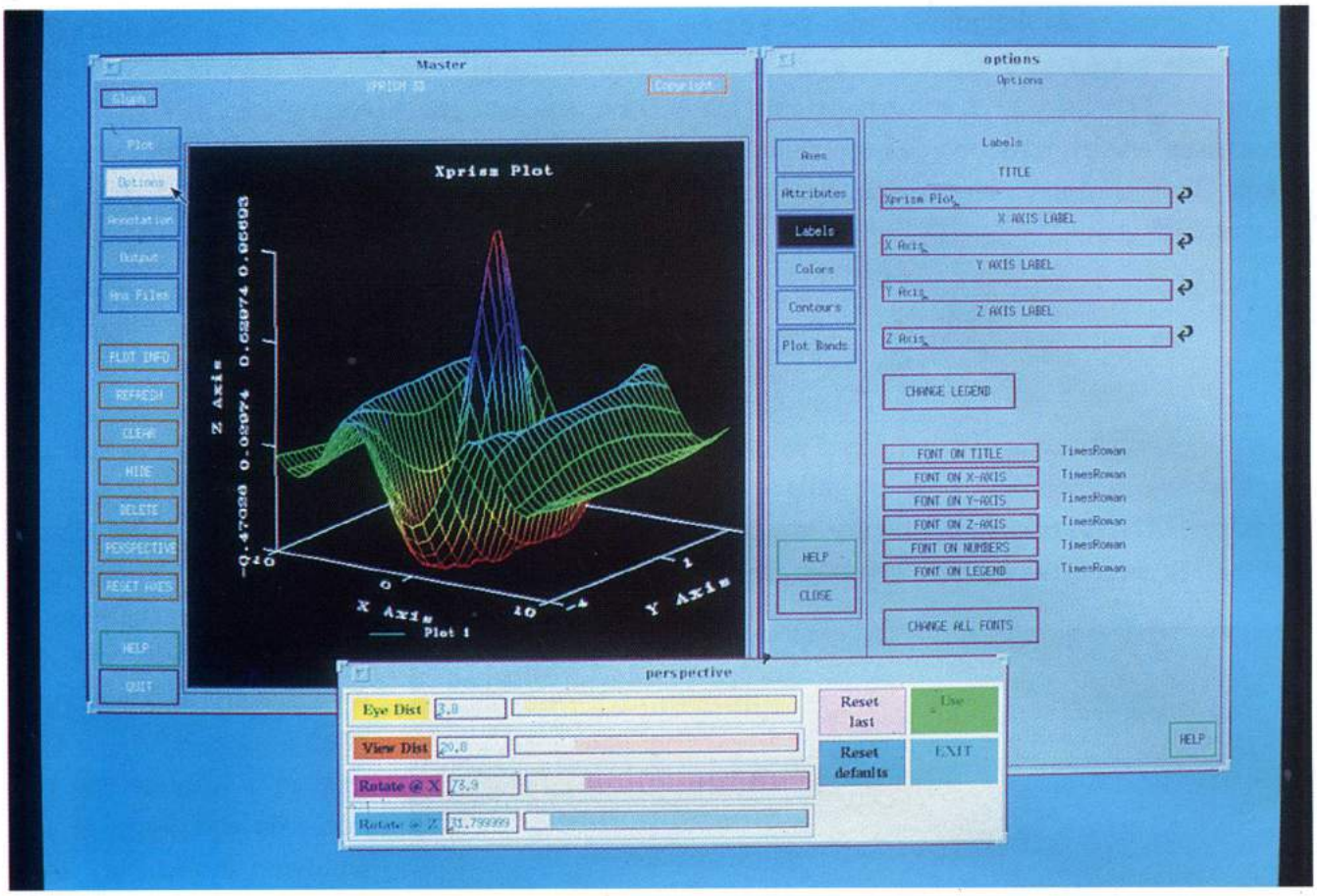
– kreślenie rysunków związanych z komputerowym wspomaganiem projektowania.

Najbardziej znanym programem służącym do rysowania wykresów jest **StatGraphics,** który dodatkowo może dokonywać analizy statystycznej danych. Dla rysowania trójwymiarowych wykresów – szczególnie ciekawych, ale i wyjątkowo trudnych (patrz fotografia na następnej stronicy) –

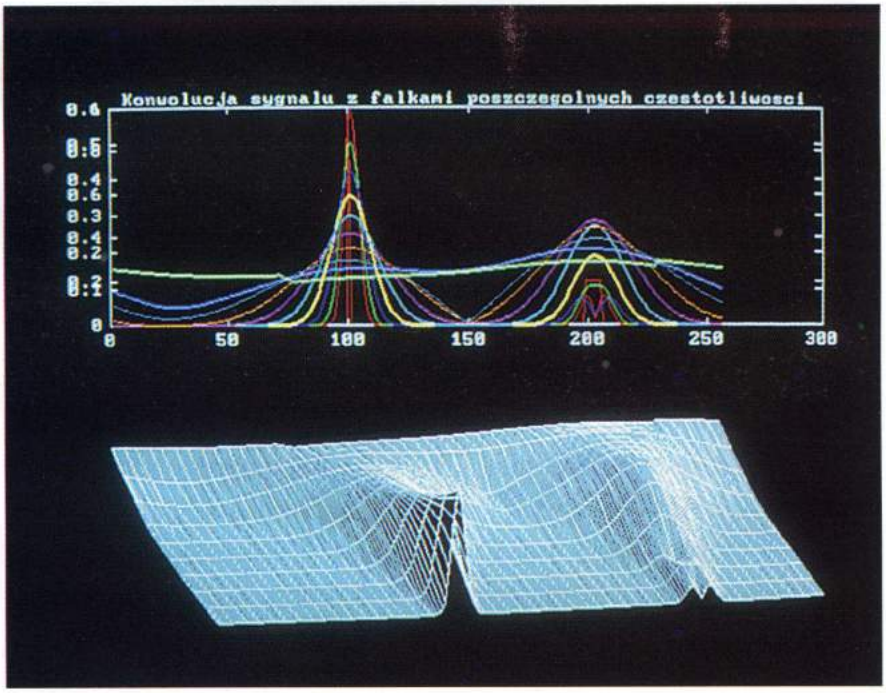
---

[1] Bardziej skomplikowane programy kalkulacyjne zapewniają ochronę wybranych danych przed odczytaniem przez niepowołane osoby. Zawartość wskazanych komórek, kolumn lub wierszy może być uwidoczniona na ekranie tylko na żądanie uprawnionych osób.

[2] Jeśli przewiduje się konieczność korzystania z tej możliwości – należy wcześniej użyć polecenia **GRAPHICS**.

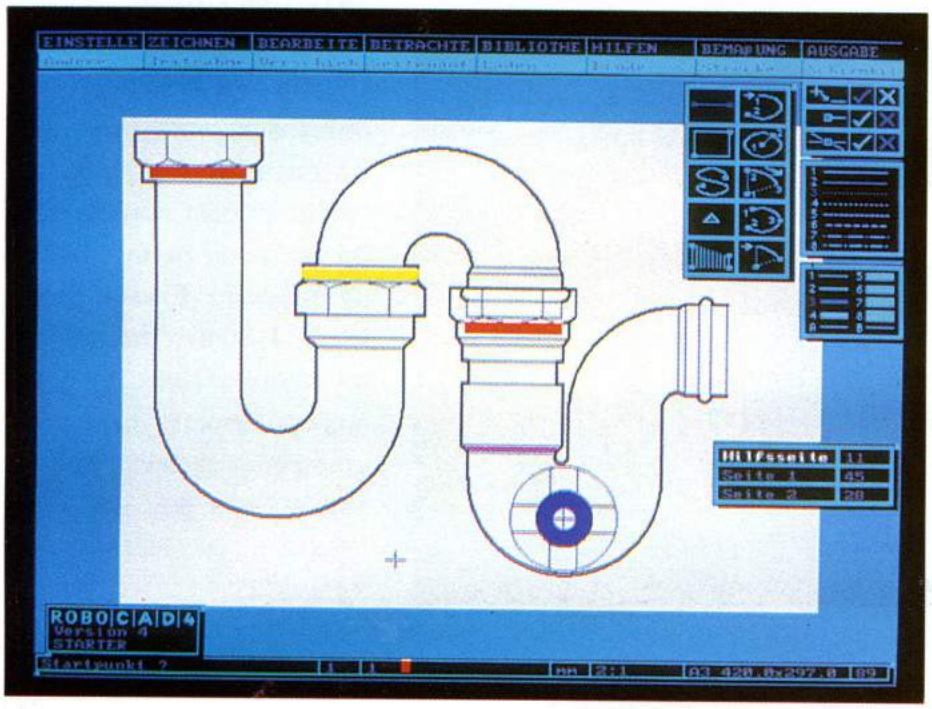


służyć może na przykład szeroko znany **Surfer** albo **Graph 3D**. Za pomocą takich programów graficznych można tworzyć wykresy ilustrujące dane w jednym z kilku przyjętych formatów. Najczęściej stosuje się wykresy dwuwymiarowe, diagramy kołowe, histogramy oraz ich różne odmiany. Tego typu rysunki noszą często nazwę grafiki prezentacyjnej (patrz fotografia) i są w podstawowym zakresie dostępne także w programach omówionych wyżej arkuszy kalkulacyjnych.



Inne typowe zastosowanie grafiki komputerowej wiąże się z kreśleniem rozmaitych schematów, na przykład schematy algorytmów pięknie rysuje program **FlowCharting II+** (jego "produkty" zobaczysz na fotografiach w rozdziale 6), a **Slide Writer** przygotowany jest do tworzenia rysunków, które potem można użyć jako przeźrocza. Proste rysunki można także tworzyć i w ciekawy sposób łączyć z ładnym liternictwem w takich popularnych programach jak **Print Shop** lub **Print Master**, a z nowszych **Page Maker, Corel Draw, PhotoStyler** czy zgrabny, wchodzący w skład systemu Windows, program **Paintbrush** (już kilka razy oglądałeś fotografie rysunków wykonywanych tym właśnie programem). Wymienione programy oferują Ci szeroką gamę gotowych rysunków, które

możesz wykorzystać przygotowując sobie rozmaite zaproszenia, ogłoszenia, wywieszki itp., a także dzięki opcji **Graphic Editor** pozwalają tworzyć własne rysunki. Wzięte ze zbioru gotowych wzorów lub samodzielnie opracowane rysunki możesz dowolnie rozmieszczać, wiązać z tekstem, powiększać, powielać itp. Używa się do tego myszki, traktując jej ruchy podobnie jak ruchy pędzla, ołówka lub gumki (do usuwania zbędnych linii). Program pomaga rysować idealnie proste linie, kreśli na życzenie koła, pomaga w powielaniu typowych fragmentów rysunku itp. W sumie programy omawianej tu grupy pozwalają na tworzenie ładnej grafiki przy minimalnym wysiłku i z tego powodu są szeroko i chętnie stosowane. Wymaga to oczywiście odrobiny wprawy, ale efekt końcowy jest wart wysiłku, jaki włożysz w nauczenie się, jak zaprzęgać posiadany program do pracy. Wytworzyła się zresztą z tego cała dziedzina, zwana **Desktop publishing**[1], w której używając komputera i odpowiedniego programu można tworzyć wydawnictwa (druki, gazetki, ulotki reklamowe itp.) o jakości nie gorszej niż z profesjonalnej drukarni.



Jednym z najpopularniejszych (w skali autentycznie światowej) programów graficznych jest pakiet **AutoCAD**, opracowany w **1983** roku przez firmę *Autodesk Inc.* Program ten miał w **1992** roku 335 tys. zarejestrowanych użytkowników[2]. Głównym przeznaczeniem pakietu jest tworzenie rysunków związanych z automatyzacją prac projektowych i inżynierskich (skrót **CAD** zawarty w nazwie pakietu jest szeroko znanym określeniem dziedziny *Computer Aided Design*, czyli projektowania wspomaganego komputerowo). Rozważany pakiet ma jednak możliwość tworzenia dowolnych rysunków, a więc nie tylko części maszyn i detali architektonicznych (do czego jest głównie wykorzystywany), ale również trójwymiarowych, cieniowanych (specjalny program **Auto SHADE**) oraz animowanych (program **AutoFLIX**) rysunków dowolnych obiektów.

3.9. Pakiety zintegrowane

Powstanie pakietów zintegrowanych wynikło bezpośrednio z konieczności korzystania z tych samych danych przez kilka różnych programów użytkowych. W wielu instytucjach korzysta się z edytora tekstowego, z arkusza kalkulacyjnego, z bazy danych czy wreszcie z programów graficznych. Wszystkie te programy tworzą własne struktury danych, a przecież bardzo często dane te się pokrywają (np. na fragmencie bazy danych należy wykonać obliczenia za pomocą programu

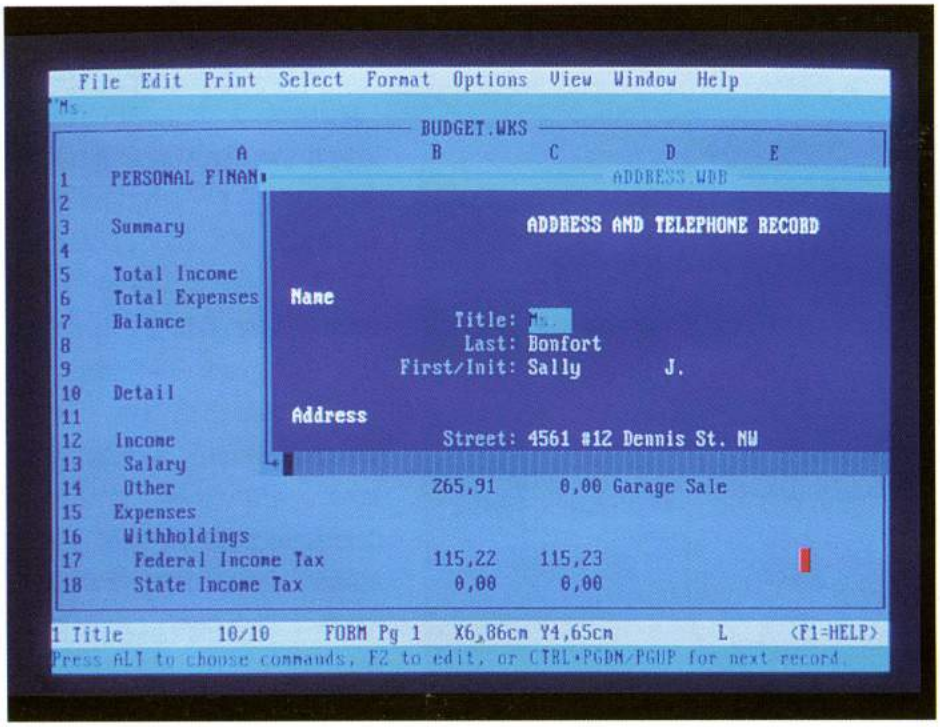
---

[1] W literaturze przedmiotu podawane są sprzeczne informacje na temat tego, kto i kiedy po raz pierwszy zastosował komputer do zadań małej poligrafii. Jednak najczęściej przyjmuje się, że twórcą koncepcji *desktop publishing* był **Paul Brainerd**, który w **1985** roku opracował wykorzystywany do dziś program **Page Maker.**

[2] Wydawane są nawet specjalne periodyki, poświęcone wyłącznie zagadnieniom związanym z programem AutoCAD i jego zastosowaniami. W Europie wydawany jest miesięcznik zatytułowany *CADUser*, w którym użytkownicy programu AutoCAD wymieniają doświadczenia, prezentują swoje osiągnięcia i dowiadują się o rozwoju używanego programu, a w Polsce drukowany jest periodyk *CAD–forum.*

kalkulacyjnego, wyniki przedstawić w formie graficznej i jeszcze na dodatek uzupełnić opisem słownym). Tego typu potrzeby doprowadziły do opracowania programów, które łączą w sobie cechy wszystkich wymienionych. Zestaw takich programów, spełniających różne funkcje, lecz nawzajem do siebie dopasowanych i współdziałających ze sobą i zdolnych przekazywać sobie dane, nazwano **pakietem zintegrowanym**.

Posługiwanie się pakietem zintegrowanym jest znacznie prostsze niż użycie kilku oddzielnych programów realizujących poszczególne funkcje pakietu, gdyż ujednolicony jest sposób obsługi poszczególnych składowych pakietu. Największą zaletą zintegrowanego oprogramowania jest jednak łatwość przesyłania danych pomiędzy procedurami pakietu oraz możliwość tworzenia dokumentów, w których informacje przetworzone przez te procedury sąsiadują ze sobą. Pakiety zintegrowane są bardzo złożonymi programami i wymagają komputerów o dużej mocy obliczeniowej. Klasycznym przykładem pakitu zintegrowanego był do niedawna znany i lubiany program **Framework**[1], firmy *Ashton–Tate*, twórcy programu **dBase**[2]. Obecnie najpopularniejszym programem o cechach pakietu zintegrowanego jest **MS Works** (widoczny na fotografii).



Program ten scala w postaci jednego programu następujące możliwości:

- redagowanie tekstów;
- kalkulację tabelową;
- zarządzanie bazą danych;
- graficzną prezentację wyników;
- programy telekomunikacyjne.

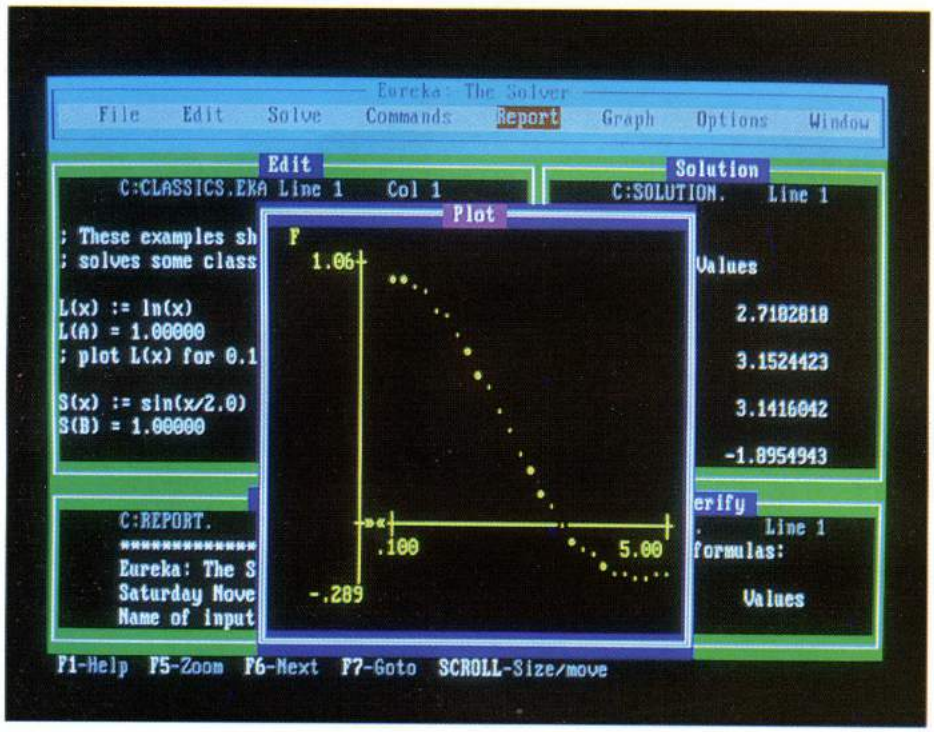
Scalenie wymienionych funkcji oznacza, że – przykładowo – pisząc tekst możemy go w każdej chwili, nie wychodząc z programu, uzupełniać dowolnymi danymi z bazy danych. Dane te mogą być dowolnie przetwarzane z wykorzystaniem możliwości, jakich dostarcza arkusz kalkulacyjny, a forma ich prezentacji może być wzbogacona dzięki zastosowaniu środków grafiki komputerowej. Wyniki po zredagowaniu i przetworzeniu mogą być w bardzo ładnej formie wydrukowane względnie przesłane do innego posiadacza komputera z wykorzystaniem wbudowanych w pakiet możliwości tzw. elektronicznej poczty.

---

[1] Z innych programów, mających podobne właściwości i zbliżony zakres tematyczny, wymienić można **Enable** firmy *The Software Group* lub też **Symphony** firmy *Lotus*.

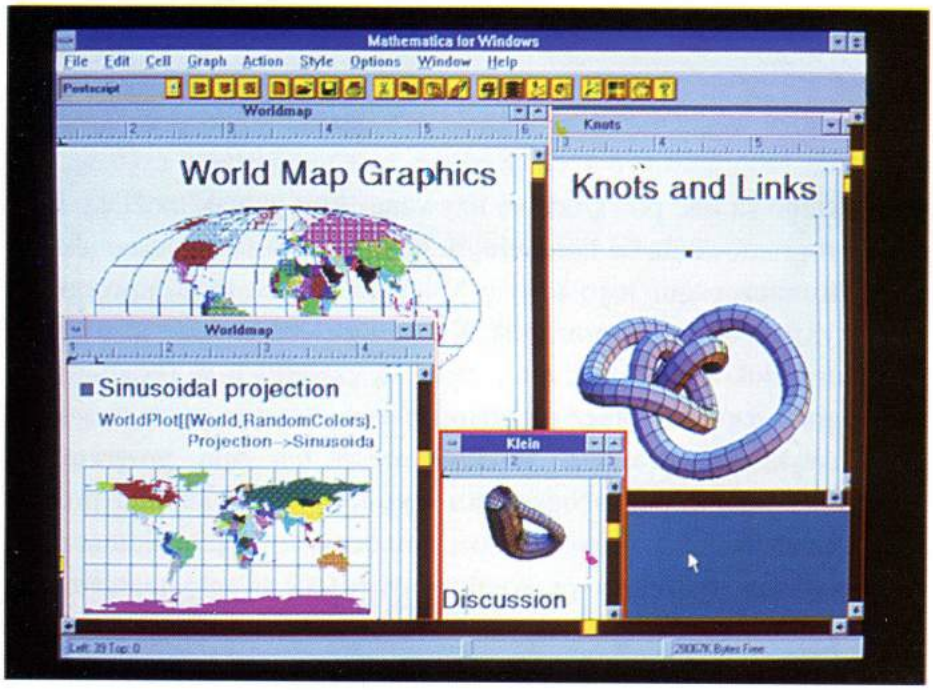
[2] Jedną z zalet tego programu był fakt, że jeśli komuś jego podstawowe możliwości nie wystarczały – mógł swobodnie dopisać własne programy obsługi i przetwarzania danych, wykorzystując wbudowany w program **Framework** specjalizowany język programowania o nazwie **FRED**.

3.10. Programy matematyczne



Od początku swego istnienia komputery były i są wykorzystywane do obliczeń matematycznych. Takie było podstawowe zastosowanie komputerów i od tego wzięła się nazwa *computer*, czyli maszyna licząca. Ze względu na powszechność tego zastosowania powstała ogromna liczba programów komputerowych przeznaczonych do rozmaitego rodzaju obliczeń. Wchodzą one w skład różnych pakietów i bibliotek, których – ze względu na ilość i różnorodność po prostu nie sposób wymienić[1]. Pojawiły się także próby budowy oprogramowania, które nie tylko rozwiązuje zlecane przez użytkownika zadania, lecz współdziała z nim przy ich formułowaniu oraz dostarcza możliwości graficznych związanych z prezentacją wyników obliczeń w najdogodniejszej do interpretacji formie.

Przykładem takiego programu jest pokazany na górnej fotografii program **Eureka.** Jest to produkt firmy *Borland* określany też jako **The Solver**, czyli rozwiązywacz problemów[2]. Program ten posiada wygodny dla użytkownika system rozwijalnych menu (*pull down menu*), pozwalający sterować jego pracą metodą kolejnych wyborów z listy proponowanych ewentualności, a także zawiera liczne teksty wyjaśniające, oferowane na każde żądanie, przy czym zakres i charakter wyjaśnień dobierany jest automatycznie do tego, co w danej chwili robisz, a więc i do tego, czego zapewne potrzebujesz.

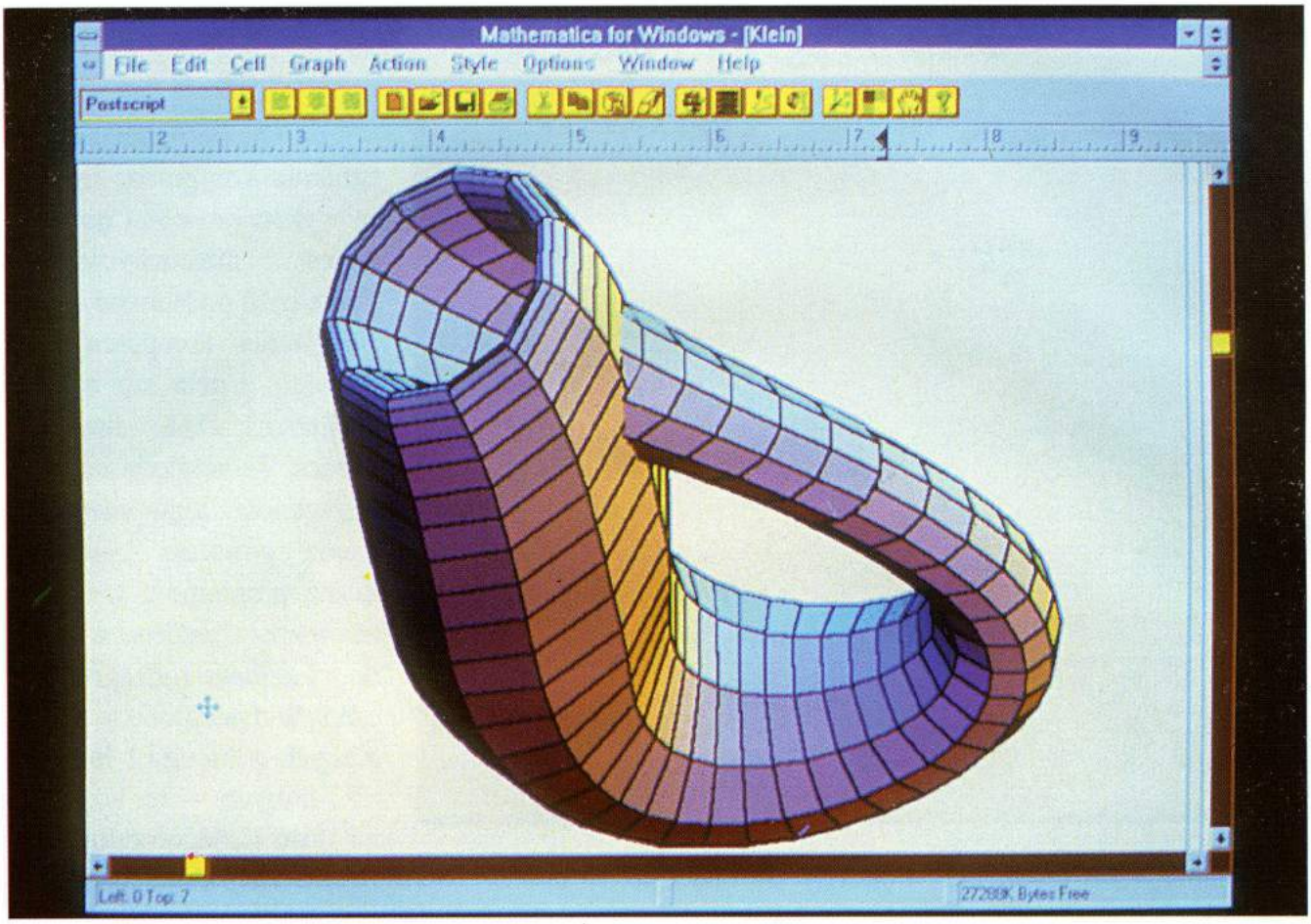


Inne wygodne programy matematyczne to **MathCAD, Mathlab** i szlagier ostatnich lat – fantastyczny program **Mathematica** firmy *Wolfram Research*, którego możliwości pokazano na fotografii obok i na następnej stronicy.

---

1 Np. znana biblioteka **Turbo Tools** związana z językiem **Turbo Pascal** firmy *Borland*.

2 Opis możliwości i sposobów użytkowania programu **Eureka** znaleźć można w książce: *R. Tadeusiewicz, Eureka, WNT 1992.*



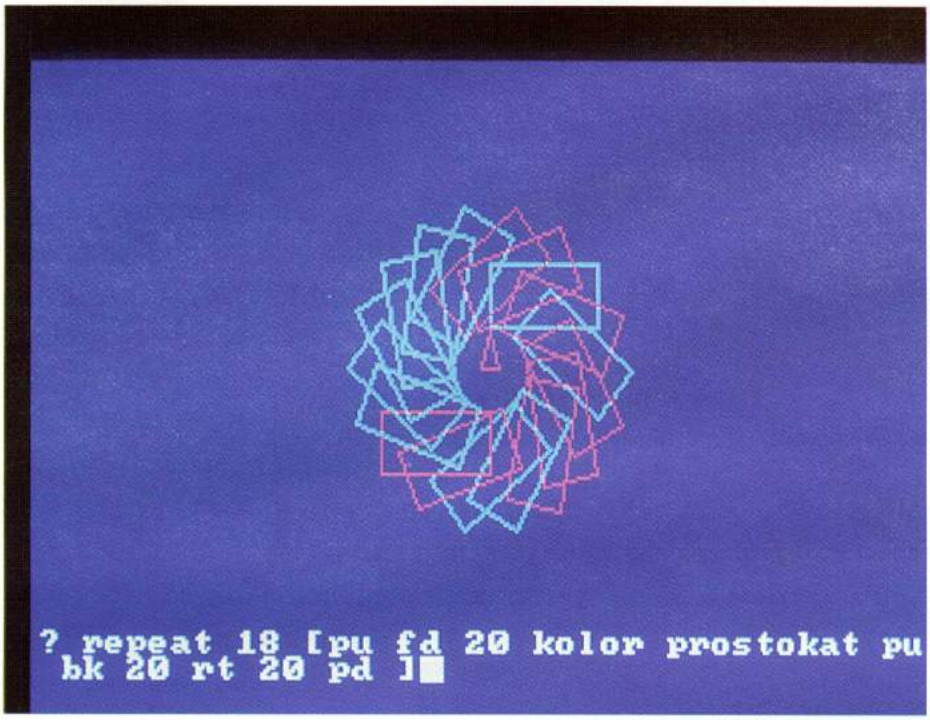
3.11. Programy narzędziowe

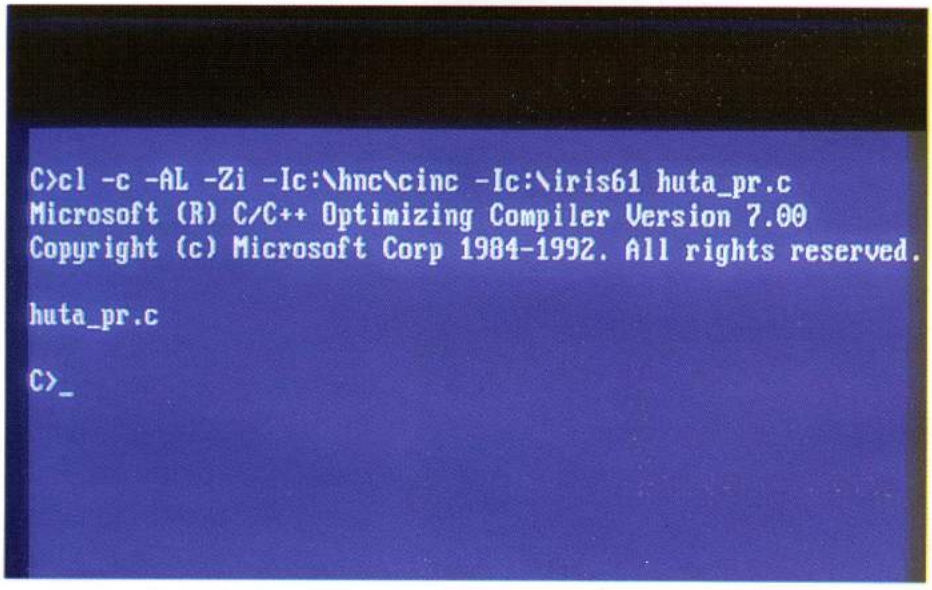
Programy narzędziowe służą głównie do tworzenia nowych programów, są więc przeznaczone dla tych bardziej doświadczonych i bardziej wymagających użytkowników komputera, do których Ty się chyba **jeszcze** nie zaliczasz. Jednak pewna wiedza na temat oprogramowania narzędziowego też się przyda, zatem ten rozdział trzeba potraktować jako pigułkę z wiadomościami z egzotycznego świata **prawdziwych informatyków.**

W skład oprogramowania narzędziowego wchodzą **translatory języków programowania.** Języków tych jest kilkadziesiąt. Pisanie programów w tych językach jest łatwiejsze i szybsze, niż tworzenie ich w języku komputera, dlatego są one powszechnie używane. Aby jednak możliwe było korzystanie z jakiegokolwiek języka programowania na konkretnym komputerze – konieczne jest u–życie **translatora, czyli programu tłumaczącego**. Jego użycie jest niezbędne z tego powodu, że **program napisany w dowolnym języku programowania jest dla komputera zawsze absolutnie niezrozumiały**[1]. Musi on więc najpierw dokonać **tłumaczenia** poleceń zawartych w programie, a dopiero potem może je wykonać. Program napisany przez programistę w wybranym przez niego je–zyku programowania nazywa się zwykle **programem źródłowym,** a ten sam program po przetłumaczeniu go na język maszynowy nazywa się **programem wynikowym.** Komputer pracuje więc nad nowym programem dwukrotnie: najpierw dokonując (za pomocą translatora) tłumaczenia programu źródłowego na wynikowy, a potem wykonując program wynikowy (z uwzględnieniem da–nych) celem uzyskania potrzebnych wyników.

[1] Komputer może przyjmować i wykonywać programy wyłącznie napisane w jego języku wewnętrznym, natomiast każdy z języków programowania jest dla niego językiem obcym.

Translatory można podzielić na **kompilatory** i **interpretery**. Kryterium tego podziału związane jest ze wzajemnym stosunkiem etapów: tłumaczenia programu i jego wykonania. Jeśli **cały** program źródłowy przetłumaczony zostaje na język maszynowy, a dopiero potem następuje jego wykonanie – wówczas uważamy, że translator jest **kompilatorem**. Jeśli natomiast proces tłumaczenia **przeplata się** z procesem wykonywania programu (na przykład tłumaczone są pojedyncze instrukcje programu źródłowego i natychmiast następuje ich wykonanie), wówczas translator nazywa się **interpreterem**. Każda z tych technik ma swoje zalety, warto więc je znać, aby świadomie (w razie potrzeby) wybierać.





Interpreter jest wygodniejszy w przypadku, kiedy przewidujesz wielokrotne poprawianie i korygowanie programu[1], ponieważ przechowuje cały czas **program źródłowy**, więc wprowadzanie poprawek jest łatwe i naturalne. Wygodne jest także i to, że przy korzystaniu z interpretera widzisz na ekranie równocześnie program i wynik jego działania – patrz fotografia obok. Niestety ta wygoda jest okupiona znacznym spowolnieniem pracy, bo trzeba te same instrukcje tłumaczyć wielokrotnie.

Jeśli więc masz rozwiązać za pomocą komputera skomplikowane zadanie, wymagające dużej liczby obliczeń, jeśli zamierzasz napisany program bez żadnych zmian wielokrotnie wykonywać, jeśli wreszcie piszesz duży program, zajmujący niemal całą pamięć i nie pozostawiający miejsca na interpreter – wówczas użyj kompilatora (patrz fotografia obok, pokazująca obraz z ekranu podczas procesu kompilacji programu napisanego w języku C). Ta technika udostępnia translatorowi program źródłowy w całości, w rezultacie może on wykryć większość ewentualnych błędów. Ponadto kompilator może optymalizować[2] kod wynikowy, to

[1] Przykładowo badasz nowy algorytm i doskonalisz jego działanie próbując, poprawiając i sprawdzając efekty, albo uczysz się programowania i liczysz się z koniecznością wielokrotnego poprawiania programu z powodu błędów.

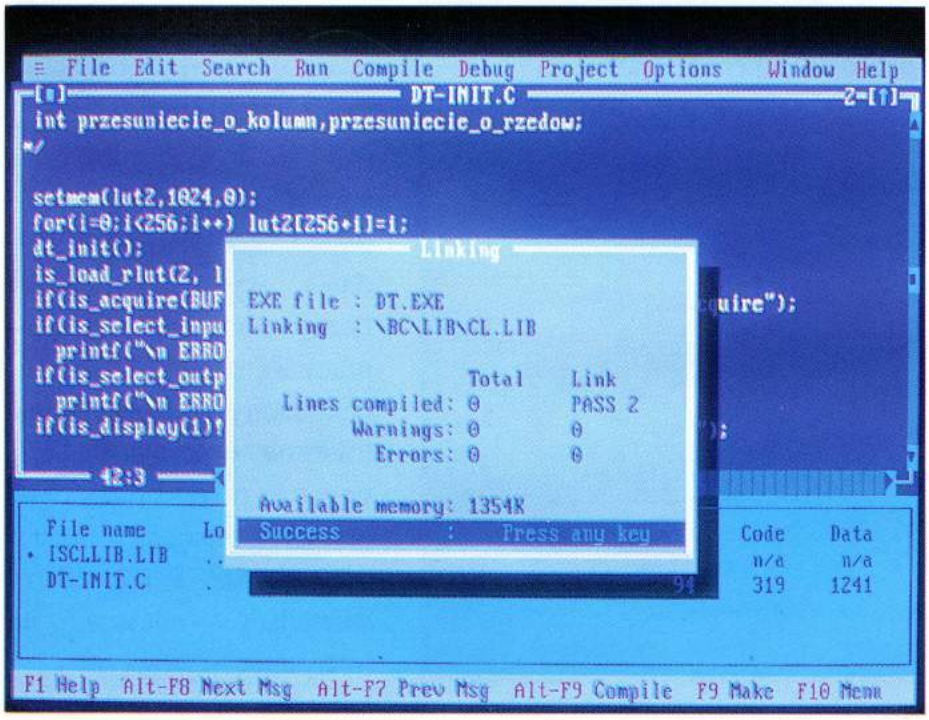
[2] Nawiasem mówiąc optymalizacja dokonywana jest zwykle na żądanie, gdyż praca kompilatora przy poszukiwaniu możliwości optymalizacji programu trwa bardzo długo. W połączeniu z wcześniej podanym faktem, że w ogóle translacja jest bardziej czasochłonna, niż wykonanie programu, stosowanie optymalizacji w sposób mechaniczny może doprowadzić do tego, że zyskasz ułamki sekund na czasie wykonania, tracąc całe minuty na bardziej wyrafinowaną pracę translatora.

znaczy tak poprawiać program napisany przez Ciebie, by uzyskać dalsze przyspieszenie jego działania lub zapewnić maksymalnie ekonomiczną gospodarkę pamięcią.

Program otrzymywany z kompilatora (program wynikowy) wytwarzany jest zwykle na którymś z urządzeń pamięci zewnętrznej komputera (typowo – na dysku magnetycznym), ponieważ w trakcie kompilacji pamięć operacyjna zajęta jest przez (bardzo duży z reguły) program kompilatora i (ewentualnie) źródłowy tekst przetwarzanego programu. Program po kompilacji wymaga uzupełnienia[1], co przeprowadza program, nazywany **linker** (polska nazwa "program łączący" jest niestety mało popularna). Ekran komputera podczas pracy tego programu przedstawiono na fotografii.

```
C>link /CO /ST:4096 huta_pr.obj, huta_pr.exe, NUL, c:\hnc\clib\luisl.lib
clib\lutil.lib c:\iris61\isc11ib.lib

Microsoft (R) Segmented Executable Linker  Version 5.30
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Definitions File [nul.def]:
Microsoft Debugging Information Compactor  Version 4.00.00
Copyright(c) 1987-1992 Microsoft Corporation

Line/Address size   =     1668
Public symbol size  =    11660
Initial symbol size =     3904
Final symbol size   =     1212
Global symbol size  =     3032
Initial type size   =     5446
Compacted type size =     5784

C>_
```

Wada techniki kompilacyjnej polega w pierwszym rzędzie na tym, że nawet najdrobniejsza zmiana w programie wymaga przejścia całego cyklu pracy[2]. Przy uruchamianiu programu jest to bardzo duża niedogodność, gdyż stale w użyciu muszą być kolejno różne programy: edytor do wprowadzania poprawek, kompilator do tłumaczenia kolejnych wersji tworzonego programu i linker. To ciągłe "żonglowanie" wieloma programami jest tak niewygodne, że w wielu wypadkach użytkownik rezygnuje ze stosowania kompilacji na rzecz wygodniejszej, choć mniej wydajnej techniki interpretacyjnej. Alternatywą jest koncepcja[3] firmy *Borland* tzw. **zintegrowane środowisko programowe**, (patrz obok) zawierające wszystkie niezbędne programy: edytor, kompilator, linker itd. Programy te "same" się ładują w miarę potrzeby – na przykład wykrycie błędu



---

[1] Ta sprawa znowu warta jest kilku słów komentarza. Pisząc program w określonym języku, z reguły używamy jako elementarnych poleceń operacji o dość dużym stopniu złożoności. Przykładowo tworząc wyrażenia korzystamy z funkcji sin lub log, a przy pisaniu wyników lub czytaniu danych nie zastanawiamy się nad tym, jak dokonać zamiany liczb z postaci dziesiętnej na dwójkową (w pamięci maszyny). Tymczasem do obliczenia wartości odpowiednich funkcji lub do wykonania potrzebnych czynności wymagane są konkretne programy – i programy te są brane ze zbiorów zwanych bibliotekami. Można wyróżnić biblioteki kompilatora (dostarczające podstawowych modułów uzupełniających program w wyżej omówionym zakresie), biblioteki standardowe, zawierające podprogramy dla realizacji typowych czynności (odwracanie macierzy, rozwiązywanie równań, całkowanie wyrażeń itp.) oraz własne biblioteki użytkownika. Wszystkie te biblioteki muszą być w momencie ładowania programu przeszukane, a potrzebne moduły muszą uzupełnić tekst programu.

[2] Z ponowną kompilacją **całego** programu, ładowaniem, uzupełnianiem tekstu bibliotekami itd.

[3] Zawarta w szeregu bardzo udanych i lubianych kompilatorów nazywanych **Turbo Pascal, Turbo C, Turbo Prolog.**

powoduje automatyczne uruchomienie edytora i załadowanie do niego tekstu programu celem wprowadzenia poprawek. Może jest to metoda przezwyciężenia klasycznej dwoistości interpreterów i kompilatorów?

Do programów narzędziowych pożytecznych dla zaawansowanego użytkownika należą programy umożliwiające śledzenie zawartości rejestrów procesora i wybranych komórek pamięci (lub zmiennych programu) podczas krokowego (rozkaz po rozkazie) wykonywania programu. Programy tego typu służą do poszukiwania błędów w tworzonych programach i stąd się bierze ich nazwa: **debugger** ("odpluskwiacz"). Z kolei do odczytania i modyfikacji kodu programu wynikowego służą programy nazywane **deassemblerami.** Pokazują one, jakim rozkazom odpowiada aktualna zawartość pamięci. Umożliwia to między innymi "zajrzenie" do systemu operacyjnego lub "podglądnięcie" sposobu działania translatora.

3.12. "Przyjazność" oprogramowania

Jedną z cech nowoczesnego oprogramowania jest jego "przyjazność"[1]. Ponieważ jest to ważna właściwość wszystkich dobrych programów – przeto na zakończenie tego rozdziału poświęcimy jej nieco uwagi. Postulat wymagający, by programy komputerowe były łatwiejsze w użyciu (*user friendly*) sformułowano stosunkowo dawno[2]. Od niedawna jednak istnieją warunki po temu, by tego typu postulat mógł być spełniony. Wiąże się to głównie ze wzrostem możliwości graficznych komputera ("przyjazność" zakłada graficzną komunikację człowieka z komputerem), a także z faktem rosnącej dostępności sprzętu komputerowego.

Przyjazne oprogramowanie legitymuje się zespołem czynników, określanych w literaturze jako **WIMP**. Skrót ten można rozszyfrować jako: **windows** (okienka), **icons** (ikony), **mouse** (myszka) i **pull–down menus** (rozwijalne menu). Uzupełnieniem atrybutów WIMP bywa w przyjaznym oprogramowaniu stale dostępna i możliwa do natychmiastowego użycia "ściągawka" (tzw. *context sensitive help*) oraz opcja znana pod dość skomplikowanym skrótem **WYSIWYG** (*What You See Is What You Get,* co po polsku bywa tłumaczone jako masz dokładnie to, co widzisz).

---

[1] Naukową podstawę dla systemów "przyjaznych" stanowiły prowadzone w latach sześćdziesiątych badania w ramach tzw. *Augmentation Research Project* (SRI Int.). Właśnie w tym projekcie po raz pierwszy jasno sformułowano myśl, że skoro w systemie człowiek – komputer centralną częścią jest człowiek, to jego możliwości psychofizyczne i preferencje powinny być podstawą wszelkich działań.

[2] Uważa się, że początki idei przyjaznego oprogramowania datują się od prac **Alana C. Kaya** z *Xerox Palo Alto Research Center* (**PARC**), a praktycznym wyrazem tych idei był zaproponowany przez niego język programowania **SMALLTALK**.

4. Zastosowania

4.1. Uwagi wstępne

Opisane podstawowe elementy systemu komputerowego, to znaczy sprzęt (rozdz. 2) i oprogramowanie (rozdz. 3) pozwalają obecnie spojrzeć na zagadnienie **zastosowań** tej techniki. Trzeba bowiem pamiętać, że ani komputer ani program nie są nigdy celem samym w sobie: ich sens zawarty jest w tym, do czego je **zastosujesz**. Już w poprzednim rozdziale wspominałem o zastosowaniach takich, jak redagowanie tekstów, wykorzystanie baz danych, używanie arkuszy kalkulacyjnych. Tak więc nie należy utożsamiać sfery zastosowań informatyki **wyłącznie** z niżej opisanymi zagadnieniami – chociaż im właśnie poświęciłem osobny rozdział, aby zaakcentować ich znaczenie[1].

Każda klasyfikacja zastosowań komputerów ma zawsze charakter niepełny i arbitralny. Jest ona niepełna, gdyż wciąż rodzą się nowe zastosowania i arbitralna, gdyż przyjmując inne kryteria można je poklasyfikować inaczej. Podane niżej uwagi mają na tyle ogólny i orientacyjny charakter, że zapewne nie przeszkodzą Ci w samodzielnym wyrobieniu sobie zdania na temat zastosowań techniki komputerowej, ale pomogą w "osadzeniu" nowych wiadomości w pewnych ustalonych ramach.

Zanim przystąpimy do dyskusji zagadnień szczegółowych celowe jest wprowadzenie pewnego ogólnego podziału systemów komputerowych ze względu na sposób ich stosowania. Otóż w każdej dziedzinie zastosowań komputer musi mieć dostarczane określone dane i produkuje określone wyniki. Biorąc za podstawę sposób dostarczania danych i metodę dystrybucji wyników możemy wyróżnić generalnie dwa typy systemów.

---

[1] Zabawne i interesujące może być przypomnienie faktu, że możliwości wszechstronnych zastosowań komputerów okazały się najbardziej zaskakujące dla ... ich producentów. Często cytowana jest, pochodząca z **1948** roku, wypowiedź **Thomasa J. Watsona**, prezesa firmy IBM (obecnie 85% światowej produkcji komputerów!): *"Świat potrzebuje najwyżej tuzin komputerów"*.

Pierwsze nazywane są **systemami o działaniu bezpośrednim** lub częściej z angielska systemami ***on–line*** (patrz fotografia obok). Ich cechą charakterystyczną jest fakt, że **dane są wprowadzane bezpośrednio tam, gdzie powstają**[1], **a wyniki są dostarczane bezpośrednio tam, gdzie są potrzebne**[2]. Komputer jest tu włączony w obieg informacji i jest jego niezbędnym elementem – stąd angielska nazwa.

Drugi typ systemu (nie pokazny, bo jest już dziś anachronizmem) mamy w przypadku, kiedy komputer znajduje się w wydzielonym ośrodku obliczeniowym, do którego wszystkie niezbędne dane **są dostarczane w postaci tradycyjnych (papierowych) dokumentów** i dopiero w samym ośrodku wprowadzane są do komputera przez autoryzowany personel z wykorzystaniem maszynowych nośników informacji. Podobnie wyniki produkowane przez komputer na drukarce **rozsyłane są do zainteresowanych** w postaci odpowiednich dokumentów – na przykład pocztą albo za pomocą specjalnych posłańców. Ten drugi przypadek określany jest terminem ***off–line*** (polskiego terminu brak).

Porównując obie wymienione wyżej metody pracy, stwierdzamy, że pierwsza z nich jest zdecydowanie wygodniejsza dla użytkowników komputera, natomiast związana jest ze zwiększonymi wymaganiami odnośnie wyposażenia i mocy obliczeniowej komputera. Druga – przeciwnie – gwarantuje dużą wydajność wykorzystania komputera, któremu zadania można dostarczać równomiernym, rytmicznym strumieniem, lecz wiąże się z dużym dyskomfortem dla korzystających z usług komputera ludzi.

Jak wspomniałem na wstępie, różnorodnych zastosowań komputerów jest bardzo wiele, zatem nie sposób ich wszystkich wyczerpująco omówić. W tym rozdziale skupimy uwagę na kilku zaledwie – za to szczególnie ważnych. Oczywiście lista omówionych zastosowań nie jest kompletna, gdyż kompletna być nie może. Nie omówię– między innymi – zastosowań komputerów w małej poligrafii (tzw. *desktop publishing*), roli komputerów w edukacji i dziesiątków innych zagadnień szczegółowych. Jednak nawet ten zakres zagadnień, który omówiłem, już skłania do refleksji, że komputer nadaje się – właściwie do wszystkiego!

4.2. Obliczenia numeryczne

Klasyczne i najstarsze zastosowanie komputera wiązało się z zadaniem wykonywania obliczeń. Z reguły były to skomplikowane obliczenia prowadzone według złożonego algorytmu i ukierunkowane na uzyskanie jakiegoś jednorazowego ustalonego wyniku. Przykłady takich obliczeń wiążą się z reguły z naukami ścisłymi (zwłaszcza fizyką i chemią) z techniką lub z wojskowością (pierwsze komputery wykorzystywano do precyzyjnego obliczania torów lotu pocisków). Wskażmy na podstawowe cechy takich zastosowań.

---

1 Np. w kasie sklepowej lub przy wadze w magazynie.

2 Np. na biurko dyrektora albo do zaopatrzeniowca, który ma uzupełnić zapasy.

- Obliczenia mają charakter jednorazowy. Po obliczeniu wyniku nie ma potrzeby dalszego wykorzystywania tego samego programu, gdyż kolejne zadanie wymaga stworzenia nowego programu.
- Algorytm obliczeń jest skomplikowany, a liczba operacji, jakie trzeba wykonać, aby otrzymać wymagany wynik jest ogromna. Bardzo ważna jest więc duża szybkość procesora i stosowanie języków dających możliwość zaprogramowania najbardziej złożonych algorytmów.
- Liczba danych wejściowych, potrzebnych do rozpoczęcia obliczeń jest niewielka. Mała jest także liczba produkowanych przez komputer wyliczeń. Nie ma żadnych szczególnych wymagań odnośnie postaci danych czy formy wyników.

Na fotografiach pokazano obraz wyposażenia komputerowego dużego naukowego centrum obliczeniowego "Cyfronet". Podana charakterystyka implikuje następujące wymagania odnośnie sprzętu (w "Cyfonecie" jest nim widoczny na fotografii tzw. superkomputer CONVEX) i oprogramowania dla takiego ośrodka:

- zasadnicze znaczenie mają tu parametry procesora (szybkość

obliczeń, dokładność, pojemna pamięć operacyjna umożliwiająca ulokowanie dużego i skomplikowanego programu itp.);

- małe wymagania wiążą się z urządzeniami peryferyjnymi (nie jest konieczne stosowanie szybkich urządzeń wprowadzających dane lub szczególnie wyrafinowanych urządzeń wyprowadzających wyniki, bardzo pojemnych pamięci masowych i rozbudowanych sieci telekomunikacyjnych);
- w oprogramowaniu najważniejsze są sprawne i wydajne translatory wielu różnych języków programowania, a także inne elementy oprogramowania narzędziowego wspomagającego tworzenie coraz to nowych programów.

Ponieważ ten typ zastosowań (to znaczy obliczenia numeryczne) dominował przez pierwsze dziesięciolecia rozwoju informatyki, nic dziwnego, że postęp w zakresie sprzętu odbywał się nierównomiernie: obserwowaliśmy niewiarygodny rozwój mocy obliczeniowej jednostki centralnej przy umiarkowanym postępie w zakresie właściwości i cen urządzeń peryferyjnych. Dopiero w latach 80. rozwój innych zastosowań komputerów wymusił nową skalę wartości, co natychmiast zaowocowało postępem w zakresie urządzeń peryferyjnych. Małe, sprawne i tanie drukarki, dyski, ekrany – to efekt przesunięcia uwagi konstruktorów, zafascynowanych dotychczas procesorem, na urządzenia peryferyjne. Obliczenia numeryczne były i są nadal najambitniejszym obszarem zastosowań informatyki. To tutaj rodzą się i rozwijają podstawowe idee i koncepcje zarówno w zakresie sprzętu, jak i oprogramowania. Jednak z punktu widzenia liczby zaangażowanych komputerów czy liczby użytkowników – jest to margines zastosowań techniki obliczeniowej.

4.3. Przetwarzanie danych

Niekwestionowany prymat pod względem liczby komputerów ma dziedzina **przetwarzania danych.** Wszelkie zastosowania biurowe, związane z przyjmowaniem i rejestrowaniem określonych informacji oraz sporządzaniem na ich podstawie raportów, zestawień, wyciągów, analiz itp. – to właśnie przetwarzanie danych. Na fotografii obok pokazano komputery pracujące w wielkim banku, a na drugiej stronie zobaczyć można komputer pracujący w małym sklepiku. Także na następnej stronicy pokazano skomputeryzowany gabinet dyrektora i komputery na dworcu kolejowym. Obszarów zastosowań jest tu co niemiara, wymienię jedynie dla przykładu kilka z nich:

- gospodarka finansowa w przedsiębiorstwach i obsługa kont w bankach,
- ewidencja obrotów magazynowych w fabrykach i w sklepach,
- zagadnienia kadrowe,
- problematyka remontów,
- ewidencja zapasów,
- wspomaganie zarządzania,

- operatywne informowanie kierownictwa o stanie instytucji.

Można wprowadzić pewną klasyfikację tych zadań, wyróżniając:

1. Zastosowania "operacyjne", wspomagające bieżące funkcjonowanie instytucji (np. obsługa kont w bankach, ewidencja materiałowa, rejestracja sprzedaży wraz z fakturowaniem i rozliczaniem odbiorców itp.).
2. Wspomaganie decyzji polegające na analizie danych i symulowaniu procesów zachodzących w przedsiębiorstwie w celu sprawdzenia skutków różnych decyzji.
3. Planowanie strategiczne będące kompleksowym przetwarzaniem danych na temat przewidywanej przyszłości.
4. Wspomaganie prac biurowych.[1]

W warunkach polskich dominuje i długo jeszcze dominować będzie pierwszy typ zadań, przeto dalsze uwagi w zasadniczy sposób odnosić się będą do tego zastosowania. Jednak ograniczenie rozważań do jednego podobszaru zadań przetwarzania danych nie oznacza ograniczenia sfery naszych zainteresowań. Przeciwnie, właśnie te zadania stanowią najbardziej powszechny model wykorzystania komputera, decydujący o sposobie widzenia tego urządzenia przez większość użytkowników i determinujący politykę handlową oraz preferencje techniczne największych producentów komputerów.

Właśnie **powszechność zastosowań** powoduje tak duże zainteresowanie przetwarzaniem danych, gdyż system informatyczny, który nadaje się do zastosowania zarówno w małym sklepie, jak i w wielkiej fabryce, do ewidencji pacjentów w szpitalu i do księgowości w banku,

[1] Zadanie to obejmuje narzędzia usprawniające tworzenie i wykorzystanie podręcznych zbiorów danych, opracowywanie korespondencji, obliczenia i inne formy przetwarzania tekstów, a przede wszystkim komunikację (przesyłanie pomiędzy stanowiskami "dokumentów elektronicznych" bez użycia papieru).

do obsługi sprawozdawczości wymaganej przez zwierzchników i do własnej kontroli zapasów, a nawet mogący znaleźć zastosowanie w szkole (patrz fotografia obok) – taki system się doskonale sprzedaje!

Od strony intelektualnej systemy przetwarzania danych nie są szczególnie fascynujące[1] – zwłaszcza w zestawieniu z wyżej omówionymi zastosowaniami obliczeń numerycznych. Ich cechy charakterystyczne są następujące:

- proces przetwarzania danych jest bardzo "płytki": na pojedynczej jednostce danych wykonuje się najwyżej kilka operacji nie wykraczających poza podstawowe działania arytmetyczne[2],
- liczba przetwarzanych danych jest duża, co warunkuje celowość użycia komputera,
- wprowadzane dane pochodzą od różnych ludzi i dlatego mogą zawierać błędy – konieczne są rozbudowane systemy kontroli,
- wyniki pracy komputera stanowią dokumenty dostarczane ludziom nie mającym zawodowej styczności z komputerem, dlatego bardzo ważna jest czytelność i estetyka wydruków (przy pracy w trybie *off–line*) lub optymalne z ergonomicznego punktu widzenia rozmieszczenie informacji na ekranie przy korzystaniu z systemu *on–line*,
- konieczne jest oparcie przetwarzania na bardzo dużych bazach danych gromadzonych w pamięciach masowych, do których zalecany jest dostęp swobodny,
- z reguły wykorzystywany jest stale jeden program, raz napisany (lub zaadaptowany do rozważanego zadania) i nie zmieniany potem latami[3].

Z podanej charakterystyki wynika, że wymagania sprzętowe i programowe, związane z omawianym tu zastosowaniem, są zdecydowanie odmienne niż w przypadku systemów dla obliczeń numerycznych. Na plan pierwszy wybijają się wymagania dotyczące urządzeń peryferyjnych: liczne i szybkie czytniki danych przygotowanych na nośnikach maszynowych, dobrej jakości wydajne drukarki, szybkie i pojemne pamięci masowe. W przypadku systemu pracującego w trybie *on–line* potrzebne są dodatkowo liczne konsole rozmieszczone w punktach wprowadzania danych i wyprowadzania wyników oraz stosowna sieć łączności. Wprawdzie ani charakter wykonywanych operacji, ani ich liczba nie dorównują złożoności i liczbie obliczeń w systemach numerycznych, jednak kolosalna masa danych, z jakimi muszą borykać się systemy przetwarzania, a także konieczność obsługi wielu końcówek, które potencjalnie mogą zgłaszać się równocześnie, zmuszają do stosowania dużych (w sensie mocy obliczeniowej) i wydajnych procesorów także i w tym zastosowaniu.

W stosunku do oprogramowania systemów przetwarzania danych stwierdzić należy, że ważny jest tu dobry system operacyjny (obsługa wielodostępu) oraz gotowe programy do konkretnych celów (zarządzanie danymi, generowanie raportów itp.). Natomiast oprogramowanie narzędziowe jest praktycznie zbędne, ponieważ rozważane systemy pracują przeważnie jako stałoprogramowe.

---

1 Dotyczy to zwłaszcza zadań wymienionych w punkcie 1.

2 Na przykład dodanie nowej dostawy do stanu zapasów lub obliczenie należnych odsetek od udzielanego przez bank kredytu.

3 Oczywiście od tej reguły bywają wyjątki: fluktuacje otoczenia (na przykład zmienne przepisy finansowe), a także zmiany potrzeb użytkowników (powstające w miarę zdobywania przez nich umiejętności korzystania z narzędzi informatycznych) – mogą wymuszać pewne modyfikacje używanych programów, chociaż nie tak radykalne, jak w przypadku systemów dla obliczeń numerycznych.

4.4. Bazy danych

Specyficznym rodzajem systemu przetwarzania danych jest baza danych. O tym rodzaju zastosowań komputera była już mowa w podrozdz. 3.5, ale temat ten jest tak ważny, że należy go powtórnie omówić w kontekście przeglądu zastosowań systemów informatycznych, czyli właśnie w tym rozdziale. Wynika to z jednego, obserwowanego od wielu lat faktu: chęć uporządkowania posiadanych informacji w formie komputerowej bazy danych jest najczęściej występującym powodem zakupu pierwszego komputera w określonej instytucji. Potem, gdy komputer już jest – przychodzi czas na inne zastosowania, jednak decyzja o zakupie jest zwykle wynikiem zmęczenia ciągłym poszukiwaniem wciąż ginących dokumentów i świadomością, że tylko automatyzacja gromadzenia i wyszukiwania danych może istotnie podnieść efektywność pracy biurowej – niezależnie od tego, czy wiąże się ona z rozliczeniami finansowymi, gospodarką materiałową, ewidencją remontów, katalogiem części zamiennych czy wykazem własnych pracowników.

Schemat systemu informatycznego będącego bazą danych jest w istocie identyczny z opisanymi wyżej schematami systemów przetwarzania danych, gdyż uwarunkowania tu występujące są w istocie identyczne z tymi, jakie występują w systemie przetwarzania: duże strumienie wejściowych i wyjściowych danych, które trzeba wprowadzać, wyprowadzać i gromadzić. Jedyna różnica polega na tym, że w systemie bazy danych przetwarzanie danych jest bardzo "płytkie" – najczęściej dane są po prostu wyszukiwane zgodnie z określonym kluczem i udostępniane użytkownikowi (w trybie *on–line*) lub drukowane w zbiorczym raporcie (w trybie *off–line*).

Jednak fakt braku jakichkolwiek obliczeń wykonywanych na informacjach zgromadzonych w bazie danych nie zmniejsza w istotny sposób zapotrzebowania na moc obliczeniową komputera, ponieważ nakład pracy związany z wyszukiwaniem i (zazwyczaj także wymaganym) sortowaniem danych jest porównywalny z tym, jaki jest wymagany w systemach przetwarzania. W dodatku często na danych wyszukanych w bazie trzeba wykonywać pewne obliczenia (na przykład statystyczne), a to zwiększa i tak spore wymagania odnośnie mocy jednostki centralnej.

Bazy danych dzieli się na scentralizowane (z pojedynczym komputerem gromadzącym i aktualizującym informacje i licznymi komputerami udostępniającymi dane zainteresowanym użytkownikom) oraz na rozproszone (z wieloma "inteligentnymi" węzłami, z których każdy gromadzi dane potrzebne z punktu widzenia jego bieżącej pracy, a w razie potrzeby może "importować" dane z odległych węzłów, jeśli tego wymaga aktualnie postawione pytanie użytkownika). Bazy scentralizowane są technicznie prostsze i łatwiejsze do tworzenia i aktualizacji, natomiast bazy rozproszone mogą lepiej służyć zdecentralizowanej administracji i mogą minimalizować liczbę przesyłanych między węzłami informacji.

Kierunek rozwoju systemów baz danych, obserwowany w rozwiniętych i uprzemysłowionych krajach świata, zdecydowanie wskazuje na tendencję do zastępowania baz scentralizowanych bazami rozproszonymi. Jednak w Polsce przy kosztownym i niewydolnym systemie telekomunikacji długo jeszcze dominować będą bazy scentralizowane.

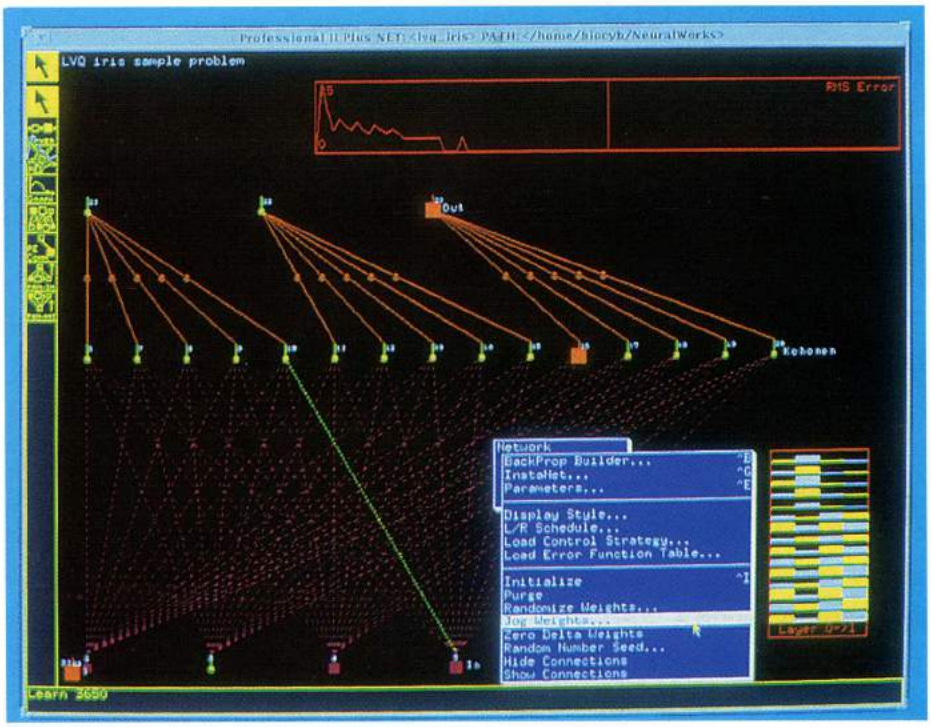
4.5. Symulacja i modelowanie

Komputer może służyć do modelowania różnych systemów i zjawisk, dzięki czemu przygotowując jakieś działanie możemy sprawdzić jego skutki na modelu symulacyjnym, co jest znacznie bezpieczniejsze i tańsze, niż podejmowanie określonych prób w praktyce.

Komputerowe modele służyć mogą trzem zasadniczym celom:

➢ **prognostycznym**[1]; ➢ **diagnostycznym**[2]; ➢ **dydaktycznym**[3].

W ekonomii wykorzystywane są głównie funkcje prognostyczne i dydaktyczne budowanych modeli, natomiast funkcja diagnostyczna ma większe znaczenie przy wykorzystywaniu modelowania w technice (na przykład badanie przyczyn wypadków lotniczych), w biologii (zastępowanie uciążliwych i trudnych badań żywych organizmów eksperymentem symulacyjnym) i w fizyce (próby wyjaśnienia za pomocą badań modelowych procesu ewolucji wszechświata lub zjawisk zachodzących w niemierzalnych ułamkach sekundy wewnątrz jądra atomowego).



Zasada działania symulacyjnego modelu dowolnego systemu lub zjawiska polega na tym, że komputer oblicza w kolejnych krokach wartości sygnałów opisujących modelowany system. Najczęściej eksperymentator podaje sygnały wejściowe, a komputer oblicza i drukuje (lub prezentuje w formie wykresów) sygnały wyjściowe i wewnętrzne. Bardzo ważnym elementem procesu symulacji komputerowej jest wybór kroku[4]. Możliwe jest stosowanie **kroków o jednakowej długości** lub **kroków dostosowanych do zdarzeń zachodzących w modelowanym systemie.** Odpowiednio do tego podziału wyróżnia się **symulację ciągłą** (patrz przykład na górnej fotografii) i **symulację dyskretną** (przykład na fotografii obok).

Przy symulacji ciągłej model komputerowy naśladuje jak najwierniej wszystkie procesy zachodzące w rozważanym systemie (na przykład tor lotu rakiety), a krok symulacji wybiera się tak, żeby nie

---

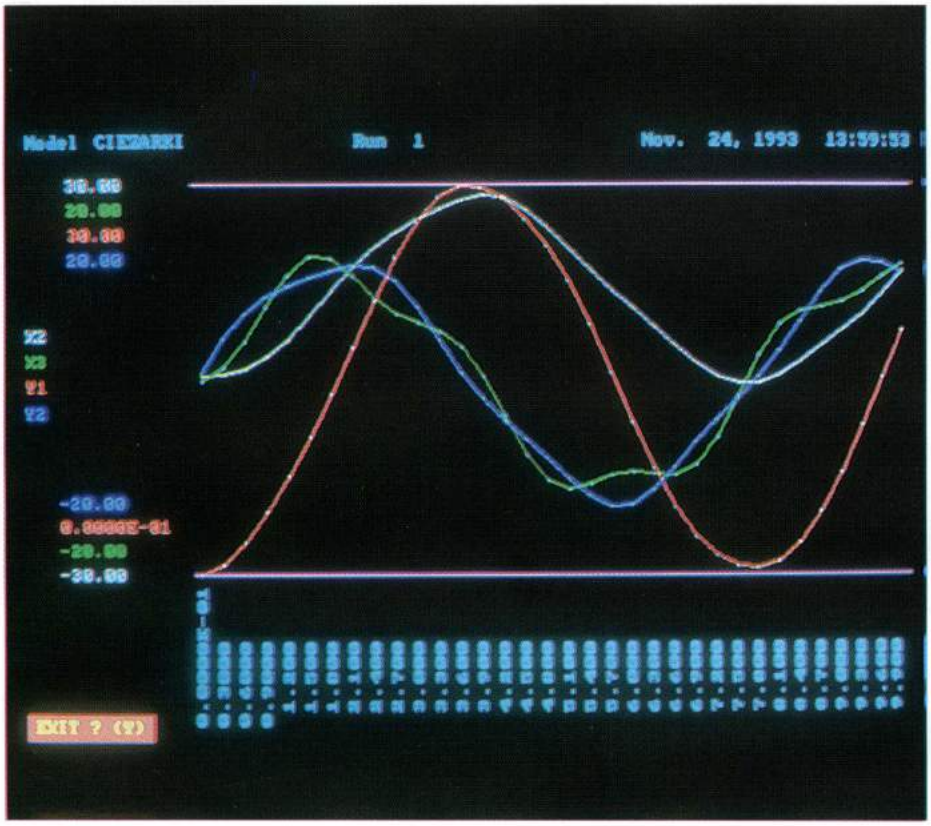
1 Za pomocą modelu można zbadać, co się zdarzy w przyszłości, i w oparciu o tą wiedzę można zmodyfikować podejmowane działania lub zoptymalizować tworzoną strukturę.

2 Model pomaga znaleźć przyczyny obserwowanych zjawisk lub dokładniej oraz taniej prześledzić ich przebieg.

3 Model pozwala tanio i bezpiecznie zdobyć doświadczenie wymagane przy wykonywaniu wielu prac.

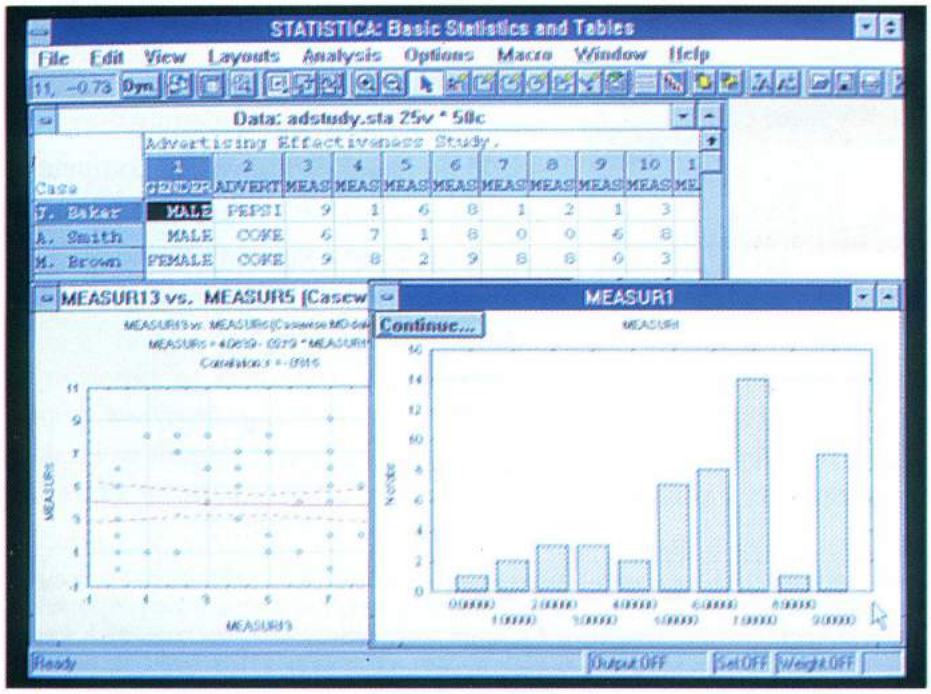
4 Krok modelowania odpowiada odcinkowi czasu, upływającemu w rozważanym systemie pomiędzy kolejnymi momentami, w których wyznaczany jest w modelu komplet wszystkich sygnałów.

zakłócić ciągłości modelowanych zjawisk (w praktyce ma on wtedy wartość ustaloną i z reguły bardzo małą). Przy symulacji dyskretnej wyróżnia się w modelowanym systemie tak zwane zdarzenia (na przykład pojawienie się klienta w sklepie) i dostosowuje się krok modelowania do zmiennego rytmu zachodzących zdarzeń. Podział na modele ciągłe i dyskretne jest tak ważny, że do celów symulacji opracowano szereg specjalistycznych języków programowania, ale oczywiście oddzielnie rozwijane są języki do symulacji ciągłej, a osobno języki modelowania dyskretnego. Inaczej się także korzysta z wyników symulacji ciągłej, a inaczej z rezultatów uzyskanych przy symulacji dyskretnej.



W symulacji ciągłej eksperyment symulacyjny dostarcza ogromnej liczby wyników, ponieważ ustalony i bardzo mały krok symulacji powoduje, że komputer produkuje miliony liczb, które z reguły trzeba najpierw zapamiętać na dyskach magnetycznych, a dopiero potem można je analizować i interpretować, do czego z reguły konieczne jest przedstawienie tych wyników w **formie graficznej** – jak na fotografii obok. Stosuje się wykresy, histogramy, wielobarwne mapy, rysunki trójwymiarowe, animację komputerową itp.

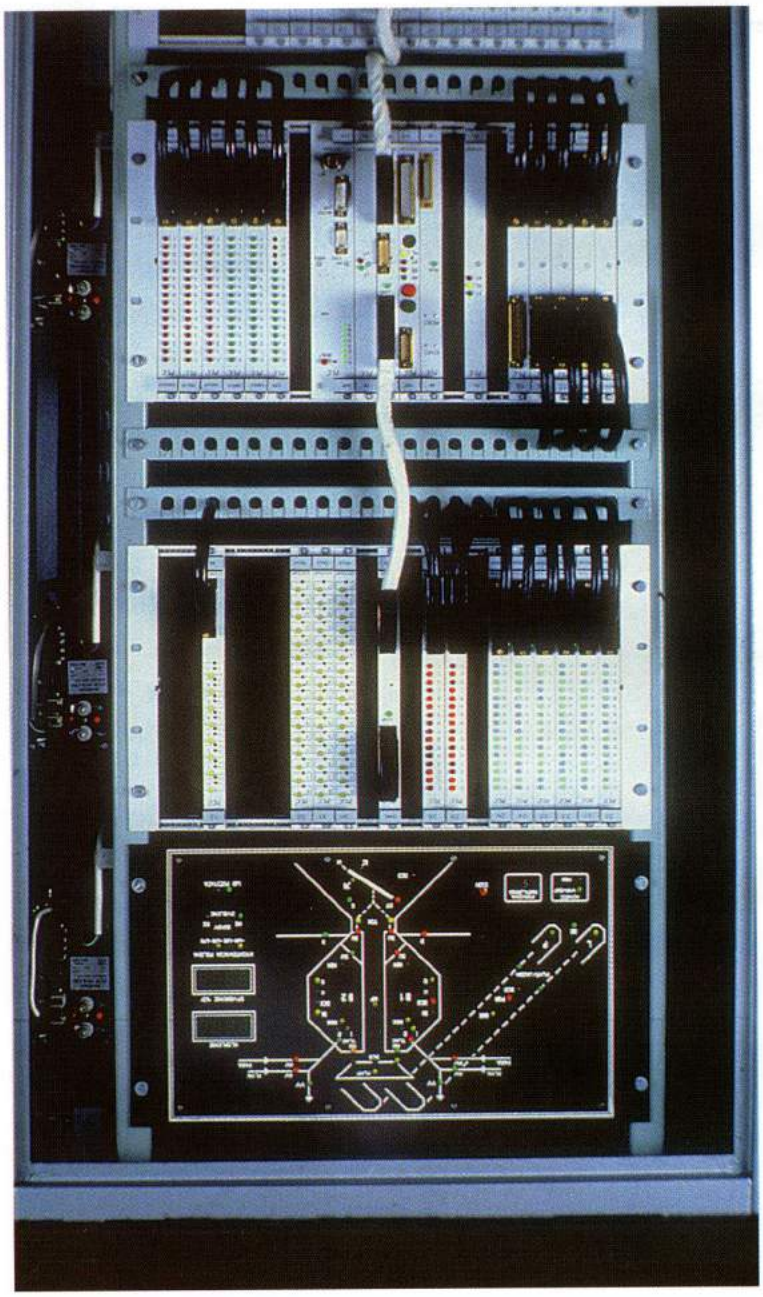
W modelach dyskretnych liczba wyników uzyskiwanych z pojedynczego eksperymentu jest znacznie mniejsza (ponieważ rejestruje się tylko zachodzące w systemie zdarzenia). W modelach tych uwzględnia się z reguły **indeterminizm**, czyli przypadkowość pewnych zdarzeń i ich atrybutów (na przykład czasu trwania), zatem korzystając z symulacji dyskretnej wykonuje się zwykle ciąg badań symulacyjnych (wiele razy modeluje się ten sam system w tych samych warunkach), a wyniki opracowuje się metodami statystyki matematycznej. Przykład takiego opracowania widoczny jest na fotografii obok.



Z punktu widzenia wymagań stawianych systemowi komputerowemu symulacja porównywalna

jest z omówioną wcześniej dziedziną obliczeń numerycznych[1]. Natomiast odmienność zadań symulacji od zadań numerycznych polega na produkowaniu przez program symulacyjny ogromnej liczby wyników.

4.6. Sterowanie procesów



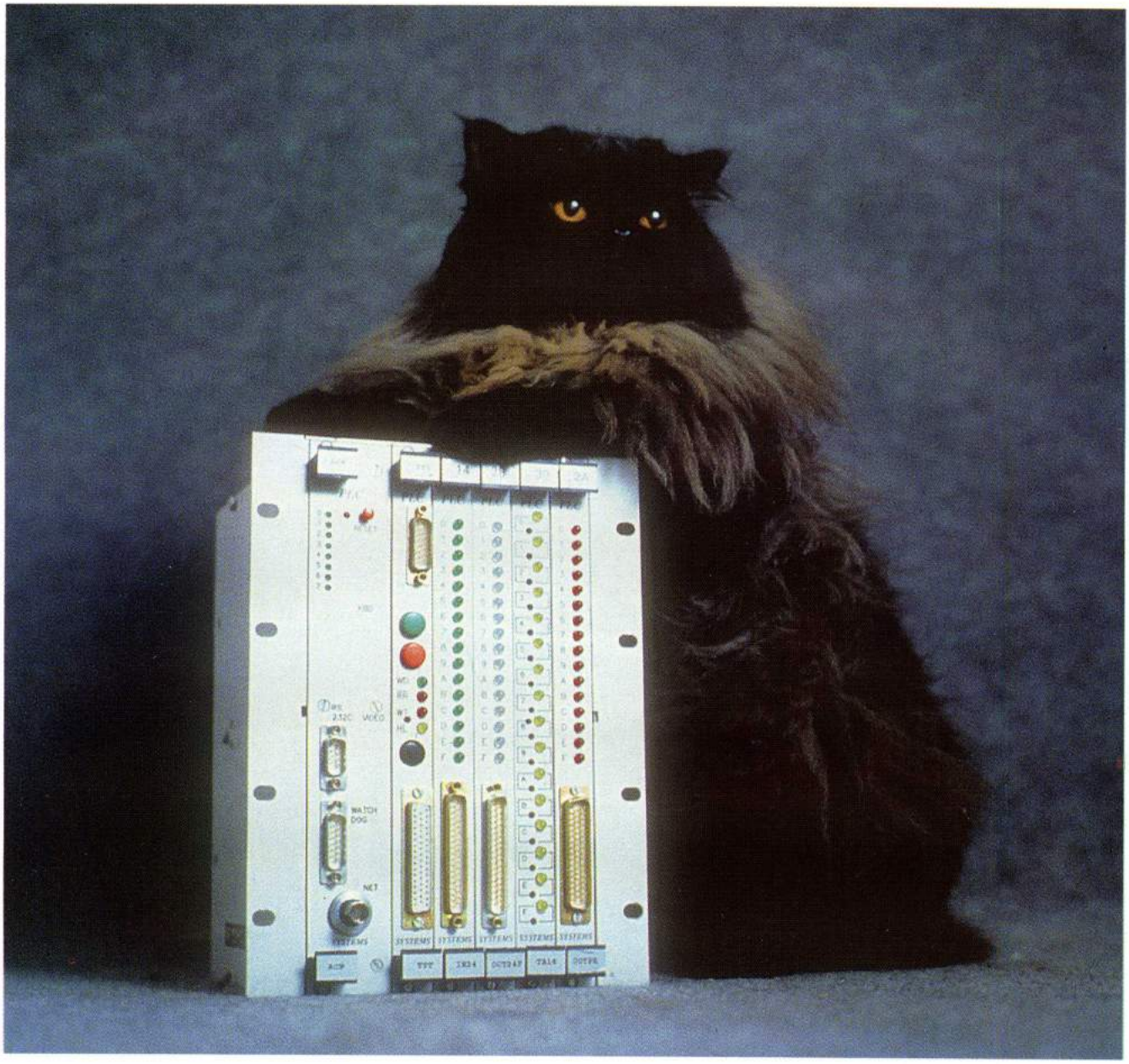
Całkowicie odmiennym od wszystkich już omówionych jest zastosowanie komputera do sterowania procesów. Na fotografii obok możesz zobaczyć taki komputerowy system do sterowania procesu metalurgicznego – prawda, że jest całkiem niepodobny do zwykłych komputerów? Jego "sercem" jest specjalny komputer pokaznany na następnej stronicy (kot nie jest elektroniczny, tylko służy do pokazania, jakie to wszystko jest małe!).

Pojęcie **proces** należy tutaj rozumieć maksymalnie szeroko: będzie nim każdy proces wytwórczy lub transportowy, a także zautomatyzowane magazynowanie wyrobów lub ich pakowanie za pomocą robotów. Jako sterowanie procesów można uznać wykorzystanie komputera do automatycznego prowadzenia samolotu, statku lub nawet samochodu. W tej kategorii mieści się także sterowanie eksperymentem laboratoryjnym, elektrownią, siecią energetyczną lub rakietą kosmiczną. W pewnym sensie do kategorii sterowania komputerowego należą także systemy wspomagające zarządzanie, a nawet systemy nauczające – jeśli do nauczania użyje się komputera, a ucznia traktować się będzie jak podlegający sterowaniu system.

Omawiane zastosowanie implikuje wiele specyficznych wymagań i uwarunkowań, które teraz spróbuję w skrócie omówić.

- Systemy sterowania wymagają specjalnych urządzeń do sprzężenia komputera ze sterowanym procesem. Urządzenia te, nazywane zwykle **sprzęgiem przemysłowym** (ang. *interface,* popularne jest także spolszczenie **"interfejs"**) mają za zadanie przekształcenie sygnałów z takiej postaci, w jakiej rejestrują je czujniki i przetworniki pomiarowe w rozważanym procesie, do

[1] Proces obliczania sygnałów występujących w modelu jest związany z wykonywaniem ogromnych ilości obliczeń, chociaż z reguły polega na wielokrotnym wykonywaniu tych samych działań "w kółko". Oznacza to, że wysoce pożądanym atrybutem jest przy symulacji komputerowej jak największa szybkość działania jednostki centralnej komputera. Z praktyki większości ośrodków obliczeniowych wiadomo zresztą, że obliczenia symulacyjne potrafią "zatkać" najszybsze komputery na wiele godzin i wykazują najwyższe wskaźniki zapotrzebowania na pamięć operacyjną.



takiej postaci, w jakiej może je przyjąć komputer. W uproszczeniu można powiedzieć, że przekształcenie to polega na:

➢ **próbkowaniu**[1], ➢ **konwersji A/C**[2], ➢ **multipleksowaniu**[3].

- Rola komputera w sterowanym procesie może polegać na samym tylko gromadzeniu danych i raportowaniu (na przykład dyspozytorowi) stanu procesu, względnie możliwe jest bezpośrednie sterowanie za pomocą komputera[4].

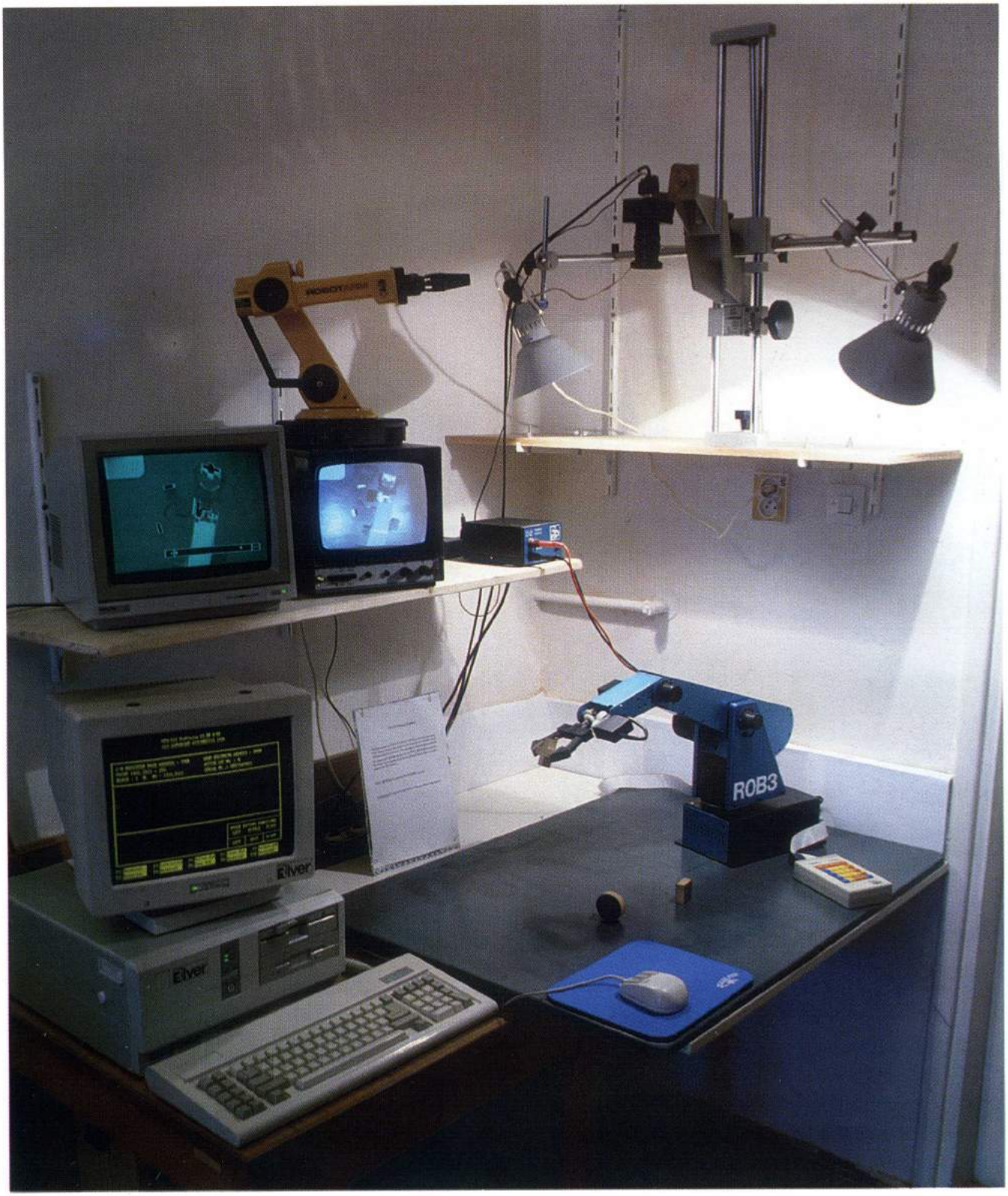
---

[1] **Próbkowanie** przekształca **sygnał ciągły** w **sygnał dyskretny** (to znaczy taki, którego wartości są określone jedynie w ustalonych, wybranych momentach czasu). Pojawia się przy tym problem, jaki ma być odstęp czasu pomiędzy kolejnymi próbkami. Mniejszy odstęp ("gęściejsze" próbkowanie) sprzyja dokładniejszemu śledzeniu sygnału, lecz prowadzi do większego obciążenia komputera, który musi wówczas przetwarzać więcej liczb. **Optymalne** próbkowanie może być określone na podstawie tzw. twierdzenia **Shannona**, którego omówienie jest tu niestety niemożliwe ze względu na brak miejsca.

[2] **Konwersja A/C** polega na przypisaniu **sygnałom** ustalonych **kodów wartości.** W wyniku konwersji sygnał, który w rzeczywistości zmienia się w sposób **ciągły i analogowy**, zamieniony zostaje na **kod** przesyłany do komputera, który przy ograniczonej liczbie bitów prowadzi do jedynie przybliżonego odwzorowania wejściowego sygnału (powstaje tzw. **szum kwantowania**). Mimo tej wady konwersja jest konieczna, gdyż tylko w postaci dyskretnych kodów można przesłać sygnał do komputera i tam go przetwarzać.

[3] **Multipleksowanie** polega na szybkim przełączaniu wejścia komputera miedzy licznymi wyjściami sterowanego procesu. Na skutek istnienia znacznej dysproporcji między powolnymi zjawiskami zachodzącymi w każdym realnym procesie, a szybkimi obliczeniami komputera – możliwe jest "obieganie" setek źródeł kontrolowanych sygnałów przez jeden kanał wejściowy komputera.

[4] W pierwszym przypadku mówimy niekiedy o systemach **CRPD** (centralnej rejestracji i przetwarzania danych) a w drugim o systemach BSC (bezpośredniego sterowania cyfrowego, częściej znanych jako systemy **DDC** od angielskiego określenia *Direct Digital Control*). Możliwe są także sytuacje pośrednie (na przykład systemy **doradzające** jak sterować procesem).



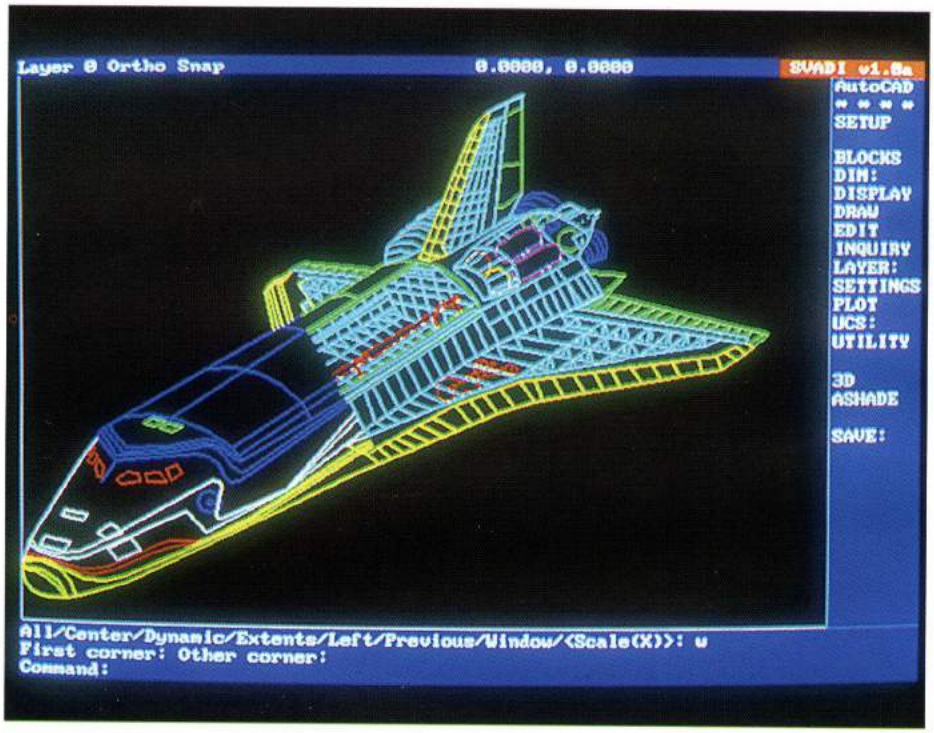
- Specyficzną cechą oprogramowania systemów do sterowania procesów jest konieczność tak zwanej pracy **w czasie rzeczywistym** (*real time*). Termin ten oznacza, że komputer musi uwzględniać w obliczeniach czynnik czasu i wysyłać sygnały sterujące dokładnie w tym momencie, kiedy są potrzebne – ani wcześniej, ani później. Jednym z parametrów każdej procedury obliczeniowej jest w tym wypadku czas odczytywany z wewnętrznego zegara, a wszystkie algorytmy muszą być tak dobrane, by mieściły się w dopuszczalnych interwałach czasowych[1].

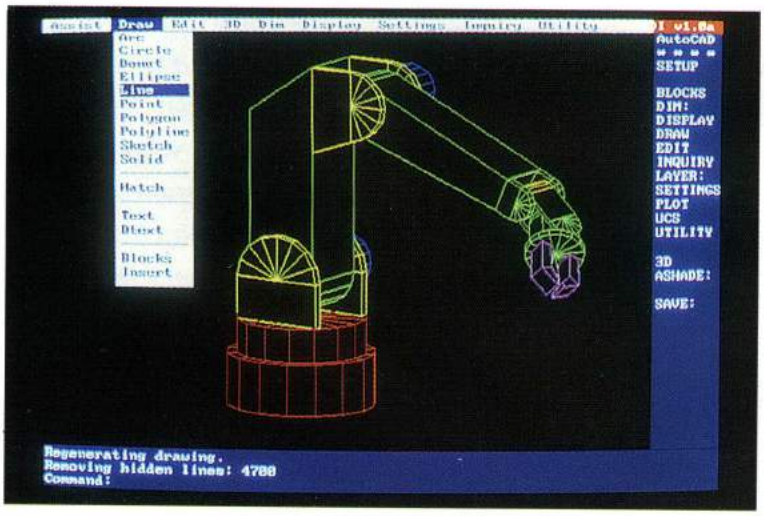
[1] Jest to wymóg bardzo kategoryczny, gdyż spóźniony sygnał sterujący może być przyczyną groźnej katastrofy, przeto prowadząc obliczenia trzeba każdorazowo pamiętać najlepszy z dotychczas uzyskanych wyników, gdyż może się zdarzyć, że konieczne będzie wysłanie sygnału sterującego **natychmiast** – nawet kosztem znacznej utraty dokładności.

- Centralnym problemem przy sterowaniu procesów jest **niezawodność.** Awaria komputera w każdym zastosowaniu jest przykrym wypadkiem, lecz w systemach sterowania (na przykład samolotu pasażerskiego) może oznaczać katastrofę[1].

Podana charakterystyka wskazuje na dominujące znaczenie sprzętu w komputerowym sterowaniu procesów. Warto podkreślić, że chodzi tu głównie o sprzęt specjalistyczny, (m.in. *interface)* związany z tym właśnie zastosowaniem. Oprogramowanie systemów sterowania ma drugorzędne znaczenie ze względu na fakt jednorazowego w praktyce pisania programu dla określonej instalacji sterującej i stałego wykonywania przez komputer tego właśnie jednego programu. Programy sterujące pisze się zazwyczaj w asemblerze, a ich cechy powinny odpowiadać omówionym, specyficznym wymaganiom wynikającym z faktu współpracy komputera z realnym procesem.

4.7. Projektowanie wspomagane komputerowo





Z tematyką sterowania procesów za pomocą maszyn cyfrowych ścisły związek ma, bardzo burzliwie rozwijająca się ostatnio, problematyka projektowania wspomaganego przez komputer. Dyscyplina ta, zwana w skrócie **CAD** (*Computer Aided Design),* obejmuje zarówno projektowanie maszyn i urządzeń (samochodów, samolotów, koparek itp.), jak i układów elektronicznych (między innymi komputerów). Dotyczy to również takich dziedzin, jak architektura, urbanistyka, budownictwo lądowe i wodne, a nawet produkcja butów i konfekcji i wielu innych. Komputer wspomaga projektanta przy obliczaniu parametrów tworzonej konstrukcji, pomaga w wyborze gotowych elementów wchodzących w skład konstrukcji (śruby, łożyska), a także "bierze na siebie" całość prac związanych z tworzeniem rysunków technicznych projektu. Służyć do tego mogą bardzo duże i precyzyjnie kreślące plottery (fotografia na następnej stronicy).

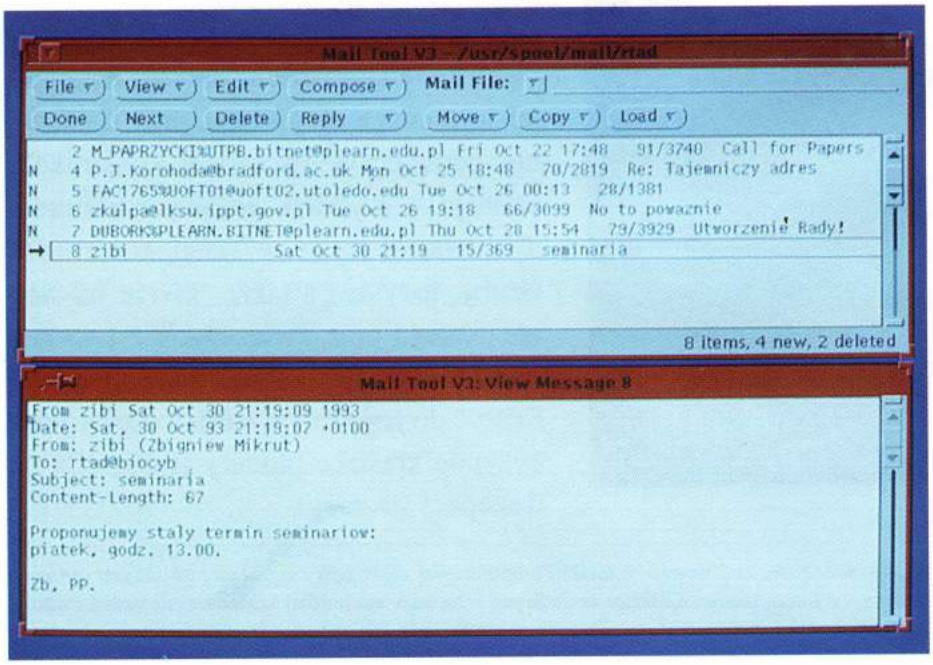
[1] Dlatego zresztą sterowanie procesów jako dziedzina zastosowań techniki komputerowej rozwinęła się później od innych tu omawianych obszarów: po prostu pierwsze komputery były urządzeniami bardzo zawodnymi i dopiero lata bardzo kosztownych badań i udoskonaleń (stymulowanych zresztą pierwotnie głównie przez zastosowania militarne) spowodowały tak znaczny wzrost pewności działania komputerów, że w wielu współcześnie używanych instalacjach przemysłowych komputer jest najmniej zawodnym ogniwem.

Łącząc ze sobą system **CAD** oraz zautomatyzowany system wytwórczy sterowany przez komputer, możemy otrzymać **system komputerowo wspomaganego wytwarzania** (**CAM** – *Computer Aided Manufacturing*), w którym wystarczy podać pomysł jakiegoś urządzenia, a komputer zaprojektuje je, a następnie całkowicie automatycznie wykona.

Podobne postępowanie można kontynuować: Do systemu **CAD + CAM** dołączyć można jeszcze **CIO** (skomputeryzowane biuro, ang. *Computer Integrated Office*) i **CIM** (skomputeryzowany system zarządzania, ang. *Computer Integrated Management*). Z połączenia tego typu systemów powstają już dziś całe "bezludne fabryki", oznaczane niekiedy jako **CII** (skomputeryzowane zakłady wytwórcze, ang. *Computer Integrated Industry*, oznaczane także niekiedy – niezbyt szczęśliwie ze względu na zdublowanie skrótu – jako **CIM** od *Computer Integrated Manufacturing*), które prawdopodobnie będą miały ogromny wpływ na kształt przyszłego świata i strukturę społeczeństwa przyszłości.

4.8. Korzystanie z poczty elektronicznej

Łączenie komputerów w sieci o coraz większym zasięgu prowadzi do pojawienia się nowej formy usług komputerowych w postaci tak zwanej **poczty elektronicznej** (znanej szeroko pod angielską nazwą *e–mail*). Dzięki poczcie elektronicznej można bardzo szybko i stosunkowo tanio przesyłać informacje do dowolnego punktu kuli ziemskiej, przy czym ten sposób prowadzenia korespondencji odznacza się znacznie większą szybkością i znacznie większą niezawodnością, niż inne sposoby telekomunikacji.



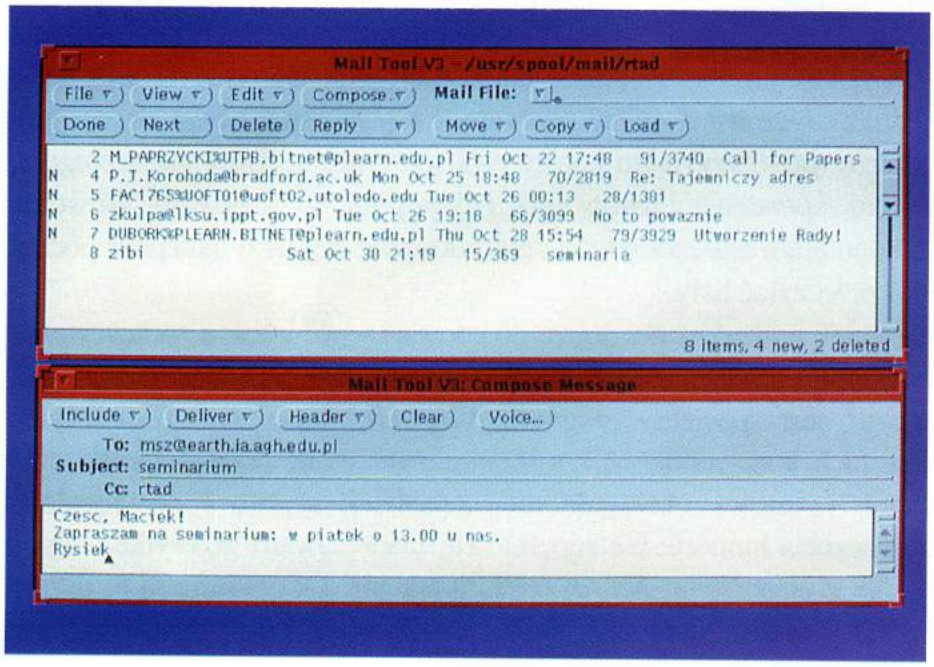
Zasada działania poczty elektronicznej jest bardzo prosta. Jeśli komputer ma w swoim wyposażeniu specjalny program do obsługi poczty elektronicznej, tzw. **mailer** (patrz fotografia obok), wówczas wystarczy uruchomić ten program i postępować zgodnie z jego wskazówkami.

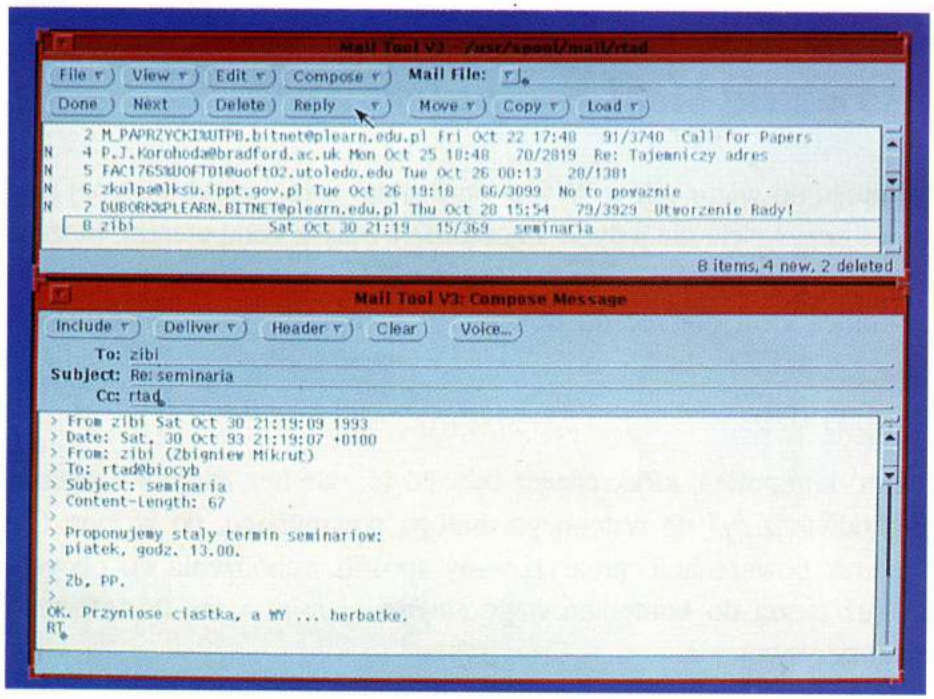
Adres odbiorcy przesyłki jest jego jednoznacznym identyfikatorem w sieci komputerowej. Na ogół składa się on ze skrótu

nazwiska odbiorcy, symbolu "@" oraz nazwy instytucji. Przykładowo można to obejrzeć w adresie sieciowym autora książki. Adres ten (osiągalny z dowolnego punktu kuli ziemskiej) ma postać:

RTAD @ BIOCYB.IA.AGH.EDU.PL

Kolejne fragmenty adresu są łatwe do odczytania. RTAD to skrót imienia i nazwiska, BIOCYB to identyfikator komputera, na którym założona jest "skrzynka na listy" (SUN3 w Zakładzie Biocybernetyki), IA to Instytut Automatyki, AGH to nazwa uczelni, EDU sygnalizuje, że chodzi o pion Edukacji Narodowej i wreszcie PL to identyfikator kraju.

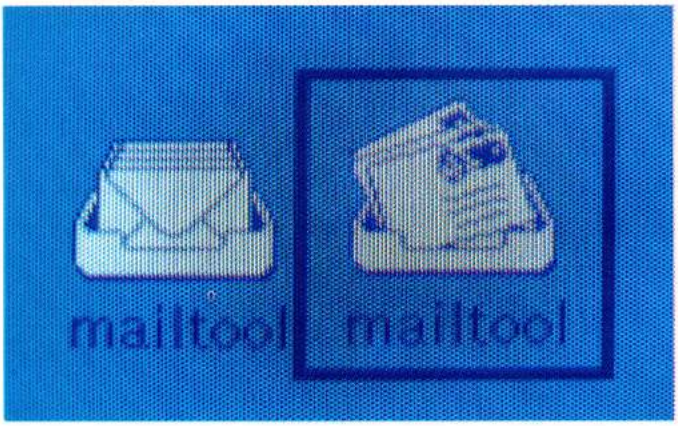




Adres sieciowy korespondenta, do którego chcesz przesłać elektroniczną pocztę musisz znać przed wysłaniem wiadomości[1]. Potem możesz go już łatwo stosować przy wysyłaniu kolejnych listów (patrz fotografia obok), ponieważ *mailer* zapamięta adresy korespondentów, do których często wysyłasz listy i wystarczy tylko wskazać właściwą pozycję w usłużnie podsuniętym wykazie, by cały adres został bezbłędnie skopiowany do nagłówka aktualnie wysyłanego listu. Mój *mailer* ma dodatkowo taką miła opcję, która pozwala natychmiast odpowiedzieć na aktualnie czytany list. Jeśli odczytuję list z poczty, która nadeszła na mój adres, to wystarczy że – jak na fotografii obok – wydam polecenie ***Reply*** (odpowiedz) i podam treść odpowiedzi, a program sam poprawnie zaadresuje przesyłkę, odczytując adres nadawcy z czytanego przeze mnie listu.

Poczta elektroniczna działa metodą "podaj dalej". Komputer wysyłający pocztę kontaktuje się ze swoim najbliższym węzłem sieci i przekazuje mu tekst. Węzeł wybiera najbliższy wolny węzeł w odpowiednim kierunku (na przykład wysyłając korespondencję do USA z reguły korzysta się z węzłów niemieckich, holenderskich lub francuskich), który odbiera przesyłkę i kieruje ją dalej, do kolejnego wolnego węzła itd. Może to trwać dość długo, zwłaszcza w okresach szczytowego ruchu,

[1] Najmniejsza niedokładność w adresie spowoduje, że przesyłka nie dotrze do adresata i wróci jako tak zwany *dead letter*.



kiedy linie i węzły sieci są bardzo obciążone. Wreszcie jednak przesyłka dociera do komputera, w którym zdefiniowana jest "skrzynka pocztowa" adresata (w praktyce jest to katalog na dysku, zwykle nazywający się *maildir*, w którym umieszczane są przesyłane informacje jako pliki tekstowe). Adresat może być przy klawiaturze swojego komputera i wówczas *mailer* powiadomi go o nadejściu listu specjalnym sygnałem (patrz fotografia obok). Jeśli ma czas – może przeczytać list od razu, jeśli nie – poczta zaczeka. Wygodne jest jednak to, że list zostanie odebrany i zapamiętany nawet wtedy, gdy użytkownika nie ma przy komputerze (przy wysyłaniu poczty elektronicznej do USA jest to reguła, gdyż z powodu różnicy czasu moi korespondenci z reguły śpią, gdy ja piszę do nich listy). Wówczas użytkownik zaczynając pracę z komputerem następnego dnia rano dostanie sygnał o nadejściu poczty i może – w miarę wolnego czasu – przeczytać listy.

Jeśli wysłany list z jakichkolwiek powodów nie dotrze do adresata – wówczas u nadawcy elektronicznej poczty pojawia się informacja o tym smutnym fakcie w postaci tzw. *dead letter*. Nadawcą takiego "elektronicznego nekrologu" jest specjalny program, nazywany "demonem pocztowym" *(Mailer Deamon)*, który sprawdza ruch korespondencji i wykrywa ewentualne braki potwierdzenia odbioru nadanej przesyłki. Demon pocztowy obsługuje ruch przesyłek szybko i niezawodnie, wykazując przy tym nawet nieco poczucia humoru: Na zupełnie źle zaadresowaną przesyłkę reaguje listem, w którym pisze, że nigdy nie słyszał *(never heard)* o takim adresie. Oczywiście są to odpowiedzi z góry zaprogramowane przez twórców "demona", ale bardzo ożywiają i uprzyjemniają pracę z elektroniczną pocztą.

4.9. Zastosowania sieci komputerowych

Sieci komputerowe służyć mogą do wielu celów. Wyżej omówiłem usługi elektronicznej poczty. Jest to zastosowanie ważne, ale oczywiście nie jedyne. Łącząc się z innym komputerem, możesz (jeśli masz do tego prawo!) zaglądać do jego plików z danymi, kopiować je i wykonywać swoje programy. Oczywiście możesz za pomocą sieci dotrzeć do dowolnego komputera na kuli ziemskiej, pisząc

FTP *adres*

gdzie *adres* jest sieciowym adresem[1] komputera, który chcesz odwiedzić, ale bez znajomości hasła musisz zwykle ograniczyć swoje "odwiedziny" do wstępnego dialogu powitalnego, po którym Cię nieuchronnie wyrzucą. Istnieje jednak powszechnie praktykowany sposób wchodzenia do obcych, odległych komputerów, należący już raczej do komputerowego *savoir–vivre'u*, a nie do informatycznej wiedzy – tym niemniej wart przyswojenia.

Odwiedzając obcy komputer, który – jak sądzisz – zawiera przydatne dane, podajesz – w odpowiedzi na pytanie **LOGIN:** – identyfikator ***anonymous*** (anonim) albo czasem ***guest*** (gość). Jeśli gościnni właściciele komputera przewidzieli taką ewentualność – system zaprasza nas do wejścia, warunkiem jest zwykle podanie (jako hasła) swojego adresu sieciowego. Po takim gościnnym wejściu do nieznanego komputera możesz na ogół odczytać interesujące Cię pliki lub nawet skopiować je (poleceniem ***get***) – jeśli oczywiście uprawnienia gościa na to pozwolą. Natomiast bardziej rozległa

[1] O tym , jak zdobyć adres komputera, który może mieć interesujące Cię dane lub programy napisałem niżej.

eksploatacja odległego komputera możliwa jest wyłącznie w przypadku posiadania na nim własnego konta. Przy odrobinie wprawy można to robić z prawie każdym komputerem na kuli ziemskiej.

Po co to robić?

W sieciach mieszczą się liczne bazy danych i programy, które są dostępne za darmo i dostarczają wielu cennych informacji każdemu, kto się po nie zgłosi. Takie publicznie dostępne bazy danych znane są zwykle pod nazwą "*anonymous ftp centers*", a ich wykazy (ciągle uzupełniane o nowe pozycje) są dostępne w sieci w postaci łatwych do uzyskania plików informacyjnych. Są to długie wykazy (ja mam skatalogowanych ponad 2000 adresów węzłów sieci), których przytaczanie tu jest mało celowe (bardzo szybko się zmieniają). Przykładowo podam jedynie adres sieciowy bogatego *servera* (komputera dostarczającego informacje), znajdującego się w **Uniwersytecie Illinois** (*Urbana–Champaign*)

a.cs.uiuc.edu

Warto może także wspomnieć o archiwum programów nazwanym SIMTEL, w którym znajduje się ponad 10 tysięcy dostępnych (za darmo i legalnie!) programów dla IBM PC. Specyficznego smaku tej informacji dodaje fakt, że SIMTEL jest instalacją wojskową znajdującą się w *White Sands Missile Range, New Mexico*. Dostęp do SIMTEL–a można uzyskać między innymi za pośrednictwem serwerów z grupy TRICKLE w sieci EARN.

Szukanie potrzebnych banków informacji w sieci jest bardzo uproszczone dzięki istnieniu specjalnych systemów poszukujących, z których najbardziej znanym jest ARCHIE. Do systemu ARCHIE możesz się podłączyć pisząc polecenie

TELNET *adres*

Następnie możesz zadać mu pytanie o lokalizację informacji na zadany temat. Jeśli napiszesz – przykładowo – **prog Australia,** to po kilku minutach dostaniesz listę wszystkich miejsc na świecie, o których wie ARCHIE, w których są pliki lub katalogi zawierające słowo "Australia". Adresy sieciowe kilku systemów ARCHIE są następujące (w nawiasach podano, gdzie się te systemy znajdują; ma to znaczenie raczej ciekawostkowe, gdyż za pomocą sieci komputerowej można się połączyć z dowolnym punktem kuli ziemskiej z równą łatwością):

archie.doc.ic.ac.uk *(Imperial Col. London, UK),*
archie.funet.fi *(FUnet, Helsinki, Finlandia),*
archie.au *(Deaking, Geelong, Australia),*
archie.mcgill.ca *(McGill, Montreal, Canada).*

Wysławszy zapytanie do ARCHIE, otrzymujesz po chwili raport, gdzie się interesujący plik mieści[1]. Wystarczy tylko potem połączyć się ze wskazanym komputerem (poleceniem **FTP**), podać ścieżkę i plik – aby po chwili mieć go już na swoim komputerze. W ten sposób (przy odrobinie szczęścia) można dotrzeć do dowolnego programu lub pliku danych dostępnego na świecie w ciągu kilkunastu minut! Istnieją też inne sposoby korzystania z informacji zawartych w światowych bazach danych nawet w przypadku nieznajomości ich lokalizacji. W **CERN**–ie (międzynarodowy instytut badań jądrowych) funkcjonuje system **W3** (*World Wide Web*), za pomocą którego możliwy jest hipertekstowy[2] dostęp do dowolnej informacji. Tańsze (i łatwiej dostępne) są podobnie działające (ale bez hipertekstu) programy typu **Gopher** (*Go–for–it*) lub **WAIS** (*Wide Area Information Systems*). Bazy danych dostępne przez sieć dotyczą literalnie wszystkiego. Oto kilka przykładów:

- **Agricultural Info** – wiadomości rolnicze,
- **Baseball Scores** – wyniki rozgrywek baseballowych,

[1] Raport podaje nazwę komputera i jego adres sieciowy oraz nazwę katalogu (wraz z pełną ścieżką dostępu) i dokładną nazwę potrzebnego pliku.

[2] Idea hipertekstu wprowadzona została do informatyki w latach 60. przez **Teda Nelsona**. Polega ona na tym, że użytkownik komputera w trakcie czytania jakiegoś dokumentu (tekstu) może wskazać na dowolny jego element (np. niezrozumiałe słowo) i zażądać wyjaśnień. Czytając te wyjaśnienia może stawiać dalsze pytania itd. Hipertekst jest bardzo wygodny dla użytkownika, ale bardzo skomplikowany dla osoby, która musi napisać program realizując tę ideę. Dlatego zwykle stosowane są ograniczone hiperteksty (wyjaśnienia można uzyskać tylko na temat niektórych, z góry ustalonych, słów). Przykładem takich "pseudohipertekstów" bywają niektóre systemy podpowiedzi ("*help*") – na przykład w programach *Borlanda*.

- **CHAT** – informacje na temat AIDS,
- **GenBank** – informacje genetyczne,
- **STIS** – bank danych naukowych i technicznych,
- **Webster** – słownik ortograficzny (angielski),

Niestety, nie wszystkie bazy danych są dostępne za darmo. Za dostęp do niektórych trzeba płacić (na przykład **Chemical Abstract** lub **Medline**) i to całkiem sporo (od 40 do 500 $ za godzinę korzystania z bazy). W przyszłości należy oczekiwać znacznie większej liczby usług sieciowych, które będą na wyższym poziomie niż obecnie, ale będą dostępne za opłatą. Niektórzy twierdzą, że inwestowanie w płatne usługi sieciowe jest najbardziej dochodowym interesem. Być może, chociaż na pewno żal trochę będzie dzisiejszego, pionierskiego okresu, kiedy wszystko było jeszcze bardzo prymitywne, ale dostępne jak powietrze – za darmo...

Na szczęście jeszcze dziś w sieci komputerowej dostępne są *gratis* różne informacje, nawet dzieła literatury pięknej[1]. Dostępne są komputerowe wersje słowników (wielojęzycznych, ortograficznych, synonimów itp.), a także encyklopedii i atlasów geograficznych (chociaż dostęp do nich ograniczony jest zwykle do posiadaczy odpowiednich kart bibliotecznych). Już wkrótce łatwiej będzie wertować książki poprzez sieć na ekranie swego komputera, niż szukać[2] ich w tradycyjnych bibliotekach!

Większość dużych uniwersytetów na Zachodzie tworzy i udostępnia lokalne sieci, za pomocą których można się dowiedzieć "co, gdzie, kiedy?" na Uczelni i w mieście (cenne, zwłaszcza gdy się jest nowicjuszem!). Bazy takie, znane jako **CWIS** (*Campus–Wide Information System)* zawierają informacje o wszystkich wydarzeniach na Uniwersytecie, o możliwej do podjęcia pracy, o kupnie i sprzedaży różnych przedmiotów, o imprezach kulturalnych i spotkaniach dyskusyjnych itp[3]. Dostęp do wielu komputerowych zasobów danych i programów jest już dziś łatwy i powszechny, a będzie zapewne coraz łatwiejszy. Jedyny problem to czas, jaki jest potrzebny żeby przejrzeć katalog wszystkich ofert i wybrać interesujące pliki. Nie jest to problem błahy, bo katalogi potrafią być bardzo duże[4]. Czyżby groziło nam, że pomimo coraz większej liczby dostępnych danych będziemy wiedzieli coraz mniej?

Korzystający z sieci komputerowych mają do dyspozycji szereg biuletynów informacyjnych (np. *US Headline News, Clarinet News*) oraz szereg czasopism (np. wydawane po polsku **Donosy**). Niektóre z tych elektronicznie rozprowadzanych czasopism mają charakter otwarty (zamieszczane są w nich praktycznie wszystkie nadsyłane materiały), inne natomiast są starannie recenzowane i mają nadany numer ISDN. Tematyka tych periodyków jest zaskakująco szeroka i bardzo niekiedy odległa od problematyki komputerowej. Niżej podano kilka przykładowych tytułów czasopism rozprowadzanych w sieci (wraz z krótką ich charakterystyką), aby zilustrować to zjawisko.

- **ART COM** – pismo na temat współczesnej sztuki i zagadnień komunikacji społecznej,
- **Bryn Mawr Classical Review** – magazyn zawierający recenzje klasycznych prac starożytnych autorów łacińskich i greckich;
- **Dragonzine** – publikuje opowiadania typu fantastycznego;
- **The Purple Thunderbolt of Spode** – pismo wyznawców sumeryjskiego kultu OTIS,

Oczywiście krążących w sieci pism jest znacznie więcej, przytoczona próbka ma jedynie pokazać, jak bardzo bywają zróżnicowane.

---

[1] Realizowany w USA projekt badawczy o nazwie **Guthenberg** zmierza do przeniesienia do elektronicznych bibliotek wszystkich dzieł literackich, które nie są już objęte prawami autorskimi. Przeniesiono już dzieła Szekspira, Dantego i innych klasyków, a także księgi religijne (Biblia, Koran, Księga Mormonów). Australijski projekt **Coombs** przenosi z kolei do komputerowych baz danych między innymi klasyczne pisma buddyjskie i taoistyczne.

[2] Również bibliografie tematyczne najłatwiej dostępne są przez sieci komputerowe. Takich baz danych o wydanych książkach i artykułach (głównie naukowych) jest bardzo wiele, przykładowo można tu wymienić LIDO MAILSERVER, mieszczący się na Uniwersytecie w Saarbruecken (RFN), gromadzący dziesiątki tysięcy pozycji bibliograficznych dotyczących sztucznej inteligencji.

[3] Najbardziej znaną taką bazą danych jest FREENET zarządzana przez *Case Western Reserve University*, do której można się łatwo dostać poprzez adres sieciowy **yfn.ysu.edu**. Korzystając z tej bazy, można uzyskać liczne informacje na temat Uniwersytetu, miasta Cleveland i jego okolicy.

[4] Na przykład katalog bazy **fu.net** ma wielkość 9 MB (podkreślam: sam katalog!).

Piszę o tym tak obszernie, gdyż – wbrew pozorom – korzystnie z sieci jest dziedziną, w której działać mogą nie tylko profesjonaliści, ale również amatorzy. Obok profesjonalnych sieci (takich jak **EARN** lub **INTERNET**) są bowiem w użyciu sieci amatorskie. Najbardziej znana jest sieć **FIDO,** zawierająca (w 1992 roku) 7000 węzłów w Ameryce Północnej, 2500 w Europie, 500 w Australii i Oceanii, 100 w Ameryce Łacińskiej, 100 w Afryce i 450 w Azji. Podstawą działania sieci FIDO są tak zwane **BBS** (*Bulletin Boards Services*), czyli Elektroniczne Tablice Ogłoszeń, w których użytkownicy sieci umieszczają i z którego pobierają publicznie dostępne programy, teksty, pliki danych itp.

Sieć FIDO jest ogólnoświatowym forum dyskusyjnym, gdzie gromadzi się i wymienia opinie na różne tematy: naukowe, techniczne[1], filozoficzne i inne (np. dowcipy) w sposób niczym właściwie nie ograniczony. Działalność sieci FIDO opiera się na amatorskiej pracy tak zwanych *sysopów* – operatorów skrzynek BBS, ale pieczę nad całą siecią ma międzynarodowa organizacja IFNA (*The International Fido Net Association*). W Polsce jest obecnie około 20 węzłów sieci FIDO, a szkoda – bo włączenie do takiej sieci wymaga posiadania jedynie dowolnego komputera[2], linii telefonicznej i modemu.

W profesjonalnych sieciach komputerowych idea elektronicznego forum dyskusyjnego jest także dostępna. Za pomocą poczty elektronicznej i specjalnych programów (zwanych listserver) dostępnych jest mnóstwo tak zwanych list dyskusyjnych. Ich pełny wykaz na początku 1992 był plikiem o rozmiarze 900 kB! Znając nazwę listy dyskusyjnej obejmującej bliską nam tematykę,możemy wysyłać do niej nasze teksty oraz czytać, co inni dyskutanci mają do powiedzenia. Jest to bardzo wygodna forma wymiany myśli. Przykładowe listy dyskusyjne dostępne w Polsce przez sieć EARN mają nazwy:

- **AMERSTDY** – lista dyskusyjna na temat studiów w Ameryce,
- **POLAND-L** – lista dyskusyjna na tematy polskie.

Formą dyskusji bardzo rozpowszechnioną w sieciach komputerowych jest USENET. Jest to zbiór grup dyskusyjnych bez centralnego archiwum, w którym zainteresowani użytkownicy przesyłają sobie wzajemnie informacje na ustalony temat. Aktualnie zarejestrowanych jest w USENET ponad 700 grup dyskusyjnych, a statystyki z maja 1992 dowodzą, że informacje krążące w sieci USENET mają ponad 2,5 miliona użytkowników dziennie! Tematyka dyskusji jest zwykle sygnalizowana nazwą grupy dyskusyjnej, na przykład **comp** – to dyskusja na temat komputerów, **sci** – tematyka naukowa, **soc** – tematyka społeczna, **news** – nowości, **talk** – plotki, **rec** – rekreacja, hobby, **alt** – tematy alternatywne, **misc** – rozmaitości, itp[3]. Najczęściej czytane grupy dyskusyjne to: **alt.sex** – (bez tłumaczenia!), 360.000 czytelników dziennie, **misc.jobs.offered** – proponowane miejsca pracy, 250.000 czytelników dziennie, **news.announce.newuser** – dla nowicjuszy, 230.000 czytelników dziennie.

Najwięcej czasu zużywa sieć na przesyłanie plików **alt.pictures.erotica.binary**. Wrodzone poczucie przyzwoitości nie pozwala mi na komentowanie tego faktu!

Na koniec jeszcze jedna informacja. W USENET funkcjonuje polecenie **FAQ** (*Frequently Asked Questions*), pozwalające uzyskać wykaz najczęściej zadawanych pytań. Pozwala ono zorientować się, o co ludzie najczęściej pytają (i na co jest w sieci najwięcej gotowych odpowiedzi). Gorąco polecam Ci poczytanie w wolnej chwili takich zbiorów pytań i odpowiedzi. Bardzo ciekawe i pouczające!

---

[1] Wiele firm komputerowych oferuje swoje usługi za pomocą BBS-ów, na przykład wraz z kartą muzyczną SoundBlaster otrzymuje się hasło do BBS, w której są zgromadzone programy dla tej karty. Innym przykładem są tak zwane BIX-y zorganizowane przez znane pismo komputerowe BYTE jako forum dyskusyjne pozwalające na kontakt z autorami artykułów i wydawcami pisma.

[2] Do sieci FIDO dołączają się amatorzy mający komputery IBM PC, Atari, Commodore czy nawet ZX Spectrum!

[3] Przykładowe grupy dyskusyjne, w których ja uczestniczę: **comp.ai** – dyskusja na temat sztucznej inteligencji, **comp.ai.neural-nets** – sieci neuronowe, **comp.society** – wpływ komputerów na społeczeństwo, **sci.bio** – nowości nauki w biologii, **sci.math** – nowości matematyczne, **sci.electronics** – elektronika, **sci.econ** – nauki ekonomiczne, **misc.education** – metody nauczania i systemy edukacyjne, **talk.philosophy.misc** – różne (ciekawe!) dywagacje filozoficzne, **rec.food.cooking** – coś dla smakoszy!

5. Odrobina teorii

5.1. Uwagi wprowadzające

Informatyka kojarzy się głównie z działaniami na wskroś praktycznymi: budową sprzętu komputerowego lub jego programowaniem – i tylko te sprawy zapewne Cię interesują. Niemniej, jak mawiał wielki **Albert Einstein** – *nie ma nic bardziej praktycznego niż dobra teoria* – zatem pozwól, że na zakończenie przedstawię Ci krótki wykład przynajmniej **niektórych** teoretycznych aspektów informatyki. Nie musisz oczywiście tego czytać, ale zapewniam Cię – wiele zyskasz studiując także i ten rozdział[1].

5.2. Teoria informacji

Punktem wyjścia w całej informatyce jest pojęcie **informacji**. Jest to pojęcie trudne do pełnego opisania i wyczerpującego scharakteryzowania, ponieważ można mówić o różnych aspektach informacji: o jej znaczeniu, o wartości, o nośnikach na których informację zapisano itd. Nas jednak interesować będzie głównie **ilościowy aspekt informacji**, czyli odpowiedź na pytanie:

Jak zmierzyć ilość informacji?

Podstawy takiej ilościowej teorii informacji dał **C. Shannon** w **1948** roku. Prześledźmy zasadnicze elementy jego wywodów. Punktem wyjścia do sformułowania miary informacji jest u Shannona określenie **miary niepewności**. Jeśli uda się ilościowo wyrazić niepewność związaną z pewnym wydarzeniem A, to wówczas nadejście komunikatu B, zmniejszającego tę niepewność, będzie mogło być rozpatrywane w następujący sposób:

Ilość informacji I zawarta w komunikacie B o zdarzeniu A równa jest różnicy pomiędzy **początkową** niepewnością zdarzenia A a niepewnością jaka **pozostaje** na temat wydarzenia A po nadejściu komunikatu B:

$$I(A\ B) = H(A) - H(A/B)$$

Oznaczenia $H(A)$ użyto do zapisu początkowej (pierwotnej) niepewności zdarzenia A, natomiast $H(A/B)$ jest niepewnością jaka pozostaje mimo odebrania komunikatu B[2].

Jak wynika z przytoczonych sformułowań, kluczowym problemem w teorii informacji jest określenie miary niepewności H. Kolejnym, bardzo celnym pomysłem Shannona było powiązanie

---

[1] Niestety, nie udało mi się uniknąć w tym rozdziale kilku wzorów matematycznych. Podobno każdy wzór użyty w popularnej książce zmniejsza liczbę jej Czytelników o połowę, mam jednak nadzieję, że mimo to liczba moich Czytelników nie zmaleje do zera!

[2] Można sobie wyobrazić sytuację, kiedy $H(A/B) = 0$, wówczas komunikat całkowicie wyjaśnia sytuację, jednak można także rozpatrywać sytuację kiedy $H(A/B) = H(A)$ i wówczas komunikat nie niesie żadnej użytecznej informacji albo nawet $H(A/B) > H(A)$ i wówczas po otrzymaniu komunikatu wiemy jeszcze mniej, niż przed jego odebraniem – czyli mamy do czynienia z **dezinformacją.**

miary niepewności $H(A)$ zdarzenia A z **prawdopodobieństwem** $p(A)$ tego zdarzenia. Punktem wyjścia były tu trzy bardzo naturalne i oczywiste postulaty:

1. Niepewność zdarzenia pewnego wynosi zero:

   jeśli $p(A)=1$, to $H(A)=0$

2. Im mniejsze prawdopodobieństwo, tym większa niepewność:

   jeśli $p(A) > p(C)$, to $H(A) < H(C)$

3. Jeśli rozważane zdarzenie A jest złożeniem dwóch niezależnych zdarzeń C i D, wówczas niepewność takiego zdarzenia jest równa sumie niepewności zdarzeń składowych:

   $$H(A) = H(C) + H(D)$$

Na podstawie tych postulatów bez trudu można ustalić, jaka powinna być postać matematycznej formuły opisującej niepewność:

$$H(p) = -\log p$$

Pozostaje otwarta kwestia **jednostek**, w jakich wyraża się niepewność H, a tym samym i informację I. Jednostki te zależą od podstawy używanego logarytmu: użycie najłatwiej dostępnych logarytmów dziesiętnych prowadzi do jednostek nazywanych **dit** (*decimal information unit*), cenione przez teoretyków logarytmy naturalne pozwalają operować jednostką nazywaną **nit** (*natural information unit*), jednak najczęściej używane są jednostki nazywane **bit**[1] (*binary information unit*), stosowane gdy podstawą logarytmu jest liczba dwa.

Sytuacja, w której zdarzeniu A można przypisać jedno określone prawdopodobieństwo, jest nadmiernie uproszczona. Znacznie bardziej celowe jest rozważenie możliwości występowania (z różnymi prawdopodobieństwami) różnych wariantów zdarzenia A. Przykładem takiej sytuacji jest czytanie dowolnego tekstu (na przykład tej książki). Zdarzeniem, które w takim przypadku można rozważać jest **następna litera tekstu**, a wariantami mogą być: litera A, litera B, itd. Dla ogólności rozważań przyjmijmy, że możliwych jest N wariantów występujących odpowiednio z prawdopodobieństwami $p_1, p_2, p_3, ..., p_N$, przy czym oczywiście któryś z wariantów musi wystąpić, zatem

$$\sum_{i=1}^{N} p_i = 1$$

Średnia niepewność związana z omówioną sytuacją może być wyliczona jako:

$$H(A) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Podany wzór jest ważny i często używany[2]. Spróbujmy zbadać, jaki musi być rozkład prawdopodobieństw p_i, aby niepewność była największa z możliwych? Okazuje się, że rozkład powinien być równomierny, to znaczy prawdopodobieństwa powinny spełniać warunek $p_1 = p_2 = p_3 = ... = p_N = 1/N$. Niepewność wówczas wynosi

$$H_{max}(A) = \log_2 N$$

Dlaczego mnie to interesuje? Otóż wiedząc, jaka jest **rzeczywista** niepewność $H(A)$ wystąpienia jednej litery w tekście i porównując ją z maksymalną możliwą niepewnością $H_{max}(A)$ możemy określić, czy dany system przekazywania informacji (na przykład język) koduje przekazywane informacje oszczędnie, czy nie. Ilość informacji $I(A\ B)$ można bowiem wyznaczyć na dwa sposoby:

$$I(A\ B) = H(A) - H(A/B) = H(B) - H(B/A)$$

---

[1] Uważny Czytelnik może w tym momencie zacząć protestować. Wszak termin bit był wcześniej używany w innym kontekście, jako nazwa pewnego podzespołu komputera, a mianowicie takiego najmniejszego fragmentu pamięci, w którym można zapamiętać pojedyncze zero lub pojedynczą jedynkę! Wbrew pozorom nie ma tu żadnej sprzeczności. Zastanówmy się bowiem, czemu odpowiada bit rozumiany jako jednostka ilości informacji: będzie to pojemność takiego komunikatu, który usunął niepewność wyrażającą się możliwością wyboru jednej z dwóch jednakowo prawdopodobnych ewentualności, gdyż $H = \log_2 p = 1$, gdy $p = 0{,}5$. Zatem zgadza się – **bit** rozumiany jako fragment struktury komputera może przechowywać dokładnie jeden **bit** informacji.

[2] Za jego pomocą wyznacza się ilość informacji zawartą w dowolnych przekazach słownych, kodach komputerowych i zestawieniach liczbowych, ale także w obrazach, muzyce lub ... molekułach DNA zawierających kod genetyczny.

Oznaczenia występujące w drugiej części wzoru są dość łatwe do odszyfrowania: ***H(B)*** to entropia[1] komunikatu, a ***H(B/A)*** to niepewność dotycząca postaci komunikatu w momencie, kiedy zdarzenie, którego komunikat dotyczy, jest całkowicie zdeterminowane. Z przytoczonej zależności wynika, że

$$I(A\,B) \leq H(B)$$

ponieważ zawsze

$$H(B/A) \geq 0$$

Komunikat ma więc tym większą pojemność informacyjną, im większa jest jego własna entropia. Z tego powodu przy wyborze sposobów kodowania (por. następny podrozdział) dążymy do **maksymalizacji** entropii komunikatów. Dla każdego systemu kodowania możemy oszacować jego maksymalną informacyjną pojemność H_{max} i rzeczywistą entropię H. Ponieważ zwykle $H < H_{max}$ pojawia się niekorzystne zjawisko: każdy symbol (np. każda litera tekstu) przenosi mniej informacji, niż by mogła. Jest to strata: dla przesłania lub zapamiętania tej samej ilości informacji trzeba użyć większej liczby symboli – a to kosztuje[2]. Zjawisko to nazywamy **nadmiarem** albo **redundancją** (oznaczenie ***R***) i możemy oszacować wzorem

$$R = \frac{H_{\max} - H}{H_{\max}}$$

Oszacowanie redundancji ***R*** dla wielu naturalnych systemów przekazywania informacji daje zaskakujące wyniki. Oto na przykład dla języka polskiego stwierdzono, że (biorąc pod uwagę pojedyncze litery) H = 5,26 bita/literę, a H_{max} = 1 bit/literę. Zatem nadmiarowość języka polskiego wynosi około 81%, a to oznacza, że **4/5 wszystkiego co mówimy lub piszemy jest, z informacyjnego punktu widzenia, zbędne.**

Pierwsze podejrzenie, jakie się w związku z tym nasuwa, to przypuszczenie, że w podanych oszacowaniach mieści się jakiś błąd. Jednak z literatury można stwierdzić, że podobne badania prowadzono dla innych języków: najpierw angielskiego (pionierskie prace Shannona prowadzone były w USA i w naturalny sposób dotyczyły języka angielskiego), potem niemieckiego, francuskiego, szwedzkiego, rosyjskiego ... Za każdym razem uzyskiwano podobne[3] wyniki oszacowań, co daje do myślenia[4]. To nie może być przypadek, lecz jakaś głębsza prawidłowość.

Tak jest w istocie. Redundancja zawarta w języku naturalnym służy do tego, by uczynić komunikację między ludźmi maksymalnie **niezawodną.** Dzięki temu, że nie wszystkie zestawy liter są sensownymi słowami, możemy poprawnie odczytać i właściwie zrozumieć słowo, które napisano z błędem. Przykładowo widząc napis:

KRWKÓW

bez trudu rozpoznajemy błąd i potrafimy powiedzieć, jaka litera została zamieniona. Natomiast znajdując w opisie wypadku drogowego informację, że samochód miał numer

KRW 3463

w żaden sposób nie możemy się domyślić, że w istocie był to numer **KRA 3463** i tylko chochlik drukarski wymienił czcionki. Dzieje się tak właśnie dlatego, że w systemie, jakim są znaki rejestracyjne samochodów, **redundancji nie ma.**

Wiedząc o redundancji języka naturalnego, możemy poszukiwać metod takiego rejestrowania (na przykład w pamięci komputera) tekstów języka naturalnego, by eliminując (chociaż częściowo)

---

[1] Ponieważ istnieje bezpośredni związek pomiędzy pojęciem **niepewności** wprowadzonym w teorii informacji i pojęciem **entropii** używanym przez fizyków – zwykle stosuje się te dwa terminy zamiennie.

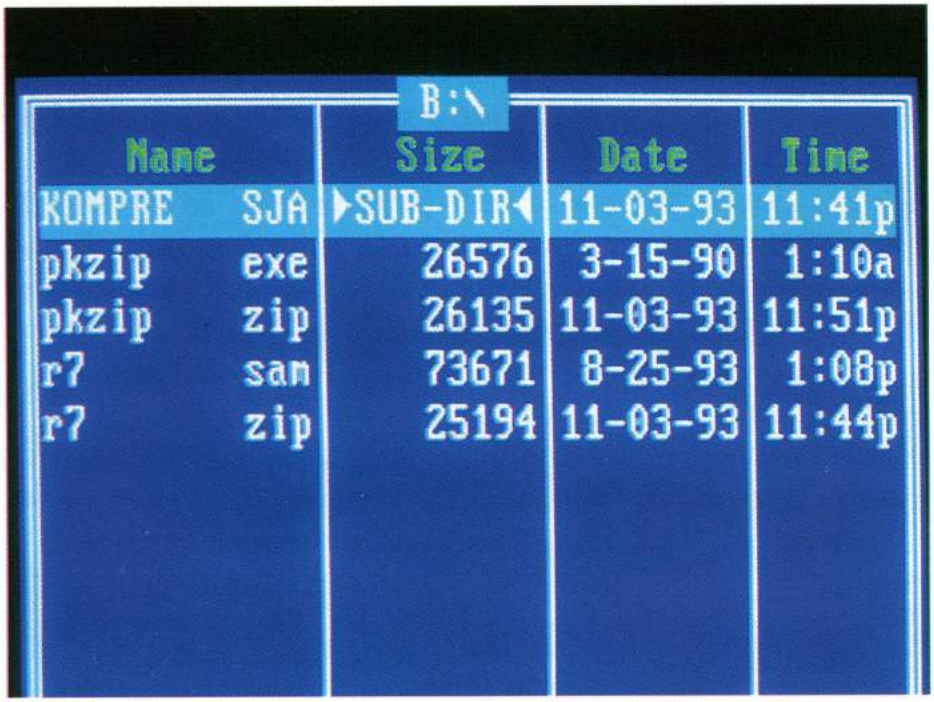
[2] Informacja zajmuje więcej miejsca na dysku lub wymaga dłuższego czasu podczas transmisji, a za to się płaci!

[3] Oczywiście poszczególne języki różnią się od siebie wielkością redundancji, na przykład język angielski ma mniejszą redundancję niż język polski, co ma ten (między innymi) skutek, że tekst przetłumaczony z angielskiego na polski zajmuje zawsze więcej miejsca, niż tekst pierwotny. Ma to związek z problemem "polonizacji" wielu programów – początkowo używane w programie hasła angielskie (na przykład pozycje menu lub części podpowiedzi "*help*") po przetłumaczeniu nie mieszczą się w przewidzianych dla nich oknach na ekranie. Są jednak języki jeszcze bardziej nadmiarowe, niż język polski – na przykład język portugalski.

[4] Co więcej, dokonywano podobnych obliczeń dla "języka" afrykańskich bębnów, tak zwanych tam–tamów, za pomocą których tubylcy mieszkający w nieprzebytej dżungli porozumiewają się na odległość wielu kilometrów. I w tym wypadku także okazało się, że redundancja wynosiła nie mniej niż 80%.

nadmiar, doprowadzić do "gęściejszego" upakowania wiadomości. Z drugiej jednak strony, wiedząc o "protekcyjnej" roli redundancji możemy ją celowo wprowadzać do przesyłanych wiadomości, aby zmniejszyć ich podatność na zakłócenia. Będzie to dokładniej przeanalizowane w kolejnym podrozdziale. W tym miejscu natomiast wspomnijmy o pewnym praktycznym zastosowaniu przytoczonych wniosków wynikających z teorii informacji. Otóż przechowując na dyskach komputerowych rozmaite teksty, mamy niemiłą świadomość marnowania znacznej części cennej przestrzeni (w dosłownym znaczeniu tego słowa – dyski są bardzo drogie!). Czy nie można by wykorzystać faktu, że w tekstach tych zawarty jest ogromny nadmiar informacji – i zmniejszyć jakoś ich objętość?

Oczywiście, że można to zrobić! Służą do tego specjalne programy, tak zwane archiwizatory. Jest ich wiele, ARJ, PKARC, LHARC, PKZIP[1] to przykładowe nazwy kilku częściej spotykanych. Programy te pozwalają automatycznie przekształcić treść dowolnego pliku dyskowego w taki sposób, żeby zajmowała najmniejszą możliwą powierzchnię na dysku. Metody stosowane przy tym są rozmaite, stosunkowo często wykorzystuje się na przykład kodowanie zaproponowane przez Huffmana. Nie wchodząc w szczegóły techniczne stosowanych algorytmów, stwierdzić trzeba, że redukują one objętość plików tekstowych średnio o około 60%, co stanowi wprawdzie jedynie część możliwego do uzyskania efektu (nadmiarowość języka naturalnego przekracza wszak 80%!), jednak w stosunku do tekstów nie upakowanych oznacza bardzo istotny zysk, gdyż na tej samej dyskietce można zmieścić ponad dwukrotnie większą liczbę danych. Zobaczmy bliżej, jak się to odbywa.



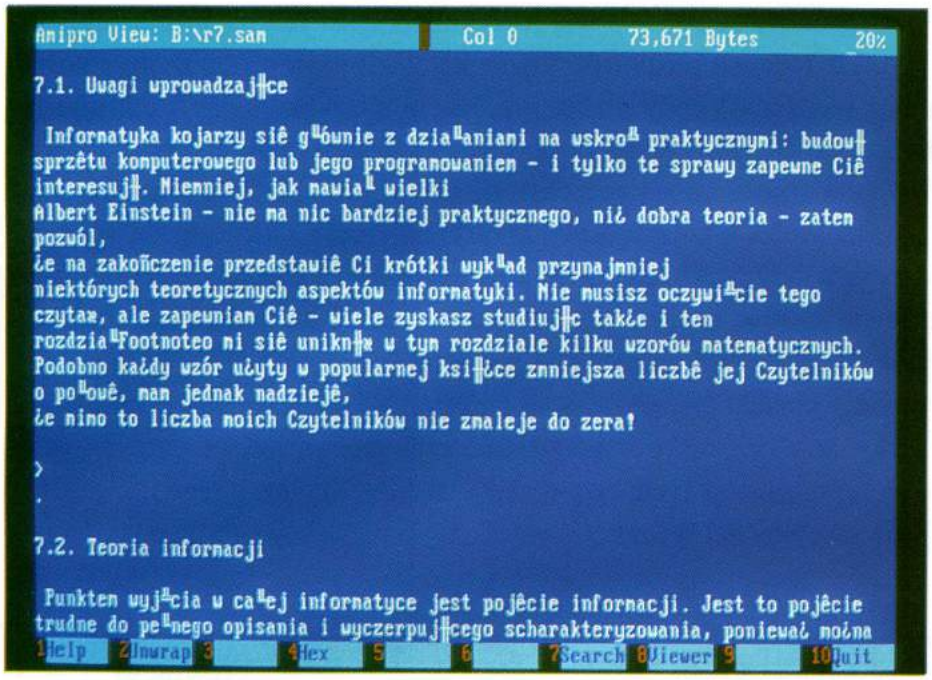
Na rysunku obok podano obraz katalogu, z którego można odczytać, że pewien tekst (konkretnie – tekst tego rozdziału książki nazwany R7.SAM) zajmuje w formie nie upakowanej ponad 70 tysięcy bajtów. Po upakowaniu (programem PKZIP) powstał plik R7.ZIP o trzykrotnie mniejszej objętości. Warto zauważyć, że podobny zabieg wykonany na pliku nie zawierającym redundancji (przykładem takiego pliku może być dowolny program, na przykład sam "pakowacz" PKZIP.EXE) nie daje już tak efektownych wyników (por. podaną objętość plików PKZIP.EXE i PKZIP.ZIP).

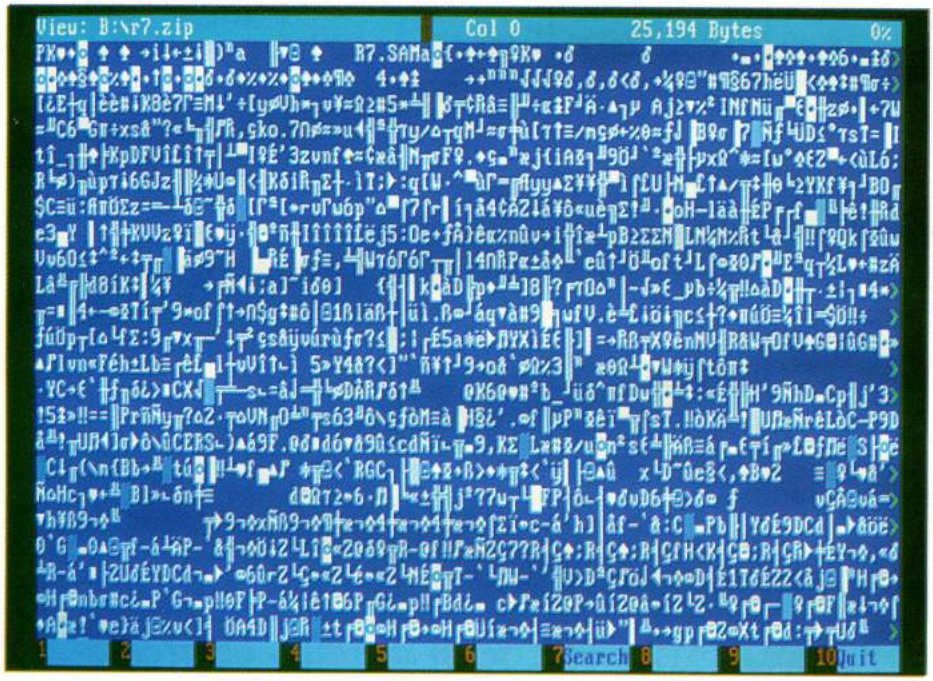
Na kolejnych rysunkach (na następnej stronicy) pokazano zawartość plików R7.SAM i R7.ZIP. Widać, że tekst po zakodowaniu staje się nieczytelny, gdyż litery lub sekwencje liter zastąpione są zupełnie przypadkowymi (na pozór) zestawami bajtów. Przy idealnej kompresji każdy z tych bajtów powinien występować z tą samą częstością, co powoduje kompletną nieczytelność zapisu. Program PKZIP nie jest taki mądry i dlatego w pokazanym na rysunku fragmencie kodu można wykryć pewną regularność.

Zasada działania programów archiwizujących (typu omawianego tu PKZIP) polega w głównej mierze na stosowaniu oszczędnego kodowania powtarzających się znaków lub ich grup. Ilekroć program napotyka grupę identycznych znaków (na przykład serię odstępów w tekście) – zastępuje tę grupę dwubajtową kombinacją złożoną z kodu znaku i liczby określającej krotność powtarzania znaku.

---

[1] Autorem wielu popularnych programów archiwizujących (PKZIP, PKARC, PKPAK) jest **Phil Katz,** właściciel firmy *PKWare.*





W podobny sposób można także oszczędnie kodować inne rodzaje informacji, na przykład obrazy. Obraz w systemie komputerowym może być zapamiętany w postaci wektorowej albo w postaci rastrowej. Przy prezentacji rastrowej (częściej obecnie stosowanej) obraz zapamiętuje się w ten sposób, że rejestruje się w pamięci komputera barwę i jasność każdego punktu ekranu. Ponieważ punktów tych jest bardzo dużo (wynika to z przyjętej rozdzielczości), zatem obrazy należą do najbardziej "pamięciochłonnych" obiektów zapamiętywanych przez komputery. Ale przecież w przypadku obrazu wyjątkowo dużo sąsiadujących ze sobą punktów ma te same atrybuty (na przykład wszystkie one są punktami tła) – zatem przy zastosowaniu nawet najprostszych metod kompresji można przy obrazach uzyskać wyjątkowo duże współczynniki oszczędności pamięci. Warto o tym pamiętać!

5.3. Teoria kodów

W informatyce stale mamy do czynienia z różnymi **kodami**. Kody liter i kody liczb, kody klawiatury i kody ekranu, kodowanie danych przed przesłaniem ich za pomocą łącza i dekodowanie po odebraniu. Wszystko to sprawia, że informatyk często pisze albo mówi o kodach – nie zawsze do końca wiedząc, z czym naprawdę ma do czynienia. Aby więc te pojęcia usystematyzować, musimy obecnie odwołać się do precyzyjnej definicji.

Kodem nazywamy ustalony zbiór symboli (nazywany zwykle alfabetem) oraz zbiór reguł pozwalających poszczególnym łańcuchom symboli przypisać określone wiadomości. Dla konkretyzacji niektórych dalszych rozważań wprowadzimy kilka prostych oznaczeń. Niech **A** oznacza rozważany **alfabet**. Wówczas symbole należące do tego alfabetu można w sposób ogólny oznaczać jako a_1, a_2, a_3, ..., a_N. **Łańcuchy** symboli kodowych, którym przypisuje się określone wiadomości, powstają przez proste złożenie tych symboli. Zbiór wszystkich łańcuchów (**wyrazów kodowych**), które można

utworzyć z elementów alfabetu **A** oznaczany będzie $\mathbf{A}^*$. Dla jednolitości dalszych rozważań zakładać będziemy, że do zbioru $\mathbf{A}^*$ należy także łańcuch pusty (nie zawierający ani jednego symbolu). W informatyce stosowane są (prawie bez wyjątków) kody dwójkowe, zatem nie popełniając błędu możemy założyć, że alfabet **A** składa się z dwóch symboli ***0*** oraz ***1*** i jest ustalony.

Mając ustalony **kod** i określone **wiadomości,** możemy mówić o procesie **kodowania,** czyli ustalania symboli kodowych dla wiadomości wymagających przesłania[1], zarejestrowania w pamięci systemu lub wypisania na urządzeniach zewnętrznych. Odwrotna czynność, to znaczy odzyskiwanie określonych wiadomości na podstawie kodów, nazywa się **dekodowaniem.** Kluczem do techniki kodowania jest reguła wiążąca wyrazy kodowe z rozważanymi wiadomościami. Zależnie od tego, jak jest ona zbudowana, możemy mówić o kodach mających pewne użyteczne właściwości, co zilustruję przykładem różnych sposobów kodowania dziesięciu wiadomości symbolizowanych przez cyfry od **0** do **9.** Dowolność wyboru kodu dla reprezentacji tych cyfr wynika między innymi z tego, że możliwych wyrazów kodu jest więcej niż wiadomości do zakodowania. Wybrane kody zebrałem w formie tabeli.

Cyfra	ASCI	BCD	Aiken	Gray	Watts	BCDP	2z5	Johnson
0	00110000	0000	0000	0000	0000	0000**0**	00011	00000
1	00110001	0001	0001	0001	0001	0001**1**	00101	00001
2	00110010	0010	0010	0011	0011	0010**1**	01001	00011
3	00110011	0011	0011	0010	0010	0011**0**	10001	00111
4	00110100	0100	0100	0110	0110	0100**1**	00110	01111
5	00110101	0101	1011	0111	1110	0101**0**	01010	11111
6	00110110	0110	1100	0101	1010	0110**0**	10010	11110
7	00110111	0111	1101	0100	1011	0111**1**	01100	11100
8	00111000	1000	1110	1100	1001	1000**1**	10100	11000
9	00111001	1001	1111	1101	1000	1001**0**	11000	10000

Najchętniej stosowanym kodem jest kod ASCII, jest on jednak nieco zbyt kosztowny (wymaga aż 8 bitów). Dlatego dla kodowania samych cyfr dziesiętnych jest chętnie używany kod nazywany **BCD** (*Binary Coded Decimals*). W kodzie tym kolejnym cyfrom odpowiadają bezpośrednio ich dwójkowe odpowiedniki, wyznaczone zgodnie z ogólnymi regułami zamiany liczb dziesiętnych na binarne. Kod ten ma wadę, próba wykorzystywania go w technice komputerowej ujawniła wielką niedogodność, jaką jest w tym kodzie fakt, że wszystkie przejścia od kodu cyfry N do kodu cyfry N+1 dokonywane są przez dodanie (zgodnie z zasadami arytmetyki dwójkowej) jedynki do kodu liczby N – **z wyjątkiem** przejścia od 9 do 0. Właśnie ten wyjątek stanowił przeszkodę przy konstruowaniu pierwszego komputera świata i nie przypadkowo kod, który jest wolny od tej wady, związany jest z nazwiskiem **Aikena**, budowniczego pierwszego komputera **Mark I** (czwarta kolumna tablicy).

---

[1] Z jednego podsystemu komputera do innego podsystemu względnie na zewnątrz systemu (na przykład do człowieka).

W wielu zastosowaniach praktycznych kody BCD lub Aikena są niewygodne, gdyż przy przechodzeniu między kodem liczby N i kodem liczby N+1 następuje w nich zmiana na pozycjach wielu bitów[1]. Opisanej niedogodności można zapobiegać stosując inny kod. Jest to tak zwany kod **Graya**, w którym sąsiednie pozycje kodowe różnią się wartością tylko jednego bitu (patrz kolumna 5 tablicy). Jeszcze lepsze własności ma kod **Wattsa** (kolumna 6 tablicy), w którym także stany 9 i 0 różnią się tylko na pozycji pojedynczego bitu.

Wszystkie 4–bitowe kody używane do odwzorowywania cyfr 0–9 miały wspólną wadę: jeśli chociaż jeden bit w tych kodach uległ **przekłamaniu,** to znaczy został zamieniony z jedynki na zero lub z zera na jedynkę, wówczas powstawał kod innej wiadomości i faktu przekłamania nie można było wykryć ani skorygować. Tymczasem w każdym realnym systemie informacyjnym stale dochodzi do przekłamań: podczas przesyłania zakodowanych informacji powodem przekłamań mogą być towarzyszące transmisji zakłócenia, podczas przetwarzania informacji źródłem przekłamań mogą być zawodnie działające elementy elektroniczne komputera, a podczas przechowywania informacji w pamięci przekłamania powstają pod wpływem oddziaływania zewnętrznych czynników fizycznych na używany nośnik pamięci. Poszukiwanie i ewidencjonowanie przyczyn przekłamań wyprowadzi nas bardzo daleko poza przedmiot rozważań książki, zapamiętajmy jednak podstawowy fakt: projektując kody, musimy liczyć się z tym, że w każdej chwili dowolny bit może zamienić się z 0 na 1 lub odwrotnie. Sztuka tworzenia dobrych kodów polega na tym, by zabezpieczyć się przed przekłamaniami. Nie jest to sprawa prosta[2].

Zastanówmy się, dzięki czemu można wykryć błąd,a nawet skorygować przekłamaną wiadomość uzyskując prawidłową odpowiedź. Otóż źródłem tego efektu może być jedynie fakt istnienia nadmiaru, czyli redundancja kodu[3]. Mamy więc gotową odpowiedź na pytanie, co należy zrobić, aby tak zabezpieczyć kod przed przekłamaniami, by każda pomyłka mogła być wykryta (w kodzie BCD takie wykrycie mogło być uzyskane jedynie czasami). Trzeba wprowadzić celowy nadmiar.

Najprostszy sposób wprowadzenia nadmiaru polegać może na zwiększeniu liczby bitów w stosowanym kodzie. Przykładowo, zamiast 4 użyjemy 5 bitów. Jeśli dodatkowo wprowadzimy jakąś porządkującą regułę, którym bity te muszą podlegać, wówczas możliwe będzie wykrycie ewentualnego przekłamania – właśnie poprzez zakłócenie tej reguły. Najprostszą, ale chętnie używaną regułą jest tak zwana **zasada parzystości** (opisywana również jako tzw. **bit parytu**). Zasada ta polega na konsekwentnym stosowaniu zasady, że w każdym wyrazie kodowym musi być parzysta liczba jedynek. Praktyczne zabezpieczenie realizacji tej zasady osiąga się poprzez dodanie do dowolnego czterobitowego kodu (na przykład BCD) dodatkowej pozycji (na przykład na końcu kodu), w której dopisywana jest jedynka, jeśli w oryginalny kodzie jest nieparzysta liczba jedynek, względnie dopisywane jest zero w przeciwnym przypadku. Po takim uzupełnieniu wszystkie wyrazy kodowe mają parzystą liczbę jedynek, jeśli więc pojawi się jakikolwiek kod z nieparzystą liczbą jedynek – można go bez wahania zakwalifikować jako błąd. Kod powstający z kodu BCD po dodaniu bitu parzystości oznaczany jest niekiedy jako **BCDP** (por. kolumnę 7 tablicy).

Dysponując pięciobitowym słowem, można budować także inne kody z zabezpieczeniem. Przykładem innego pomysłowego kodu jest kod **dwa–z–pięciu**, w którym zasada polega na

---

[1] Rozważmy przejście od kodu cyfry 7 do kodu cyfry 8 w kodzie BCD. Zostają w tym momencie zmienione **wszystkie** bity, gdyż od wyrazu kodowego 0111 trzeba przejść do wyrazu kodowego 1000. Wyobraźmy sobie, że nie nastąpi to równocześnie, to znaczy niektóre zmiany zajdą wcześniej, a inne później. Możliwe jest zatem pojawienie się w stanach przejściowych dowolnego kodu – na przykład 0011. Podana przykładowa sytuacja zajdzie, jeśli druga (od lewej) pozycja zmieni się, a pozostałe jeszcze nie. Będzie to bardzo mylące, gdyż pojawi się niespodziewanie kod cyfry 3. Szczególnie niebezpieczne są kody BCD przy sterowaniu procesów (por. rozdz. 4.4).

[2] Przykładowo w kodzie BCD zamiana jednego bitu może spowodować przejście poprawnego wyrazu 0111, będącego kodem cyfry 7, na niepoprawny wyraz 0011, oznaczający cyfrę 3. Przy odrobinie szczęścia może jednak nastąpić wykrycie przekłamania, na przykład jeśli w podanym wyżej kodzie 0111 nastąpi zamiana pierwszego bitu z 0 na 1 – wówczas powstanie wyraz 1111 nie będący sensownym kodem żadnej wiadomości. Co więcej, zakładając, że domyślamy się, iż tylko jeden bit uległ przekłamaniu, możemy także odtworzyć poprawną wiadomość, jaka była przesłana zakłóconym łączem: jedynym kodem, z którego poprzez zmianę jednego bitu otrzymać można przekłamany kod 1111, jest właśnie 0111.

[3] Ponieważ nie wszystkie czterobitowe kombinacje są sensownymi kodami określonych wiadomości – przez uzyskanie kombinacji "nonsensownej" mogliśmy w poprzednim przykładzie wykryć i skorygować błąd.

tworzeniu tylko takich pięciobitowych kombinacji, w których dokładnie dwie pozycje są jedynkami. Tabela dla tego kodu podana została w ósmej kolumnie[1]. Jeszcze inny pięciobitowy kod z częściowym zabezpieczeniem zbudowany jest według zasady "przejazdu" zwartego łańcucha jedynek przez pole kodu. Jest to tak zwany kod Johnsona (kolumna 9).

5.4. Teoria języków i gramatyk formalnych

Rozważając w poprzednim podrozdziale teorię kodów, wprowadziliśmy dwa istotne pojęcia, które i teraz będą nam przydatne. Z jednej strony mówiliśmy o **alfabecie** określonych symboli, a z drugiej strony zdefiniowaliśmy **wyrazy kodowe**, rozumiane jako łańcuchy symboli pochodzących z określonego alfabetu. O ile jednak poprzednio interesowaliśmy się sensem poszczególnych wyrazów, wiążąc (poprzez reguły używanego kodu) wyrazy kodowe z określonymi wiadomościami, o tyle obecnie zajmować nas będzie poprawność poszczególnych wyrazów. Język bowiem (każdy, a w szczególności język programowania) rozpatrywać można jako podzbiór zbioru wszystkich możliwych wyrazów, zbudowanych z liter przyjętego alfabetu. O tym, czy jakiś wyraz jest poprawnym zdaniem rozważanego języka, decyduje **gramatyka**[2]. Gramatyka jest więc traktowana jak filtr – jedne napisy **akceptuje** jako zgodne z regułami języka, a inne **odrzuca.** Proces akceptacji lub odrzucenia dokonywany jest automatycznie, na podstawie podstawiania za jedne symbole innych symboli. Pełna matematyczna definicja gramatyki jest dość złożona i nie będzie tu przytaczana[3], jednak podstawowe pojęcia spróbuję tu prosto i przystępnie przedstawić, ponieważ pewna orientacja w tych zagadnieniach jest potrzebna przy poznawaniu języków programowania[4].

Definiując gramatykę jakiegoś języka, trzeba najpierw podać wykaz symboli, z których można budować zdania tego języka[5]. Jeśli jakiś napis zawiera chociaż jeden niedozwolony symbol[6] – musi być odrzucony bez dalszej analizy. Jednak nie każdy napis zbudowany z poprawnych wyrazów jest poprawnym zdaniem. Gramatyka musi więc opierać się na ogólnych regułach[7], pokazujących jaka jest poprawna struktura zdania. Przy budowie tych reguł używa się zbioru pomocniczych symboli nazywanego **alfabetem nieterminalnym**. Elementy tego alfabetu używane są do zapisywania pewnych **ogólnych** reguł gramatycznych i odgrywają w gramatyce formalnej podobną rolę, jak w klasycznej gramatyce takie pojęcia, jak *rzeczownik* albo *orzeczenie*. Ponieważ symbole nieterminalne nie wchodzą w skład zdań języka, przeto nazywa się je niekiedy symbolami *metajęzykowymi*[8]. Jednym z ważniejszych symboli metajęzykowych jest w matematycznym opisie gramatyki tak zwany **aksjomat,** nazywany też korzeniem języka. W gramatyce dotyczącej języka naturalnego aksjomat można utożsamiać z ogólnym pojęciem "*poprawne zdanie*", a w przypadku języków programowania aksjomat to stwierdzenie "*prawidłowo zbudowana instrukcja*"[9].

---

[1] Na podobnej zasadzie funkcjonują znane kody pasmowe, za pomocą których identyfikuje się towary w sklepach.

[2] Badaczem, który zbudował podstawy lingwistyki matematycznej, był **Noam Chomsky.** Program badawczy Chomsky'ego przewidywał zbudowanie matematycznych podstaw gramatyki języka **naturalnego** (angielskiego), co się jednak nie powiodło, gdyż języki naturalne mają zbyt skomplikowaną i niekonsekwentną składnię, by się ją dało opisać matematycznie.

[3] Jeśli Cię to interesuje – przeczytaj którąś z książek znakomitego polskiego specjalisty, **prof. Andrzeja Bliklego.** Możesz też skorzystać ze skryptu: *R. Tadeusiewicz "Wstęp do informatyki", Akademia Ekonomiczna w Krakowie (wyd. III – 1993).*

[4] Translatory języków programowania tłumaczą napisane przez człowieka programy na język komputera, posługując się matematyczną definicją języka programowania jako główną podstawą swego działania.

[5] W ścisłej terminologii matematycznej ten wykaz nazywa się **alfabetem terminalnym.**

[6] Symbole, z których budowane są poprawne zdania (symbole terminalne), kojarzyć trzeba raczej z **wyrazami** języka naturalnego, a nie z literami, bo gramatyka bada poprawność budowy zdań (lub instrukcji języka programowania), a "budulcem" do tworzenia zdań są właśnie wyrazy, a nie litery. Ocena poprawności składania wyrazów z oddzielnych liter jest natomiast domeną ortografii.

[7] W matematyce reguły te rozważane jako specyficzne odwzorowania jednych napisów w inne.

[8] Metajęzyk to "język do opisu języka". W matematyce da się zdefiniować oddzielny język, w którym zapisane są reguły gramatyki, nie "mieszający się" z językiem podlegającym opisowi. W językach naturalnych jest to niemożliwe (do opisu gramatyki języka polskiego używa się ... języka polskiego), co prowadzi do różnych kłopotliwych paradoksów.

[9] Dla odróżnienia elementów terminalnych od nieterminalnych wprowadza się często zasadę, że napisy będące symbolami nieterminalnymi pisze się *kursywą* (pochylonymi literami), zaś symbole terminalne grubymi literami, a jeśli są to symbole jednoznakowe – używa się dodatkowo cudzysłowów.

Kluczową rolę w każdej gramatyce odgrywają reguły przekształcania symboli metajęzykowych w napisy zawierające inne symbole metajęzykowe i – ewentualnie – dopuszczalne symbole należące do języka. Reguły te nazywane są (niezbyt szczęśliwie) **produkcjami**. Użyteczność produkcji polega na tym, że dokonując zamiany jednych symboli na inne – zgodnie z zasadami podanymi przez produkcje – nie wykraczamy nigdy (z definicji) poza opisany gramatyką język. Dlatego chcąc zbadać, czy pewne zdanie należy do języka, możemy wyjść od aksjomatu i poszukiwać takiego łańcucha produkcji, by po pewnej liczbie przekształceń uzyskać wymagane zdanie. Taka technika dowodzenia, że zdanie należy do języka nazywa się **metodą generacyjną**. Możliwe jest także podejście odwrotne: Biorąc za początek elementy zdania i stosując zawarte w gramatyce produkcje "od prawej do lewej", można próbować "zwinąć" zdanie do aksjomatu. Taka technika nazywa się **metodą redukcyjną**. Jeśli uda się z aksjomatu generacyjnie wywieść potrzebne zdanie lub jeśli uda się je zredukować do aksjomatu – wówczas mamy dowód, że zdanie to jest poprawne w rozważanym języku. Niepowodzenie próby redukcji lub generacji dowodzi niepoprawności zdania.

Najlepiej w tym momencie odwołać się do przykładu. Zbudujemy gramatykę prostego języka, pozwalającego na budowę wyrażeń arytmetycznych. Dla prostoty ograniczymy przykład do użycia zaledwie trzech zmiennych i dwóch działań: dodawania i mnożenia (wprowadzenie większej liczby zmiennych, a także dowolnych stałych liczbowych tudzież pozostałych działań jest bardzo proste i oczywiste, możesz to zatem zrobić w ramach ćwiczeń). Niebanalnym elementem rozważanego języka będzie natomiast możliwość stosowania nawiasów. Oto elementy potrzebnej gramatyki[1], której aksjomatem będzie symbol nieterminalny *wyrażenie*:

wyrażenie = *wyrażenie* "+" *składnik*
wyrażenie = *składnik*
składnik = *składnik* "*" *czynnik*
składnik = *czynnik*
czynnik = "(" *wyrażenie* ")"
czynnik = "**x**"
czynnik = "**y**"
czynnik = "**z**"

Jak widać, zaproponowana gramatyka jest prosta, ma jednak zdolność definiowania całkiem skomplikowanych wyrażeń. Przykładowo udowodnijmy, że wyrażenie

x + y * (z + x)

należy do języka definiowanego przez tę gramatykę. Wybierzemy drogę generacyjną i wychodząc od aksjomatu języka spróbujemy zbudować wywód przytoczonej formuły[2].

wyrażenie = *wyrażenie* + ***składnik*** = *wyrażenie* + *składnik* * ***czynnik*** =
= *wyrażenie* + *składnik* * (***wyrażenie***) =
= *wyrażenie* + *składnik* * (*wyrażenie* + *składnik*) =
= **x + y * (z + x)**

Ostatni krok wywodu nieco "zagęściliśmy", dokonując od razu podstawienia za *wyrażenie* dwukrotnie *składnik*, potem czterokrotnie za *składnik* podstawiliśmy *czynnik* i wreszcie wszystkie symbole nieterminalne *czynnik* podmieniliśmy, używając potrzebnych symboli terminalnych; prawidłowy wywód powinien tu przewidywać oddzielne kroki.

Łatwo wykazać, że każde poprawne (z punktu widzenia algebry) wyrażenie zawierające wymienione wyżej symbole terminalne – będzie zaakceptowane przez podaną gramatykę. Łatwo także

---

[1] Do jej zapisu użyjemy powszechnie używanej notacji nazwanej **BNF**, którą wprowadzili po raz pierwszy **F. Backus** i **P. Naur** przy definiowaniu gramatyki języka ALGOL. Obecnie w powszechnym użyciu jest notacja rozszerzona, zwana **EBNF** (*Extended Backus – Naur Formalism*), której jednak w naszych prostych przykładach nie musimy wprowadzać.

[2] W każdym kroku wywodu stosuje się jedną produkcję z podanej listy. Żeby ułatwić śledzenie wywodu – za każdym razem wyróżniam pogrubionym drukiem ten symbol nieterminalny, który jest w następnym kroku zastępowany.

upewnić się, że gramatyka odrzucać będzie wyrażenia niepoprawnie zbudowane, gdyż nie da się zbudować wywodu dla takich (przykładowo) wyrażeń:

x + * y
) z + y (
x y + z
z +

Zachęcam Cię gorąco, abyś podjął trud prześledzenia, dlaczego przytoczone wyżej niepoprawne wyrażenia będą odrzucone przez tę gramatykę.

Ze względu na częste stosowanie w informatyce metody zapisu gramatyk formalnych są stale doskonalone. Na przykład w celu skrócenia zapisów wprowadza się czasami symbol pionowej kreski | dla rozdzielenia pisanych w jednej linii zapisów różnych następników tego samego symbolu nieterminalnego. Przykładowo, cztery ostatnie produkcje w naszej gramatyce można krócej zapisać w postaci

czynnik = "(" *wyrażenie* ")" | **x** | **y** | **z**

Inne udoskonalenie polega na zastosowaniu specjalnych symboli dla zapisu powtarzania pewnych fragmentów zapisu lub wyboru jednego z elementów. Powtarzany dowolną liczbę razy[1] fragment definicji określonego pojęcia ujmowany jest w nawiasy "{" oraz "}". Pozwala to na uniknięcie mało czytelnych na ogół definicji rekurencyjnych, takich jak użyte w opisie rozważanej przez nas prostej gramatyki określenia *wyrażenia* lub *składnika.* Oto jak te definicje można teraz zapisać:

wyrażenie = *składnik* {"+" *składnik*}
składnik = *czynnik* {"*" *czynnik*}

Ostatnie udoskonalenie polega na wprowadzeniu specjalnego oznaczenia elementów **opcjonalnych**[2]. Takie "uznaniowo" wprowadzane elementy produkcji oznacza się nawiasami "[" i "]". W opisanej wyżej gramatyce takich elementów nie było, natomiast bardzo prostym przykładem może tu być definicja liczby całkowitej

liczba = [−] *cyfra* {*cyfra*}

Wśród gramatyk formalnych wyróżnić można pewną klasyfikację. Dotychczas rozważaliśmy produkcje, w których po lewej stronie znaku "=" był używany pojedynczy symbol nieterminalny. Oznaczało to, że odpowiedniego podstawienia dokonać można niezależnie od kontekstu, w jakim rozważany symbol wystąpił. Z tego względu rozważane gramatyki nosiły nazwę **gramatyk bezkontekstowych.** Istnieje jednak znacznie szersza i ogólniejsza klasa **gramatyk kontekstowych,** w których o możliwości zastosowania danej produkcji decyduje kontekst, w jakim występuje dany symbol[3].

5.5. Teoria automatów

Proces analizy gramatycznej, dokonywany między innymi w translatorach języków programowania z wykorzystaniem gramatyk formalnych, przeprowadzany jest zwykle drogą wiodącą poprzez tak zwane **automaty skończone**[4]. Ponadto automaty stanowią bardzo przydatne modele szerokiej klasy procesów przetwarzania informacji w systemach operacyjnych i programach użytkowych[5].

[1] W szczególności być może w ogóle nie użyty, co odpowiada zerowej liczbie powtórzeń!

[2] To znaczy takich, które mogą wystąpić lub nie.

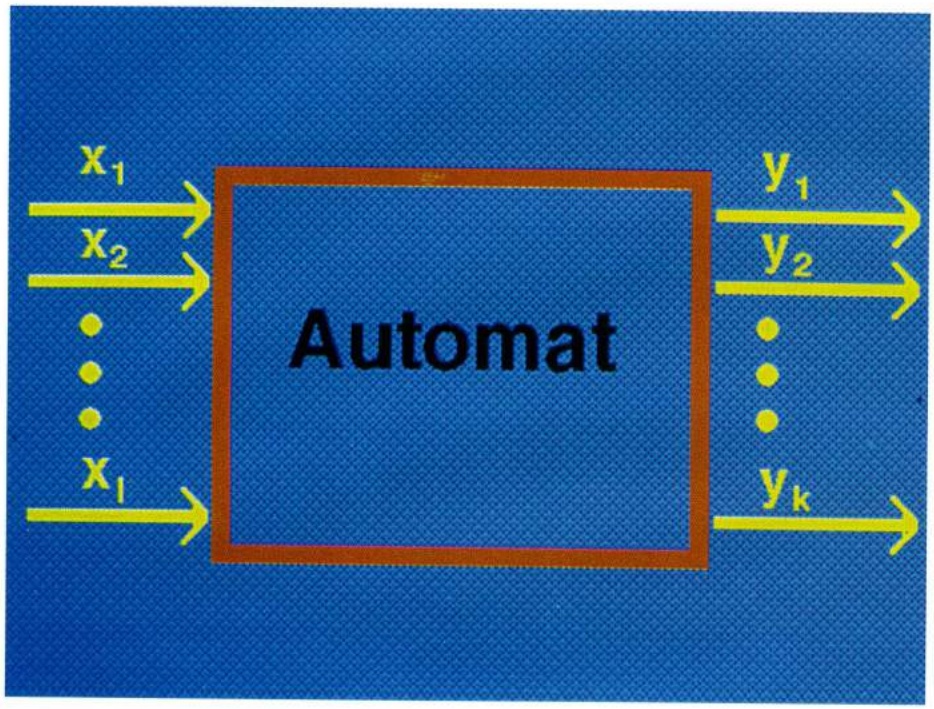
[3] Można wykazać, że klasa języków definiowanych przez gramatyki kontekstowe jest istotnie szersza, niż klasa języków definiowanych przez gramatyki bezkontekstowe. Przykładowo, wprowadzając język **ALGOL 68** musiano zdefiniować dla niego specjalną gramatykę kontekstową, zaś do zdefiniowania tej gramatyki wprowadzono specjalną notację **Van Wijngaardena.** Jednak dla większości języków programowania gramatyki bezkontekstowe są całkowicie wystarczające, natomiast złożoność gramatyk kontekstowych ogranicza ich przydatność z punktu widzenia techniki translacji (dla wspomnianego tu języka ALGOL 68 nigdy nie zbudowano powszechnie dostępnego translatora, gdyż było to zadanie zbyt skomplikowane dla większości komputerów).

[4] Odpowiedni automat jest zwykle nieodłącznym elementem tak zwanego *parsera*, czyli algorytmu dokonującego wstępnej analizy tłumaczonego tekstu (programu).

[5] Nie bez kozery szczególnie często spotyka się książki o tytułach ***Automaty i gramatyki, Języki formalne i automaty skończone, Teoria automatów z wprowadzeniem do lingwistyki matematycznej*** itp.

W dodatku zarówno jednostka centralna komputera, jak i wszystkie jego elementy pomocnicze (urządzenia peryferyjne) mogą być rozpatrywane jako automaty skończone, a teorii automatów skończonych używa się do projektowania systemów łączności, komputerów i mikroprocesorów[1].

Automatem skończonym jest układ mający (w ogólnym przypadku) l wejść i k wyjść, przy czym zasadniczym założeniem jest, że sygnały x występujące na poszczególnych wejściach należą do pewnego skończonego zbioru symboli wejściowych X i podobnie sygnały wyjściowe y należą do skończonego zbioru symboli wyjściowych Y. Właśnie ze względu na skończone zbiory symboli wejściowych i wyjściowych mówi się o automatach **skończonych.** Automaty takie reprezentowane są zwykle poprzez schematy blokowe, podobne do podanego na rysunku.



Automat mający tylko sygnały wejściowe i sygnały wyjściowe jest układem o dość ubogich możliwościach. W szczególności w automacie takim identyczny zestaw sygnałów wejściowych wywoła zawsze taki sam sygnał wyjściowy, uzasadnione jest więc przekonanie o prymitywnym i bardzo "płytkim" przetwarzaniu informacji przez ten typ automatu. Dla automatów tego typu wprowadzono nazwę **układy kombinacyjne.** Zajmują się nimi często elektronicy projektujący tak zwane układy logiczne – między innymi dla potrzeb informatyki. Jednak znacznie ciekawsze od układów kombinacyjnych są automaty obdarzone pamięcią, mające tradycyjną nazwę **układów sekwencyjnych.** W automatach takich reakcja na określony sygnał wejściowy zależna może być od historii automatu, czyli od tego, jakie sygnały odbierał automat w przeszłości. Dla formalnego odwzorowania wiedzy automatu o jego historii wprowadza się zazwyczaj pojęcie tak zwanych **stanów wewnętrznych**. Stany te, oznaczane **s,** wspólnie z sygnałami wejściowymi **x** określają sygnały wyjściowe **y**. Równocześnie automat otrzymujący sygnał wejściowy **x** zmienia swój stan wewnętrzny ze stanu **s** do **s'.** W ten sposób zachowanie automatu (wyrażające się jego sygnałami wyjściowymi) nie jest już jednoznacznie zdeterminowane przez sygnały wejściowe, możliwe jest więc ciekawsze i bogatsze działanie automatu i jego współdziałanie ze środowiskiem.

Zaletą automatów skończonych jest fakt, że funkcje mogą w nich być przedstawione za pomocą prostej tabelki, której kolejne wiersze odpowiadają wszystkim możliwym kombinacjom sygnałów wejściowych **x** i stanów wewnętrznych **s,** a kolumny podają: sygnały **x**, stany **s**, sygnały **y** oraz nowe stany **s'**. Opiszmy w ten sposób przykładowy automat. Odpowiednia tabelka podana jest na następnej stronicy.

Posługując się taką tabelą, możemy dowiedzieć się o rozważanym automacie praktycznie wszystkiego. Możemy w szczególności zawsze łatwo określić, jaka będzie sekwencja sygnałów wyjściowych przy określonej sekwencji sygnałów wejściowych.

---

[1] Jak z tego wynika, podstawowa wiedza na temat automatów skończonych jest informatykowi wprost niezbędna i podane informacje należy traktować jako encyklopedyczne wprowadzenie w temat, możliwe do poszerzenia przez skorzystanie z podanej na końcu literatury.

x	s	y	s'
A	1	X	2
B	1	Z	3
A	2	Z	3
B	2	Y	2
A	3	Y	3
B	3	X	3

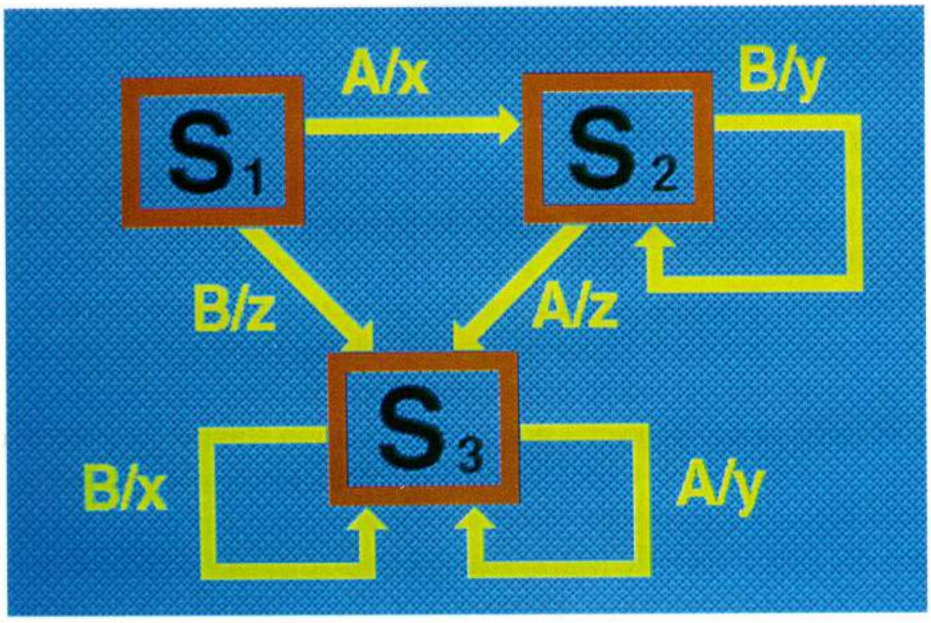
Przykładowo wyobraźmy sobie, że automat znajduje się w stanie 1 i otrzymuje na wejściu sygnały:

A B B B A B A A A B A

Przy odrobinie wysiłku możesz na podstawie przytoczonej tabeli stwierdzić, że na wyjściu powstanie ciąg sygnałów:

X Y Y Y Z X Y Y Y Y X Y

Spróbuj prześledzić, jak ten ciąg powstaje!



Reprezentacja automatu za pomocą tabelki jest mało przejrzysta, dlatego chętnie odwołujemy się do grafowej reprezentacji automatu. Reprezentacja taka powstaje w ten sposób, że tworzy się graf o tylu węzłach, ile stanów ma rozważany automat. Z każdego węzła (utożsamianego z określonym stanem) wychodzi tyle gałęzi, ile jest możliwych słów wejściowych. Gałęzie te są utożsamiane z działaniami automatu zachodzącymi pod wpływem sygnałów wejściowych. Przy każdej gałęzi pisze się symbol wejściowy, pod wpływem którego następuje określone przejście, oraz (po symbolu "/") symbol wyjściowy, jaki przy tym powstaje, a krawędzie grafu prowadzą do węzłów odpowiadających docelowym stanom (s') automatu. Graf dla opisanego wyżej automatu pokazano na rysunku.

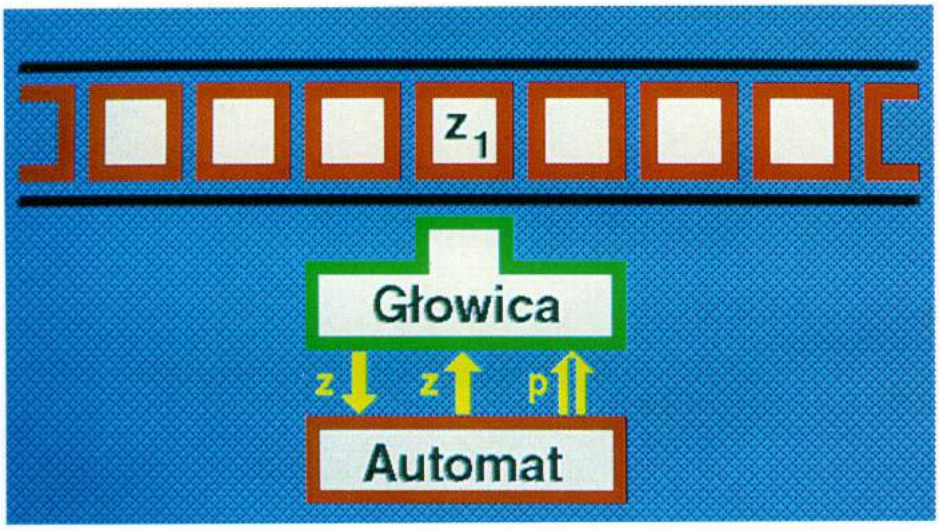
Teoria automatów jest dziedziną bardzo obszerną: można mówić o automatach logicznych (realizujących dowolne funkcje logiki matematycznej – takie właśnie wchodzą w skład każdego komputera), można rozważać automaty probabilistyczne (podane są prawdopodobieństwa pojawienia się określonych sygnałów i określonych stanów) i wiele innych. Dla nas ciekawe i inspirujące jest stwierdzenie, że **automat może działać według założonego algorytmu,** czyli może być realizatorem tego algorytmu. Dla różnych konkretnych algorytmów można tak dobrać elementy automatu, że będzie on realizatorem tych algorytmów. Jednak **można udowodnić, że nie każdy algorytm może**

być realizowany za pomocą automatu[1]. Z tym większym zainteresowaniem powinieneś przestudiować kolejny rozdział, w którym opisałem system, który ma tę własność, że **może wykonać każdy algorytm.**

5.6. Maszyna Turinga

Jak wspomniano w zakończeniu poprzedniego podrozdziału, automat nie może być realizatorem dowolnego algorytmu, natomiast ma taką możliwość automat połączony – jak to pokazano na rysunku poniżej – z potencjalnie nieskończoną pamięcią, w której zawarta jest informacja o metodzie działania algorytmu. Budowę takiego tworu zaproponował w **1937** roku **Allan M. Turing**, matematyk brytyjski, którego prace na temat metod obliczania, algorytmów i możliwości maszyn wyprzedzały epokę komputerów i torowały jej drogę. Maszyna Turinga składa się z trzech części:

- nieskończenie długiej taśmy, podzielonej na kwadraty, na której zapisane mogą być dowolne symbole z pewnego ustalonego alfabetu;
- głowicy, która może odczytywać symbol z tego kwadratu taśmy, który znajduje się aktualnie pod nią oraz może w tym kwadracie wpisać dowolny inny symbol;
- automatu, który otrzymuje od głowicy symbol aktualnie odczytany z taśmy, wysyła do głowicy symbol do zapisania a ponadto (dodatkowym wyjściem) steruje ruchami głowicy, wysyłając sygnały {LEWO, STOP, PRAWO}.



Schemat maszyny Turinga przedstawia fotografia obok. Samej maszyny nie udało mi się sfotografować, gdyż nikt jej jeszcze nie zbudował. I jest całkiem pewne, że nie zbuduje! Brakuje bowiem – bagatela – nieskończenie długiej taśmy. reprezentującej **potencjalnie** nieskończenie wielką pamięć. Tylko tyle i aż tyle dzieli nas od możliwości wykonywania dowolnego algorytmu.

Od momentu ogłoszenia pierwszych prac Turinga badaczy najbardziej fascynował inny problem: znalezienia **uniwersalnej maszyny Turinga**, gdyż zależnie od wyboru parametrów automatu, maszyna ta może realizować pewne konkretne algorytmy, zaś innych – nie. Pierwszą uniwersalną, to znaczy zdolną do wykonania dowolnego algorytmu, maszynę Turinga –wymyślił i zaproponował **Shannon** (ten od teorii informacji), natomiast obecnie badaczy interesuje problem zbudowania minimalnej uniwersalnej maszyny Turinga – to znaczy konstrukcji charakteryzującej się najmniejszą wartością iloczynu liczby stanów automatu i liczby symboli alfabetu. Najlepszy znany dziś wynik odpowiada wartości 24 (4 symbole i 6 stanów) i jest związany z nazwiskiem **A. Trittera**. Czy jest to rzeczywiste optimum? Nie wiadomo.

Może Ty Czytelniku uwiecznisz swoje imię w historii informatyki, podając lepsze rozwiązanie?

---

[1] Na przykład bardziej znanym algorytmem, który nie może być zrealizowany za pomocą żadnego automatu, jest algorytm wydzielania z ciągu liczb tych, które są pełnymi kwadratami. Dowód tego stwierdzenia jest zbyt obszerny i zbyt trudny, żeby go tu przytaczać – samo twierdzenie jest jednak prawdziwe.

6. Zakończenie

Ograniczona objętość tej książki nie pozwala na dalsze wzbogacanie jej treści, chociaż wiele zagadnień zostało zaledwie zasygnalizowanych, a na inne w ogóle nie starczyło miejsca. Dlatego zachęcamy Cię gorąco, jeśli pracowicie dotarłeś aż do tego miejsca, abyś nie ustawał w wysiłkach i kształcił się nadal. We wstępie wskazaliśmy na celowość korzystania z periodyków: naukowych i fachowych, a także tych przeznaczonych głównie dla hobbystów, zaś na następnej stronicy podana jest literatura, z której korzystaliśmy podczas pisania książki i która w wielu zagadnieniach oferuje znacznie obszerniejszą i bardziej wyczerpującą wiedzę niż to co udało się zawrzeć w tej książce.

Musisz pamiętać – i to jest chyba najważniejsze – że **w informatyce nie ma nic stałego** (oprócz stałości zmian). Książka pomogła Ci w zdobyciu podstaw i ułatwi korzystanie z dalszej literatury. Jednak **stałe** czytanie specjalistycznej prasy komputerowej, poszukiwanie nowości, stała gotowość do wychodzenia naprzeciw kolejnym rewelacjom, jakie nieść będzie postęp w zakresie sprzętu, oprogramowania i zastosowań techniki komputerowej – to najważniejsze cechy prawdziwego informatyka. Musisz bowiem nieustannie pamiętać o tym, co ponad 40 lat temu napisał twórca cybernetyki, **Norbert Wiener:**

"(...) każda maszyna jest tylko tyle warta,

ile wart jest człowiek, który ją obsługuje (...) "

Kiedy pisał te słowa, nikt nie słyszał o komputerach, jednak ich powstanie i rozwój potwierdziły słuszność tezy czcigodnego profesora. I dlatego w zakończeniu tej książki chcemy Ci szczerze i serdecznie życzyć, żeby **Twoje** komputery – były bardzo wiele warte!

7. Literatura

1. Davies D.W., Barber D.L.A.: *Sieci teleinformatyczne*, WNT, Warszawa 1979.
2. Gookin D.: *DOS dla opornych*, Oficyna Wydawnicza Read Me, Warszawa 1993.
3. Graham N.: *Introduction to Computer Science* (Third Edition), West Publ. Co., San Francisco 1987.
4. Harel D.: *Rzecz o istocie informatyki – algorytmika.* WNT, Warszawa 1992.
5. Kozdrowicz T.: *IBM PC i PC DOS,* PWN, Warszawa 1991.
6. Lewis A.: *Microsoft Excel 4.0,* WNT, Warszawa 1993.
7. Linde P.: *Notron Commander*, Oficyna Wydawnicza Read Me, Warszawa 1993.
8. Livingston B.: *Sekrety Windows 3.1.* Wibet, Pruszków 1993.
9. Łopuch B.: *Poznaj swój komputer*, Lynx–SFT, Warszawa 1993.
10. Mano M. M.: *Architektura kompiuterów*. WNT, Warszawa 1988.
11. Niedzielska E., Skwarnik M. (ed.): *Projektowanie systemów informatycznych.* PWE, Warszawa 1993.
12. Porębski W.: *IBM dla każdego*, Oficyna Wydawnicza Read Me, Warszawa 1993.
13. Rafa J.: *MS–DOS dla dociekliwych*, Lynx–SFT, Warszawa 1993.
14. Rocki M.: *Elementy informatyki dla szkół podstawowych*, Wydawnictwa Edukacyjne, Warszawa 1993.
15. Sikorski W.: *System DOS 5.0 dla początkujących,* Oficyna Wydawnicza Read Me, Warszawa 1992.
16. Sikorski W.: *Przewodnik po IBM,* WNT, Warszawa 1993.
17. Sikorski W.: *Leksykon komputerowy dla początkujących,* WNT, Warszawa 1993.
18. Tadeusiewicz R.: *Komputerowe Przygody – Atari LOGO,* WNT, Wydanie 2, Warszawa 1991.
19. Tadeusiewicz R.: *Wstęp do informatyki,* Skrypt uczelniany AE, Wydanie 3, Kraków 1993.
20. Tadeusiewicz R.: *Eureka.* Seria PPP, WNT, Wydanie 3, Warszawa 1992.
21. Tadeusiewicz R.: *Programowanie w języku PL/1*, Skrypt uczelniany AGH, Kraków 1984.
22. Tadeusiewicz R.: *Edytory tekstowe dla ATARI*, Informatyka Mikrokomputerowa, SOETO, Warszawa 1988.
23. Tadeusiewicz R.: *Programowanie w języku LOGO,* Skrypt uczelniany WSP, Kraków 1992.
24. Tanenbaum A. S.: *Sieci komputerowe.* WNT, Warszawa 1988.
25. Walat A.: *Elementy informatyki dla szkół średnich,* Wydawnictwa Edukacyjne, Warszawa 1993.
26. Weitzman C.: *Systemy minikomputerowe – struktura i zastosowanie.* WNT, Warszawa 1979.

WYDAWNICTWA NAUKOWO-TECHNICZNE
00-048 Warszawa, Mazowiecka 2/4
TEL./FAX 26-82-93

INFORMATYKA

⇒ ***Oferujemy za zaliczeniem pocztowym następujące książki:***

• *Abelson H., diSessa A.A.*: GEOMETRIA ŻÓŁWIA. KOMPUTER JAKO ŚRODEK DO POSZUKIWAŃ MATEMATYCZNYCH	54.000 zł
• *Barta J., Markiewicz R.*: GŁÓWNE PROBLEMY PRAWA KOMPUTEROWEGO	70.000 zł
• *Bentley J.*: PEREŁKI OPROGRAMOWANIA	53.500 zł
• *Buren Van Ch.*: ZRÓB TO W WINDOWS	85.000 zł
• *Christopher, Jr.K.W., Feigenbaum B.A., Saliga S.O.*: DOS 5. COMMAND REFERENCE PO POLSKU	90.000 zł
• *Delannoy C.*: ĆWICZENIA Z JĘZYKA C++	90.000 zł
• *Dunn S., Bermant Ch., Berst J.*: WINDOWS 3.1. 101 WSKAZÓWEK I TRIKÓW	50.000 zł
• *Ferbrache D.*: PATOLOGIA WIRUSÓW KOMPUTEROWYCH	120.000 zł
• GRAFIKA KOMPUTEROWA. METODY I NARZĘDZIA. *Praca zbiorowa*	210.000 zł
• *d'Hardancourt A.*: MICROSOFT® ACCESS	70.000 zł
• *Harel D.*: RZECZ O ISTOCIE INFORMATYKI. ALGORYTMIKA	65.000 zł
• *Harris W.*: BAZY DANYCH NIE TYLKO DLA LUDZI BIZNESU	120.000 zł
• *Kernighan B.W, Ritchie D.M.* Język ANSI C	150.000 zł
• *MacInnes J.* dBASE IV	50.000 zł
• *Nelson K.Y.*: WINDOWS WERSJA 3.1	60.000 zł
• *Nelson K.Y.*: WORDPERFECT W ŚRODOWISKU WINDOWS	90.000 zł
• *Schieb J.*: MS-DOS 6	90.000 zł
• *Silberschatz A., Peterson J.L., Galvin P.B.*: PODSTAWY SYSTEMÓW OPERACYJNYCH	140.000 zł
• *Sillescu D.*: WORDPERFECT 6.0	80.000 zł
• *Weston L.*: OD EDYTORA DO EDYTORA	60.000 zł
• *Will-Harris D.*: RECEPTY DR. DANIELA NA DOLEGLIWOŚCI WINDOWS	46.000 zł
• *Wood L.*: POCZTA ELEKTRONICZNA. MODEMY I PROGRAMY	160.000 zł

MIKROKOMPUTERY

• *Frelek B.*: COMMODORE 64	47.000 zł
• *Holland R.*: TESTOWANIE I DIAGNOSTYKA SYSTEMÓW MIKROKOMPUTEROWYCH	90.000 zł
• *Małysiak H., Pochopień B., Wróbel E.*: MIKROKOMPUTERY KLASY IBM PC	48.000 zł
• *Wolisz A.*: PODSTAWY LOKALNYCH SIECI KOMPUTEROWYCH	
t. 1 – Sprzęt sieciowy	50.000 zł
t. 2 – Oprogramowanie komunikacyjne i usługi sieciowe	50.000 zł

BIBLIOTEKA INŻYNIERII OPROGRAMOWANIA

• *Iglewski M., Madey J., Matwin S.*: PASCAL. STANDARD	48.000 zł
• *Klin M.Ch., Pöschel R., Rosenbaum K.*: ALGEBRA STOSOWANA DLA MATEMATYKÓW I INFORMATYKÓW	60.000 zł
• *Mirkowska G., Salwicki A.*: LOGIKA ALGORYTMICZNA DLA PROGRAMISTÓW	34.000 zł
• *Mykowiecki T.*: dBASE, FOXBASE, BAZY DANYCH	33.500 zł
• *Robling Denning D.E.*: KRYPTOGRAFIA I OCHRONA DANYCH	135.000 zł
• *Silvester P.P.*: SYSTEM OPERACYJNY UNIX™	44.000 zł
• *Tsichritzis D.C., Lochovsky F.H.*: MODELE DANYCH	25.000 zł

ĆWICZ Z NAMI

• *Booth D.*: SYMPHONY 2 (*zarządzanie i operowanie informacjami*)	50.000 zł
• *Botto F.*: PARADOX (*nowoczesna relacyjna baza danych*)	50.000 zł
• *Lewis A.*: MICROSOFT EXCEL 4.0	60.000 zł

Zamówienie na wybrane książki proszę przesyłać do Działu Handlowego WNT:
DH WNT, skr. poczt. 359, 00-950 Warszawa