1
Poznajemy nasz mikrokomputer

W pierwszym rozdziale zapoznamy się ze sposobem uruchamiania i obsługiwania mikrokomputerów *Meritum* i *Spectrum*. Pierwsze dwa podrozdziały dotyczą *Meritum*, dwa ostatnie *Spectrum*. Podrozdział 1.3 dotyczy zagadnień wspólnych dla większości mikrokomputerów z tym jednak, że wszystkie podane tam przykłady poleceń pochodzą z zestawu poleceń realizowanych przez *Meritum*. Ze względu na sposób podania materiału, użytkowników *Spectrum* zachęcam do przeczytania rozdziału w całości.

1.1
Uruchamiamy mikrokomputer Meritum

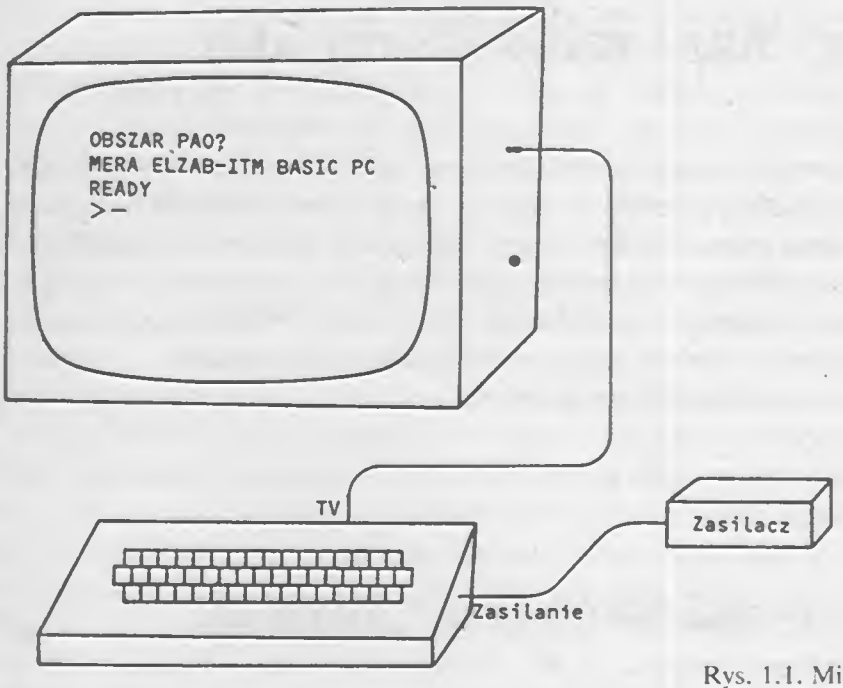
Mikrokomputer *Meritum* mieści się w pudełku wielkości dużej książki. **Po dołączeniu zasilacza i włączeniu napięcia mikrokomputer jest gotów do pracy — już działa.** Klawiatura wskazuje sposób naszego porozumiewania się: wszystkie polecenia i informacje dla mikrokomputera będziemy pisać na klawiaturze. Swoje odpowiedzi mikrokomputer napisze (wyświetli) na ekranie telewizora, połączonego odpowiednim kablem z gniazdem na jego obudowie. Ściśle biorąc, mikrokomputer *Meritum* jest przystosowany do bezpośredniej współpracy z monitorem, czyli telewizorem, z którego usunięto układy wielkiej częstotliwości. Jeżeli nie mamy monitora, możemy wykorzystać dowolny odbiornik telewizyjny, łącząc go z komputerem poprzez specjalny adapter OTV dostarczany na zamówienie przez producenta komputera. Telewizor musimy dostroić do częstotliwości pracy adaptera, znajdującej się w zakresie 3-4 kanału telewizyjnego. Wymienione urządzenia: mikrokomputer z klawiaturą, zasilacz i monitor (lub telewizor z adapterem OTV) tworzą minimalny użyteczny zestaw komputera osobistego (rys. 1.1). I to jest wszystko, co musimy wiedzieć, aby przystąpić do pracy. Zacznijmy od czegoś prostego.

Pierwszym obrazem wyświetlonym na ekranie po włączeniu napięcia jest napis:

OBSZAR PAO? -

stanowiący zapytanie komputera, czy chcemy zarezerwować część pamięci na cele specjalne. Nie chcemy; przyciskamy więc klawisz z napisem *ENTER*, na co komputer odpowiada tekstem

MERA ELZAB - ITM BASIC PC
READY
> –



Rys. 1.1. Mikrokomputer *Meritum*

z którego ważne są jedynie słowa: *BASIC* i *READY*. Pierwsze z nich informuje, że możemy porozumiewać się z komputerem w języku *Basic*, drugie stanowi potwierdzenie gotowości komputera do pracy. Technika komputerowa, w tym mikrokomputery, powstała i rozwinęła się w USA, stąd większość słów i zwrotów używanych w komunikowaniu się z komputerami jest zaczerpnięta z języka angielskiego. Słowo *READY* znaczy po polsku *GOTÓW*. Wyświetlony na ekranie znak - (kreska), nazywany **kursorem**, wskazuje miejsce, w którym pojawią się pisane przez nas polecenia. Jeżeli chcemy, aby komputer coś zrobił, musimy mu bardzo dokładnie powiedzieć, czego oczekujemy. Wydajemy więc polecenie (tzn. piszemy na klawiaturze)

PRINT 5

co po polsku znaczy *DRUKUJ 5*. Na ekranie nic się nie dzieje, dlaczego? Dlatego, że komputer wykona polecenie dopiero wtedy, gdy upewni się, że już napisaliśmy je do końca. **Znakiem zakończenia jest naciśnięcie klawisza** *ENTER* (po polsku: *WPROWADŹ*), **którym musimy kończyć każde polecenie i każdą linię programu.**

Naciskamy więc *ENTER* i na ekranie pojawia się wynik

5

wraz ze znanym już tekstem
READY
> –
sygnalizującym gotowość wykonania dalszych poleceń.

A co będzie, jeżeli się pomylimy? Sprawdźmy. Po napisaniu błędnego polecenia, np.
PRIND 5
(i naciśnięciu klawisza *ENTER*) na ekranie ukaże się napis
? SN ERROR
stanowiący komunikat o błędzie. Słowo *ERROR* znaczy *BŁĄD*, a pozostałe dwie litery określają typ błędu; *SN* oznacza błąd w składni polecenia. Dalsza część wyświetlonego napisu
READY
> –
upewnia nas, że komputer jest gotów przyjąć następne polecenia. Pełna lista kodów błędów jest podana w fabrycznej instrukcji komputera.

Wszystko wygląda prosto, warto jednak zastanowić się przez chwilę, co właściwie komputer zrobił. A wykonał on wiele skomplikowanych czynności. Najpierw **odczytał tekst polecenia, wyświetlając go jednocześnie na ekranie** (klawiatura nie jest połączona z telewizorem — aby znak pojawił się na ekranie, musi zostać wysłany tam przez komputer). Po odebraniu znaku *ENTER* przeanalizował tekst i stwierdził, jakie zawiera polecenie. Dopiero po analizie tekstu i rozpoznaniu polecenia wydruku (*PRINT*) **obliczył wartość argumentu i wyświetlił wynik na ekranie**. Komunikat o wystąpieniu błędu w drugim przykładzie był spowodowany niezrozumiałym dla komputera słowem *PRIND*. Zwróćmy uwagę na zwrot „obliczył wartość argumentu". Aby go wyjaśnić, napiszmy kolejne polecenie
PRINT 5+2
Po naciśnięciu klawisza *ENTER* na ekranie pojawi się wynik
7
A więc komputer nie wyświetla tekstu umieszczonego za słowem kluczowym *PRINT*, lecz rozpoznając tam **wyrażenie arytmetyczne** oblicza jego wartość i wyświetla wynik. Reguły obliczania wartości wyrażeń są zgodne z regułami arytmetyki *:
— najpierw potęgowanie (oznaczone symbolem ↑, np. $3 \uparrow 2$ oznacza 3^2);
— następnie mnożenie (znak *) i dzielenie (znak /);
— na końcu dodawanie (znak +) i odejmowanie (znak –).
Działania w nawiasach są wykonywane w pierwszej kolejności.
Jeżeli więc napiszemy polecenie
*PRINT (4 + 3↑2 – 6 * (8 + 6))/7*
zawierające wyrażenie $(4 + 3^2 - 6(8 + 6))/7$, to na ekranie otrzymamy wynik
–10.1429

---

* Dokładniejszy opis wyrażeń arytmetycznych podano w p. 2.2.

Zwróćmy uwagę, że **do oddzielenia części całkowitej od ułamkowej jest użyta kropka (.) a nie przecinek**. Tę samą konwencję musimy zachować umieszczając niecałkowite liczby w treści poleceń. Użycie przecinka w tej roli nie zostanie przez komputer zrozumiane.

Napiszmy teraz dla porównania polecenie

PRINT "5+2"

Tym razem na ekranie zostanie wyświetlony napis

5+2

Oczywiście po naciśnięciu klawisza *ENTER*; w dalszym ciągu nie będziemy już o tym warunku przypominać. Umieszczenie dowolnego napisu w cudzysłowie powoduje, że jest on traktowany jako tekst, tzn. łańcuch znaków, z których jest zbudowany. Obydwa rodzaje argumentów, tzn. liczby i wyrażenia arytmetyczne oraz teksty mogą być łączone w tym samym poleceniu, np.

PRINT "DODAWANIE:", "5+2 ROWNA SIE"; 5+2

Polecenie to zawiera trzy argumenty:

— dwa teksty: *"DODAWANIE:"* i *"5+2 ROWNA SIE"*

— jedno wyrażenie: *5+2*

Argumenty są oddzielone (odseparowane) znakami: , (przecinek) oraz ; (średnik). Rola tych znaków jest różna:

; (średnik) — powoduje wyświetlenie następnego argumentu bezpośrednio za poprzednim,

, (przecinek) — jest znakiem tabulacji przesuwającym miejsce wyświetlenia następnego argumentu na pozycję 16, 32 lub 48 znaku w wierszu.

Po naciśnięciu klawisza *ENTER* na ekranie pojawi się więc napis

DODAWANIE: 5+2 ROWNA SIE 7

Powróćmy na chwilę do błędów i pomyłek. **Jeżeli zauważymy pomyłkę przed naciśnięciem klawisza *ENTER*, to możemy błąd poprawić.** Naciskając klawisz ze znakiem ← (strzałka w lewo) powodujemy cofanie kursora i kasowanie kolejnych znaków. W miejsce skasowanych możemy wpisać nowe. Naciskając klawisz funkcyjny *SHIFT* i jednocześnie klawisz ze strzałką ← cofamy kursor na początek linii z jednoczesnym skasowaniem wszystkich napisanych znaków.

Podsumowanie

— umiemy uruchomić komputer: wystarczy dołączyć monitor lub telewizor i włączyć zasilanie;

— umiemy wykonywać obliczenia typu kalkulatorowego i wyświetlać teksty: za pomocą polecenia *PRINT*; wiemy też jak budować wyrażenia arytmetyczne;

— wiemy, że każde polecenie kończymy naciskając klawisz z napisem *ENTER*;

— umiemy poprawiać błędnie napisany tekst polecenia: błędy kasujemy naciskając klawisz ← i dopisując tekst poprawny.

Pytania i zadania

1. Jakie urządzenia są niezbędne do uruchomienia mikrokomputera *Meritum*?
2. Które z podanych wyrażeń są poprawne, a które błędne:
   1 (((1 + 2) + 4.1)*5) (1,2+2) 22.3↑2.1/8
3. Napisz polecenia powodujące obliczenie wszystkich wyrażeń z pytania 2.
4. Napisz polecenie, które wyświetli na ekranie: osiem gwiazdek, przerwę o długości równej ośmiu gwiazdkom, osiem gwiazdek.
5. Napisz polecenie powodujące wyświetlenie na ekranie własnego imienia.

1.2
Piszemy program

Wszystkie polecenia pisane przez nas w p. 1.1 były przez komputer natychmiast wykonywane i zapominane. Poza najprostszymi przypadkami nie jest to zbyt dogodne. Kilkakrotne wykonanie tego samego ciągu poleceń wymaga za każdym razem napisania go od nowa. Znacznie dogodniejsza jest na ogół możliwość rozdzielenia etapu wprowadzania poleceń i późniejszego ich wykonania. Zauważmy, że ciąg poleceń dokładnie określa wymagane działanie maszyny. Stanowi on **program** działania komputera. Polecenia wchodzące w skład programu są zwyczajowo nazywane **instrukcjami**.

W języku *Basic* programem jest ciąg instrukcji opatrzonych numerami. Po rozpoczęciu wykonania programu instrukcje są wykonywane kolejno, w kolejności wyznaczonej przez ich numerację. Na przykład program

```
10 PRINT, "  *"
20 PRINT, " ***"
30 PRINT, "*****"
```

narysuje na ekranie trójkąt równoramienny o podstawie złożonej z pięciu gwiazdek.

W czasie pisania programu każdą linię musimy kończyć naciśnięciem klawisza *ENTER*. Umieszczenie numerów przed instrukcjami informuje komputer, że są to instrukcje programu, które należy zapamiętać, a nie kolejne polecenia, które należy natychmiast wykonać. Wykonanie zapamiętanego programu rozpocznie się dopiero po napisaniu polecenia

```
RUN
```

(oczywiście bez numeru). Polecenie to, jak wszystkie inne polecenia, jest realizowane przez mikrokomputer natychmiast, powoduje więc rozpoczęcie wykonania aktualnie zapamiętanego programu.

Na razie znamy tylko jedną instrukcję języka *Basic*: instrukcję *PRINT*.

Zanim poznamy dalsze, musimy poświęcić chwilę uwagi argumentom instrukcji, którymi mogą być nie tylko liczby, wyrażenia arytmetyczne i teksty, ale również **zmienne**. Pojęcie zmiennej znamy z matematyki: jest to pewien obiekt, któremu możemy nadać dowolną wartość. Rozumienie zmiennej w języku *Basic* jest takie samo. Również forma zapisu zmiennych jest zbliżona do postaci przyjętej w matematyce *, tzn. możemy je nazywać *X*, *Y*, *A*, *A1* itp. Istotną nowością jest natomiast to, że wartościami niektórych zmiennych mogą być nie liczby, ale teksty. Zmienne, których wartościami są teksty noszą nazwę **zmiennych tekstowych** i są wyróżniane przez dodanie na końcu nazwy znaku *$* (symbol dolara), np. *X$*, *A$*, *Z$*. Nadanie wartości zmiennej może nastąpić w wyniku wykonania różnych instrukcji programu. Poznamy jedną z nich.

Instrukcja *INPUT* — wprowadzania danych, pozwala nadawać zmiennym wartości pisane na klawiaturze w czasie wykonania programu. Argumentami instrukcji *INPUT* mogą być pooddzielane przecinkami nazwy dowolnych zmiennych. W chwili pisania instrukcji wartości, które zostaną nadane zmiennym są jeszcze nieznane. Dla ilustracji napiszemy następujący program działający wyłącznie na tekstach

```
10 PRINT "JAK MASZ NA IMIE?"
20 INPUT X$
30 PRINT "HALO "; X$;", MILO MI CIE POZNAC."
```

Jeżeli mamy przed sobą komputer, to możemy prześledzić sposób wykonania programu. **Wykonanie programu zaczyna się zawsze od linii o najmniejszym numerze** — w naszym przykładzie 10. Znajduje się tu instrukcja *PRINT*, której wykonanie powoduje wyświetlenie na ekranie podanego w cudzysłowie tekstu. **Bezpośrednio po zakończeniu wykonania instrukcji komputer rozpoczyna wykonanie następnej instrukcji, o następnym numerze.** W naszym przypadku jest to instrukcja *INPUT*, której wykonanie przebiega w dwóch etapach. Najpierw komputer wyświetla w nowym wierszu znak zapytania (*?*), aby pokazać, że oczekuje na wprowadzenie danych, potem przechodzi do oczekiwania na napisanie przez nas wartości zmiennej wymienionej w treści instrukcji — w naszym przykładzie *X$*. Wykonanie naszego programu do tego miejsca spowoduje więc wyświetlenie na ekranie fragmentu konwersacji

```
JAK MASZ NA IMIE?
? -
```

Teraz nasza kolej. Podajemy swoje imię, które komputer zapamiętuje jako wartość zmiennej tekstowej *X$*, po czym przechodzi do wykonania ostatniej instrukcji — *PRINT*. W rezultacie, cała konwersacja widoczna na ekranie ma postać

```
JAK MASZ NA IMIE?
? KRZYS
```

---

* Dokładniejszy opis nazw zmiennych zostanie podany w p. 2.2.

HALO KRZYS, MILO MI CIE POZNAC
READY
> –

Ostatnie dwa wiersze zawierają znany nam komunikat kwitujący poprawne zakończenie wykonania programu i sygnalizujący gotowość komputera do przyjęcia nowego polecenia (lub nowego programu).

Poprawne wykonanie programu jest możliwe tylko wtedy, gdy nie zawiera on błędów. W czasie wykonywania programu komputer sprawdza poprawność wszystkich instrukcji i sygnalizuje zauważone błędy komunikatem złożonym z dwuliterowego kodu typu błędu, słowa *ERROR* i numeru wiersza, w którym błąd wystąpił. Po wyświetleniu komunikatu, komputer przechodzi w stan oczekiwania na dalsze polecenia lub linie programu. Wyjątkowym przypadkiem jest wykrycie błędu składniowego. Błędna instrukcja musi zostać poprawiona, komputer przechodzi więc automatycznie do trybu edycji, w którym jest możliwe poprawienie dowolnych znaków w linii programu. Na ekranie pojawia się numer błędnej instrukcji, np.

> RUN
SN ERROR IN 30
30 –

i komputer oczekuje na wprowadzenie poleceń edycyjnych. Opis edytora (podany w fabrycznej instrukcji komputera) jest dość skomplikowany, a jego znajomość nie jest niezbędna. Na kilku dalszych stronach poznamy prostszy sposób zmiany całych linii programu, a informację o edytorze ograniczymy do stwierdzenia, że wyjście z trybu edycji następuje po naciśnięciu klawisza *ENTER*.

Po zakończeniu wykonania program jest nadal przechowywany w pamięci mikrokomputera i może zostać ponownie uruchomiony tym samym poleceniem *RUN*. Zróbmy to nadając zmiennej *X$* np. wartość Iza. Na ekranie powstanie obraz konwersacji

JAK MASZ NA IMIE?
? IZA
HALO IZA, MILO MI CIE POZNAC

zakończony tym samym komunikatem stwierdzającym zakończenie wykonywania programu.

Powróćmy na chwilę do etapu pisania programu. Każda instrukcja ma tu swój numer, przy czym w naszym programie numery zmieniają się nie co 1, lecz co 10. Świadczy to o naszej przezorności, gdyż dzięki temu możemy łatwo dopisywać nowe instrukcje i wstawiać je przed, lub pomiędzy już istniejące. Na przykład, możemy na początku konwersacji przedstawić nasz komputer. W tym celu napiszmy instrukcję

5 PRINT "DZIEN DOBRY, TU KOMPUTER MERITUM"

Po odczytaniu instrukcji mikrokomputer przejrzy cały zapamiętany program, znajdzie miejsce, w którym według kolejności numeracji powinna znaleźć się nowa linia i wstawi tam napisaną instrukcję. W rezultacie ostatnio na-

pisana instrukcja znajdzie się na początku programu. Możemy to łatwo sprawdzić pisząc polecenie

LIST

które powoduje wyświetlenie, czyli wylistowanie na ekranie całego zapamiętanego programu

5 PRINT "DZIEN DOBRY, TU KOMPUTER MERITUM"
10 PRINT "JAK MASZ NA IMIE?"
20 INPUT X$
30 PRINT "HALO "; X$;" , MILO MI CIE POZNAC"

W ten sam sposób możemy dodać linie o numerach np. *11* lub *28*, które zostaną wstawione w miejsca wskazywane przez ich numerację. A co będzie, jeżeli oznaczymy linię, którą chcemy wpisać, numerem już istniejącej linii programu, np.

30 PRINT X$; " ? A CO TO ZA POGANSKIE IMIE?"

Komputer zrobi to samo co poprzednio — wstawi nową linię do programu w miejsce wynikające z numeracji. Tym samym „stara" linia z numerem *30* zostanie zniszczona. Jeżeli masz dostęp do komputera, możesz to natychmiast sprawdzić za pomocą polecenia *LIST*, które spowoduje wyświetlenie na ekranie nowego tekstu programu ze zmienioną linią *30*. W taki sam sposób możemy zmieniać i inne linie programu. Możemy też zbędne linie kasować, pisząc sam numer,np. *20*

Jest to tzw. pusta linia, która zastępuje „starą" linię z numerem *20*, powodując jej usunięcie.

Możliwość dopisywania i wymiany linii będziemy wykorzystywać do wprowadzania zmian i poprawek do napisanych programów. Zanim jednak zmodyfikujemy nasz program, zapoznajmy się z nieco inną postacią instrukcji *INPUT*. W rozszerzonej wersji tej instrukcji jako pierwszy argument podajemy tekst umieszczony w cudzysłowie i oddzielony od dalszych argumentów średnikiem. W chwili wykonania instrukcji komputer wyświetli cały tekst, kończąc go znakiem zapytania (?), po czym przejdzie do oczekiwania na wprowadzenie danych. Dopiszmy do naszego programu

10 INPUT "JAK MASZ NA IMIE?"; X$

Po uruchomieniu programu poleceniem *RUN* na ekranie pojawi się pytanie

DZIEN DOBRY, TU KOMPUTER MERITUM.
JAK MASZ NA IMIE? –

Tak jak w poprzednich przykładach możemy teraz podać swoje imię i otrzymać uprzejmą odpowiedź komputera.

Odpowiednie numerowanie linii programu może być prowadzone automatycznie przez komputer. Polecenie *AUTO*, np.

AUTO 50, 10

rozpoczyna numerowanie linii przez komputer poczynając od numeru podanego jako pierwszy parametr polecenia (tu *50*) ze zwiększeniem numeru co linia o wartość podaną jako drugi parametr (tu *10*). Powrót do trybu bezpośre-

dniego wykonywania poleceń wymaga naciśnięcia klawisza *BREAK* (*przerwij*). Na ekranie pojawi się

READY

>–

sygnalizując gotowość do przyjęcia kolejnych poleceń lub dowolnie numerowanych instrukcji.

W czasie listowania programu (polecenie *LIST*) zawartość ekranu jest przesuwana do góry z usuwaniem szczytowych linii. Szybkość przesuwania uniemożliwia obejrzenie programu, który nie mieści się w całości na ekranie. W tej sytuacji konieczne jest ograniczenie w poleceniu *LIST* zakresu wyświetlania przez podanie numerów linii, które mają być wyświetlone. Na przykład

LIST 100—150

spowoduje wyświetlenie na ekranie linii o numerach z zakresu *100...150*. Przerwanie listowania programu jest zawsze możliwe przez naciśnięcie klawisza *BREAK*.

Podsumowanie

— wiemy, jak pisać program: jako ciąg numerowanych instrukcji, przy czym numerację prowadzimy sami albo prowadzi ją automatycznie mikrokomputer po otrzymaniu polecenia *AUTO*;
— wiemy, jak usuwać, zmieniać lub dodawać instrukcje programu: pisząc nowe z odpowiednią numeracją;
— umiemy program wylistować: poleceniem *LIST*, wiemy, jak ograniczyć zakres listowania i jak je przerwać: poleceniem *BREAK*;
— umiemy uruchomić program: poleceniem *RUN*;
— znamy dwie instrukcje języka *Basic*: *INPUT* i *PRINT*.

Pytania i zadania

1. Jakie elementy mogą być argumentami instrukcji *PRINT*? Jakimi znakami oddzielamy argumenty?
2. Napisz program, który odczyta wartość dwóch liczb wprowadzonych z klawiatury i wyświetli wartość sumy tych liczb.
3. Jakie elementy mogą być argumentami instrukcji *INPUT*? Jakimi znakami oddzielamy argumenty?
4. Opisz sposób wykonania programu

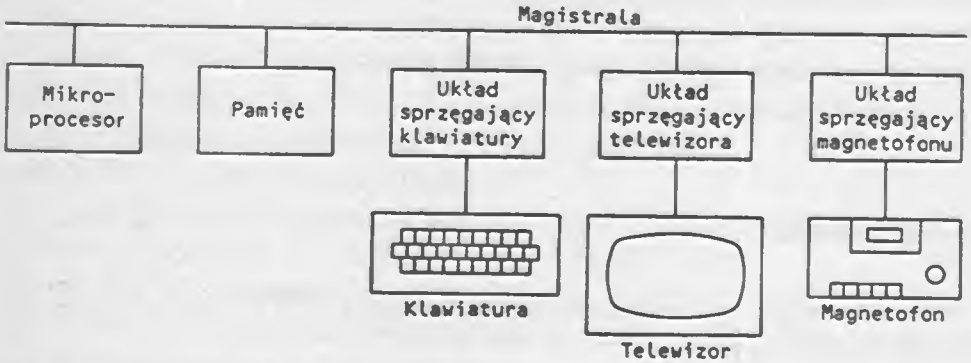
   10 INPUT "IMIE I WIEK;X$;Y

   20 PRINT X$;"MA";Y;" LAT"

5. W jaki sposób usunąć zbędne linie programu?
6. Do czego służą polecenia *AUTO* i *LIST*?
7. Czy można kilkakrotnie wykonać ten sam program? Jakiego polecenia trzeba użyć dla uruchomienia programu?

1.3 Przechowywanie programów

Centralnym elementem mikrokomputera (rys. 1.2) jest **mikroprocesor**. Jest to niewielkie urządzenie zdolne do wykonywania dowolnych ciągów instrukcji tworzących program jego działania. Zarówno program jak i wszystkie dane oraz otrzymane wyniki są przechowywane w **pamięci**, stanowiącej drugi istotny element mikrokomputera. W czasie pracy mikroprocesor nieustannie pobiera z pamięci instrukcje i dane potrzebne do ich wykonania, a następnie zapisuje w pamięci otrzymane wyniki. Pamięć wykorzystywana w taki sposób nosi nazwę **pamięci operacyjnej**. Pamięć operacyjna jest połączona z mikroprocesorem zespołem przewodów tworzących tzw. **magistralę** mikrokomputera. Magistrala stanowi główne łącze, po którym przepływa cała informacja magazynowana i przetwarzana wewnątrz komputera. Trzecim elementem mikrokomputera jest klawiatura. Jest ona dołączona do magistrali za pośrednictwem **układu sprzęgającego**. W podobny sposób (za pośrednictwem specjalnego układu sprzęgającego) jest dołączony do magistrali monitor lub telewizor.



Rys. 1.2. Podstawowe elementy mikrokomputera

Obszar pamięci operacyjnej wykorzystywany przez programy napisane w języku Basic jest podzielony funkcjonalnie na szereg pól. Z naszego punktu widzenia najistotniejsze są trzy:
— pole pamięci programu,
— pole pamięci zmiennych,
— pole pamięci ekranu.

Bezpośrednio po włączeniu zasilania wszystkie trzy pola pamięci są puste. Napisanie programu powoduje umieszczenie go w polu pamięci programu i zarezerwowanie w polu pamięci zmiennych miejsca na wszystkie zmienne występujące w tym programie. Polecenie *RUN* powoduje wykonanie dwóch czynności:

— wyzerowanie pola pamięci zmiennych,
— rozpoczęcie wykonywania instrukcji programu od początku.

Po zakończeniu wykonywania programu zawartość obydwu pól pamięci nie jest kasowana, lecz pozostaje w takim stanie, jaki powstał po wykonaniu ostatniej instrukcji.

Program jest zapisywany w pamięci wraz z numerami instrukcji. Dlatego możliwe jest również rozpoczęcie wykonywania programu od linii o określonym numerze. Numer ten podajemy jako argument w poleceniu *RUN*, np. polecenie

RUN 30

spowoduje wyzerowanie całego pola pamięci zmiennych i rozpoczęcie wykonywania programu od instrukcji oznaczonej numerem *30*. Podobny efekt osiągniemy pisząc polecenie

GOTO 30

(co znaczy *"idź do 30"*) z tą różnicą, że nie nastąpi tu wyzerowanie pola pamięci zmiennych. Jedyną wykonaną czynnością będzie przejście do wykonywania programu poczynając od instrukcji oznaczonej numerem *30*.

Rozpoczęcie wprowadzania nowego programu zmienia jedynie zawartość tych linii poprzedniego programu, których numery pokrywają się z numerami nowo wprowadzanych linii. Może to prowadzić do „zaśmiecenia" programu resztkami poprzednich programów. Aby tego uniknąć możemy postąpić dwojako:

albo zacząć numerację linii nowego programu od numeru wyższego, np. o 100 od ostatniego w starym programie,

albo wyzerować (opróżnić) pole pamięci programu poleceniem

NEW

(co znaczy *nowy*), które zeruje jednocześnie pole pamięci zmiennych.

Wyzerowanie pola pamięci zmiennych następuje również jako efekt wykonania polecenia

CLEAR

(co znaczy *wyczyść*), które nie zmienia jednak zawartości pola pamięci programu. W mikrokomputerze *Meritum* polecenie *CLEAR* wydziela dodatkowo w polu pamięci zmiennych miejsce przewidziane na przechowywanie zmiennych tekstowych. Standardowy przydział (*CLEAR* bez argumentu) wynosi 50 znaków. Wykonanie polecenia, np.

CLEAR 100

spowoduje zarezerwowanie miejsca wystarczającego do zapamiętania 100 znaków tekstu. Zmiany rezerwacji miejsca przeznaczonego na zmienne tekstowe dokonujemy zazwyczaj po komunikacie błędu

OS ERROR
sygnalizującym wyczerpanie dotychczas zarezerwowanego na ten cel miejsca.

Pole pamięci ekranu służy do przechowywania odpowiednio zakodowanej informacji określającej obraz aktualnie wyświetlany na ekranie telewizyjnym. Wyzerowanie pola pamięci ekranu poleceniem

CLS

powoduje wymazanie obrazu z ekranu.

Istotną cechą elementów elektronicznych, z których jest wykonana pamięć mikrokomputera jest tzw. ulotność. To znaczy **informacja jest przechowywana tylko przy włączonym napięciu zasilającym. Wyłączenie napięcia zasilającego powoduje natychmiastowe skasowanie całej zawartości pamięci.** Łatwo to sprawdzić wprowadzając do pamięci jakiś program, odłączając na chwilę zasilacz i po ponownym dołączeniu zasilania pisząc polecenie *RUN*. Żaden program nie zostanie wykonany. Również próba wykonania polecenia *LIST* wykaże brak jakiegokolwiek programu w pamięci. Ta właściwość pamięci mikrokomputera jest bardzo niedogodna w praktyce. Do niektórych programów chcielibyśmy mieć stały dostęp, a trudno przecież wpisywać je wciąż od nowa z klawiatury. Rozwiązaniem problemu jest wyposażenie mikrokomputera w dodatkową pamięć trwałą, która przechowuje informację również przy wyłączonym zasilaniu. Najtańszym rodzajem pamięci o tej właściwości są kasety magnetofonowe. Nagranie piosenki to nic innego jak trwałe zapamiętanie jej tekstu i melodii. W taki sam sposób można przechowywać w kasetach teksty programów i wartości danych. Dołączenie magnetofonu do mikrokomputera wymaga oczywiście specjalnego układu sprzęgającego. Mikrokomputer *Meritum*, podobnie jak większość innych komputerów domowych, jest standardowo wyposażony w układ sprzęgający przystosowany do współpracy z magnetofonem kasetowym. W przypadku mikrokomputera *Meritum* może to być dowolny magnetofon, choć pożądane jest, aby był wyposażony w licznik taśmy, który bardzo ułatwia odszukanie na taśmie potrzebnego programu.

Kasety magnetofonowe okazały się dogodnym nośnikiem informacji: są tanie, a jednocześnie bardzo pojemne. Stąd większość programów przenoszonych między komputerami takimi jak *Meritum* lub *Spectrum* jest przenoszona właśnie na kasetach. Niestety, na ogół przenoszenie programów możliwe jest tylko między komputerami tego samego typu.

Zauważmy, że jakkolwiek taśma magnetofonowa pełni rolę pamięci komputera, to jednak jest to pamięć innego rodzaju niż pamięć operacyjna. **Procesor nie może pobierać i wykonywać instrukcji wprost z magnetofonu.** Przed wykonaniem zapisanego na taśmie programu, program ten musi być przepisany do pamięci operacyjnej. Taśma magnetofonowa pełni rolę **pamięci zewnętrznej** komputera. Do przesyłania programu między pamięcią operacyjną a zewnętrzną służą specjalne polecenia *CSAVE* i *CLOAD*.

Polecenie *CSAVE* (od ang. *SAVE — zabezpiecz*) powoduje zapisanie całego programu na taśmie magnetofonowej i nadanie mu nazwy podanej w cu-

dzysłowie za słowem kluczowym *CSAVE*. **Przed naciśnięciem** klawisza *ENTER* kończącego polecenie należy włączyć magnetofon nastawiony na nagrywanie. Należy też zwrócić uwagę, aby nie próbować zapisu na początkowym odcinku rozbiegówki.

Polecenie *CLOAD* (od ang. *LOAD — ładuj*), którego argumentem jest podana w cudzysłowie nazwa programu, powoduje wyszukanie programu na taśmie, odczytanie go i zapisanie w pamięci operacyjnej. Magnetofon, nastawiony na odtwarzanie, należy włączyć dopiero **po naciśnięciu** klawisza *ENTER* kończącego polecenie. Inną rolę pełni polecenie *CLOAD?* wykonane po operacji *CSAVE*. Polecenie to również powoduje wyszukanie i odczytanie programu z taśmy, nie ładuje go jednak do pamięci, lecz porównuje z jej zawartością. Ewentualna niezgodność jest kwitowana komunikatem *BAD* (*źle*) sygnalizującym błąd w operacji zapisu.

Sprawdźmy działanie nowo poznanych poleceń. Mamy jeszcze w pamięci operacyjnej ostatni program z p. 1.2, a jeżeli nie, to wprowadźmy jeden z podanych tam programów. Przewijamy taśmę do początku, odwijamy rozbiegówkę, piszemy na klawiaturze polecenie

CSAVE "A"

włączamy magnetofon nastawiony na „zapis" i dopiero teraz naciskamy klawisz *ENTER*. Po zakończeniu operacji, co jest potwierdzane przez mikrokomputer komunikatem

READY

>–

przewijamy taśmę do początku, piszemy polecenie kontroli

CLOAD? "A"

naciskamy klawisz *ENTER* i włączamy magnetofon na „odczyt". W chwili wyszukania przez mikrokomputer programu na taśmie, w prawym górnym rogu ekranu pojawiają się dwie gwiazdki (**). W czasie trwania odczytu druga gwiazdka miga. Zasygnalizowanie przez mikrokomputer błędu (komunikat *BAD*) oznacza dla nas konieczność ponownego zapisania programu, najlepiej na innym odcinku taśmy. Zakończenie kontroli bez wykrycia błędu

READY

>–

upewnia nas o prawidłowym zapisaniu programu na taśmie. Możemy teraz spokojnie wyłączyć mikrokomputer wiedząc, że w każdej chwili możemy program załadować od nowa. W tym celu przewijamy taśmę, piszemy polecenie

CLOAD "A"

naciskamy *ENTER* i włączamy magnetofon nastawiony na odtwarzanie. Jeżeli na taśmie jest zapisanych kilka różnych programów, mikrokomputer rozpoczyna poszukiwanie programu o podanej w poleceniu nazwie. W chwili wykrycia początku jakiegoś programu na ekranie pojawiają się dwie gwiazdki (**). W razie niezgodności nazwy program jest pomijany, a na ekranie w miejscu pierwszej gwiazdki jest wyświetlana pierwsza litera jego nazwy. W

czasie ładowania programu druga gwiazdka miga. Prawidłowy odczyt programu kończy się komunikatem

READY

>-

Polecenie *RUN* uruchamia wykonanie programu. Nazwy używane w poleceniach *CSAVE*, *CLOAD?*, *CLOAD* mogą być co najwyżej sześcioznakowe, z tym jednak, że **mikrokomputer rozpoznaje tylko pierwszą literę**. Należy o tym pamiętać, gdyż inaczej może nastąpić pomylenie nagranych na taśmie programów.

Podsumowanie

— znamy podstawowe elementy mikrokomputera: mikroprocesor, pamięć operacyjną, urządzenia wejścia-wyjścia (klawiatura i telewizor) i pamięć zewnętrzną (magnetofon);
— znamy podział pamięci operacyjnej i umiemy posługiwać się poleceniami opróżniania różnych pól pamięci: *NEW*, *CLEAR*, *CLS* i uruchamiania programu: *RUN*, *GOTO*;
— wiemy, do czego służy magnetofon, i umiemy go obsługiwać: polecenia *CSAVE*, *CLOAD?*, *CLOAD*.

Pytania i zadania

1. Czym się różni pamięć operacyjna od pamięci zewnętrznej?
2. Wyjaśnij działanie poleceń *NEW*, *CLEAR* i *CLS*.
3. W czasie wykonania pewnego programu pojawił się komunikat
   OS ERROR
   jak usunąć zasygnalizowany błąd?
4. Czym różnią się polecenia: *RUN 50* i *GOTO 50*?
5. Napisz program z pytania 4 w p. 1.2, po czym:
   a) zapisz go na taśmie magnetofonowej,
   b) sprawdź poprawność zapisania,
   c) dopisz do programu jakąś instrukcję, np.
      15 PRINT "ZROZUMIALEM"
      i spróbuj teraz wykonać polecenie *CLOAD?*,
   d) załaduj program z kasety i wykonaj go.

2.6
Niskorozdzielcza grafika mikrokomputera Meritum

Podstawowe instrukcje służące do wprowadzania i wyprowadzania liczb i tekstów, tzn.: *PRINT*, *INPUT* i *READ* są zdefiniowane w standardzie języka *Basic* i działają identycznie we wszystkich komputerach. Oprócz tych instrukcji każdy komputer wykonuje zestaw dodatkowych, niestandardowych instrukcji wejścia-wyjścia związanych ze specyficznymi właściwościami wykorzystywanych urządzeń. Przykładem takich instrukcji mogą być instrukcje graficzne pozwalające na wyświetlanie rysunków na ekranie telewizora. Różnice między możliwościami różnych mikrokomputerów są w tym zakresie duże, ze względu na różną organizację wyświetlania obrazu.

Przy wyświetlaniu liczb i tekstów ekran monitora mikrokomputera *Meritum* jest podzielony na 16 linii po 64 znaków każda. Łączna liczba pozycji znakowych na ekranie wynosi więc 1024. Każda pozycja znakowa ma swój numer, przy czym pozycje w pierwszej linii są numerowane z lewa na prawo od 0 do 63, w drugiej linii od 64 do 127, itd., aż do ostatniej linii, w której pozycje są numerowane od 960 do 1023. Jeżeli umówimy się numerować linie od 0 do 15 i kolumny od 0 do 63, to numer pozycji znakowej w *I*-tej linii i *J*-tej kolumnie jest dany wzorem

$P = 64 * I + J$

Numery pozycji mogą być wykorzystane do wskazania miejsca wyświetlania danego znaku na ekranie monitora. Służy do tego instrukcja *PRINT* z funkcją @, o postaci

PRINT @ pozycja, lista elementów

w której *lista elementów* może być zapisana w taki sam sposób jak w zwykłej instrukcji *PRINT*, a wartość wyrażenia *pozycja* określa miejsce na ekranie, w którym zostanie wyświetlony pierwszy znak, czyli od której rozpocznie się wyświetlanie.

Na przykład program

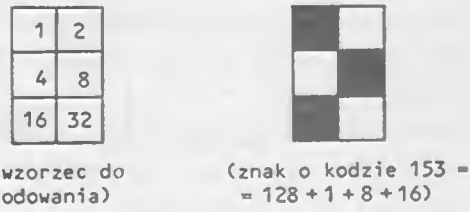
```
10 CLS
20 PRINT @ 350, "A"
30 PRINT @ 413, "A   A"
40 PRINT @ 476, "A       A"
50 PRINT @ 540, "AAAAA"
60 PRINT @ 604, "A       A"
70 PRINT @ 668, "A       A"
80 GOTO 80
```

spowoduje wyświetlenie pośrodku pustego ekranu wielkiej litery *A* zbudowanej z liter *A* normalnego rozmiaru.

Przy redagowaniu informacji wyświetlanej na ekranie użyteczna jest też

funkcja *TAB* przesuwająca w obrębie jednej linii miejsce wyświetlania znaku. Argumentem funkcji *TAB* jest numer kolumny wskazujący miejsce następnego wyświetlanego znaku. W poprzednim programie linia 70 powodowała wyświetlenie pierwszej litery *A* w kolumnie 28, a drugiej w kolumnie 32. Działanie programu nie zmieni się, jeżeli zastąpimy poprzednią linię 70 nową

70 PRINT @ 668, "A"; TAB (32) "A"



Rys. 2.19. Znak graficzny mikrokomputera *Meritum*

W instrukcji *PRINT*, oprócz typowych znaków pisarskich, mogą być też używane specjalne znaki graficzne (które mogą również być włączane do zmiennych tekstowych), pozwalające na sporządzanie prostych rysunków. Każdy **znak graficzny** zajmuje na ekranie jedną pozycję znakową. Jest on zbudowany z sześciu punktów (rys. 2.19), z których każdy może być jasny (niewidoczny na tle ekranu) lub ciemny. 64 znaki graficzne o kodach z zakresu 128...191 odpowiadają wszystkim możliwym kombinacjom punktów jasnych i ciemnych. Kod znaku może być obliczony na podstawie wzorca kodowania pokazanego na rys. 2.19. Również typowe znaki pisarskie mają swoje kody, przy czym mikrokomputer *Meritum* wykorzystuje znormalizowany system kodowania określony przez międzynarodowy standard *ASCII*. Zgodnie z tym standardem kody 0...31 odpowiadają rozmaitym znakom sterującym sposobem wyświetlania, kody 32...127 odpowiadają znakom widocznym na ekranie (*Meritum* nie wykorzystuje wszystkich kodów), a kody powyżej 127 mogą być wykorzystane dowolnie. Aby wykorzystać kod znaku, np. w instrukcji *PRINT*, należy najpierw zamienić go na jednoznakowy tekst, zawierający znak odpowiadający temu kodowi. Zadanie to wykonuje **funkcja *CHR$***. Na przykład program

```
10 CLS: PRINT : PRINT : PRINT
20 FOR I = 0 TO 9
30 FOR J = 0 TO 15
40 PRINT CHR$ (16 * I + J + 32);
50 NEXT J
60 PRINT
70 NEXT I
```

spowoduje wyświetlenie w 10 liniach i 16 kolumnach wszystkich znaków odpowiadających kodom 32...191.

Traktując punkt (znaku graficznego) jako najmniejszy element ekranu możemy uważać, że cała powierzchnia ekranu jest podzielona na 6144 punkty ułożone w 48 wierszach (numerowanych z góry na dół od 0 do 47) i 128 kolu-

mnach (numerowanych od 0 do 127). Stan każdego punktu może być ustalony albo przez wykonanie instrukcji *PRINT* wyświetlającej znaki graficzne, albo indywidualnie przy użyciu jednej z dwóch **instrukcji zmiany stanu punktu**

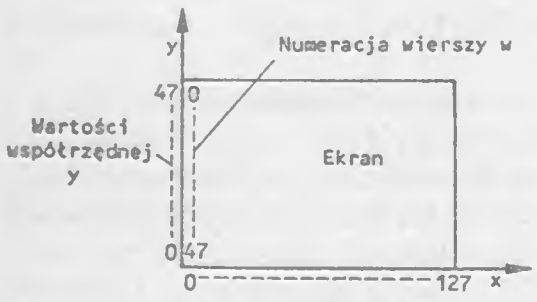
SET (w,x) — zapal

RESET (w,x) — zgaś

Parametrami obydwu instrukcji są współrzędne (*w* — numer wiersza, *x* — numer kolumny) zapalanego lub gaszonego punktu ekranu. Stan każdego punktu możemy też sprawdzić przy użyciu **funkcji badania stanu**

POINT (w, x)

której wartością jest 0, jeżeli punkt jest „zgaszony" (jasny), lub 1, jeżeli punkt jest „zapalony" (ciemny).



Rys. 2.20. Ekran mikrokomputera *Meritum* jako obraz pierwszej ćwiartki układu współrzędnych *x0y*

Zaciemniając odpowiednie punkty możemy rysować na ekranie nieskomplikowane obrazki. W większości przypadków będziemy traktować ekran jako obraz pierwszej ćwiartki kartezjańskiego układu współrzędnych *x0y*. Pewną niewygodę stanowi sposób adresacji punktów ekranu: w pionie numeracja punktów wzrasta z góry na dół, przeciwnie niż wartość współrzędnej *y* układu (rys. 2.20). Przy kreśleniu rysunków musimy pamiętać o konieczności przeliczania współrzędnej *y* rysowanych punktów na numer wiersza *w*

$$w = 47 - y$$

stanowiący argument instrukcji *SET* i *RESET*. Na przykład, korzystając z równania prostej

$$y = kx$$

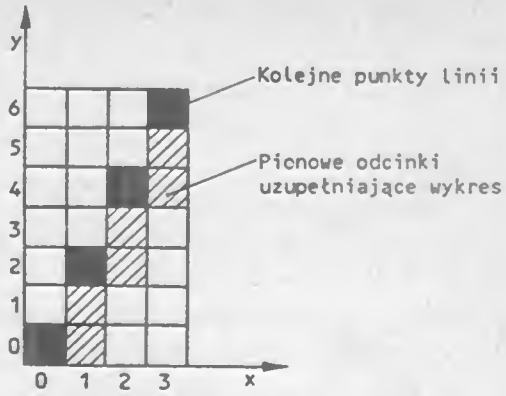
możemy napisać program rysujący odcinek o zadanym nachyleniu *k*.

```
10 INPUT "NACHYLENIE =" ; K
20 CLS
30 FOR X = 0 TO 127
40 Y = INT (K * X)
50 IF Y>47 THEN GOTO 10
60 SET (X, 47 - Y)
70 NEXT X
80 GOTO 10
```

Program działa w miarę dobrze tylko dla wartości współczynnika *k* nie większych niż 1. Dla wartości większych z odcinka zostaje zaledwie kilka punktów. Przyczyną jest duży skok zmiennej *y*, który dla uzyskania ciągłości linii należy wypełnić punktami (rys. 2.21), zmieniając nasz program na przykład do postaci

```
10 INPUT "NACHYLENIE="; K
```

```
20 CLS: Y1 = 0
30 FOR X = 0 TO 127
40 Y2 = INT (K * X)
45 FOR Y = Y1 TO Y2
50 IF Y>47 THEN GOTO 10
55 SET (X, 47 - Y)
60 NEXT Y
65 Y1 = Y2
70 NEXT X
80 GOTO 10
```



Rys. 2.21. Wykres prostej $y = 2x$ na siatce punktów

Przy okazji nowa wersja łagodzi nieco „schody" widoczne przy wykreślaniu odcinków ukośnych. Zauważmy jeszcze jedną pułapkę: wartości k równej 1 powinno odpowiadać nachylenie odcinka do poziomu równe 45°, tymczasem na ekranie jest ono dużo większe. Wynika to z faktu, że ekran monitora nie jest kwadratowy ani w sensie długości boków (3:4), ani w sensie liczby punktów w pionie i w poziomie (48:128). Poprawa efektu wymaga skorygowania obliczonej wartości współrzędnej y, zgodnie ze współczynnikiem

$$\frac{4}{3} \cdot \frac{48}{128} = \frac{1}{2}$$

To znaczy, w poprzednim programie musimy zastąpić instrukcję z linii 40 instrukcją

```
40 Y2 = INT (K * X/2)
```

Pomimo wszystkich poprawek rezultat nie jest rewelacyjny, a co gorsza, jest niemożliwy do poprawienia. Schodkowy kształt odcinka wynika po prostu ze zbyt małej rozdzielczości (zbyt małej liczby elementarnych punktów) obrazu w mikrokomputerze *Meritum*.

Podsumowanie

— Użycie funkcji @ i *TAB* w instrukcji *PRINT* pozwala na łatwe redagowanie wyświetlanych na ekranie wyników;
— znaki graficzne (o kodach 128...191) oraz instrukcje: *SET*, *RESET*, *POINT*, których argumentami są współrzędne punktów ekranu, pozwalają na sporządzanie prostych rysunków;
— możliwości grafiki niskorozdzielczej nie są duże, ze względu na zbyt małą liczbę punktów.