Wskazówka. Użyj w programie funkcji NWDZIELNIK. Możesz porównać swój program z programem zapisanym w pliku TPULAMKI.PAS.

10.11. Procedury graficzne

10.11.1. Wprowadzenie

Jeżeli w Twojej szkolnej pracowni komputerowej jest używany arkusz kalkulacyjny, to zapewne wygodniej jest korzystać z możliwości, które on oferuje. Samodzielne pisanie programów służących do tworzenia wykresów nie jest trudne, ale jeżeli wykres taki miałby być opisany i wyskalowany, to wszystko razem mogłoby okazać się dość pracochłonne. Niekiedy jednak wygodniej jest skorzystać z możliwości graficznych oferowanych przez język programowania, w szczególności wówczas, gdy na ekranie miałby być odtworzony ruch. Pełny opis możliwości graficznych systemu Turbo Pascal jest dość obszerny. Tu przedstawimy jedynie możliwości przykładowe.

Na ekran pracujący w trybie graficznym trzeba spojrzeć jako na swojego rodzaju prostokątną tablicę, której elementami są punkty (piksele). Liczba elementów zależy od użytej karty graficznej. Zazwyczaj w poziomie jest nie mniej niż 640 punktów, a w pionie nie mniej niż 348, z wyjątkiem komputerów wyposażonych w kartę CGA, która daje możliwość użycia tylko 200 punktów w pionie. Poszczególne punkty są numerowane od zera, a więc największy numer jest o jeden mniejszy od liczby punktów w danym kierunku.

Jeżeli Twój komputer zawiera kolorowy monitor, to każdemu punktowi możesz nadać kolor; jeżeli monochromatyczny, to możesz nadać stopień szarości. Nie zawsze masz jednak tu taką dowolność jak w przypadku tekstów, możesz bowiem dysponować np. tylko dwoma lub tylko czterema kolorami jednocześnie. Co więcej, przy większej liczbie kolorów możesz dysponować mniejszą liczbą punktów.

Funkcje graficzne są zawarte w module GRAPH. Procedura DETECTGRAPH rozpoznaje rodzaj zainstalowanej karty graficznej (sterownika) i tryb pracy karty.

```
detectgraph(sterownik, tryb);
```

Karcie VGA odpowiada sterownik nr 9, EGA — 3 (EGA64 — 4), CGA — 1, a karcie Hercules — 7. Interpretacja trybu zależy od karty, np. tryb nr 2 karty VGA oznacza użycie 16 kolorów przy rozdzielczości 640 na 480 punktów.

Tryb graficzny jest inicjowany wywołaniem procedury INITGRAPH. Ma ona trzy parametry: sterownik i tryb (oba typu *integer*) oraz parametr będący łańcuchem, wskazujący katalog dyskowy ze sterownikami ekranu

(pliki te można rozpoznać po rozszerzeniu BGI); w podanym przykładzie jest to podkatalog BGI w katalogu TP na dysku D.

Liczbę punktów dostępnych wzdłuż poszczególnych osi można odczytać za pomocą funkcji GETMAXX i GETMAXY. Pracę w trybie graficznym kończy się instrukcją wywołania procedury CLOSEGRAPH.

Do rozpoznania parametrów swojego komputera możesz posłużyć się programem TP-31.

Program TP-31

```
program TP31;
  {rozpoznaje sterownik graficzny, tryb pracy karty
   oraz rozdzielczosc ekranu}
Uses Graph;
var sterownik, tryb, x, y: integer;
begin
  DetectGraph(sterownik, tryb);
  InitGraph(sterownik, tryb, 'D:\TP\BGI');
      {tu podaj sciezke do pliku sterownika *.bgi w Twoim
                                                komputerze}
  x:=GetMaxX;
  y:=GetMaxY;
  CloseGraph;
  WriteLn('Sterownik graficzny ', sterownik);
  WriteLn('Tryb graficzny      ', tryb);
  WriteLn('MaxX ',x,'          MaxY ',y);
  ReadLn
end.
```

Zestaw kolorów zależy od stosowanego sterownika graficznego i trybu. Przyporządkowanie kolorów liczbom może być zmieniane przez użytkownika.

Liczba dostępnych kolorów zależy od trybu pracy. W standardowych trybach dla kart VGA i EGA masz do dyspozycji 16 kolorów oznaczanych liczbami od 0 do 15 (15 odpowiada kolorowi białemu — *white*).

Ćwiczenie 10-34

Napisz procedurę inicjującą pracę w trybie graficznym.

10.11.2. Podstawowe operacje graficzne

System Turbo Pascal umożliwia Ci nadanie barwy pojedynczemu pikselowi, poprowadzenie linii między dwoma wskazanymi pikselami, zakreślenie okręgu o podanym promieniu ze wskazanego punktu itp. Wszystkie te operacje wymagają od Ciebie świadomego wskazania konkretnego punktu. Położenie

punktu określasz przez podanie jego współrzędnych, oznaczanych jak w matematyce jako para liczb (x,y). Współrzędne x i y są liczbami całkowitymi. Zakres każdej ze współrzędnych zaczyna się od zera, a kończy na liczbie o jeden mniejszej od liczby punktów (na przykład dla karty VGA na 639 i 479).

Uwaga. Współrzędna pozioma rośnie w prawo, a pionowa w dół (a więc odwrotnie niż jesteś przyzwyczajony!). Zapamiętaj zatem, że punkt o współrzędnych (0,0) znajduje się w lewym górnym(!) rogu ekranu.

Przy obliczaniu współrzędnych punktu za pomocą wyrażeń matematycznych należy zwrócić uwagę, żeby wynik był liczbą całkowitą; może to wymagać użycia funkcji TRUNC.

Do nadania punktowi określonego koloru służy procedura PUTPIXEL (umieść piksel). Ma ona trzy parametry: dwie współrzędne punktu i kolor.

```
  PutPixel(0,0,1);
  PutPixel(600,400,7);
end.
```

Ćwiczenie 10-35

A. Napisz program zawierający wywołanie procedury przejścia do trybu graficznego, instrukcję „zapalenia" na ekranie jednego punktu, na przykład o współrzędnych (150,150), oraz instrukcję powodującą pozostanie w trybie graficznym do czasu naciśnięcia klawisza przez użytkownika. Zastanów się, w jakim miejscu ekranu punkt powinien się znaleźć.

B. Uruchom swój program i sprawdź, czy położenie punktu jest zgodne z oczekiwaniami. W razie problemów porównaj go z programem TP-32.

Program TP-32

```
program TP32;
Uses Graph, Crt;
var sterownik, tryb, x, y: integer;
begin
  DetectGraph(sterownik, tryb);
  InitGraph(sterownik, tryb, 'D:\TP\BGI');
      {tu podaj swoja ścieżkę do pliku sterownika *.bgi}
  PutPixel(150,150,7);
  Repeat until KeyPressed;
  CloseGraph
end.
```

C. Zmień współrzędne punktu, tak abyś zorientował się, w jakim kierunku przesuwa się punkt przy ich zwiększaniu.
D. Nadaj kilku punktom różne wartości koloru i sprawdź, czy faktycznie mają taki kolor, jakiego się spodziewałeś.
E. Jeżeli lubisz nieoczekiwane efekty, to napisz program, w którym współrzędne punktu oraz kolor byłyby losowane i program działałby w pętli, „zapalając" nowe punkty do chwili naciśnięcia klawisza przez użytkownika.

10.11.3. Linie, okręgi i inne figury

System Turbo Pascal zawiera instrukcje rysowania niektórych figur: linii prostych, okręgów, prostokątów.

Do wykreślenia odcinka linii prostej służy procedura LINE (linia). Ma ona cztery parametry typu całkowitego (x1,y1,x2,y2), przy czym (x1,y1) są współrzędnymi jednego końca linii, a (x2,y2) — drugiego jej końca.

Procedura RECTANGLE (prostokąt) służy do rysowania prostokątów. Ma ona cztery parametry typu całkowitego (x1,y1,x2,y2), przy czym (x1,y1) są współrzędnymi lewego górnego rogu prostokąta, a (x2,y2) — prawego dolnego.

Procedura CIRCLE (okrąg) służy do rysowania okręgów. Ma ona trzy parametry (x,y,r), przy czym dwa pierwsze (całkowite) wyznaczają środek okręgu, a trzeci *r* (typu *word*) — jego promień mierzony w pikselach.

Ćwiczenie 10-36

Napisz i uruchom program rysujący:

A. Trójkąt o wierzchołkach: w środku ekranu, w prawym górnym rogu i w prawym dolnym.
B. Prostokąt o wierzchołkach: (10, 10), (410, 10), (410, 210) i (10, 210).
C. Kwadrat o boku 70 i środku w punkcie (200, 120).
D. Koło wpisane w kwadrat określony w punkcie C.
E. Koło opisane na kwadracie określonym w punkcie C.
F. Układ współrzędnych łącznie z kreseczkami na obu osiach: krótszymi co 20 jednostek i dłuższymi co 100 jednostek.
G. Linię prostą tak dobraną (eksperymentalnie), żeby przechodziła blisko czterech punktów: (100, 180), (50, 190), (200, 140), (300, 120).

W systemie Turbo Pascal są też dostępne procedury do rysowania wielokątów, łuków, słupków (takich jak na wykresach arkuszy kalkulacyjnych) i wykresów kołowych.

Linia może być ciągła, przerywana lub kropkowana; oprócz linii normalnej grubości są też linie pogrubione. Kolor linii określa się oddzielną instrukcją. Fragmenty powstałych figur objęte liniami zamkniętymi mogą być wypełniane wzorami i kolorami.

Można umieszczać na wykresach opisy tekstowe, posługując się przy tym różnymi krojami pisma, łącznie z krojami ozdobnymi. Pełny opis możliwości graficznych jest o wiele obszerniejszy od przedstawionego. Jeżeli zamierzałbyś wykorzystywać je w swoich programach, to musiałbyś zapoznać się z nimi w podręczniku użytkownika systemu Turbo Pascal. Przykłady użycia znajdziesz ponadto w pomocy (*help*).

10.11.4. Wykres funkcji

Sporządzimy teraz wykres funkcji, np. jednego okresu znanej Ci funkcji trygonometrycznej *sinus*.

Pierwsza decyzja dotyczy **zakresu wartości** argumentu funkcji (x) i jej wartości (y), jakie mają być odwzorowane na wykresie. Takie same decyzje podejmujesz podczas tworzenia wykresu na papierze. Decydujesz wówczas również, jak duży ma być wykres; czy ma zajmować całą kartkę czy jej część, a dokładniej, jaka ma być skala wykresu. Podobną decyzję musisz podjąć, sporządzając wykres na ekranie. **Skalę wykresu** wyrażasz jednak nie w odniesieniu do centymetrów, lecz do punktów ekranu.

Zastanów się chwilę nad zakresem i skalą wykresu funkcji sinus. Jeden okres funkcji odpowiada zmianie jej argumentu od 0 do 2π, czyli do ok. 6,28. Jeżeli posłużysz się podstawowym trybem karty VGA, to współrzędne poziome punktów ekranu będą się zmieniać od 0 do 639. Stosunek obu liczb jest nieco większy od 100, można zatem do odwzorowania funkcji sinus posłużyć się taką skalą dla zmiennej niezależnej.

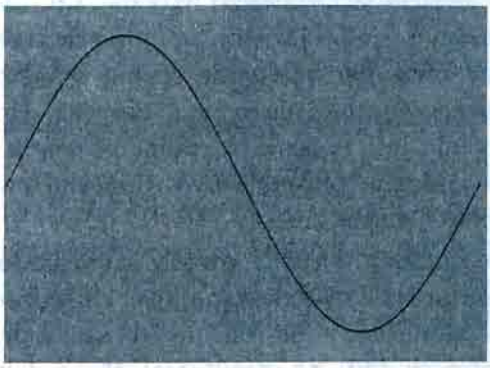
Argumentem funkcji sinus będzie współrzędna pozioma punktu ekranu podzielona przez 100; argumentowi 2π będzie odpowiadać punkt o współrzędnej 628 ($314/100 = 6,28$).

Wartości funkcji sinus zmieniają się od -1 do 1, a więc o 2. Zmienna pionowa wykresu zmienia się od 0 do 479. Jeżeli chcesz wykorzystać dobrze powierzchnię ekranu, to powinieneś przyjąć skalę bliską 200 (nieco mniejszą od 240). Oś pozioma wykresu powinna być pośrodku ekranu, na poziomie 240. W tym celu trzeba dodać liczbę 240 do wartości funkcji. Należy jeszcze pamiętać, że współrzędne punktów ekranu rosną z góry do dołu, i zmieniać znak współczynnika, przez który mnożyć się będzie wartość funkcji.

Otrzymany wynik działań arytmetycznych dla każdej współrzędnej wykresu należy poddać działaniu funkcji TRUNC, żeby otrzymana liczba rzeczywista została zamieniona na odpowiadającą jej liczbę całkowitą.

Ćwiczenie 10-37

Napisz program sporządzający wykres funkcji sinus zgodnie z podanymi wskazówkami i uruchom go. Czy tworzony wykres jest podobny do przedstawionego na rys. 10.1?



Rys. 10.1. Wykres sporządzony za pomocą programu TP-33

Jeżeli dostrzegasz wyraźne różnice w kształcie wykresu (lub masz trudności z uruchomieniem programu), to przeanalizuj działanie swojego programu i ewentualnie porównaj go z programem TP-33 (zapisanym w pliku TP33.PAS).

Program TP-33

```
program TP33;
  {Wykres funkcji sinus}
uses Graph, Crt;
var i: integer;

procedure InicjacjaGraficzna;
var sterownik, tryb: integer;
begin
  DetectGraph(sterownik, tryb);
  InitGraph(sterownik, tryb, 'D:\TP\BGI')
      {tu podaj sciezke do pliku sterownika *.bgi w Twoim
                                                komputerze}
end;

begin
  InicjacjaGraficzna;
  for i:=0 to 628 do
    PutPixel(i, Trunc (240 - 200 * sin(i/100)),7);
```

```
  repeat until KeyPressed;
  CloseGraph
end.
```

Po takim ćwiczeniu nie powinieneś mieć większych trudności ze sporządzeniem trudniejszych wykresów.

Jeżeli punktem wyjścia jest zjawisko fizyczne, np. ruch punktu umieszczonego na kole toczącym się po prostej, to najpierw trzeba sporządzić opis matematyczny, żeby otrzymać równania ruchu, i następnie dobrać skalę odwzorowania poszczególnych zmiennych. Zauważ, że trajektoria ruchu punktu po płaszczyźnie jest opisana za pomocą trzech zmiennych: x i y oraz *czas*. Trzecia zmienna nie musi być widoczna na ekranie, niemniej i dla niej musisz dobrać zakres zmian.

Typowe szkolne zadanie o ruchu dwóch samochodów wyjeżdżających z miast A i B można przedstawić następująco. Odległość między miastami odwzorujemy na osi pionowej; jeżeli jest nią 300 km, a przeznaczymy na nią 300 pikseli (z dostępnych 480), to otrzymamy skalę: 1 km na piksel. Czas odwzorujemy za pomocą zmiennej t w proporcji: 200 na godzinę. Na osi poziomej użyjemy takiej samej proporcji: 200 pikseli na godzinę; przy 640 pikselach możemy odwzorować ponad trzy godziny.

Torem ruchu samochodu poruszającego się ruchem jednostajnym jest na naszym wykresie prosta. Prędkości 80 km/h jednego samochodu odpowiada zmiana współrzędnej y wykresu o 80 przy zmianie x o 200, co daje współczynnik 0.4; podobnie prędkości 60 km/h drugiego samochodu odpowiada współczynnik 0.3. Samochód wolniejszy wyjeżdża później o pół godziny, musimy zatem odpowiednio skorygować czas (o 100).

Program TP-34 tworzący wykres ruchu (zapisany w pliku TP34.PAS) zawiera, oprócz instrukcji wykreślenia osi wykresu, następujący fragment odpowiedzialny za przedstawienie trajektorii ruchu samochodów:

Program TP-34

```
{fragment programu TP34 -  rysowanie wykresu ruchu
                           dwoch samochodow}
 for t := 0 to 600 do
   begin
     PutPixel (10 + t, Trunc(400 - 0.4 * t),7);
     if t >= 100 then PutPixel (10 + t, Trunc(100 + 0.3 *
                                              (t - 100)),7)
   end
```

Zmienna t zmienia się od 0 do 600 (trzy godziny). Zwiększeniu zmiennej t o jednostkę odpowiada zwiększenie współrzędnej poziomej obu zapalanych

punktów także o jednostkę. Dodana liczba 10 powoduje odsunięcie wykresu od lewej krawędzi ekranu, co umożliwia m.in. ładniejsze przedstawienie układu współrzędnych.

Przy obliczaniu współrzędnych pionowych wykresu są użyte wyliczone współczynniki 0.4 i 0.3. Współrzędne te są zmieniane w zakresie od 100 do 400. Wyruszenie drugiego samochodu później o pół godziny jest przedstawione za pomocą przesunięcia argumentu t o 100 zarówno we wzorze na współrzędną pionową, jak i w warunku na rysowanie trajektorii ($t > 100$).

Komentarz do wykresów przedstawionych na rys. 8.5.

Jeżeli chciałbyś sporządzić wykresy podobne do przedstawionych na rys. 8.5, to przyda Ci się krótki komentarz. Ruch punktu umieszczonego na toczącym się kole jest złożeniem dwóch ruchów: prostoliniowego ruchu osi koła i obrotowego ruchu punktu względem osi koła. Oba ruchy są ze sobą powiązane w taki sposób, że przebyciu w ruchu prostoliniowym dystansu równego obwodowi koła odpowiada jeden jego obrót (wówczas kątowi αt odpowiada dystans $R\alpha t$, gdzie R jest promieniem koła. Równania ruchu oddalonego od środka koła o r można zapisać następująco:

$$x = R\alpha t + r sin\alpha t$$
$$y = R + r cos\alpha t$$

Program rysujący trajektorię dla przykładowych wartości R i r jest zapisany w pliku TPKOLO.PAS.

Ćwiczenie 10-38

A. Wykonaj program rysujący ruch ciała rzuconego rzutem poziomym z wysokości h z daną prędkością początkową, i ruch drugiego ciała poruszającego się rzutem ukośnym z innego miejsca i rozpoczynającego ruch nieco później. Dobierz eksperymentalnie kąt, pod jakim drugie ciało rozpoczyna swój ruch, tak żeby oba ciała spotkały się (tzn. żeby w tej samej chwili były dostatecznie blisko siebie). Punkt spotkania zaznacz na wykresie. (Porównaj swój program z programem zapisanym w pliku TPRZUT.PAS).

B. Wykonaj program rysujący ruch punktu znajdującego się na kole toczącym się po kole. (Porównaj swój program z programem zapisanym w pliku TPDWAKOL.PAS.)

10.11.5. Ruchomy obiekt

Wykresy niektórych trajektorii były sporządzane zapewne w takim tempie, że mogłeś obserwować ruch ciała (zależy to od szybkości działania Twojego komputera). Ogólnie biorąc, możesz imitować na ekranie zmiany zachodzące

w czasie w niektórych procesach fizycznych, chemicznych lub innych. W razie potrzeby możesz dodawać instrukcje mające spowolnić wykonywanie programu. Możesz także „gasić" punkt szybko po jego zapaleniu.

Możesz także nie określać z góry warunku zakończenia obliczeń, jak w pokazywanych do tej pory przykładach, lecz założyć, że przerwanie pętli nastąpi po naciśnięciu klawisza przez użytkownika.

Dla przykładu program przedstawiający ruch punktu opisany zależnościami:

$$x = 2^{-0.05t} sin(\omega t)$$
$$y = 2^{-0.05t} cos(2\omega t)$$

gdzie ω jest prędkością kątową, może mieć następującą postać

Program TP-35

```
program TP35;
  {Wykres ruchu punktu}
uses Graph, Crt;
var t, Amplituda: real;
    x, y: integer;
procedure InicjacjaGraficzna;
var sterownik, tryb: integer;
begin
  DetectGraph(sterownik, tryb);
  InitGraph(sterownik, tryb, 'D:\TP\BGI')
      {tu podaj sciezke do pliku sterownika *.bgi w Twoim
                                                komputerze}
end;

function Potega(x,y:real):real;
begin
  potega := EXP(y * LN(x))
end;

begin
  InicjacjaGraficzna;
  t:=0.0;
  repeat
    Amplituda := Potega(2, -0.05*t);
    x := Trunc(320 + 200 * Amplituda * Cos(2 * t));
    y := Trunc(240 - 200 * Amplituda * Sin(2 * t));
    PutPixel (x,y,7);
    Delay(5);
    t := t + 0.01
  until KeyPressed;
```

```
  CloseGraph
end.
```

Program będzie działał dopóty, dopóki nie naciśniesz jakiegoś klawisza. Skala wykresu została dobrana następująco. Wartość funkcji mieści się w zakresie od -1 do 1. Środek tego obszaru został przesunięty do środka ekranu (320, 240), a wartości zmiennych są mnożone przez 200.

Uwaga. Szybkość sporządzania wykresu zależy od szybkości działania Twojego komputera (typu procesora, wyposażenia w koprocesor i inne). Jeżeli nie jest bardzo szybki, to możesz zobaczyć, jak wykres funkcji sporządzany jest na Twoich oczach. Jeżeli komputer jest szybki, a chcesz zobaczyć ruch, możesz sztucznie spowolnić tworzenie wykresu np. przez wywoływanie w pętli procedury DELAY, podobnie jak w programie TP-35.

Ćwiczenie 10-39

A. Sprawdź wpływ procedury DELAY na działanie programu. Możesz ją usunąć lub zmienić jej parametr.
B. Zmodyfikuj program TP-35 mnożąc argumenty funkcji SIN i COS przez różne liczby całkowite (np. argument funkcji sinus przez 2, a cosinus przez 3). Możesz ewentualnie zmienić ponadto SIN na COS lub odwrotnie.
C. Zmodyfikuj program TP-35 w taki sposób, żeby przedstawiał równania znanych Ci funkcji (np. funkcji wymienionych przy omawianiu zadania 8-7 w rozdz. 8).

10.12. Odczytywanie danych i wyprowadzanie wyników

10.12.1. Otwieranie i zamykanie pliku

System Turbo Pascal ma wiele różnych procedur służących do zapisywania danych do różnego typu plików. Ograniczymy się tu do jednej instrukcji: WRITE (WRITELN). W razie potrzeby wiadomości te możesz uzupełnić samemu.

Plik przed dokonaniem zapisu lub odczytu musi zostać **otwarty**. Operacja otwarcia pliku w systemie Turbo Pascal musi być poprzedzona skojarzeniem zmiennej typu plikowego z fizycznym plikiem danych. Użycie zmiennej plikowej wymaga odpowiedniej deklaracji.

Do skojarzenia pliku z fizycznym zbiorem danych służy procedura ASSIGN. Ma ona dwa argumenty: identyfikator pliku i wyrażenie łańcuchowe.

A oto przykład jej użycia:

```
var SymbolPliku: text;
...
Assign(SymbolPliku,'dane1.dat')
```

W instrukcjach programu odnoszących się do tego pliku występuje od tej pory identyfikator pliku, a nie jego nazwa. Jak pamiętasz typ *text* odnosi się do pliku tekstowego. Zamiast nazwy pliku dyskowego można w procedurze ASSIGN wstawić symbol urządzenia, taki jak stosowany w poleceniach DOS (na przykład PRN — drukarka).

Uwaga. Nazwa pliku w wywołaniu procedury ASSIGN może być określona zmienną łańcuchową.

Do otwierania pliku nowego, a także pliku, który ma być zapisany od nowa, służy procedura REWRITE; jej argumentem jest identyfikator pliku. Instrukcja jej wywołania w odniesieniu do pliku DANE1.DAT po przypisaniu temu plikowi podanego wyżej identyfikatora ma postać REWRITE(*SymbolPliku*). Do otwierania pliku skojarzonego ze zbiorem już istniejącym służą procedury RESET i APPEND. Ta pierwsza umożliwia odczytanie pliku od początku, ta druga — dopisanie nowych danych na końcu pliku tekstowego. Są wywoływane podobnie jak REWRITE, tzn. ich argumentem jest przypisana plikowi zmienna typu plikowego (*SymbolPliku*).

Do zamykania plików służy procedura CLOSE, np. CLOSE(*SymbolPliku*). Wprawdzie system zamyka pliki kończąc wykonywanie programu, niemniej zaleca się, żeby plik zamykać instrukcją CLOSE, kiedy tylko zapisywanie danych zostało zakończone (bez czekania na koniec programu), zmniejsza to bowiem niebezpieczeństwo przypadkowej utraty danych.

Po zamknięciu pliku „zwolnioną" zmienną plikową *SymbolPliku* można skojarzyć z innym fizycznym zbiorem danych, ale zaleca się nie czynić tego, ze względu na mniejszą czytelność programu i większą szansę popełnienia błędu. Przyjmij zatem zasadę stosowania różnych nazw dla różnych plików.

10.12.2. Wyprowadzanie wyników do pliku

Procedury WRITE i WRITELN umożliwiają zapisywanie wyników bezpośrednio do plików dyskowych określonego typu, przy czym WRITELN jest stosowana tylko przy zapisie do plików tekstowych. Obie procedury są wywoływane tak samo jak przy drukowaniu na ekranie, z jedną tylko różnicą, że po otwarciu nawiasu przeznaczonego na zapisanie parametrów, jako pierwszy parametr podaje się identyfikator pliku; jeżeli są dalsze parametry, to stawia się za nim przecinek.

Możesz przerobić dowolny ze swoich programów w taki sposób, żeby wyniki zapisywał w pliku. Przed pierwszą z instrukcji drukowania wyników musisz umieścić instrukcję otwarcia pliku do zapisu, a za ostatnią — instrukcję zamknięcia go.

Zapis do pliku tekstowego

Zaczniemy od zapisu do pliku tekstowego, przerabiając jeden z poznanych przez Ciebie programów.

Ćwiczenie 10-40

A. Zmodyfikuj program TP-5 w taki sposób, żeby komunikat pokazywany na ekranie zapisywał się ponadto w pliku tekstowym TP05.WYN. Czy Twój program jest podobny do programu TP-36?

Program TP-36

```
program TP36;
  {Obliczanie obwodu kola
     - zapis do pliku (zmodyfikowany program TP-5)}
const
   DwaPi = 2 * Pi;
var
   promien:   real;
   SymbolPliku: text;
begin
  Write ('Podaj promien kola ');
  ReadLn (promien);
  Assign(SymbolPliku,'tp05.wyn');
  Rewrite(SymbolPliku);
  Write ('Obwod kola o promieniu ', promien:7:2);
  WriteLn (' wynosi: ', DwaPi * promien :11:5);
  Write (SymbolPliku,'Obwod kola o promieniu ', promien:7:2);
  WriteLn (SymbolPliku,' wynosi: ', DwaPi * promien :11:5);
  Close(SymbolPliku);
  ReadLn
end.
```

B. Upewnij się, że w katalogu roboczym nie ma pliku TP05.WYN (jeżeli jest, to usuń go). Uruchom program i sprawdź, że w katalogu powstał plik o tej nazwie zawierający komunikat programu z wprowadzonym przez Ciebie promieniem koła i obliczonym przez program wynikiem. W celu stwierdzenia istnienia pliku TP05.WYN i obejrzenia jego zawartości nie musisz wychodzić z systemu Turbo Pascal, lecz możesz

otworzyć plik za pomocą edytora systemu (File, Open), podając oczywiście nazwę pliku.

C. Wykonaj ponownie polecenia z punktu B, zmieniając wprowadzaną liczbę. Sprawdź, czy plik zawiera oba komunikaty programu czy jeden, a jeżeli jeden, to czy ostatni czy poprzedni.

D. Zmodyfikuj program TP-36 (na TP-36A) zmieniając instrukcję otwierania pliku z REWRITE na APPEND. Uruchom program kilkakrotnie i sprawdź zawartość pliku jak w punkcie C.

E. Zastanów się nad zmodyfikowaniem w podobny sposób programu TP-18 (wyznaczającego NWD i NWW dla danych dwóch liczb) w celu zgromadzenia w pliku pewnej ilości wyników. Zwróć uwagę, że (w odróżnieniu od programu TP-5) wprowadzane dane nie są drukowane instrukcją WRITE. Jeżeli uważasz to za przydatne, to dodaj instrukcję drukowania w pliku danych wprowadzonych przez użytkownika.

Za pomocą tak użytych instrukcji WRITE i WRITELN będziesz mógł zapisywać znaki w tworzonych plikach (zadania: 8-12, 8-13 i 8-14).

Zapis do pliku liczbowego

Sam wynik obliczeń mógłbyś zapisać także w pliku zdefiniowanym jako plik liczb rzeczywistych. Różnica zapisu wiąże się ze sposobem przedstawiania liczb w komputerze. W pliku tekstowym liczba jest przedstawiana w zapisie dziesiętnym, podczas gdy w komputerze jest pamiętana w postaci dwójkowej. Zapisanie liczby dziesiętnej wymaga użycia większej lub mniejszej liczby znaków zależnie od wielkości liczby (na przykład 1,0 i 6457,239102), podczas gdy w komputerze liczby tego samego typu są zapisywane zawsze za pomocą takiej samej liczby bajtów.

Zmodyfikuj program TP-36, tak by zapisywał sam wynik liczbowy do pliku zdefiniowanego jako `file of real`. Trzeba zmienić deklarację zmiennej plikowej oraz wywołanie procedur zapisu do pliku; ponadto trzeba wprowadzić dodatkową zmienną rzeczywistą, ponieważ w procedurze zapisu muszą wystąpić zmienne, a nie wyrażenia. Dla łatwiejszego porównywania wyników modyfikujemy także nazwę pliku, do którego następuje zapis.

```
var promien, obwod: real;
  SymbolPliku: file of real;
...
  Write (SymbolPliku, promien, obwod);
```

Zauważ, że do zapisywania liczb w pliku liczbowym posługujemy się procedurą WRITE (a nie WRITELN), i oczywiście zapisujemy same liczby,

a nie łańcuchy. Z tego względu pomijamy instrukcje formatowania liczb, ponieważ dotyczą one zapisu w postaci dziesiętnej.

Ćwiczenie 10-41

A. Utwórz program na podstawie podanych wskazówek. Porównaj go z programem zapisanym w pliku TP36L.PAS (wyniki są w nim zapisywane do pliku TP05L.WYN).
B. Uruchom program, zapisz wielkości promienia i obwodu widoczne na ekranie. Obejrzyj pod edytorem zawartość utworzonego pliku wynikowego. Czy umiesz rozpoznać w nim liczby znane Ci z ekranu?
C. Uruchom kilkakrotnie program, sprawdzając za każdym razem wielkość powstałego pliku. Porównaj ją z liczbą znaków zajętych na przedstawienie liczb na ekranie. Jak ma się wielkość pliku do liczby bajtów przeznaczonych na zapis liczb (tab. 10.1)?

Zapis do pliku rekordowego

Zmiennej typu rekordowego używałeś w programie TP-22. Zmodyfikuj go teraz w taki sposób, żeby dodatkowo zapisywał rekord do pliku rekordowego. Zmienną plikową zdefiniuj jako `file of ksiazka` (`ksiazka` jest nazwą utworzonego typu zmiennej rekordowej). Procedurą WRITE zapisz cały rekord *Biblioteka*[1], a nie poszczególne jego pola. Porównaj swój program z zapisanym w pliku TP22M.PAS.

Ćwiczenie 10-42

A. Uruchom program. Sprawdź, jaką postać ma informacja w utworzonym pliku wynikowym.
B. Zmodyfikuj zawartość rekordu i sprawdź, czy ma to wpływ na wielkość pliku.

Zauważ, że zapisywanie całych rekordów do pliku typu rekordowego wymaga prostszych instrukcji niż zapisywanie ich do pliku tekstowego.

10.12.3. Odczytywanie danych z pliku dyskowego

Procedury READ i READLN umożliwiają odczytywanie danych bezpośrednio z pliku dyskowego. Przed użyciem ich do tego celu musisz taki plik otworzyć (RESET), przedtem wiążąc z plikiem zmienną typu plikowego (ASSIGN). W procedurach musisz używać tej zmiennej jako pierwszego parametru, podobnie jak używałeś go w procedurze WRITE. Po zakończeniu odczytywania musisz plik zamknąć (CLOSE).

Za pomocą procedur READ i READLN odczytuje się kolejne znaki z pliku tekstowego, pojedynczo lub całymi łańcuchami. Koniec ciągu znaków odczytywanego za jednym wywołaniem procedury READLN wyznacza miejsce, w którym znajdują się znaki oznaczające koniec linii (odpowiednik naciśnięcia klawisza *Enter* podczas wprowadzania z klawiatury). Za pomocą procedury READ odczytuje się dane zapisane w plikach nietekstowych zdefiniowanych jako pliki złożone z elementów konkretnego typu.

Odczytywanie pliku tekstowego znak po znaku

Do odczytywania znaków wygodnie jest posłużyć się procedurą READ. Za pomocą programu TP-37 będziesz odczytywać pojedyncze znaki z pliku, którego nazwę podasz. Przeanalizuj działanie tego programu.

Program TP-37

```
program TP37;
  {odczytywanie pliku znak po znaku}
var
  NazwaPliku: string;
  IdPliku:  text;
  NumerZnaku: longint;
  Znak: char;
begin
  Write('Podaj nazwe pliku  ');
  ReadLn(NazwaPliku);
  Assign(IdPliku,NazwaPliku);
  reset(IdPliku);
  NumerZnaku := 0;
  if Eof(IdPliku) then Writeln('Plik ',NazwaPliku,' jest pusty')
    else
      begin
        repeat
          Read(IdPliku,Znak);
          Inc(NumerZnaku);
          WriteLn(NumerZnaku:6,'  kod znaku ',Znak,' = ',
                                                  Ord(Znak))
        until Eof(IdPliku);
        WriteLn('Odczytano ', NumerZnaku,' Znakow')
      end;
  close(IdPliku);
  ReadLn
end.
```

Po czynnościach wstępnych i otwarciu pliku do odczytu (RESET) następuje sprawdzenie za pomocą funkcji EOF, czy nastąpił już koniec pliku; jeżeli

tak, to funkcja przybiera wartość logiczną Prawda. Jeżeli koniec pliku został wykryty bezpośrednio po jego otwarciu, to drukowany jest komunikat, że plik jest pusty, w przeciwnym razie (ELSE) odczytywane są kolejne znaki w pętli, która jest przerywana, kiedy nastąpi koniec pliku.

Ćwiczenie 10-43

A. Odczytaj program TP-37, uruchom go i odczytaj za jego pomocą plik DANE1.DAT.

B. Odczytaj za pomocą programu TP-37 kilka innych (krótkich) plików. Zastanów się, co byś zmienił w programie, żeby był lepiej przystosowany do odczytywania większych plików.

C. Zmodyfikuj program TP-37 w taki sposób, żeby po odczytaniu takiej ilości znaków, która powinna zapełnić ekran, program zatrzymywał się i czekał na naciśnięcie:
   - wyróżnionego klawisza w celu zakończenia programu,
   - dowolnego innego klawisza w celu kontynuowania odczytu pliku.

   Uwaga. Do sprawdzenia, czy liczba znaków osiągnęła kolejną wielokrotność zadanej liczby, możesz posłużyć się dzieleniem całkowitym. (Porównaj swój program z programem zapisanym w pliku o nazwie TP37M.PAS.)

D. Rozpoznaj w odczytywanym pliku znaki końca linii (powrotu karetki i nowej linii).

E. Odczytaj plik ZNST1 za pomocą programu TP-37 lub za pomocą programu zmodyfikowanego przez Ciebie i porównaj wyniki z przedstawionymi na rys. 4.23.

F. Odczytaj (krótki) plik z polskimi znakami (np. VGA.TAG lub jeden z pozostałych plików znajdujących się na załączonej dyskietce). Zwróć uwagę na kody znaków. Czy możesz rozpoznać (sprawdzić) standard kodowania?

Odczytywanie łańcucha znaków z pliku tekstowego

Odczytywałeś już zmienne łańcuchowe posługując się procedurą READLN. System traktował naciśnięcie przez Ciebie klawisza *Enter* jako zakończenie łańcucha. Zauważ, że to samo dotyczyło wprowadzania kilku liczb. Podobnie jest przy odczytywaniu danych z pliku. System traktuje napotkane w nim znaki końca linii (zazwyczaj jest to para znaków o kodach 13 i 10) jako koniec łańcucha. Koniec pliku możesz rozpoznawać podobnie jak w programie TP-37.

Ćwiczenie 10-44

A. Napisz program odczytujący z pliku kolejne łańcuchy znaków. Czy Twój program jest podobny do programu TP-38? (Cały program znajduje się w pliku TP38.PAS.)

Program TP-38

```
Program TP38;       {fragment}
  ....
        repeat
          ReadLn(IdPliku,Lancuch);
          Inc(NumerLancucha);
          WriteLn(NumerLancucha:4,'  ',Lancuch)
        until Eof(IdPliku);
        WriteLn('Odczytano ', NumerLancucha,' Lancuchow')
```

B. Uruchom program i odczytaj zawartość pliku TP05.WYN.

C. Odczytaj zawartość pliku PLIKDIR.TXT utworzonego poleceniem systemu operacyjnego `DIR > plikdir.txt`. Jeżeli drukowany na ekranie numer łańcucha przeszkadza Ci, to usuń go z instrukcji WRITELN.

Odczytywanie danych z plików nietekstowych

Dysponujesz plikami nietekstowymi: liczbowym i rekordowym, utworzonymi przez programy TP-36L i TP-22M (lub podobnymi opracowanymi przez Ciebie). Do odczytania tych plików możesz posłużyć się tymi samymi programami po wprowadzeniu niewielkich zmian. Możesz porównać swoje programy z programami zapisanymi w plikach TP22MC.PAS i TP36LC.PAS.

Ćwiczenie 10-45

Zmodyfikuj program TP-36LC, tak by liczby odczytane z pliku liczbowego zapisywał w pliku tekstowym w jednej linii, oddzielone średnikiem.

Odczytywanie liczb i tekstów z pliku tekstowego

Z pliku zawierającego dane można odczytywać informację większymi porcjami, a nie tylko znak po znaku. W szczególności można odczytywać łańcuchy znaków oraz liczby. Pewnym problemem jest zakończenie odczytywania danych z pliku we właściwym momencie; jeżeli się tego nie uczyni, to program zatrzyma się sygnalizując błąd. Wygodnie jest znać liczbę odczytywanych elementów. Jeżeli liczba elementów nie jest znana, to można postąpić w sposób analogiczny do stosowanego przy odczytywaniu pojedynczych znaków (sprawdzanie wartości funkcji EOF).

Można w ten sposób odczytywać:

- teksty do zmiennych łańcuchowych i liczby do zmiennych numerycznych;
- kilka liczb umieszczonych w jednym wierszu (trzeba w wywołaniu procedury READLN uwzględnić w tym celu odpowiednią liczbę zmiennych);
- dane przygotowane w taki sposób, że po jednej lub kilku liczbach w tym samym wierszu jest umieszczony tekst (w takiej sytuacji ostatnia ze zmiennych na liście argumentów procedury READLN musi być zmienną tekstową).

Podczas odczytywania pliku zawierającego wiele wierszy danych, dane te można umieszczać w wierszach odpowiednich macierzy.

Plik o nazwie DANE2.DAT, znajdujący się na załączonej dyskietce w katalogu PLIKI\DANE_DAT, zawiera kilka wierszy danych, przy czym w każdym wierszu jest kolejno: liczba rzeczywista, liczba całkowita oraz tekst (imię), na przykład

```
34.56    132    Krzysztof
```

Dane te można odczytać za pomocą programu zawierającego instrukcję podobną do

```
Read(IdPliku,LiczbaRzecz, LiczbaCalk, Lancuch)
```

Ćwiczenie 10-46

A. Napisz własny program odczytywania do tablic danych o przedstawionej postaci i uruchom go (lub odczytaj program TPCZYTAJ). Odczytaj za jego pomocą plik DANE2.DAT z załączonej dyskietki. Porównaj otrzymane wyniki z zawartością pliku.

B. Zmodyfikuj plik DANE2.DAT dodając jedną lub kilka pustych linii i zobacz, jak wpływa to na działanie programu.

C. Zmodyfikuj program w taki sposób, żeby mógł odczytywać pliki zawierające do 60 liczb całkowitych zapisanych po jednej w wierszu. Sprawdź jego działanie odczytując plik DANE3.DAT.

D. Zmodyfikuj program utworzony w punkcie C, aby zapisywał odczytane liczby do pliku liczbowego. Sprawdź działanie programu odczytując plik DANE3.DAT.

E. Zmodyfikuj program z punktu A w taki sposób, żeby mógł odczytywać pliki zawierające po 4 liczby w wierszu, przy czym druga z liczb jest rzeczywista, a pozostałe są całkowite. Sprawdź jego działanie odczytując plik DANE4.DAT.

F. Zmodyfikuj program utworzony w punkcie E, aby zapisywał odczytane czwórki liczb w pliku jako rekordy. Sprawdź jego działanie odczytując plik DANE4.DAT.

G. Zmodyfikuj program z punktu A w taki sposób, żeby mógł odczytywać pliki zawierające w każdym wierszu liczbę całkowitą oraz tekst. Odczytywane dane umieszczaj w tablicy odpowiedz (liczbę całkowitą) i pytanie (tekst). Sprawdź działanie programu odczytując plik o nazwie DANE5.DAT.

10.13. Przykładowe programy

Możesz już napisać wszystkie programy odpowiadające zadaniom z rozdz. 8. Omówimy tu przede wszystkim koncepcję ich utworzenia, dodając niekiedy (fragmentaryczne) wyjaśnienia. Jeżeli w rozdziale 8 został przedstawiony algorytm rozwiązania zadania, to potraktuj tworzenie programu jako dodatkowe ćwiczenie z tzw. kodowania programu za pomocą instrukcji języka programowania.

Program przekształcania pliku — zadanie 8-13

Zadanie 8-13 dotyczy odczytania parzystej ilości liczb zapisanych w pliku po jednej w linii i zapisania ich w innym pliku po dwie obok siebie. Znasz już wszystkie potrzebne instrukcje na tyle dokładnie, że powinieneś poradzić sobie z napisaniem programu bez dodatkowych wskazówek. Napisz go i sprawdź jego działanie na danych zapisanych w pliku DANE6.DAT. Możesz porównać swój program z programem zapisanym w pliku TPDWLICZ.PAS.

Program testu — według algorytmu 15 do zadania 8-11

Schemat postępowania z wybieraniem pytań testowych z tablicy jest przedstawiony w algorytmie 15, napisanie zasadniczego zrębu programu nie powinno przedstawiać zatem trudności.

Odczytywanie danych możesz oprzeć na programie opracowanym w punkcie D ostatniego ćwiczenia (możesz przekształcić go na procedurę, lecz nie musisz). Do testowania programu możesz użyć pliku DANETEST.DAT. Porównaj swój program z programem TP-39; jego zasadniczy fragment przedstawiono poniżej (cały program jest zawarty w pliku TP39.PAS).

Program TP-39

```
program TP39;
  {test tekstowy - pytania odczytywane z pliku, zadawane losowo}
uses Crt;
```

```
var
  NazwaPliku: string;
  IdPliku:  text;
  DobreOdp, Numer, NumerLos, NumerOdp, NumerPom : integer;
  NumerWiersza: integer;
  Kontynuacja: char;
  Zadane: array[1..40] of Boolean;
  Pytanie: array[1..40] of string;
  Odpowiedz:  array[1..40] of integer;
procedure DanePoczatkowe;
begin
     {odczytanie danych,
      wstawienie wartosci poczatkowych do tablic Zadane}
  DobreOdp := 0;
  Numer := 1;
  NumerPom := 0
end; {DanePoczatkowe}
begin                                    {program glowny}
  DanePoczatkowe; {Kroki 1 i 2}
  repeat
    ClrScr;
    repeat                                  {Krok 3}
      NumerLos := 1 + Trunc(40 * Random)
    until not Zadane[NumerLos];
    WriteLn (Pytanie[NumerLos]);            {Krok 4}
    Zadane[NumerLos] := True;
    Write('Podaj numer odpowiedzi ');  {Krok 5}
    ReadLn(NumerOdp);
    if NumerOdp = Odpowiedz[NumerLos] then
      Inc(DobreOdp);
    if Numer <= 20 then                {Kroki 6 i 7}
      begin
        WriteLn('Czy chcesz odpowiedziec na nastepne
                                                pytanie? t/n ');
        Kontynuacja := UpCase(ReadKey);
        if Kontynuacja = 'T' then Inc(Numer)
      end
  until (Numer > 20) or (Kontynuacja <> 'T');
  ClrScr;                                          {Krok 8}
  WriteLn ('Udzieliles ',Dobreodp,' prawidlowych odpowiedzi');
  WriteLn ('na ',Numer,' zadanych pytan');
  ReadLn
end.
```

Program TP-39 możesz samodzielnie ulepszyć. Na przykład podawanie nazwy pliku z pytaniami możesz zastąpić wybieraniem nazwy przedmiotu.

Liczba pytań do różnych przedmiotów nie musi być identyczna. Możesz także umożliwić ładniejsze przedstawianie na ekranie odpowiedzi do wyboru — w oddzielnych liniach — przez osobne umieszczenie tych odpowiedzi w pliku danych (każda w oddzielnej linii) i oddzielne umieszczanie ich w tablicy (mógłbyś użyć jednej tablicy z dodatkowym indeksem, oznaczającym numer odpowiedzi). Liczba odpowiedzi do wyboru nie musiałaby być identyczna dla wszystkich pytań; musiałbyś jednak podawać w pliku danych, i zapamiętywać w programie, liczbę odpowiedzi do poszczególnych pytań.

Program sporządzania mapy funkcji — zadanie 8-10

Koncepcja odwzorowywania funkcji dwóch zmiennych na ekranie z użyciem kolorów została przedstawiona w rozdz. 8. Trzeba skorzystać z trybu graficznego, który umożliwia stosowanie kilku kolorów. Z rozdzielczości ekranu w wybranym trybie wynika sposób wzajemnego powiązania zmiennych niezależnych x i y funkcji ze współrzędnymi ekranu, np. przy rozdzielczości równej 640 pikseli zmienna x z zakresu $0 \leq x < 10$ powinna być mnożona przez 64 (i zmniejszana do wartości całkowitej). Do przyporządkowania kolorów trzeba użyć instrukcji warunkowych. Możesz porównać sporządzony przez siebie program z programem zapisanym w pliku o nazwie TPMAPA.PAS.

Czas działania programu przy obliczaniu wartości koloru oddzielnie dla każdego punktu ekranu może być znaczny. Jeżeli tak się okaże, to zmniejsz odpowiednio liczbę punktów w poziomie i pionie, pokrywając obrazem jedynie część ekranu (albo przypisując dany kolor kilku sąsiednim punktom). Spróbuj teraz zmienić granice przedziałów odpowiadających poszczególnym kolorom, a także, jeśli pozwala na to karta graficzna, użyć większej liczby kolorów.

Programy zmiany kodów znaków w pliku — zadanie 8-12

Programy te możesz oprzeć na programie TP-37 realizującym odczytywanie pliku znak po znaku. Do szyfrowania możesz użyć prostego algorytmu zmniejszającego wartość kodu o 1 lub o wielkość podaną przez użytkownika. (Tekst „To jest szyfr" po zmniejszeniu kodów o 1 wyglądałby następująco: „Sn idrs ryxeq". A jak byś odszyfrował słowo: „Jnlotsdq"?) Uruchamiając program sprawdź m.in., czy zaszyfrowany plik po odszyfrowaniu jest identyczny z plikiem oryginalnym. Do zamiany standardu polskich znaków użyj instrukcji CASE (rozdz. 10.7.3). Możesz porównać sporządzone przez siebie programy z programami zapisanymi w plikach TPSZYFR.PAS i TPMAZISO.PAS (zamiana standardu z kodu Mazovia na ISO Latin 2).

Program do dodawania „pisemnego" — zadanie 8-8

Dodawanie „pisemne", tak proste na papierze, może Ci sprawić nieco trudności. Musisz się zdecydować, jak chcesz reprezentować poszczególne cyfry w pamięci programu. Zauważ, że nie są to wyłącznie cyfry, lecz także „miejsca puste", traktowane przy dodawaniu jako zera. Naturalną strukturą odpowiadającą papierowi kratkowanemu jest tablica dwuwymiarowa. Jej elementami mogą być liczby całkowite (do przedstawiania cyfr posłużą liczby od 0 do 9, a do zapamiętania miejsca pustego np. liczba ujemna). Można też posłużyć się zmiennymi łańcuchowymi. Zapisanie w tablicy wprowadzonych przez użytkownika liczb, które należy dodać, wymaga rozbicia liczb wielocyfrowych na liczby jednocyfrowe — oznaczające pozycje jedności, dziesiątek, setek itd. Do tego celu należy posłużyć się dzieleniem całkowitym. Pisząc swój program możesz przyjrzeć się rozwiązaniom zastosowanym w programie zapisanym w pliku TPDODPIS.PAS.

Jeśli chcesz, to po uruchomieniu swojego programu lub przeanalizowaniu działania programu z dyskietki zastanów się, co należałoby w nim zmienić, żeby realizował (ilustrował) odejmowanie liczb. A może jest potrzebny program ilustrujący wykonywanie innych działań arytmetycznych ...? Zastanów się także nad przystosowaniem go do dodawania w innych systemach pozycyjnych, przede wszystkim w systemie dwójkowym.

10.14. Podsumowanie

Doszliśmy do miejsca, gdzie zaczyna się użycie języków programowania na poziomie zaawansowanym. Większego znaczenia nabiera stosowanie procedur i funkcji. Programista musi brać pod uwagę takie problemy, jak zakres obowiązywania poszczególnych deklaracji, wspólne używanie zmiennych i tablic przez różne funkcje i procedury, a nawet definiowanie procedur i funkcji wewnątrz innych procedur. Jeżeli zamierzasz kontynuować naukę pisania programów za pomocą języka wysokiego poziomu, to będziesz musiał wiadomości z tego tematu przyswoić sobie w nieco większym zakresie. Stoi przed tobą otworem dziedzina określana jako **programowanie strukturalne**. Chodzi tu o nadawanie programom wyraźnej struktury w wyniku używania procedur i funkcji. Druga taka dziedzina, a raczej jej odmiana, to tzw. **programowanie obiektowe**. W tym wypadku określa się nie tylko funkcje i procedury, ale także obiekty, z którymi są związane.

Jeżeli nie zamierzasz kontynuować nauki programowania, to i tak zdobyte wiadomości i umiejętności powinny umożliwić Ci napisanie niewielkich programów na własne potrzeby. Poza tym jesteś już lepiej przygotowany do rozwiązywania problemów, z jakimi możesz się zetknąć.