Zauważ, że zmienna *Biblioteka* ma 200 rekordów, natomiast w programie użyliśmy zaledwie jednego z nich. Jeżeli interesuje Cię posługiwanie się rekordami, to możesz ten program samodzielnie rozbudować.

10.8.4. Pliki

Będziesz się posługiwał plikami dyskowymi. Musisz wówczas zadeklarować ich typ. Plik jest dla programu ciągiem elementów tego samego rodzaju, którego liczba elementów nie jest stała i może się zmieniać w trakcie wykonywania programu. W definicji typu występują słowa FILE OF, np.

```
type dane = file of integer;
     wyniki = file of char;
```

Można też użyć samego słowa FILE bez OF i następującej po nim części; oznacza to, że plik nie ma zdefiniowanej struktury. Nie można definiować pliku, którego elementami byłyby pliki (*file of file*).

W systemie Turbo Pascal istnieje też zdefiniowany typ plikowy o nazwie TEXT (plik tekstowy); jego elementami są znaki zapisane w liniach, możesz zatem używać tego typu w deklaracjach zmiennych.

Typ elementów pliku może być uprzednio zdefiniowany w programie, na przykład po zdefiniowaniu typu rekordowego *ksiazka* można zadeklarować posłużenie się zmienną plikową *file of ksiazka*.

Plikowi odpowiada fizyczny zbiór danych o dostępie sekwencyjnym. Oznacza to, że w każdej chwili istnieje dostęp tylko do jednego elementu pliku, a dostęp do innych elementów wymaga wykonania pewnych operacji. Zbiorem takim jest nie tylko plik dyskowy, ale także zbiór danych w pamięci operacyjnej komputera oraz danych wprowadzanych lub wyprowadzanych przez urządzenia zewnętrzne (klawiatura, monitor, drukarka).

Posługiwanie się plikami, ich odczytywanie i zapisywanie, będzie omówione w rozdz. 10.12.

10.9. Funkcje i procedury standardowe

10.9.1. Moduły

Funkcje i procedury standardowe są zgrupowane w modułach. Moduł **System** jest dostępny zawsze bez specjalnych zabiegów. Zawiera on funkcje arytmetyczne, funkcje do wykonywania zamiany typów zmiennych, funkcje porządkowe, funkcje do wykonywania operacji na łańcuchach i in. Korzystanie z funkcji i procedur zawartych w innych modułach wymaga już pewnych działań z Twojej strony.

W systemie Turbo Pascal znajduje się procedura czyszczenia ekranu CLRSCR (*clear screen* — czyść ekran). Dodaj na początku któregoś ze swoich programów instrukcję wywołania tej procedury (kończąc średnikiem) i spróbuj otrzymany w ten sposób program skompilować (w książce uzupełnimy w ten sposób program TP-1).

```
program TP23;
begin
  ClrScr;
  WriteLn ('Pierwszy komunikat: koniec programu!')
end.
```

Kompilacja się nie powiedzie i zobaczysz komunikat:

```
Error 3:  Unknown identifier.
```

oznaczający

Błąd 3: Nieznany identyfikator.

Program tłumaczący informuje Cię w ten sposób, że nazwy (ciągu liter) CLRSCR nie rozpoznał jako znanej mu. Przyczyną takiego sposobu potraktowania procedury CLRSCR jest to, że nie ma jej w module SYSTEM, do którego system sięga bez specjalnych deklaracji. System musi być poinformowany, że powinien użyć odpowiedniego modułu. Służy do tego deklaracja USES (używa), po której wskazujesz nazwy modułów, z których zamierzasz korzystać.

Procedura CLRSCR znajduje się w module CRT. Zmodyfikuj więc program, dopisując tę deklarację jako pierwszy element po nagłówku programu:

Program TP-23

```
program TP23;
  {wersja uzupelniona - deklaracja USES CRT}
uses Crt;
begin
  ClrScr;
  WriteLn ('Pierwszy komunikat: koniec programu!')
end.
```

Tym razem kompilacja powinna się powieść.

Po wyjściu z systemu odszukaj plik o nazwie CRT (jeżeli możesz, posłuż się programem wspomagającym odszukiwanie plików, np. takim jak Norton Commander). Zobaczysz, że plik ten ma rozszerzenie TPU. Moduły programowe w systemie Turbo Pascal mają właśnie takie rozszerzenie. Dotyczy to zarówno modułów gotowych, jak i modułów tworzonych przez użytkownika.

W tej chwili możesz patrzeć na moduł jako na coś, o czym trzeba pamiętać (co „utrudnia Ci życie"). W rzeczywistości moduły ułatwiają pisanie programów w sposób przejrzysty, co ma szczególne znaczenie podczas tworzenia dużych programów. Jeżeli będziesz kontynuował naukę posługiwania się systemem Turbo Pascal, to dowiesz się, jak tworzyć moduły. Na razie powinieneś zdawać sobie sprawę z ich istnienia i umieć korzystać z gotowych.

Uwaga. Przed użyciem funkcji lub procedury należącej do systemu Turbo Pascal warto się upewnić, czy znajduje się ona w module **System**, czy też w jakimś innym module i w jakim. Moduł **Crt**, z którym już się zetknąłeś, zawiera funkcje i procedury służące do obsługi ekranu i klawiatury. Funkcje służące do sporządzania wykresów znajdują się w module **Graph**, a do wyprowadzania wyników na drukarkę — w module **Printer**. Moduł **Dos** umożliwia dostęp do funkcji systemu operacyjnego.

Wszystkie te informacje są dostępne w pomocy. W informacji o funkcji lub procedurze jest podany moduł, w którym się ona znajduje; jeżeli moduł nie jest wymieniony, to znaczy że funkcja jest dostępna w module **System**.

10.9.2. Standardowe funkcje matematyczne

Do obliczeń możesz wykorzystywać funkcje matematyczne dostępne w systemie Turbo Pascal. Należą do nich takie funkcje, jak: **pierwiastek kwadratowy** (SQRT), **funkcja kwadratowa** (SQR) i **wartość bezwzględna** (ABS), **funkcje trygonometryczne** (SIN i COS), **logarytm naturalny** (LN) i **funkcja wykładnicza** (EXP). Oprócz funkcji trygonometrycznych jest też **funkcja odwrotna do funkcji tangens** (ARCTAN); jej argumentem jest dowolna liczba rzeczywista, a wartością kąt (w radianach) o takiej wartości, że tangens tego kąta jest równy danej liczbie.

Jeżeli posługiwałeś się arkuszem QuattroPro lub innym w wersji angielskojęzycznej, to widzisz, że nazwy funkcji często są takie same jak w arkuszu kalkulacyjnym (choć nie zawsze). Zauważ, że dwie podobne nazwy SQR i SQRT są użyte na oznaczenie dwóch różnych funkcji: kwadratowej i pierwiastkowej.

Chcąc obliczyć wartość funkcji dla konkretnego argumentu, musisz ten argument napisać w nawiasach po nazwie funkcji, np. SQR(x) (podobnie jak w arkuszu kalkulacyjnym). Argumentem funkcji może być wyrażenie, np. SIN(x+y). Wartości funkcji możesz używać w wyrażeniach, na przykład 0,5*(b-SQRT(delta)).

Teraz umiesz już dokończyć program realizujący algorytm 3 do zadania 8-1.

Ćwiczenie 10-19

Napisz program podający rozwiązanie równania kwadratowego: $ax^2 + bx + c = 0$.

Za pomocą podanych funkcji możesz realizować potęgowanie x^y dla niecałkowitych wartości wykładnika y i dodatniej podstawy x. Należy użyć dwóch funkcji: wykładniczej EXP(x), podnoszącej do potęgi x liczbę e (stałą matematyczną), oraz logarytmiczną LN(x), wyznaczającą logarytm naturalny (czyli przy podstawie e) swojego argumentu x. Do obliczenia potęgi x^y posłuż się wzorem:

$$x^y = e^{y\ln(x)}.$$

Chcąc się przekonać o jego prawdziwości oblicz logarytm obu jego stron.

Ćwiczenie 10-20

A. Napisz program TP-POTEGA realizujący potęgowanie x^y. Czy użyłeś podobnego wyrażenia: `z := EXP(y * LN(x))`? Sprawdź wyniki dla kilku par danych x i y.

B. Dla kilku całkowitych wartości wykładnika y porównaj wyniki otrzymywane tą metodą z uzyskiwanymi za pomocą wielokrotnego mnożenia (w pętli FOR).
(Dokładność obliczeń zależy od wyposażenia komputera w koprocesor, a w razie jego braku — od sposobu jego emulacji.)

10.9.3. Liczby losowe

W systemie Turbo Pascal znajduje się funkcja RANDOM generująca liczby losowe. Za każdym kolejnym jej użyciem ma ona inną wartość. Podobnych funkcji używałeś w arkuszu kalkulacyjnym (LOS, RAND). Wartością funkcji RANDOM bez parametru jest liczba rzeczywista nieujemna mniejsza od 1. Jeżeli funkcję RANDOM wywołasz z parametrem o wartości całkowitej (typu *word*), to jej wartościami będą liczby z przedziału od 0 do wartości o 1 mniejszej od wartości parametru, np. RANDOM(100) daje liczby od 0 do 99 (RANDOM(0) daje 0).

Program TP-24

```
program TP24;
  {Liczby losowe}
begin
  Randomize;
  WriteLn (Random:9:4);
  WriteLn (Random:9:4);
```

```
  WriteLn (Random(100):4);
  WriteLn (Random(100):4)
end.
```

Procedura RANDOMIZE wywołana w pierwszej linii programu służy do losowego inicjalizowania generatora liczb losowych.

Ćwiczenie 10-21

A. Uruchom program i sprawdź, czy powtarzającym się w programie identycznym instrukcjom odpowiadają takie same liczby.
B. Uruchom kilkakrotnie program TP-24 i porównaj otrzymywane za każdym razem cztery liczby. Sprawdź, czy się powtarzają przy kolejnych uruchomieniach programu.
C. Usuń instrukcję RANDOMIZE (lub obejmij ją nawiasami klamrowymi) i powtórz punkt B.

Możesz już napisać program TP-25 do zadania 8-11 według algorytmu 12.

Program TP-25

```
program TP25;
  {realizujacy algorytm 12 dla zadania 8-11
   test z tabliczki mnozenia}
uses Crt;
var
  Licz,DobreOdp,Czynnik1,Czynnik2,Iloczyn:   integer;
  JeszczeJedna:                              string;
begin
  ClrScr;
  Randomize;
  Licz := 0;
  DobreOdp := 0;
  repeat
    Czynnik1 := 2 + random(9);
    Czynnik2 := 2 + random(9);
    Write ('Podaj wynik mnozenia', Czynnik1,' przez',
                                        Czynnik2,' ');
    ReadLn (Iloczyn);
    if Czynnik1 * Czynnik2 = Iloczyn then
      begin
        DobreOdp := DobreOdp + 1;
        WriteLn ('Odpowiedz poprawna')
      end
    else
      WriteLn('Odpowiedz bledna. Iloczyn jest rowny
                                    ',Czynnik1*Czynnik2);
```

```
    Licz := Licz + 1;
    Write ('Czy jeszcze jedno pytanie? (t/n) ');
    ReadLn (JeszczeJedna)
  until JeszczeJedna <> 't';
  WriteLn;
  Write ('Na ', Licz, ' zadanych pytan ');
  WriteLn ('udzieliles ', DobreOdp,' prawidlowych odpowiedzi');
  ReadLn
end.
```

10.9.4. Zamiana typów

W systemie Turbo Pascal oprócz funkcji znanych Ci z matematyki są dostępne funkcje związane z zamianą typów zmiennych.

Do zamiany liczb rzeczywistych na całkowite przez zaokrąglenie w dół służy funkcja TRUNC (*truncate* — obetnij) o wartościach typu *longint*, np. TRUNC(7.6)=7, TRUNC(7.1)=7. Oprócz niej jest funkcja ROUND (zaokrągl), wykonująca zaokrąglanie w górę lub w dół, zależnie od tego, gdzie popełnia się mniejszy błąd, np. ROUND(7.6)=8, ROUND(7.1)=7.

Zamiana w drugą stronę nie jest taka ważna, ponieważ w razie użycia w wyrażeniu liczby całkowitej zamiast rzeczywistej konwersja typu jest wykonywana przez system samoczynnie.

Jest także funkcja CHR przyporządkowująca kodowi znaku ($0 \div 255$) sam znak ASCII. Przekształcenie odwrotne można wykonać za pomocą funkcji ORD.

Napisz program TP-26 (lub wczytaj go z pliku TP26.PAS) i uruchom go.

Program TP-26

```
program TP26;
  {drukowanie zestawu znakow}
uses Crt;
var kod:  integer;
begin
  ClrScr;
  for kod := 32 to 255 do Write (Chr(kod),' ');
  ReadLn
end.
```

Ćwiczenie 10-22

A. Zmodyfikuj program TP-26 w taki sposób, żeby po napisaniu każdych kolejnych 32 znaków kończył linię. Sprawdź jego działanie. Możesz go porównać z programem zapisanym w pliku TP26M.PAS.

B. Poszerz zakres drukowanych znaków o znaki o kodach od 0 do 31.

C. Napisz program, który w odpowiedzi na podaną przez użytkownika liczbę n (≥ 32) wydrukuje na ekranie 20 znaków o kodach kolejnych począwszy od danego numeru n, podając obok znaku numer jego kodu.

10.9.5. Funkcje i procedury porządkowe

Są również w systemie Turbo Pascal funkcje i procedury porządkowe. Należy do nich procedura INC zwiększająca o 1 wartość argumentu — będącego zmienną całkowitą (długą). Instrukcja wywołania procedury INC, składa się z nazwy procedury i argumentu w nawiasach okrągłych:

```
Inc(x)
```

Jest ona równoważna instrukcji `x:=x+1`.

Uwaga. Podany przykład ilustruje różnicę między funkcją a procedurą. Gdyby INC była funkcją, wówczas zwiększona o 1 wartość zmiennej x byłaby „zawarta" w nazwie funkcji, a do nadawania jej zmiennej x należałoby użyć instrukcji przypisania `x := INC(x)`

Procedura DEC działa podobnie jak INC, tyle że wartość argumentu zmniejsza o 1.

Funkcja ODD służy do sprawdzania nieparzystości argumentu (będącego liczbą całkowitą); jeżeli jest nieparzysty, to wartością funkcji jest Prawda jako wartość logiczna (*true*), w przeciwnym razie — Fałsz (*false*).

10.9.6. Operacje na zmiennych łańcuchowych

Przedmiotem działań mogą być także teksty, zarówno stałe tekstowe, jak i zmienne. Podstawowym działaniem jest łączenie tekstów, zwane poprawnie konkatenacją, a potocznie dodawaniem, ponieważ są zaznaczane za pomocą tego samego symbolu *plus*, który oznacza dodawanie wyrażeń liczbowych. Dodawanie zmiennych łańcuchowych powoduje połączenie obu tekstów w jeden, bez zaznaczania przerwy. Na przykład 'Ala' + 'ma' jest równe 'Alama', a nie 'Ala ma'.

Jest wiele funkcji i procedur wykonujących operacje na zmiennych łańcuchowych.

Funkcja LENGTH podaje długość łańcucha będącego jej argumentem, np. wartością funkcji LENGTH('Beata') jest 5.

Funkcja COPY ma trzy argumenty, z których pierwszy jest łańcuchem (zmienną łańcuchową), a dwa pozostałe liczbami całkowitymi. Wartością funkcji jest łańcuch wycięty z łańcucha będącego argumentem, począwszy od pozycji wskazanej przez drugi argument, o długości określonej przez ar-

gument trzeci. Na przykład wartością funkcji COPY('Turbo Pascal',4,6) jest łańcuch 'bo Pas'.

Funkcja POS ma dwa argumenty będące łańcuchami. Jej wartością jest pozycja drugiego łańcucha, począwszy od której jest on zgodny z pierwszym łańcuchem. Wartością funkcji POS('Pascal','Turbo Pascal') jest 7.

Ćwiczenie 10-23

A. Zapoznaj się z treścią programu TP-27 i podaj, jakie powinny być wyniki jego działania, dla danych: 'Beata' i 'Krzysztof'.

Program TP-27

```
program TP27;
  {dzialania na lancuchach}
var Im1,Im2,SumaIm,Fragm:   string;
begin
  Write ('Napisz pierwsze slowo ');
  ReadLn (Im1);
  Write ('Napisz drugie slowo   ');
  ReadLn (Im2);
  SumaIm := Im1 + Im2;
  WriteLn ('Dlugosc slow = ',length(Im1),' ',
                                    length(Im2));
  WriteLn ('Suma slow = ',SumaIm);
  Fragm :=  copy(SumaIm,4,6);
  WriteLn ('Fragment sumy = ', Fragm);
  WriteLn ('Pozycja fragmentu w sumie = ',
                           pos(Fragm,SumaIm));
  ReadLn
end.
```

B. Napisz lub odczytaj program TP-27, uruchom go i sprawdź, czy daje on takie wyniki, jakich spodziewałeś się w punkcie A. Jeżeli nie, to sprawdź, jaka jest przyczyna tej różnicy.

C. Zmodyfikuj program TP-27 w taki sposób, by z łańcucha znaków odpowiadającemu informacji o jednym pliku zawartej w komunikacie polecenia DIR, np.

```
TP27      PAS          816 05-22-95  12:31a
```

„wycinał" i drukował w oddzielnych wierszach: nazwę pliku, jego wielkość i datę zapisu.

Jest jeszcze wiele innych funkcji, m.in. funkcja INSERT, umożliwiająca wstawienie jednego łańcucha w środek drugiego, i funkcja DELETE, ka-

sująca fragment łańcucha. Opis i przykłady działania tych funkcji znajdziesz w pomocy.

Do interesujących procedur należą STR i VAL. Procedura STR (*string* — łańcuch) ma dwa argumenty: numeryczny i łańcuchowy; przypisuje ona drugiemu argumentowi łańcuch złożony z takich znaków, jakie byłyby użyte do przedstawienia na ekranie liczby równej pierwszemu argumentowi. Na przykład po wywołaniu procedury STR(13,x) zmienna łańcuchowa x jest równa '13' (apostrofy podkreślają, że jest to zmienna łańcuchowa złożona z dwóch znaków: 1 i 3, a nie liczba o wartości trzynaście).

Procedura VAL (*value* — wartość) ma trzy argumenty: łańcuchowy, numeryczny i pomocniczy całkowity. Przypisuje ona drugiemu argumentowi liczbę o takiej samej wartości, jaka jest zapisana w pierwszym argumencie w postaci ciągu znaków. Wartość trzeciego argumentu zależy od tego, czy operacja zamiany została przeprowadzona prawidłowo, zaś możliwość prawidłowego jej przeprowadzenia zależy od typu zmiennej. Po wywołaniu procedury VAL('12.3',x,k) wartością zmiennej rzeczywistej x jest 123, a wartością zmiennej całkowitej k jest 0 na oznaczenie faktu, że zamiana została przeprowadzona poprawnie. Gdyby jednak na miejscu zmiennej x była użyta zmienna całkowita, wówczas zamiana nie byłaby możliwa, i zmienna k przyjęłaby wartość różną od zera.

Jeżeli zgubiłeś się w tym trochę, to rozważ, że liczba określonego typu jest zapisywana w sposób właściwy dla systemu Turbo Pascal przy użyciu określonej liczby bajtów niezależnie od jej wartości, np. liczba całkowita zawsze za pomocą dwóch bajtów, natomiast jej zapis w postaci ciągu znaków wymaga różnej liczby bajtów (znaków), np. jednego dla liczby 3, dwóch dla −3 i 15, a aż sześciu dla −20000).

Na liczbach możesz wykonywać działania arytmetyczne, podczas gdy na łańcuchach, nie. Tylko liczby możesz przez siebie mnożyć i dzielić oraz odejmować od siebie. Wprawdzie do zapisania operacji dodawania liczb i „dodawania" łańcuchów służy taki sam symbol +, ale operacje są inne. Na przykład suma arytmetyczna dwóch liczb 15 i 3 wynosi 18, niezależnie od kolejności zapisania (15+3 czy 3+15), natomiast suma łańcuchów stanowiących zapisy tych liczb: '15' i '3', jest łańcuchem powstałym przez złączenie znaków, wynoszącym zatem albo '153' albo '315', zależnie od kolejności zapisania składników „sumy".

Ćwiczenie 10-24

Odczytaj z dyskietki program zapisany w pliku TPSTRVAL.PAS. Przeanalizuj jego treść i zapisz, jak powinny wyglądać jego wyniki po wprowadzeniu liczb 15 i 3. Uruchom program i sprawdź wyniki. Ewentualne niezgodności wyjaśnij z nauczycielem.

Operacje takie mogą Ci się przydać podczas wykonywania działań na danych, gdybyś np. z łańcucha odczytanego z pliku miał wyciąć fragmenty odpowiadające różnym elementom i te fragmenty, które są zapisem liczb, zamienić na liczby.

Podczas odczytywania odpowiedzi tekstowych użytkownika może być przydatna funkcja UPCASE zamieniająca literę na odpowiadającą jej wielką literę, np.

```
UpCase(T) = T
UpCase(t) = T
```

Łatwiej jest taki tekst porównywać z wzorcem. W programach, w których kontynuacja obliczeń zależy od udzielenia przez użytkownika odpowiedzi 't' (tak), bezpieczniej byłoby zastąpić warunek `x = 't'` warunkiem `UPCASE(x) = 'T'`. Podobnie w programie TP-19.

10.9.7. Odczytywanie danych z klawiatury

Odczytywałeś już liczby i łańcuchy za pomocą procedur READ i READLN. Najczęściej były to pojedyncze zmienne, niemniej w programie TP-10 odczytywałeś także kilka liczb jedną instrukcją READLN.

Do odczytywania danych z klawiatury można posługiwać się także funkcjami: READKEY i KEYPRESSED. Obie funkcje są dostępne w module CRT.

Wartością pierwszej z nich jest znak (*char*) odpowiadający naciśniętemu klawiszowi. Wykonanie instrukcji:

```
znak := ReadKey
```

spowoduje przypisanie zmiennej *znak* znaku odczytanego z klawiatury (zmienna *znak* powinna być typu znakowego).

Funkcja KEPRESSED informuje, czy jakikolwiek klawisz został naciśnięty, jeżeli tak, to jej wartością jest Prawda. Można jej użyć m.in. do wstrzymania wykonywania programu do czasu naciśnięcia klawisza przez użytkownika, np. za pomocą następującej instrukcji pętli:

```
repeat
until KeyPressed;
```

Uwaga. Odczytywanie znaku następuje właściwie z bufora klawiatury, który zapamiętuje naciśnięte klawisze do czasu ich odczytania przez komputer. Gdyby w buforze klawiatury znajdował się jakiś znak — a może tam pozostać po wykonywanych poprzednio operacjach lub znaleźć się w następ-

pstwie nieumyślnego naciśnięcia klawisza — to do programu może być wprowadzona niewłaściwa informacja. Możesz się przed tym zabezpieczyć opróżniając bufor klawiatury. Robi się to prosto za pomocą pętli WHILE (zmienna *znak* musi być typu znakowego):

```
while KeyPressed do znak := ReadKey;
```

Za każdym obiegiem pętli jest sprawdzany warunek; jeżeli w buforze jest jakiś znak, to jest odczytywany i usuwany z bufora. Postępowanie takie jest powtarzane do chwili, gdy wszystkie znaki zostaną usunięte z bufora; wówczas wartością funkcji KEYPRESSED staje się Fałsz i system przechodzi do wykonania dalszej części programu. Wartością zmiennej *znak* pozostaje znak odpowiadający ostatnio naciśniętemu klawiszowi.

Uwaga. Naciśnięcie samego klawisza *Alt* lub *Ctrl*, lub *Shift* jest traktowane tak samo, jak gdyby żaden klawisz nie był wciśnięty; są one brane pod uwagę dopiero w połączeniu z innymi klawiszami. Podobnie nie jest brane pod uwagę naciśnięcie klawiszy: *Caps Lock*, *Num Lock*, *Print Screen*, *Scroll Lock* i *Pause*.

Używając wymienionych funkcji można zabawić się w mierzenie szybkości reakcji na pojawienie się komunikatu na ekranie. Nie byłby to prawdziwy pomiar czasu, lecz na razie jego namiastka w postaci liczenia obiegów pętli. Wielkość „jednostki" czasu zależy od komputera. Jeżeli chciałbyś ją przeliczyć na sekundy, musiałbyś przeprowadzić eksperyment (np. zmierzyć liczbę „jednostek" odmierzoną w ciągu np. 20 s).

Program TP-28

```
Program TP28;
  {czas czekania na nacisniecie klawisza}
uses Crt;
var
  znak: char;
  licznik: longint;
begin
  ClrScr;
  licznik := 0;
  while KeyPressed do znak := ReadKey;
  WriteLn ('Nacisnij szybko klawisz ');
  repeat
    Inc (licznik);
  until KeyPressed;
  znak:= ReadKey;
```

```
    Write ('Zostal nacisniety klawisz ', znak);
    WriteLn (' po ', licznik, ' ,,jednostkach''');
    ReadLn
end.
```

Tego rodzaju program mógłbyś rozbudować do programu sprawdzającego poprawność pisania na klawiaturze (mógłbyś wskazywać losowo znak, jaki ma być wprowadzony z klawiatury). Mógłbyś przy tym zmieniać losowo czas przerwy pomiędzy kolejnymi zapytaniami, mierzyć czas czekania na naciśnięcie klawisza itp.

10.9.8. Wyprowadzanie wyników na ekran

Do wyprowadzania wyników tekstowych służą procedury WRITE i WRITELN, które zdążyłeś już poznać. Podstawowym urządzeniem, do którego wyniki są wyprowadzane, jest monitor. Zapisywanie wyników w pliku dyskowym i wysyłanie ich do drukarki poznasz w rozdz. 10.12.3. Teraz poznasz procedury pomocnicze, pozwalające nadać wynikom ciekawszą formę.

Kolor wydruku

Jeżeli Twój komputer ma kolorowy monitor, to możesz określać kolor wydruku na ekranie; jeżeli natomiast masz monitor monochromatyczny VGA, to możesz określać stopień szarości. Służą do tego różne procedury zawarte w module CRT.

Procedura TEXTCOLOR określa kolor napisu. Ma ona jeden parametr. A oto przykładowe wywołania tej procedury (*blink* oznacza mruganie; *red* — czerwony; *yellow* — żółty):

```
TextColor(4);  lub  TextColor(red);
TextColor(14+blink)  lub  TextColor(yellow+blink)
```

Podobnie za pomocą procedury TEXTBACKGROUND określa się kolor tła (*green* — zielony):

```
TextBackGround(green)
```

Ćwiczenie 10-25

A. W programie testującym znajomość tabliczki mnożenia (lub innym) wprowadź różne kolory: komunikatu odpowiadającego dobrej odpowiedzi, złej odpowiedzi i komunikatowi końcowemu.

B. Sprawdź, czy zmiany koloru tła i koloru liter są ograniczone do tych fragmentów linii, które są zajmowane przez dany tekst, czy też rzutują na cały ekran edycyjny.

C. Wyjdź z programu i sprawdź, czy nowe napisy są w standardowym kolorze czy też w jednym z kolorów wprowadzonych przez Ciebie w programie.

Sygnały dźwiękowe

Procedura SOUND powoduje wydanie dźwięku o określonej wysokości (częstotliwości wyrażonej w hercach). Dźwięk trwa do czasu wykonania bezparametrowej procedury NOSOUND. Czas trwania dźwięku można określić za pomocą procedury DELAY z parametrem określającym czas wykonywania tej procedury wyrażonym w milisekundach. Wymienione procedury są dostępne w module CRT:

```
Sound(1200);
Delay(500);
NoSound
```

Ćwiczenie 10-26

A. Rozbuduj program TP-28 o sygnalizowanie dźwiękiem przekroczenia założonego czasu reakcji. Spróbuj tak dobrać wartość graniczną zmiennej *licznik*, żeby odpowiadało to czasowi reakcji ok. 0,8 s.
B. Napisz i uruchom program zmieniający częstotliwość dźwięku w pętli.

Okna tekstowe

W module CRT jest procedura umożliwiająca zdefiniowanie okna. Ma ona cztery parametry: dwa pierwsze podają numer kolumny i numer wiersza lewego górnego rogu, a dwa pozostałe — współrzędne prawego dolnego rogu.

Ćwiczenie 10-27

Odczytaj program zapisany w pliku TPOKNO.PAS. Przeanalizuj jego treść, uruchom go i sprawdź, czy to, co widzisz, odpowiada Twoim oczekiwaniom.

Napisz tekst zawierający kilka do kilkunastu słów i zobacz, jak zostaje umieszczony na ekranie. Zakończ naciśnięciem klawisza *Enter*.

10.9.9. Przerywanie wykonywania programu

Do przerywania działania programu służą m.in. procedury HALT (zatrzymaj) i EXIT (wyjdź); pierwszej z nich już używałeś. Umieszczone w programie, powodują zawsze przerwanie wykonywania go. Różnica ich działania dotyczy przypadku, gdy program zawiera procedury i funkcje utworzone przez użytkownika.

Procedura HALT działa bez zmian, tzn. przerywa wykonywanie programu bez względu na miejsce, w jakim została wywołana. Procedura EXIT wywołana wewnątrz procedury lub funkcji utworzonej przez użytkownika powoduje zakończenie wykonywania jedynie danej procedury; wykonywanie programu jest kontynuowane od miejsca, w którym procedura ta została wywołana.

10.10. Tworzenie procedur i funkcji

System Turbo Pascal umożliwia tworzenie procedur i funkcji przez użytkownika. Do napisania programów realizujących algorytmy z rozdziału 8 tworzenie procedur nie jest niezbędne, w przypadku niektórych algorytmów jest jednak korzystne. Gdy osiągniesz pewną wprawę w ich tworzeniu, okaże się być może, że zyskałeś umiejętność mającą znaczenie nie tylko dla tworzenia programów w Turbo Pascalu, lecz także i dla Twojego sposobu myślenia o złożonych problemach z różnych dziedzin.

10.10.1. Pierwsza procedura

Fragment programu można wyodrębnić i nadać mu postać procedury. W programie fragment taki można wtedy zastąpić wywołaniem tej procedury. Jeżeli zastosować takie postępowanie do poszczególnych fragmentów programu realizujących poszczególne zadania cząstkowe, to program może się składać z wywołań kilku procedur. Wówczas staje się on przejrzystszy. Ponadto utworzone procedury można ewentualnie wykorzystywać w innych programach.

Procedura musi mieć nazwę. Definicja procedury zaczyna się od słowa PROCEDURE z nazwą procedury, po którym może wystąpić lista parametrów; dalsza część procedury ma taki sam układ jak program (deklaracje i definicje, a później część wykonawcza zawarta między słowami BEGIN i END), tyle że po końcowym słowie END stawia się średnik, a nie kropkę.

Zmienne zadeklarowane w programie głównym mogą być używane w procedurze bez ponownego deklarowania. Procedurę wywołuje się przez użycie w instrukcji jej nazwy (tak samo jak procedurę standardową). Definicje procedur umieszcza się po deklaracjach zmiennych, ale przed programem głównym.

Przypomnij sobie program TP-19. Występują w nim dwa identyczne fragmenty z pętlami FOR, służące do przedstawiania na ekranie zawartości całej tablicy *Porzadek*. Fragmenty te można zastąpić odwołaniem do procedury, tworząc jednocześnie odpowiednią procedurę. Uczynimy to w pro-