```
    ELSE
        czynnik% = czynnik% + 1
    END IF
LOOP
nww% = (m% / nwd%) * n%
PRINT "NWD liczb"; m%; " i "; n%; " = "; nwd%
PRINT "NWW liczb"; m%; " i "; n%; " = "; nww%
```

Możesz być zaskoczony sposobem zapisania wzoru na NWW. Czy nie prościej byłoby zapisać zgodnie ze wzorem z rozdz. 8 wyrażenia:

```
nww% = m% * n% / nwd%
```

Rzeczywiście byłoby to bardziej naturalne, zmiana została jednak podyktowana zakresem liczb całkowitych. Uruchom program dla danych 480 i 720. Czy otrzymałeś NWD = 240 i NWW = 1440? Zmień teraz w programie instrukcję obliczania NWW na podaną wyżej i uruchom program dla tych samych danych. Czy otrzymałeś wynik? Powinieneś otrzymać komunikat o powstaniu nadmiaru w instrukcji obliczania NWW. Nadmiar ten bierze się stąd, że po zmianie instrukcji najpierw jest obliczany iloczyn zmiennych m i n (wynoszący ponad 300 000, a więc więcej niż zakres liczb całkowitych), a dopiero później jest on dzielony przez NWD (równą 240).

Jeżeli jednak zechcesz posługiwać się programem QB-17 dla większych danych, to i tak może się zdarzyć, że wystąpi nadmiar. W takiej sytuacji zmień typ liczb całkowitych na *long-integer*. Możesz zmienić tylko typ zmiennej *nww*%, ale możesz zmienić również typ pozostałych zmiennych całkowitych (w programie QB-17l zamieszczonym w pliku QB17L.BAS został zmieniony typ wszystkich zmiennych).

Ćwiczenie 9-13

Napisz program realizujący algorytm 9 (Euklidesa) i porównaj otrzymywane wyniki z otrzymanymi za pomocą programu QB-17. Jeżeli różnica w czasie obliczeń jest zauważalna, to sprawdź, który jest szybszy.

Uwaga. Łatwiej zauważysz różnicę szybkości dla dużych danych, zatem dobierz odpowiedni typ zmiennych w obu programach.

9.7. Tablice

9.7.1. Deklarowanie tablic

Oprócz wymienionych zmiennych prostych możesz używać tablic składających się ze zmiennych wymienionych typów, musisz jednak je zadeklarować. Po słowie kluczowym DIM (*dimension* — rozmiar) musisz podać nazwę

tablicy oraz jej rozmiar w nawiasach, np.

```
DIM porzadek(8)
```

Nazwa tablicy (brak przyrostka po nazwie zmiennej *porzadek*) świadczy o tym, że jej elementami są liczby rzeczywiste pojedynczej precyzji. Liczba 8 w nawiasie oznacza, że elementy tablicy są numerowane od 0 do 8 (przy podaniu jednej liczby w deklaracji wymiaru dolna granica „numerów" jest przyjmowana jako 0).

Numer elementu tablicy jest nazywany w ogólności **indeksem**. Poszczególne elementy tablicy wskazywane są za pomocą wyrażeń o postaci *porzadek*(n), gdzie n jest liczbą całkowitą od 0 do 8. Wyrażenie *porzadek*(9) jest w tym przypadku niepoprawne. Odwołując się do elementu tablicy, zamiast konkretnej liczby (stałej) możesz użyć w tym miejscu zmiennej (*porzadek*$(i\%)$), a nawet wyrażenia arytmetycznego (wzoru) o wartościach całkowitych (*porzadek*$(i\% + j\%)$).

Deklarację tablicy należy umieszczać w programie przed użyciem wyrażeń odwołujących się do elementów tablicy, najlepiej na początku programu.

Możesz w deklaracji tablicy podać także dolną granicę indeksów. Najpierw podajesz dolną granicę, potem słowo TO (do) i górną granicę, np.

```
DIM odpowiedz$ (1 TO 40)
```

Nazwa tablicy świadczy, że służy ona do przechowywania 40 zmiennych tekstowych o „numerach" od 1 do 40.

Można deklarować kilka tablic jedną deklaracją DIM. Zapisy dotyczące poszczególnych tablic należy wówczas oddzielać przecinkiem. Deklaracja

```
DIM wiek%(28), nazwisko$(28)
```

określa dwie tablice jednowymiarowe:

- WIEK — złożoną z 29 elementów typu liczb całkowitych,
- NAZWISKO — złożoną z 29 elementów tekstowych.

9.7.2. Działania na elementach tablicy

Na elementach tablic można wykonywać takie same działania jak na zmiennych danego typu. Można więc poszczególnym elementom tablicy nadawać wartości, odczytywać je i używać w obliczeniach lub przy przetwarzaniu informacji, np.

```
Imiona(3) = "Joanna"
IF NumerOdp(7) = NumerPodany THEN PRINT "Odpowiedz prawidlowa"
```

Wartość indeksu można zmieniać w pętli. Jest to wygodne przy wykonywaniu działań na wielu lub wszystkich elementach tablicy; umożliwia na

przykład ich dodawanie, obliczanie średniej, wyszukiwanie elementów spełniających dany warunek, przesuwanie. Za pomocą podanego ciągu instrukcji możesz obliczyć sumę i wartość średnią liczb zapisanych w tablicy *Temperatura*.

```
Suma = .0
FOR i=1 TO 31
    Suma = Suma + Temperatura(i)
NEXT
SredniaTemp = Suma / 31;
```

Tablicę jednowymiarową możesz wyobrażać sobie równie dobrze jako tablicę kolumnową, jak i wierszową. Dla działania programu nie ma to znaczenia. W szczególności możesz elementy tablicy przedstawiać na ekranie obok siebie lub jeden nad drugim.

Program QB-18

```
REM program QB-18 - operacje na elementach tablicy
DIM porzadek(3)
CLS
REM odczytywanie wartosci elementow tablicy
FOR i% = 0 TO 3
    PRINT "Podaj "; i%; "element tablicy ";
    INPUT porzadek(i%)
NEXT i%
REM drukowanie tablicy
FOR i% = 0 TO 3
    PRINT porzadek(i%),
NEXT i%
PRINT
REM przesuwanie w lewo wyrazen w tablicy
FOR j% = 0 TO 2
   porzadek(j%) = porzadek(j% + 1)
NEXT j%
PRINT "Tablica po przesunieciu elementow w lewo"
REM drukowanie tablicy
FOR i% = 0 TO 3
    PRINT porzadek(i%),
NEXT i%
PRINT
```

Program QB-18 ilustruje użycie tablicy o nazwie *porzadek*. Jest ona zadeklarowana jako tablica czteroelementowa (od 0 do 3). Jej elementom są kolejno nadawane wartości odczytane z klawiatury (została do tego użyta pętla FOR). Następnie wszystkie wyrazy tablicy są przedstawiane na ekranie za

pomocą instrukcji PRINT powtarzanej w pętli FOR (dodatkowa instrukcja PRINT kończy wiersz). Następnie elementy tablicy są przesuwane w lewo; zauważ, że wartości indeksów są wyznaczane za pomocą wyrażeń zależnych od wartości licznika pętli $j\%$. Efekt zmian jest pokazywany na ekranie (fragment programu drukujący tablicę jest powtórzony).

Przesuwanie w lewo polega na tym, że najpierw zmiennej *porzadek*(0) jest nadawana wartość zmiennej *porzadek*(1), następnie zmiennej *porzadek*(1) — wartość *porzadek*(2) i na końcu zmiennej *porzadek*(2) — wartość *porzadek*(3); wartość prawego skrajnego elementu (*porzadek*(3)) pozostaje niezmieniona. Zauważ, że kolejność wykonywania tych operacji nie jest obojętna.

Przeanalizuj działanie programu i sprawdź, czy uzyskiwane wyniki są zgodne z Twoimi oczekiwaniami (pamiętasz, że w chwili uruchomienia programu wszystkie zmienne są równe 0). Zwróć też uwagę na sposób drukowania wyrazów tablicy.

Ćwiczenie 9-14

A. Spróbuj przekonać się o znaczeniu poszczególnych elementów programu mających wpływ na kształt wydruku ekranowego, modyfikując program w następujący sposób:
   - usuń samodzielną instrukcję PRINT;
   - usuń przecinek za instrukcją PRINT objętą pętlą.

B. Dopisz na początku programu instrukcję nadania elementowi *porzadek*(3) niezerowej wartości, np. 111. Sprawdź, w jaki sposób następuje przesuwanie elementów tablicy. Dopisz w następnej linii za `NEXT j%` instrukcję nadawania elementowi *porzadek*(3) wartości zerowej. Uruchom ponownie program i sprawdź, czy działa zgodnie z oczekiwaniami.

C. Dopisz na końcu programu jego fragment zawierający przesuwanie w lewo i drukowanie tablicy (9 ostatnich linii programu). Uruchom program i sprawdź, czy przesuwanie wyrażeń tablicy następuje prawidłowo.

D. Zmodyfikuj program w taki sposób, żeby liczba z lewego skrajnego elementu tablicy (*porzadek*(0)) nie była tracona przy przesunięciu w lewo, lecz powracała na prawy koniec tablicy (*porzadek*(3)). Tak realizowane przesuwanie elementów tablicy jest nazywane przesuwaniem cyklicznym.

E. Zmodyfikuj program w taki sposób, żeby przesuwanie wyrazów następowało w prawo. Zauważ, że musisz zmienić kolejność ich przepisywania.

Uwaga. Wyznaczanie wartości indeksów za pomocą wyrażeń obliczanych w programie jest potencjalnym źródłem błędów wynikających z tego, że obliczony indeks nie mieści się w przedziale wartości wynikającym z deklaracji tablicy. Z drugiej strony taki sposób określania indeksów jest podstawą działania wielu algorytmów. Zatem w razie trudności z uruchomieniem programu zwróć uwagę, czy przyczyną trudności nie jest przyjęcie przez indeks wartości spoza dopuszczalnego zakresu.

9.7.3. Tablice wielowymiarowe

Tablice mogą być wielowymiarowe; w deklaracji podaje się wtedy zakresy każdego z wymiarów tablicy, oddzielając je od siebie przecinkami. Na przykład deklaracja

```
DIM uklad$(1 TO 2, 1 TO 3)
```

określa tablicę dwuwymiarową uklad$, zawierającą 6 zmiennych łańcuchowych; elementy tej tablicy są identyfikowane jako:

uklad$(1,1), uklad$(1,2), uklad$(1,3),

uklad$(2,1), uklad$(2,2), uklad$(2,3).

Liczba indeksów nie jest bezpośrednio ograniczona. Ograniczone jest miejsce w pamięci, jakie może być zajęte przez zadeklarowaną tablicę, a to zależy zarówno od liczby „komórek" tablicy, jak i od typu zmiennych. Wielkość ograniczenia zależy od wersji systemu QBasic: w systemie towarzyszącym systemowi MS DOS 6.2 tablica nie może przekraczać 64 KB. Zakres dopuszczalnych wartości indeksów odpowiada zakresowi liczb całkowitych.

Do wykonywania działań na tablicach dwuwymiarowych wygodnie jest posłużyć się zagnieżdżonymi pętlami FOR. Napisz program QB-19 lub odczytaj go z pliku QB19.BAS.

Program QB-19

```
REM program QB-19 --- tablica dwuwymiarowa
CLS
DIM tabl(3, 4)
FOR i% = 1 TO 3
    FOR j% = 1 to 4
        tabl(i%, j%) = 2 * (i% - j%)
        PRINT tabl(i%, j%),
    NEXT
    PRINT
NEXT
```

Ćwiczenie 9-15

A. Rozważ, do czego służy instrukcja PRINT bez parametrów?
B. Dopisz do programu instrukcje drukowania wszystkich elementów tablicy po raz drugi w taki sposób, żeby tablica była przedstawiona jako czterowierszowa, trójkolumnowa.
C. Zmień zakres zmian jednego z indeksów w instrukcji FOR na 5 i uruchom program ponownie. Jeżeli zobaczysz komunikat: `Subscript out of range` (indeks spoza zakresu), to potwierdź jego przyjęcie (naciśnięciem klawisza *Enter*). Zmienionego programu nie zapisuj.

9.8. Funkcje standardowe

9.8.1. Funkcje matematyczne

Do obliczeń możesz wykorzystywać funkcje matematyczne dostępne w języku QBasic. Należą do nich takie funkcje, jak: **pierwiastek kwadratowy** (SQR) i **wartość bezwzględna** (ABS), **funkcje trygonometryczne** (SIN, COS i TAN), **logarytm naturalny** (LOG) i **funkcja wykładnicza** (EXP). Podobne funkcje spotkałeś już w arkuszu kalkulacyjnym.

Chcąc obliczyć wartość funkcji dla konkretnego argumentu, musisz ten argument napisać w nawiasach po nazwie funkcji, np. SQR(x). Argumentem funkcji może być wyrażenie, np. SIN(x+y). Wartości funkcji możesz używać w wyrażeniach, np. 0.5*(b-SQR(delta)). Możesz już dokończyć program realizujący algorytm 3 do zadania 8-1.

Ćwiczenie 9-16

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

Oprócz funkcji znanych Ci z matematyki w programie QBasic są dostępne funkcje związane z samym językiem programowania i z zamianą typów zmiennych. Należy do nich funkcja INT zamieniająca liczby rzeczywiste na całkowite przez zaokrąglenie w dół, na przykład INT(7.6) = 7 (podobne funkcje stosowałeś już w arkuszu kalkulacyjnym).

9.8.2. Funkcje losowe

Jest też w systemie QBasic funkcja RND generująca liczby losowe. Za każdym kolejnym jej użyciem ma ona inną wartość. Podobnymi funkcjami operowałeś w arkuszu kalkulacyjnym (LOS, RAND).

Program QB-20

```
REM Program QB-20 - Liczby losowe
RANDOMIZE TIMER
PRINT RND
PRINT RND
PRINT INT(RND * 10)
PRINT INT(RND * 10)
```

Druga linia programu (RANDOMIZE TIMER) służy do losowego inicjalizowania generatora liczb losowych. Jeżeli nie użyjesz podobnej linii, to za każdym uruchomieniem programu będzie generowany taki sam ciąg liczb.

Ćwiczenie 9-17

A. Uruchom program QB-20 i sprawdź, czy powtarzającym się w programie instrukcjom odpowiadają takie same liczby.
B. Uruchom kilkakrotnie program QB-20 i porównaj otrzymywane liczby, czy się powtarzają przy kolejnych uruchomieniach programu.
C. Usuń instrukcję RANDOMIZE TIMER (lub poprzedź ją słowem REM) i powtórz punkt poprzedni.

Możesz już napisać program QB-21 realizujący algorytm 12 do zadania 8-11.

Program QB-21

```
REM program QB-21 - test z tabliczki mnozenia (algorytm 12)
RANDOMIZE TIMER
licz% = 0
dobreodp% = 0
DO
    czynnik1% = 2 + INT(9 * RND)
    czynnik2% = 2 + INT(9 * RND)
    PRINT "Podaj wynik mnozenia"; czynnik1%; "przez";
                                                czynnik2%
    INPUT iloczyn%$
    IF czynnik1% * czynnik2% = iloczyn% THEN
       dobreodp% = dobreodp% + 1
       PRINT "Odpowiedz poprawna"
    ELSE
       PRINT "Odpowiedz bledna. Iloczyn jest rowny"; czynnik1%
                                                * czynnik2%
    END IF
    licz% = licz% + 1
    INPUT "Czy jeszcze jedno pytanie? (t/n)"; decyzja$
LOOP UNTIL decyzja$ = "n"
```

```
PRINT "Na"; licz%; "zadanych pytan"
PRINT "udzieliles"; dobreodp%; "prawidlowych odpowiedzi"
```

9.8.3. Działania i funkcje operujące na łańcuchach

Przedmiotem działań mogą także być teksty, zarówno stałe tekstowe, jak i zmienne. Podstawowym działaniem jest dodawanie, zaznaczane za pomocą tego samego symbolu, który oznacza dodawanie wyrażeń liczbowych. Dodawanie zmiennych i stałych łańcuchowych powoduje połączenie obu tekstów w jeden, bez zaznaczania przerwy. Na przykład "Ala" + "ma" jest równe "Alama", a nie "Ala ma".

Funkcja LEN(x$) (*length* — długość) podaje długość łańcucha zmiennej łańcuchowej lub wyrażenia, np. LEN("Beata")=5. Funkcję taką możesz wykorzystywać do kontroli długości danych wprowadzanych przez użytkownika.

Funkcje LEFT$ i RIGHT$ (lewy, prawy) powodują wycięcie z łańcucha będącego ich pierwszym argumentem, łańcucha o takiej długości, ile wynosi drugi argument (liczba całkowita), przy czym pierwsza z nich wycina od lewej, a druga od prawej strony. Wycięty łańcuch jest wartością funkcji. Na przykład LEFT$("ministerstwo", 8) = "minister".

Funkcja MID$ (środek) wycina z łańcucha będącego jej pierwszym argumentem, począwszy od miejsca wskazanego drugim argumentem, łańcuch o długości określonej trzecim argumentem, np. MID$("ministerstwo", 5, 4) = "ster". Drugi i trzeci argument jest liczbą całkowitą.

Ćwiczenie 9-18

Przeanalizuj działanie programu QB-22 dla przykładowych danych: "Beata" i "Krzysztof". Jak powinny się przedstawiać wyniki dla tych danych? Uruchom program (plik QB22.BAS) i sprawdź, czy wyniki są zgodne z przewidywaniami. W razie trudności posłuż się programem z pliku QB22BK.BAS, w którym podstawiono wymienione przykładowe dane.

Program QB-22

```
REM Program QB-22  -  dzialania na lancuchach
INPUT "Wprowadz pierwszy napis ( > 4 znaki) "; Napis1$
INPUT "Wprowadz  drugi  napis  ( > 4 znaki) "; Napis1$
SumaNapisow$ = Napis1$ + Napis2$
PRINT Napis1$, Napis2$, SumaNapisow$
PRINT LEN(Napis1$), LEN(Napis2$), LEN(SumaNapisow$)
PRINT LEFT$(SumaNapisow$, 4), RIGHT$(SumaNapisow$, 7),
PRINT MID$(SumaNapisow$, 4, 3)
```

Przy odczytywaniu odpowiedzi tekstowych użytkownika mogą być przydatne funkcje LCASE$ i UCASE$ zamieniające tekst dany na tekst złożony odpowiednio z samych małych bądź wielkich liter, np.

```
LCASE$(Tak) = "tak"
UCASE$(Tak) = "TAK"
```

Latwiej taki tekst porównywać z wzorcem. Na przykład jeżeli kontynuacja obliczeń w programie zależy od udzielenia przez użytkownika odpowiedzi: "t" (tak), bezpieczniej byłoby zastąpić warunek `decyzja$ = "t"` warunkiem `LCASE$(decyzja$) = "t"`. Program będzie wówczas działał tak samo niezależnie od tego, czy użytkownik wyrazi swoją decyzję kontynuacji obliczeń wprowadzeniem "t" czy "T", w przeciwnym razie na wprowadzenie "T" program reagowałby zakończeniem działania.

9.8.4. Funkcje zmieniające typ danych

Do ważnych funkcji tej grupy należą funkcje ASC i CHR$. Argumentem funkcji ASC jest łańcuch; funkcja podaje numer kodu ASCII jego pierwszego znaku, np. ASC("A")=65, ASC("Ala")=65. Argumentem funkcji CHR$ jest liczba całkowita z przedziału od 0 do 255; wartością funkcji jest znak o kodzie równym tej liczbie, np. CHR$(65)="A". Możesz za pomocą tej funkcji drukować na ekranie znaki o konkretnych kodach.

Program QB-23

```
REM Program QB-23 - drukowanie tabeli znakow o kolejnych kodach
kod = 32
CLS
DO
    PRINT CHR$(kod); " ";
    kod = kod + 1
    IF kod MOD 32 = 0 THEN PRINT
LOOP WHILE kod < 255
```

Ćwiczenie 9-19

Napisz program, który w odpowiedzi na podaną przez użytkownika liczbę n (≥ 32) wydrukuje na ekranie 10 znaków o kodach kolejnych począwszy od danego numeru n. Drukuj kolejne znaki w kolejnych liniach, drukując obok znaku także jego kod.

Dwie funkcje STR$ i VAL służą do zamiany liczb na teksty (łańcuchy) i odwrotnie. Funkcja STR$ przyporządkowuje liczbie (wartości wyrażenia) łańcuch złożony z takich znaków, jakie byłyby użyte do przedstawienia tej liczby na ekranie. Funkcja VAL (*value* — wartość) przyporządkowuje

łańcuchowi stanowiącemu zapis liczby liczbę o takiej samej wartości. Na przykład:

wartością funkcji STR$(18.2) jest łańcuch "18.2";
wartością funkcji VAL("18.2") jest liczba 18.2.

Jeżeli zgubiłeś się w tym trochę , to rozważ, że liczba określonego typu jest zapisywana w sposób właściwy dla systemu QBasic 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). Suma arytmetyczna dwóch liczb 15 i 3 wynosi 18, natomiast suma łańcuchów stanowiących zapisy tych liczb: "15" i "3", jest łańcuchem "153". Na przykład:

wartością wyrażenia VAL("15") + VAL("3") jest liczba 18;
wartością wyrażenia STR$(15) + STR$(3) jest łańcuch "153" — złożony z trzech znaków;
wartością wyrażenia STR$(3) + STR$(15) jest łańcuch "315" — złożony z trzech znaków.

Operacje takie mogą Ci się przydać przy operowaniu 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.

9.9. Tworzenie procedur i funkcji

System QBasic 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.

9.9.1. 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. Deklaracja procedury zaczyna się od linii ze słowem SUB, po którym podaje się nazwę procedury i w nawiasach zwykłych jej argumenty, oddzielone od siebie przecinkami. W następnych liniach są

instrukcje (i ewentualne deklaracje), a jako ostatnia jest linia zawierająca słowa END SUB. Procedurę wywołuje się instrukcją CALL z nazwą procedury i argumentami w nawiasach zwykłych.

Przypomnij sobie program QB-18. Występują w nim dwa identyczne fragmenty z pętlami FOR, służące do przedstawiania na ekranie zawartości całej tablicy. Fragmenty te można zastąpić odwołaniem do procedury, tworząc jednocześnie odpowiednią procedurę. Uczynimy to w odniesieniu do drugiego fragmentu. Procedurze nadamy nazwę DRUKWYNIKOW.

Uwaga. Program możesz odczytać z pliku QB24.BAS. Jeżeli chciałbyś go przygotować, to do zapisania procedury powinieneś posłużyć się poleceniem New SUB w oknie Edit. System otworzy wtedy nowe okno dla procedury, a nawet doda słowa kluczowe. Możesz też przygotować cały program z procedurą pod zwykłym edytorem. Odczytując program system umieszcza każdą procedurę i program główny w oddzielnych oknach; wybierasz je po naciśnięciu klawisza *F2*.

Program QB-24

```
REM program QB-24 - z procedura drukowania tablicy
DECLARE SUB DrukWynikow (dane())
DIM porzadek(3)
CLS
FOR i% = 0 TO 3
    PRINT "Podaj "; i%; " element tablicy ";
    INPUT porzadek(i%)
NEXT i%
CALL DrukWynikow(porzadek())
REM przesuwanie w lewo wyrazen w tablicy
FOR j% = 0 TO 2
   porzadek(j%) = porzadek(j% + 1)
NEXT j%
PRINT "Tablica po przesunieciu elementow w lewo"
CALL DrukWynikow(porzadek())
END

SUB DrukWynikow (dane())
FOR i% = 0 TO 3
    PRINT dane(i%),
NEXT
PRINT
END SUB
```

Zastąpiwszy pętle FOR służące do drukowania tablicy *porzadek* instrukcjami wywołania procedury (CALL DrukWynikow) musisz zakończyć program in-

strukcją END. Po niej umieszczasz definicję procedury *DrukWynikow*, zaczynająca się słowem SUB z nazwą procedury i kończąca linią END SUB.

Bezpośrednio po nazwie procedury (w tej samej linii) umieszczasz w nawiasie jej argumenty. Służą one do przenoszenia danych z programu głównego do procedury. Ogólnie biorąc, mogą także służyć do przenoszenia wyników z procedury do programu głównego.

Dla zaznaczenia, że argumentem procedury jest tablica, po nazwie tablicy umieszczasz parę nawiasów. Nazwą tablicy w definicji procedury nie musi być *porzadek*; użyjemy innej nazwy (*dane*). Podkreśla to uniwersalny charakter procedury, która może być stosowana do różnych tablic.

W treści procedury umieszczasz instrukcje przeniesione z programu, tj. pętlę FOR z instrukcją PRINT oraz („pustą") instrukcję PRINT umieszczoną za pętlą.

Uwaga. Instrukcją PRINT drukujesz element tablicy *dane*, tj. tej, która występuje w definicji procedury, a nie tej, którą naprawdę zamierzasz drukować (*porzadek*).

Procedura jest wywoływana instrukcją CALL. Po słowie CALL umieszczasz nazwę procedury i za nią w nawiasach nazwę tablicy *porzadek*, tj. tej, dla której procedura ma zostać wykonana.

Argumenty używane w wywołaniu procedury są nazywane **parametrami aktualnymi**, zaś te, które są użyte w definicji procedury — **parametrami formalnymi**.

Nowym elementem programu jest deklaracja DECLARE SUB z nazwą procedury i nazwą tablicy *dane* umieszczona na początku programu. Nie musisz jej pisać, ponieważ jeśli nie ma jej w programie, to utworzy ją system QBasic podczas zapisywania pliku (dopisując ją przed całym programem, a więc także przed komentarzami).

Jeżeli uruchomisz program QB-24, to przekonasz się, że działa tak samo jak program QB-18.

Ćwiczenie 9-20

A. Prześledź bieg programów QB-24 i QB-18, używając poleceń debugera. Porównaj.

B. Utwórz procedurę przesuwania w lewo elementów macierzy i zastąp odpowiedni fragment programu głównego wywołaniem tej procedury. Możesz porównać swój program z programem zapisanym w pliku QB24DWP.BAS.

C. Powtórz kilkakrotnie wywołanie na przemian procedur przesuwania w lewo i drukowania. Sprawdź, czy przesuwanie elementów tablicy przebiega prawidłowo.

D. Utwórz procedurę odczytywania elementów macierzy i zastąp początkowy fragment programu wywołaniem jej.

9.9.2. Przekazywanie wartości przez parametry

Procedura otrzymuje informacje o wartościach zmiennych za pośrednictwem parametrów. Gdyby procedura DRUKUJX, zawierająca tylko jedną instrukcję

```
PRINT x
```

była zadeklarowana bez podania parametrów, to „nie widziałaby" zmiennej x występującej w programie głównym.

Przeanalizuj działanie programu QB-PARAM1:

```
REM  Program QB-PARAM1
INPUT "Podaj liczbe "; x
PRINT "Druk w programie "; x
DrukujX
END

SUB DrukujX()
    PRINT "Druk w procedurze "; x
END SUB
```

Rezultatem wywołania procedury DRUKUJX w programie głównym, kiedy zmienna x miałaby w programie wartość różną od 0, np. 5, byłoby wydrukowanie 0 (program jest w pliku QBPARAM1.BAS).

Gdybyś zaś w definicji procedury umieścił w nawiasach za nazwą procedury zmienną x, i tak samo uczynił w instrukcji wywołania jej w programie głównym:

```
...
DrukujX (x)
END

SUB DrukujX (x)
...
```

to rezultat byłby prawidłowy, tzn. zarówno w programie, jak i w procedurze zostałaby wydrukowana ta sama wartość.

Nazwa zmiennej w procedurze może oczywiście być inna od nazwy zmiennej użytej w instrukcji jej wywołania. Możesz dokonać stosownej zmiany w programie QB-PARAM2.

Zmienna wymieniona w definicji procedury jako parametr służy nie tylko do przekazywania danych z programu do procedury, lecz także do zmieniania za pomocą instrukcji procedury wartości zmiennych w programie głównym. Możesz się o tym przekonać analizując program QB-PARAM3, zawierający procedurę DODAJELEMENT, która zwiększa wartość drugiego parametru o wartość pierwszego.

```
SUB DodajElement (Elem, SumaEl)
  SumaEl = SumaEl + Elem
END SUB
```

Nie zawsze jest pożądane, żeby procedura zmieniała wartości zmiennych w programie głównym. Jeżeli chcesz się przed tym zabezpieczyć, to w instrukcji wywołania procedury musisz daną zmienną objąć nawiasami.

```
CALL DodajElement(Element, Suma)
CALL DodajElement(Element, (Suma))
```

Ćwiczenie 9-21

A. Przeanalizuj działanie programu QB-PARAM3 ze względu na różnicę między skutkami obu wywołań procedury. Porównaj swoje przewidywania z uzyskanymi rezultatami dla przykładowych danych.
B. Jaki byłby skutek objęcia nawiasami parametru *Element* zamiast parametru *Suma*? Sprawdź i porównaj przewidywania z uzyskanymi rezultatami.

9.9.3. Użycie programu jako procedury w innym programie

W programie do zadania 8-7 opartym na algorytmie 10 wygodnie jest posłużyć się procedurą wyznaczającą NWD i NWW dla danej pary liczb. Można ją otrzymać modyfikując (nieznacznie) program QB-17. Cały program mógłby mieć postać taką jak QB-25 (ewentualnie liczby całkowite mogą być długie, jeżeli zmieniłeś ich typ w programie QB-17).

Program QB-25

```
REM Program QB-25  dodawanie ulamkow (algorytm 10)
REM dodawania dwoch ulamkow
DECLARE SUB nwwd (n%, m%, nwd%, nww%)
CLS
INPUT "Podaj licznik i mianownik pierwszego ulamka "; l1%, m1%
INPUT "Podaj licznik i mianownik drugiego ulamka "; l2%, m2%
CALL nwwd(m1%, m2%, nwdm%, nwwm%)
PRINT nwdm%, nwwm%
```

```
nl1% = l1% * (m2% / nwdm%)
nl2% = l2% * (m1% / nwdm%)
nl% = nl1% + nl2%
CALL nwwd(nl%, nwwm%, nwd2%, nwm2%)
PRINT "Licznik ="; nl% / nwd2%, "Mianownik ="; nwwm% / nwd2%
END

SUB nwwd (n%, m%, nwd%, nww%)
REM Procedura  - wyznaczanie NWD i NWW dwoch liczb naturalnych
REM powstala z programu QB-17
czynnik% = 2
ilorazm% = m%
ilorazn% = n%
nwd% = 1
DO UNTIL ilorazm% = 1 OR ilorazn% = 1 OR czynnik% > m% OR
                                          czynnik% > n%
    IF ilorazm% MOD czynnik% = 0 AND ilorazn% MOD czynnik%
                                                = 0 THEN
        ilorazm% = ilorazm% \ czynnik%
        ilorazn% = ilorazn% \ czynnik%
        nwd% = nwd% * czynnik%
    ELSE
        czynnik% = czynnik% + 1
    END IF
LOOP
nww% = (m% / nwd%) * n%
END SUB
```

Treść procedury NWWD różni się od programu QB-17 pominięciem instrukcji odczytywania danych z klawiatury (INPUT) i drukowania wyników na ekranie (PRINT, a także CLS). Procedura otrzymuje dane z programu głównego i jemu przekazuje wyniki. Właśnie do tego celu służą argumenty (parametry) procedury.

W programie głównym procedura jest wywoływana dwukrotnie. Za każdym razem jako dwa pierwsze argumenty w instrukcji wywołania są podane nazwy zmiennych mających stanowić dane, a jako trzeci i czwarty argument — nazwy zmiennych, które mają zawierać wyniki. Podobnie jak poprzednio, nazwy tych zmiennych różnią się od nazw użytych w definicji procedury.

9.9.4. Czy stosowanie procedur jest niezbędne

Zauważ, że do utworzenia programu realizującego algorytm 10 nie jest konieczne posłużenie się procedurą, ponieważ można w tych miejscach, w których następuje wywołanie procedury, napisać taki sam fragment programu, jaki

jest zawarty w procedurze, a pominąć definicję procedury (oraz linię deklaracji procedury dodawaną przez system na początku programu).

Trzeba wówczas zastąpić nazwy tych zmiennych, które służą do napisania definicji procedury, nazwami takich zmiennych, na jakich faktycznie obliczenia mają być wykonywane. Są one wymienione w instrukcji wywołania procedury.

Zatem w miejscu, gdzie (pierwsze) wywołanie procedury ma postać: `CALL nwwd(m1%, m2%, nwdm%, nwwm%)`, podczas zastępowania tej instrukcji fragmentem programu zawartym w treści procedury zamiast zmiennej $n\%$ należałoby użyć $m1\%$, zamiast m — $m2$, zamiast $nwd\%$ — $nwdm\%$ i zamiast $nww\%$ — $nwwm\%$. (Jakich nazw użyłbyś w miejscu drugiego wywołania procedury?)

Jeżeli czujesz, że masz trudności z uchwyceniem sensu stosowania procedur, wykonaj poniższe ćwiczenie.

Ćwiczenie 9-22

A. Sporządź wersję programu QB-25, w której zastąpisz obie instrukcje wywołania procedury jej treścią (zapisz ją pod nazwą na przykład QB25BP.BAS). W razie trudności posłuż się programem zapisanym w pliku QB25BPRO.BAS.

B. Porównaj działanie obu wersji programu na tych samych danych, śledząc przebieg obliczeń instrukcja po instrukcji.

Stosowanie procedur nie jest niezbędne, zwłaszcza w niewielkich programach (programy zamieszczone w dalszej części książki często z nich nie korzystają). Jednak nawet w niewielkich programach stosowanie ich może być korzystne. Wiesz już, że program może być mniejszy i łatwiejszy do uruchomienia. Procedury opracowane dla jednych programów możesz wykorzystywać w innych swoich programach; możesz także się nimi dzielić. Rozbicie programu na oddzielne procedury ułatwia podzielenie się pomiędzy różne osoby pracą nad jego tworzeniem.

Tworzenie dużego programu nie korzystającego z procedur jest bardzo trudne, a niekiedy nawet niemożliwe. Jeżeli zainteresowałeś się programowaniem na tyle, że zamierzasz kontynuować naukę pisania programów za pomocą języka wysokiego poziomu, to staraj się przyswoić sobie wiadomości z zakresu stosowania procedur (a także omówionych dalej funkcji). Stoi przed tobą otworem dziedzina określana jako **programowanie strukturalne** (chodzi tu o nadawanie programom wyraźnej struktury przez używanie 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.

9.9.5. Funkcje

Może się zdarzyć, że potrzebowałbyś użyć w obliczeniach funkcji, jakich nie ma w systemie QBasic. Otóż możesz samemu zdefiniować nowe funkcje i stosować je w swoich programach. Argumentem tych funkcji mogą być zmienne i wyrażenia.

Jeżeli zdefiniujesz funkcję o nazwie DROGA mającą jeden argument (t), a w programie używasz zmiennej o nazwie *czas*, to we wzorach możesz pisać DROGA(czas), DROGA(czas−2), DROGA(2*czas) itd. — jednym słowem czynić to samo, co z funkcjami dostępnymi w systemie. Gdybyś np. miał używać w różnych miejscach programu wyrażeń o takiej samej postaci, lub fragmentów służących do obliczania jednego wyniku, to mógłbyś zdefiniować własną funkcję.

Funkcja jest traktowana jako odmiana procedury i podobnie jest definiowana (w oknie Edit wybierasz pozycję New FUNCTION). Różnica dotyczy sposobu przekazywania danych pomiędzy funkcją a programem głównym. Argumenty procedury mogą służyć do przekazywania informacji w obie strony (na przykład dwa pierwsze argumenty procedury NWW służą do przekazania danych z programu głównego do procedury, zaś trzeci i czwarty, do przekazania wyników z procedury do programu głównego), natomiast argumenty funkcji są przeznaczone do przekazywania danych z programu do funkcji. Do przekazywania wyników obliczeń służy nazwa funkcji.

Wśród instrukcji występujących w treści definiowanej procedury musi wystąpić instrukcja przypisania wartości nazwie procedury (w procedurze o nazwie DROGA będzie to instrukcja `DROGA=wyrażenie`). Zauważ, że podobnej instrukcji w programie głównym używać nie wolno. Nie powinno Cię to zdziwić, nie nadawałbyś przecież wartości funkcji *sinus*($\pi/4$), lecz używał tej wartości do obliczania innych wyrażeń.

Gdybyś np. w wielu miejscach swego programu podnosił do kwadratu zmienne o różnych (może długich) nazwach, to mógłbyś zdefiniować funkcję KWADR służącą do obliczania kwadratu argumentu:

```
FUNCTION kwadr (x)
    kwadr = x * x
END FUNCTION
```

Słowa `FUNCTION kwadr()` (po podaniu przez Ciebie nazwy funkcji) oraz

linię końcową tworzy system, Tobie pozostałoby zatem napisanie argumentu x w nawiasach oraz w następnej linii wyrażenia przypisującego wartość ($kwadr = x * x$). W Twoim programie mógłbyś wówczas po prostu używać tej funkcji na równi z innymi funkcjami.

Podczas zapisywania pliku system dopisałby na początku programu w osobnej linii deklarację funkcji (podobnie jak dopisywał deklaracje procedur).

Jeżeli chcesz się przekonać, że możesz używać tak zdefiniowanej funkcji, to napisz prosty program, np. obliczający wartości kwadratów dla kilku wyrażeń.

Program QB-26

```
DECLARE FUNCTION kwadr! (x!)
REM Program QB-26 - obliczanie kwadratow liczb nieparzystych
REM za pomoca funkcji KWADR
CLS
FOR i% = 1 TO 20
    PRINT i%, 2 * i% - 1, kwadr(2 * i% - 1)
NEXT
END

FUNCTION kwadr (x)
    kwadr = x * x
END FUNCTION
```

Ćwiczenie 9-23

A. Zmodyfikuj program QB-26 zastępując funkcję kwadratową następującymi funkcjami:

   - x^3
   - $\frac{x}{1+0.01x^2}$

   i zmieniając ewentualnie także argument funkcji (wyrażany w zależności od zmiennej i%).

B. Sporządź wersję programu QB-25, w której zastąpisz procedurę obliczania NWD i NWW funkcją o wartości równej NWD, zmieniając odpowiednio instrukcje wywołania procedury (zapisz ją pod nazwą QB25F.BAS).

   Wartość NWW obliczaj wówczas w programie głównym. W razie trudności posłuż się programem zapisanym w pliku QB25FUN.BAS (lub zapisanym w pliku QB25FN.BAS programem z funkcją zadeklarowaną w trochę inny sposób, spotykany w niektórych książkach o systemie QBasic).