Błąd w obliczeniach jest związany z przekroczeniem zakresu zmiennych zadeklarowanych jako całkowite (*integer*). 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). Jego powstanie jest związane z przebiegiem obliczeń, a nie z wynikiem, który mieści się w zakresie dopuszczalnym. Zauważ, że system nie sygnalizuje takiego błędu.

Ustrzeżesz się przed nim zmieniając typ przynajmniej jednej ze zmiennych m i n na całkowitą długą (*longint*). Jeżeli jednak będziesz posługiwał się programem TP-18 dla większych danych, to i tak może się zdarzyć, że wystąpi nadmiar. W takiej sytuacji zmień typ wszystkich zmiennych całkowitych na *longint* (zapisz program pod zmienioną nazwą).

Zmiana zakresu liczb nie rozwiązuje zupełnie problemu, lecz jedynie go przesuwa w stronę większych liczb. Możesz się o tym przekonać podając takie dane, których największa wspólna wielokrotność przekroczy zakres liczb całkowitych długich. Podaj np. liczby, z których jedna jest odpowiednio dużą potęgą dwóch, a druga trzech (wówczas NWW jest ich iloczynem, ponieważ nie mają wspólnego dzielnika większego od 1): 4096 i 6561, 16384 i 19683, 65536 i 59049. Czy dla ostatniej pary liczb otrzymujesz wynik prawidłowy?

Jak możesz ustrzec się przed błędem? Na przykład ogranicz zakres danych i ewentualnie sprawdzaj w programie, czy nie został przekroczony.

Ćwiczenie 10-16

Napisz program realizujący algorytm 9 (Euklidesa) i porównaj uzyskane wyniki z otrzymanymi za pomocą programu TP-18. Jeżeli różnica w czasie obliczeń jest zauważalna, to sprawdź, który jest szybszy. (Łatwiej zauważysz różnicę szybkości dla dużych danych, zatem dobierz odpowiedni typ zmiennych w obu programach.)

10.8. Tablice, rekordy, definiowane typy zmiennych

10.8.1. Tablice

Deklarowanie tablic

Tablice deklaruje się razem z innymi zmiennymi, określając typ danej zmiennej, będącej nazwą tablicy, za pomocą słowa ARRAY, po którym podaje się zakres **indeksu** lub indeksów w nawiasach kwadratowych, a następnie typ wielkości przechowywanych w tablicy. Zakres indeksu określa się

podając liczby całkowite (lub stałe), stanowiące dolną i górną jego wartość, oddzielone dwiema kropkami. Wartości te nie muszą być dodatnie (mogą także być ujemne lub równe zero). W kilkuwymiarowej tablicy zakresy poszczególnych jej indeksów oddziela się przecinkami. Przykład:

```
var Temperatura:    array[1..31] of real;
    MapaFunkcji:    array[-10..10, -10..10] of real;
    Imiona:         array[1..8] of string;
    NumerOdp:       array[1..40] of integer;
```

Poszczególne elementy tablicy są wskazywane za pomocą wyrażeń składających się z nazwy tablicy i umieszczonego w nawiasach kwadratowych wyrażenia określającego wartość indeksu. Gdy mamy kilka indeksów, wówczas poszczególne wyrażenia oddziela się przecinkami. Wartość każdego z takich wyrażeń musi być liczbą całkowitą (samo wyrażenie może być stałą lub zmienną, wyrażeniem arytmetycznym, a nawet złożonym wyrażeniem zawierającym funkcje), np.:

$Temperatura[11]$, $MapaFunkcji[-3,7]$, $Imiona[n+1]$, $NumerOdp[1+Trunc(40 * Random)]$.

W systemie Turbo Pascal liczba indeksów nie jest bezpośrednio ograniczona. Ograniczone jest (do 64 KB) miejsce w pamięci, jakie może być zajęte przez zadeklarowane zmienne łącznie z tablicami, a to zależy zarówno od liczby „komórek" tablicy, jak i od typu zmiennych.

Wartości indeksów powinny mieścić się w zadeklarowanych granicach zakresów (np. wyrażenie $Imiona[9]$ jest niepoprawne). Niespełnienie tego warunku prowadzi do błędów wykonania, które nie zawsze muszą być sygnalizowane (zależy to od ustawienia parametrów kompilacji).

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 przetwarzaniu informacji, np.

```
Imiona[3] := 'Joanna';
if NumerOdp[7] = NumerPodany then WriteLn('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* (zmienne *Suma* i *SredniaTemp* muszą być odpowiedniego typu; jakiego?).

```
Suma:= 0.0;
for i:=1 to 31 do Suma := Suma + Temperatura[i];
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 (czy wiesz jak to uczynić?).

Porządkowanie elementów tablicy

Zadanie 8-4 dotyczy uporządkowania n odczytanych liczb. Liczby takie wygodnie jest zapisać w tablicy.

Dla zadania porządkowania elementów nie jest zresztą istotne, czy są to liczby, czy np. imiona, byle dla każdych dwóch różnych elementów można było wskazać, który z nich ma być wcześniejszy. Możesz wyobrażać sobie, że masz ułożyć świadectwa szkolne według nazwisk.

Porządkowanie elementów ułożonych w pewnej (przypadkowej) kolejności wiąże się z ich przekładaniem. Po przełożeniu elementu w inne miejsce zmienia się kolejność pozostałych elementów (niektórych). Spróbujemy obecnie zrealizować jedno cykliczne przesunięcie elementów tablicy „w lewo" (przyjmujemy, że elementy tablicy są ułożone poziomo), z przeniesieniem elementu pierwszego od lewej na prawą stronę tablicy. Potraktuj to jako ćwiczenie przed napisaniem programu realizującego całe zadanie. Jako danych użyjemy liczb wprowadzonych przez użytkownika.

Program TP-19

```
program TP19;
 {tablice -- przesuwanie elementow}
var
  i,j, ElementSkrajny: integer;
  Porzadek: array[1..6] of integer;
begin
  WriteLn('Wprowadz 6 liczb calkowitych (po kazdej nacisnij
                                                    ENTER)');
  for i := 1 to 6 do ReadLn(Porzadek[i]);          {czytanie}
  for i := 1 to 6 do Write(Porzadek[i]:10);        {drukowanie}
  WriteLn;
  ElementSkrajny := Porzadek[1];
  for j := 1 to 5 do Porzadek[j] := Porzadek[j + 1];
                                                   {przesuwanie}
  Porzadek[6] := ElementSkrajny;
  for i := 1 to 6 do Write(Porzadek[i]:10);        {drukowanie}
```

```
    WriteLn;
    Readln
end.
```

Program TP-19 ilustruje użycie tablicy o nazwie *Porzadek*. Jest ona zadeklarowana jako tablica sześcioelementowa (od 1 do 6). Na początku jej elementom są nadawane wartości odczytane z klawiatury (dla przyspieszenia działania programu możesz te wartości nadawać w programie).

Następnie wszystkie wyrazy tablicy są przedstawiane na ekranie za pomocą procedury WRITE wywoływanej w pętli FOR. Na każdą z liczb jest przeznaczone po 10 znaków, co umożliwia przedstawienie całej tablicy w jednym wierszu. Dodatkowa instrukcja wywołania procedury WRITELN powoduje zakończenie wiersza.

Następnie jest zapamiętywany „lewy" element skrajny, po czym elementy tablicy są przesuwane w lewo. Zauważ, że wartości indeksów są wyznaczane za pomocą wyrażeń zależnych od wartości licznika pętli j. Na koniec zapamiętany element jest wstawiany na miejsce po prawej stronie. Efekt zmian jest pokazywany na ekranie (fragment programu drukujący tablicę jest powtórzony).

Przesuwanie w lewo polega na tym, że najpierw element *Porzadek*[1] zostaje zapamiętany poza tablicą, potem elementowi *Porzadek*[1] jest nadawana wartość elementu *Porzadek*[2], następnie elementowi *Porzadek*[2] — wartość *Porzadek*[3] itd.; na końcu elementowi *Porzadek*[5] — wartość *Porzadek*[6], a elementowi *Porzadek*[6] wartość *Porzadek*[1], przechowana w zmiennej pomocniczej *ElementSkrajny*. Zauważ, że kolejność wykonywania tych operacji nie jest obojętna.

Przeanalizuj działanie programu i sprawdź, czy uzyskiwane wyniki są zgodne z Twoimi oczekiwaniami. Zwróć też uwagę na sposób drukowania wyrazów tablicy.

Ćwiczenie 10-17

A. Spróbuj przekonać się o znaczeniu użycia instrukcji WRITELN bez parametrów, usuwając je (np. obejmując nawiasami klamrowymi).
B. Powtórz na końcu programu jego fragment zawierający przesuwanie w lewo i drukowanie tablicy. Uruchom program i sprawdź, czy przesuwanie wyrażeń tablicy następuje prawidłowo.
C. Zmodyfikuj program w taki sposób, żeby przesuwanie elementów dotyczyło tylko miejsc tablicy od 2 do 4 (a nie wszystkich sześciu).
D. Zmodyfikuj program w taki sposób, żeby przesuwanie elementów następowało w prawo. Zauważ, że musisz zmienić kolejność ich przepisywania.

Kontrola indeksów

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 zadeklarowanym w 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. System kontroluje wartość indeksów, jeżeli w opcjach kompilatora pozycja Range checking jest uaktywniona; w razie stwierdzenia błędu zakresu podczas wykonywania programu wypisuje komunikat:

```
Error 201:  Range Check Error.
```

Tablice dwuwymiarowe

Do wykonywania działań na tablicach dwuwymiarowych wygodnie jest posłużyć się zagnieżdżonymi pętlami FOR. Napisz program TP-20 lub odczytaj go z pliku TP20.PAS.

Program TP-20

```
program TP20;
 {tablica dwuwymiarowa}
var
  i,j: integer;
  tabl: array[1..8,1..15] of integer;
begin
  for i := 1 to 8 do
    for j := 1 to 15 do
      tabl[i, j] := 2 * (i - j);
  for i := 1 to 8 do
    begin
      for j := 1 to 15 do
        Write (tabl[i, j]:5);
      WriteLn
    end;
  ReadLn
end.
```

Ćwiczenie 10-18

A. Dopisz do programu instrukcje drukowania wszystkich elementów tablicy po raz drugi w taki sposób, żeby tablica była przedstawiona jako piętnastowierszowa, ośmiokolumnowa.

B. Zmień zakres zmian jednego z indeksów w instrukcji FOR na 16 i uruchom program ponownie. Czy program się wykonał, czy też wykazał błąd wykonania? Możesz sprawdzić i ewentualnie (za zgodą nauczyciela) zmienić opcję kompilatora odpowiedzialną za sprawdzanie indeksów, i ponownie wykonać program. Zmienionego programu nie zapisuj.

10.8.2. Definiowanie typów

Do definiowania typu zmiennych, jakich zamierzasz używać w programie, służy słowo TYPE. Definicja typów musi być podana przed deklaracją zmiennych.

Na przykład możesz zdefiniować typ w postaci łańcucha o określonej długości

```
type s8 = string[8]
```

i potem używać go w deklaracjach zmiennych, tak jak w przedstawionym programie TP-21.

Program TP-21

```
program TP21;
type lancuch8 = string[8];
var  aa, bb, cc: lancuch8;
begin
  aa:= 'Grzegorz';
  bb:= 'TP-22';
  cc:= 'Nauczyciel';
  WriteLn (aa,'  ',bb,'  ',cc);
  ReadLn
end.
```

Wykonaj program. Czy w zmiennej *cc* zmieściło się całe słowo „Nauczyciel", czy tylko część? Jaka? Ile miejsca na ekranie zajęła zmienna *bb*, 5 czy 8?

W programie TP-21 definiowanie typu *s8* nie jest konieczne, ponieważ wystarczyłoby zrobić to w deklaracji zmiennych,

```
var aa, bb, cc:  string[8];
```

użycie definicji typu bywa jednak czasem niezbędne, np. gdy zmienne zdefiniowanego typu mają być elementami tablicy.

```
type
   lancuch8: string[8];
var
   NazwyPlikowDyskowych: array[1..500] of lancuch8;
```

10.8.3. Rekordy

Rekord, jak wiesz z rozdziału dotyczącego baz danych, składa się z pól, przy czym każde z pól jest charakteryzowane długością i rodzajem danych. W systemie Turbo Pascal możesz tworzyć rekordy, a zatem i bazy danych. Żeby operować na rekordach musisz mieć dostęp zarówno do całych rekordów, jak i do poszczególnych pól. Z tego względu w definicji rekordu musisz zawrzeć wszystkie te informacje.

Przykładowa definicja typu rekordowego ma postać:

```
type ksiazka = record
                 Imie_Autora       :string[10];
                 Nazwisko_Autora   :string[15];
                 Tytul_Ksiazki     :string[35];
                 Sygnatura         :integer
               end;
```

Jeśli zadeklarujesz zmienną typu *ksiazka* o nazwie np. *biblioteka*, to dostęp do poszczególnych pól rekordu *biblioteka* uzyskasz używając nazwy rekordu połączonej znakiem kropki z nazwą pola, np. `Biblioteka.Nazwisko_Autora`.

Program TP-22

```
program TP22;
  {Rekordy}
uses Crt;
type ksiazka = record
                 Imie_Autora       :string[10];
                 Nazwisko_Autora   :string[15];
                 Tytul_Ksiazki     :string[35];
                 Sygnatura         :integer
               end;
var  i: integer;
     Biblioteka: array[1..200] of ksiazka;
begin
  ClrScr;
  Biblioteka[1].Imie_Autora := 'Adam';
  Biblioteka[1].Nazwisko_Autora := 'Mickiewicz';
  Biblioteka[1].Tytul_Ksiazki := 'Pan Tadeusz';
  Biblioteka[1].Sygnatura := 253;
  WriteLn (Biblioteka[1].Imie_Autora);
  WriteLn (Biblioteka[1].Nazwisko_Autora);
  WriteLn (Biblioteka[1].Tytul_Ksiazki);
  WriteLn (Biblioteka[1].Sygnatura);
  ReadLn
end.
```

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.