7. МАСИВИ

7.1. Едномерни масиви

7.1.1. Определение

Често се случва да се обработва съвкупност от данни, които са сродни по смисъл и имат стойности от един и същи тип, например оценките на студентите в една група по дадена дисциплина, факултетните номера на студентите в една група, месечните заплати на работещите в едно предприятие и т.н. Удобно е отделните компоненти на такава съвкупност да се номерират и тогава такава съвкупност от данни представлява даннова структура, наречена *променлива от тип масив* или накратко *масив*, а номерираните компоненти от съвкупността са наричат елементи на масива. Очевидно масивът е съставен тип данни и мястото на всеки елемент се определя от неговия номер, наречен индекс. Стойността на индекса може да принадлежи на всеки дискретен тип, т.е. допустимите за индекса стойности винаги представляват нарастваща последователност.

Широко използваните в математиката вектори по естествен начин се описват като едномерни масиви.

7.1.2. Дефиниране на тип масив

Тип масив се дефинира в раздела **Type** или анонимно.
Описанието на типа масив има вида

***array** [Тип на индекса] **of** Тип на елементите*

Примери:

```
Const
  Max= 100;
Type
  Index     = 0..10;
  Glavna   =  'A'..'Z';
  RealMas          = array [Index] of real;
  IntMas    = array [Glavna] of integer;
  ChMas    = array [1..45] of char;
  StrMas    = array [-Max..Max] of string[10];
```

Очевидно типът на индекса определя допустимите стойности на индекса, а от там и броя на елементите на масива, т.е. неговата дължина.

7.1.3. Константи от тип масив

Последните версии на Турбо Паскал допускат дефинирането на константи от тип масив. Например

```
Const
  Mas1: RealMas=(1.2, 2.25, -3.75, 4.5, 6, 10, -9.5, 0, 15, 25) ;
  Mas2: array[1..10] of char= ('A','B','C','D','E','G','H','K','L','M');
```

Типът RealMas на масива Mas1 е дефиниран в раздела Type по-горе, докато типът на масива Mas2 е анонимен, т.е. той е описан при самото дефиниране на масива и не му е дадено име.

Както е показано чрез примера, стойностите на елементите на масива-константа се задават в скоби. Задължително се зададат толкова стойности, колкото е броят на елементите на масива по дефиниция. Зададените стойности се явяват само начални стойности на елементите. В процеса на изпълнение на

програмата те могат да бъдат променяни, но при следващо зареждане и стартиране на програмата се възстановяват началните стойности.

7.1.4. Променливи от тип масив

Те се декларират в раздела **Var**, както следва:

```
Var
  A,B:IntMas;
  C:RealMas; D:ChMas; E:StrMas;
  F:array [Glavna] of boolean;
  G:array [Index] of 0..255;
  H:array [char] of Glawna;
  P:array [1..15] of 101..200;
```

Както се вижда, при дефинирането на масивите A,B,C,D и E е посочено името на вече дефиниран тип масив, докато при дефинирането на масивите F,G,H и P, типът е описан на място, т.е. масивите F,G,H и P са от анонимен тип.

7.1.5. Операции с масиви

7.1.5.1. Операции с константи и променливи от тип масив

Единствената разрешена операция с данните от този тип е присвояването, т.е. разрешена е операцията

A:=B,

където A и B са променливи или именувани константи от един и същи тип масив

7.1.5.2. Операции с елементи на масиви. Индекси

За елементите на даден масив са разрешени всички допустими операции за техния тип. Елементите на масивите A и B например са от тип **integer**. Следователно те могат да участват във всички операции, разрешени за данните от този тип.

Всеки елемент на масив е определен чрез името на масива (променливата от тип масив) и неговия индекс, т.е. елементът на едномерния масив има съставно име, което има вида:

Име на масив[Индекс],

където *Индекс* трябва да бъде израз от дискретен тип (нека да припомним, че константата, променливата и елементът на масив са също изрази), който съвпада с типа на индекса, посочен при дефинирането на масива.

Например, при декларации:

```
Var
  i,j,k:integer;
  X:real;
  A,B:array [1..100] of real;
  C:array [char] of integer;
  E:array [1..10] of char;
```

са допустими операторите:

```
i:=12; j:=35; k:=6;
A[12]:=12.5;        {На 12-ия елемент от масива A се присвоява 12.5}
B[sqr(K) -1]:=-2.4; {На 35-ия елемент от масива B се присвоява 2.4}
X:=A[i]+B[j];       {На X се присвоява сумата A[12] + B[35]}
C['w']:=5;          {На елемента с индекс w от масива C се присвоява 5}
E[5]:='Q';   {На 5-ия елемент от масива E се присвоява символната с-ст Q}
```

7.1.5.3. Въвеждане и извеждане на масиви

Нека да си припомним, че с операторите Read, Readln от клавиатурата могат да се въвеждат само стойностите на променливи от реален, целочислен и символен тип и тип символен низ, с операторите Write и Writeln могат да се извеждат на екрана стойностите на изрази от реален, целочислен, логически и символен тип и тип символен низ. Следователно имената на променливи от съставен тип (в това число и от тип масив) не могат да присъстват във входните списъци на операторите Read, Readln и изходните списъци на операторите Write и Writeln. Поради това масивите се въвеждат и извеждат елемент по елемент, като за целта се организират цикли. Когато елементите на масив са от тип, който също не може да се въвежда с операторите Read, Readln, Write и Writeln (логически или някакъв изброен тип), те се въвеждат чрез някакви целочислени или символни кодове.

Примери:

```
Const
    m=50; n=10;
Var
    A:array[1..m] of integer;
    B:array[1..n] of boolean;
    i,Kod:integer;
. . . . . .
    For i:=1 to m do
      begin
        Write(‘ Въведете ‘,i,’ - ия елемент:’); Readln(A[i])
      end;
. . . . . .
    For i:=1 to n do
      begin
        Write(‘ Въведете кода на ‘,i,’-ия елемент (0-false, 1-true): ’); Readln(Kod);
    If Kod =0
        then B[i]:=false
        else  B[i]:=true;
      end;
. . . . . .
```

Вместо оператора **If** може да се използва операторът **Case**.

Програма 7.1. Програма за намиране средния успех на студентите от една група по дадена дисциплина.

Подобна програма беше разгледана в пример 4.8 (глава 4). Там обаче оценките се въвеждаха една по една и в паметта във всеки момент присъстваше само една оценка (Защо?). Тук искаме да направим програмата така, че по време на изпълнението ѝ в паметта на компютъра да присъстват всички оценки. Това ще даде възможност за последващо разширяване на програмата с други обработки на същите оценки.

Програмата може да има следния вид:

```
Const
  MaxBrStud= 30;
Var
  n, BrStud: 1.. MaxBrStud;
  Suma: integer;
  Ocenki: array [1.. MaxBrStud] of 2..6;
Begin
  Write ('Задайте броя на студентите: '); Readln (BrStud);
```

```
  For n := 1 to BrStud do
  Begin
    Write ('Оценка на студент номер ',n,': '); Readln (Ocenki[n])
  end;
  Suma := 0;
  For n := 1 to BrStud do Suma := Suma + Ocenki[n];
  Writeln ('Среден успех:',Suma/BrStud :5:2)
End.
```

В горния пример, както при въвеждането на оценките, така и при намирането на сумата им, управляващата променлива на цикъла n се използува и като индекс на поредния елемент на масива Ocenki. Това е типично в много случаи при работа с масиви. В нашия пример едни и същи действия се извършват с всеки елемент на масива от първия до последния, т.е. последователността на обхождане на елементите е възходяща по отношение на техния индекс. Понякога смисълът на поставената задача изисква низходящо обхождане на масива, т.е. от последния му елемент до първия. За нашия пример това е без значение, т.е. би могло да се използва и оператор за цикъл **For** n := BrStud **downto** 1 **do** . . . . .

Програма 7.2. Програма за намиране средно-аритметичното на елементите на едномерен масив, използваща процедура за въвеждане на едномерен масив и функция за намиране на средно-аритметичното на елементите на едномерен масив.

```
Type
   Danni=real;
   Tip1=array[1..20] of danni;
Var
   Br,i:integer;
   A:Tip1;SrdAr:danni;
{Подпрограма-процедура за въвеждане на едном. масив}
Procedure InMas(Var BrEl:integer;Var El:Tip1);
  Var
    j:integer;
  Begin
    If BrEl=0
    then begin
      Write('Въведете броя на елементите на едном. масив:');Readln(BrEl)
    end;
    For j:=1 to BrEl do
      begin
        Write('Въведете елемент ',,j,' :');Readln(El[j])
      end;
  End;
{Подпрограма-функция за намиране средно-аритметичното на елементите. на едномерен масив}
Function SredAr(BrEl:integer;Var El:Tip1):danni;
  Var
    j:integer;Sum:danni;
  Begin
    Sum:=0;
    For j:=1 to BrEl do
      Sum:=Sum+El[j];
    SredAr:=Sum/BrEl
  End;
Begin
```

```
{Главна програма}
    Br:=0;  {Инициализиране с нула с цел броят да се въведе от подпрограмата}
    InMas(Br,A);      {Въвежд. на елем. на едном. масив}
    SrdAr:=SredAr(Br,A); {Намиране средно-аритм. на елем. на едном. масив}
    Writeln('Средно-аритметичното е ',SrdAr:5:2);
    Readln
End.
```

Внимателният читател трябва да е забелязал, че тази програмата решава по същество същата задача, която решава и програма 7.1 - намира средно-аритметичното на елементите на едномерен масив. Тя обаче има следните предимства:

1. Въведен е тип Danni, който в случая е тип real, но той може да бъде заменен с всеки числов тип, по този начин цялата програма много лесно може да бъде модифицирана по отношение на типа на обработвания масив.

2. Данните се въвеждат посредством процедура, която може да бъде използвана за въвеждане на едномерен масив от целочислен, реален или символен тип или от тип символен низ, т.е. тази процедура може са бъде включвана и в други програми.

3. Средно-аритметичното на елементите на едномерния масив се намира посредством функция, която също може да се използва и в други програми.

Като правило всеки програмист изгражда собствена библиотека от готови подпрограми. Това му дава възможност да създава нови програми, като ги сглобява от готови подпрограми. Новонаписана може да се окаже само главната програма. Турбо Паскал позволява да се изграждат и използват библиотеки по строго определен начин. Този въпрос е разгледан в гл. 12.

Програма 7.3. Програма, която въвежда стойностите на елементите на масива A и след това създава масивите X и Y, съдържащи съответно положителните и отрицателните елементи на дадения масив A.

```
Type
    Mas1=array[1..20] of real;
Var
    i,j,k,n,l:integer;
    A,X,Y:mas1;
Begin
    Write('Въведете броя на елементите '); Readln(n);
    For k:=1 to n do
    begin Write('Въведете ',k,'-ия елемент ');Readln(A[k]) end;
    i:=0;
    j:=0;
    For k:=1 to n do
      If A[k]>0
        then begin
               i:=i+1;
               X[i]:=A[k]
             end
        else if A[k]<0
               then begin
                      j:=j+1;
                      Y[j]:=A[k]
                      end;
    For l:=1 to i do Write(X[l]:5:2,' ');
    Writeln;
    For l:=1 to j do Write(Y[l]:5:2,' ');
```

```
    Writeln;
    Readln
End.
```

7.2. Многомерни масиви

7.2.1. Определение

Обработваната съвкупност от данни, сродни по смисъл и със стойности от един и същ тип не винаги може да се представи като една редица от стойности, т.е като едномерен масив. В някои случаи тя представлява таблица от стойности с определен брой редове и колони, например оценките на студентите от една група по няколко дисциплини, ежедневното трудово възнаграждение на работещите в едно предприятие за всички дни в месеца и т.н. Такава съвкупност от данни представлява даннова структура, наречена *променлива от тип двумерен масив* или накратко *двумерен масив*. Мястото на всеки елемент се определя от два индекса - първият указва реда, а вторият - колоната, на които принадлежи елемента. Стойността на всеки индекс може да принадлежи на всеки дискретен тип, т.е. допустимите за индекса стойности винаги представляват нарастваща последователност.

Възможно е да се наложи дефинирането и на по-сложна структура от данни, например такава, която съдържа оценките от всички изпити на студентите от няколко групи.

Очевидно сега става дума не за една таблица от оценки, а за група еднотипни таблици с оценки. Такива съвкупности от данни в програмите на Паскал се представят като тримерен масив.

Широко използваните в математиката матрици по естествен начин се описват посредством двумерни масиви.

7.2.2. Дефиниране на тип многомерен масив

Тип многомерен масив може да се опише по следният начин:

***array** [Списък от типове на индекси] **of** Тип на елементите*

Например, типът двумерен масив TipMas2 и типът тримерен масив TipMas3 могат да се дефинират по следния начин:

```
Type
  TipMas2=      array [1..20, 1..10] of integer;
  TipMas3=      array [1..15,1..20,1..10] of integer;
```

Типът TipMas2 представлва таблица с 20 реда и 10 колони, а TipMas3 - група от 15 еднотипни таблици с по 20 реда и 10 колони.

Паскал позволява многомерният масив да се разглежда и по друг начин. Например, ако оценките на един студент по всички дисциплини е един едномерен масив, то оценките на всички студенти от дадена група по същите дисциплини може да се разглежда като едномерен масив от едномерни масиви, т.е. двумерният масив може да се представи като едномерен масив от едномерни масиви. Оценките на студентите от всички групи пък можем да разглеждаме като едномерен масив от двумерни масиви. Такъв масив се нарича тримерен масив. Очевидно е, че в програмите могат да се създават и използват и четиримерни, петмерни и т. н. *многомерни масиви.* Това е показано в следния пример.

```
Type
  TipMas1=      array [1..10] of integer;
  TipMas2=      array [1..20] of TipMas1;
```

TipMas3= **array** [1..15] **of** TipMas2;

Допуска се описанието на многомерния масив да стане в разделите **Const** и **Var** при дефиниране на самите обекти от тип масив, но тогава те са от анонимен тип.

7.2.3. Константи от тип многомерен масив

Те се декларират в раздела **Const**, както следва:

```
Type
   TipMas1=array[1..4] of integer;
   TipMas2=array[1..3] of TipMas1;
Const
   Mas2:TipMas2=((3,5,-7,9), (12, -1, 4,8), (2,6,-4,15));
```

Mas2 представя таблица с три реда и четири колони. Може да се зададе и така:

```
Const
   Mas2: array [1..3, 1..4] of integer =((3,5,-7,9), (12, -1, 4,8), (2,6,-4,15));
```

7.2.4. Индекси

Всеки елемент на многомерен масив е определен чрез името на масива (променливата от тип масив) и индекси.

Съставното име на елемент на двумерен масив, дефиниран като таблица, има вида

Име на масив [Индекс1, Индекс2],

където *Индекс1 и Индекс2* могат да бъдат изрази от дискретен тип (нека да припомним, че константата, променливата и елементът на масив са също изрази), който съвпада с типа на съответния индекс, посочен при дефинирането на масива.

Например, X[3,2] е елементът, който се намира на 3-ия ред и 2-ия стълб на двумерния масив-таблица.

Съставното име на елемент на двумерен масив, дефиниран нато масив от масиви, има вида

Име на масив [Индекс1]

или

Име на масив [Индекс1][Индекс2]

където *Индекс1 и Индекс2* могат да бъдат изрази от дискретен тип, който съвпада с типа на съответния индекс, посочен при дефинирането на масива.

Например, Y[3] е третият елемент от масива от масиви и той представлява едномерен масив или третият ред от двумерния масив. Y[3][2] е вторият елемент 2-ия ред. W[2][3][5] е 5-ия елемент едномерния масив, който е 3-ти елемент от масив от едномерни масиви, който пък от своя страна е 2-ри елемент от масив от двумерни масиви.

7.2.5. Операция присвояване

Именувана константа или променлива от тип масив може да се присвои на друга именувана константа или променлива, само когато са от един и същи тип. Елемент на масив или масив може да се присвои на друг масив или на елемент от масив, когато са от един и същи тип и са декларирани като масиви от масиви. Например за декларираните по-горе масиви A, B, X, Y и W са възможни следните присвоявания:

```
A:=B; B:=A;
X[1]:=A; X[2]:=B; X[3]:=Y[1]; X[4]:=X[1]; Y[2]:=X[2];
W[1]:=X; W[2]:=Y;
W[3][1]:=A; W[3][2]:=B; W[3][3]:=X[1]; X[5]:= W[3][1];
```

7.2.6. Въвеждане и извеждане на многомерни масиви

Въвеждат се, както едномерните масиви, елемент по елемент. Но тук се налага да се създават вложени цикли, защото трябва да се стигне до елементи от прост тип, например:

```
For i:=1 to M do
  begin
  Writeln(‘ Въведете ‘,i,’-ия ред‘);
  For j:=1 to N do
    begin
    Write(‘ Въведете ‘,j,’ - ия елемент:’); Readln(G[i,j])
    end
end;
```

Програма 7.4. Програма за намиране сумата на всичките елементи на един двумерен масив с m реда и n стълба.

Сумата на всичките елементи на двумерен масив се намира като последователно се добавят към някаква предварително занулена променлива (например Suma) всички елементи на масива. Обхождането на масива може да стане по редове или по стълбове.

```
Type
   TipMas2=array[1..10,1..15]  of  real; {Двумерният масив е деклариран като таблица}
Var
   i,j,m,n : integer;
   Suma: real;
   A: TipMas2;
Begin
   Wrire(‘Задайте броя на редовете: ’); Readln (m);
   Wrire(‘Задайте броя на стълбовете: ’); Readln (n);
   For i := 1 to m do
     For j := 1 to n do
       begin
         Write(‘Задайте елемента от ‘, i, ‘- ти ред и ‘,j,’-ти стълб :’); Readln(A[i][j])
       end;
   Suma := 0;
   For i := 1 to m do
     For J := 1 to n do S := S + A[i][j];
   Writeln(‘Сумата на елементите е: ’, Suma:10:3)
End.
```

Програма 7.5. Програма за намиране средно-аритметичното във всеки ред на една матрица, използваща процедурата за въвеждане на едномерен масив и функцията за намиране на средно-аритметичното на елементите на едномерен масив от програма 7.2.

```
Type
   Danni=real;
   Tip1=array[1..20] of danni;
   Tip2=array[1..20] of Tip1; {Двумерният масив е масив от едномерни масиви}
Var
   BrRed,BrCol,i:integer;
```

```
    A:Tip2;SrdAr:Tip1;
{Подпрограма-процедура за въвеждане на едном. масив}
Procedure InMas(Var BrEl:integer;Var El:Tip1);
  Var
    j:integer;
  Begin
    If BrEl=0
      then begin
        Write('Въведете броя на елементите:');Readln(BrEl)
      end;
  For j:=1 to BrEl do
    begin
      Write('Въведете ',j,' елемент: ');Readln(El[j])
    end;
  End;
{Подпрограма-функция за намиране средно-аритм. на елем. на едном. масив}
Function SredAr(BrEl:integer;Var El:Tip1):danni;
  Var
    j:integer;Sum:danni;
  Begin
    Sum:=0;
    For j:=1 to BrEl do Sum:=Sum+El[j];
    SredAr:=Sum/BrEl
  End;
{Главна програма}
Begin
  Write('Въведете броя на редовете:'); Readln(BrRed);
  BrCol:=0;{Инициализиране с нула с цел броят да се въведе от подпрограмата}
  For i:=1 to BrRed do
    begin
      InMas(BrCol,A[i]);      {Въвеждане на елементите на i-ия ред (едном. масив)}
      Writeln('Въведени са елементите от ',i,' ред')
    end;
  For i:=1 to BrRed do
    SrdAr[i]:=SredAr(BrCol,A[i]);    {Средно-аритм. на елем. на i-ия ред }
  For i:=1 to BrRed do
    Writeln('Средно-аритметично за ',i,' ред - ',SrdAr[i]:7:2);
  Readln
End.
```

Особеното в тази програма е, че за въвеждането на всеки ред от двумерния масив е използвана подпрограма за въвеждане на едномерен масив и за намиране на средно-аритметичното на елементите от всеки ред на двумерния масив е използвана процедура за намиране на средно-аритметичното на елементите на едномерен масив.

Елементът A[i] (i-ия ред от таблицата)на двумерния масив е едномерен масив и може да бъде посочен като действителен параметър, заместващ фиктивния параметър El, който е едномерен масив от същия тип. Заместването на едномерен масив с елемент от масив от едномерни масиви е възможно, само когато двумерният масив е дефиниран като масив от едномерни масиви.

7.3. Сортиране и търсене в едномерен масив

7.3.1. Сортиране

Много обработки на масиви от данни стават по-лесно и по-бързо, когато елементите им са подредени във нарастващ (възходящ) или в намаляващ (низходящ) ред на стойностите им. Например търсенето на елемент с определена стойност е много по-лесно в подреден масив, отколкото в неподреден. Процесът на подреждане на елементите на масив се нарича **сортиране**.

Създаването на втори масив, съдържащ същите стойности като началния, но сортирани, е по-лесно, но не представлява практически интерес. За икономия на оперативна памет се предпочита пренареждане на стойностите “на място”, т.е. вътре в масива.

Ясно е, че при сортиране на масив многократно се извършват две основни операции - сравняване стойностите на два елемента и размяна на стойностите на два елемента. Всеки конкретен метод за сортировка определя стратегията на сравняване и размяна на елементите на масива. Броят на необходимите основни операции за сортирането на масив с N елемента определя ефективността (бързодействието) на метода.

Разработени са много методи за сортиране, притежаващи отделни предимства. Тук ще разгледаме три прости, но бавни метода. При това за определеност ще считаме, че сортираме във възходящ ред масив A с тип на индекса 1..M.

7.3.1.1. Сортиране чрез вмъкване

Идеята за сортиране чрез вмъкване ще изясним чрез следния пример.

Нека масива A, който трябва да сортираме е следният:

44	55	12	42	94	18

Да предположим, че първите 3 елемента вече са сортирани и масивът има вида:

12	44	55	42	94	18

Сега трябва да вмъкнем на подходящо място четвъртия елемент. Това можем да направим както следва:

1. Изтегляме четвърия елемент от масива (присвояваме A4 на променливата T);

12	44	55		94	18

T | 42 |
|---|

2. Преместваме елементите A3 и A2 с една позиция надясно, т.е. A3 присвояваме на A4 и A2 на A3, защото стойностите им са по-големи от стойността на A4. По този начин освобождаваме място в масива за вмъкване на T. Масивът A придобива вида:

12		44	55	94	18

T | 42 |
|---|

3. Вмъкваме T на свободното място, т.е. на A2 присвояваме T и масивът придобива вида:

12	42	44	55	94	18

Очевидно, за да сортираме целия масив, трябва да изпълним горните 3 операции за всички елементи от втория до последния.

Програма 7.6 Сортиране на едномерен масив чрез вмъкване

```
Type
   Danni=integer;
   Msv1=array[0..15] of danni;
Var
   i,M:integer; B:Msv1;
Procedure Vmakvane(n:integer;Var A:Msv1);
  Var
   i,j:integer; T:danni;
  Begin
   For i:=2 to n do
     begin
       j:=i; T:=A[i];
       While (j>1) and (T<A[j-1]) do
         begin
           A[j]:=A[j-1]; j:=j-1
         end;
       A[j]:=T
     end
  End;
Begin
   Write('Брой на елементите: '); Read(M);
   Writeln('Стойности на елементите:');
   For i:=1 to M do begin Write('  ',i,'  '); Readln(B[i]) end;
   Vmakvane(M,B);
   Writeln('Сортираният масив е:');
   For i:=1 to M do write(B[i],' ');
   Readln
End.
```

7.3.1.2. Сортиране чрез размяна

Този метод се състои в следното:

- сравняват се първият и вторият елемент и се разменят ако, първият е по-голям;
- сравняват се вторият и третият елемент и се разменят ако, вторият е по-голям;
- . . . . . .
- сравняват се предпоследният и последният елемент и се разменят ако, предпоследният е по-голям.

Казаното по-горе, приложено за масива:

44 55 12 42 94 18

последователно ще променя масива както е показано по-долу и ще изтика най-голямата стойност на последно (шесто) място в масива.

44	55	12	42	94	18	- след сравняване на I и II елемент
44	12	55	42	94	18	- след сравняване на II и III елемент
44	12	42	55	94	18	- след сравняване на III и IV елемент
44	12	42	55	94	18	- след сравняване на IV и V елемент

44 12 42 55 18 94 - след сравняване на V и VI елемент

Следващата задача е да се закара втората по големина стойност на предпоследно (пето) място и това може да се постигне като се повторят горните операции за първите 5 елемента.

Масивът последователно ще придобие вида:

12 42 44 18 55 94 - след разместване на първите 5 елемента

12 42 44 18 55 94 - след разместване на първите 4 елемента

12 42 18 44 55 94 - след разместване на първите 3 елемента

12 18 42 44 55 94 - след разместване на първите 2 елемента

Този метод е известен още като *метод на мехурчето*, защото, като се хвърли камък в езеро, от дъното към повърхността тръгват мехурчета, подредени по големина - най- голямото изплува най- бързо, след него е следващото и т.н.

Програма 7.7. Сортиране на едномерен масив чрез размяна

```
Type
   Danni=integer;
   Msv1=array[0..15] of danni;
Var
   i,M:integer; B:Msv1;
Procedure Razmiana(n:integer;Var A:Msv1);
  Var
   i,j:integer;T:danni;
  Begin
    For i:=1 to n-1 do
      For j:=1 to n-i do
        if A[j]>A[j+1]
        then begin
             T:=A[j]; A[j]:=A[j+1]; A[j+1]:=T
             end
  End;
Begin
   Write('Брой на елементите: '); Read(M);
   Writeln('Стойности на елементите:');
   For i:=1 to M do begin Write('  ',i,'  '); Readln(B[i]) end;
   Razmiana(M,B);
   Writeln('Сортираният масив е:');
   For i:=1 to M do write(B[i],' ');
   Readln
End.
```

Понякога масивът може да се окаже сортиран преди да са извършени всички итерации. Това означава, че по този алгоритъм понякога се върши ненужна работа. За да се избегне това, той може да се усъвършенства - итерациите се прекратяват, ако при последната не е направена нито една размяна (метод на мехурчето с флаг). Някои автори предлагат при всяка итерация да се променя посоката на преглеждане на масива (“*шейкър*” метод). И при тези усъвършенствания методът на пряката размяна не е по-ефективен от другите два, разгледани тук.

7.3.1.3. Сортиране чрез селекция

Идеята на този метод е много проста - намира се елементът с най-малката стойност и той и първият елемент си разменят стойностите. Така на първо място

вече е най-малкият елемент. Като се повтори това с подмасива, започващ от втория елемент, с подмасива, започващ от третия елемент и т.н. до подмасива, започващ от предпоследния елемент, масивът ще се окаже сортиран.

За разглеждания вече пример резултатът след всяка итерация е:

44	55	12	42	94	18	- преди началото на сортирането
12	55	44	42	94	18	- след размяна на I и III елементи
12	18	42	44	94	55	- след размяна на II и VI елементи
12	18	42	44	94	55	- след размяна на III и III елементи
12	18	42	44	94	55	- след размяна на IV и IV елементи
12	18	42	44	55	94	- след размяна на V и VI елементи

Програма 7.8. Сортиране на едномерен масив чрез селекция}

```
Type
  Danni=integer;
   Msv1=array[0..15] of danni;
Var
  i,M:integer; B:Msv1;
Procedure Selekcia(n:integer;Var A:Msv1);
  Var i,j,m:integer; T:danni;
  Begin
    For i:=1 to n-1 do
      begin
        m:=i;
        For j:=i+1 to n do
          if A[j]<A[m]
            then m:=j;
        T:=A[m]; A[m]:=A[i]; A[i]:=T
      end
  End;
Begin
  Write('Брой на елементите: '); Read(M);
  Writeln('Стойности на елементите:');
  For i:=1 to M do begin Write('  ',i,'  '); Readln(B[i]) end;
  Selekcia(M,B);
  Writeln('Сортираният масив е:');
  For i:=1 to M do write(B[i],' ');
  Readln
End.
```

7.3.2. Търсене

Често срещана в практиката задача е намирането в масив от данни мястото (индексът) на елемент със зададена стойност. В общия случай единственият начин е последователното сравняване на елементите на масива със зададената стойност, което е много бавен процес. Търсенето може значително да се ускори, ако масивът предварително е сортиран (за определеност - във възходящ ред на стойностите), защото може да се използва по-ефективен метод, например, така нареченото двоично (дихотомно) търсене.

7.3.2.1. Двоично търсене на елемент, равен на зададена стойност

Идеята на двоичното търсене се състои в последователното стесняване на половина на участъка от масива, в който се намира търсеният елемент.

Започва се с целия масив. Сравнява се търсената стойност с елемента, който е по средата на масива. Възможни са следните три случая:

- стойностите са равни - търсеният елемент е намерен;
- търсената стойност е по-малка - търсенето продължава в левия подмасив;
- търсената стойност е по-голяма - търсенето продължава в десния подмасив.

След това по същия начин се търси в подмасива, после в негов подмасив и т.н. Процесът продължава до намиране на търсения елемент или до подмасив от нито един елемент (масивът не съдържа търсената стойност).

Пример:

Нека масивът съдържа следните 10 елемента:

Индекс:	1	2	3	4	5	6	7	8	9	10
Стойност:	3	7	12	14	22	28	31	46	89	90

1. Търси се елементът със стойност 12. Процесът на търсене изглежда така:

Начален индекс	Краен индекс	Среден индекс	Стойност в средата
1	10	5	22
1	4	2	7
3	4	3	12

Търсената стойност съвпада с третия елемент.

2. Търси се елементът със стойност 13. Процесът на търсене изглежда така:

Начален индекс	Краен индекс	Среден индекс	Стойност в средата
1	10	5	22
1	4	2	7
3	4	3	12
4	4	4	14
4	3	-	-

Търсената стойност не е намерена.

Алгоритъм, основаващ се на тази идея, е реализиран като функция от логически тип, която връща стойност True или False, в зависимост от това дали в масива е намерен или не е намерен елемент, равен на зададения.

При обръщане към функцията фиктивните параметри m и A се заместват съответно с броя на елементите и името на масива, в който става търсенето; B - със стойността, която търсим в масива, или името на променливата,чиято стойност търсим в масива; NomEl с променливата, на която искаме да се присвои индексът на намерения елемент. Локалните променливи N, K и S са индекси съответно на началния, крайния и средния елемент на подмасива, в който предстои търсене. Средният индекс се определя като цялата част на полусумата на началния и крайния индекс на подмасива.

Програма 7.9. Двоично търсене на стойност в подреден едномерен масив

```
Type
  Danni=integer;
  Msv=array[1..15] of danni;
Var
  i,NomNamEl,BrEl:integer; B:Msv; X:danni;
Function DvTrsR(Br:integer;A:Msv;T:Danni;Var NomEl:integer):boolean;
  Var
```

```
    N,K,S:integer;
  Begin
    N:=1;K:=Br; DvTrsR:=false;
    Repeat
      S:=(N+K) div 2;
      If T=A[s]
        then begin
               NomEl:=s; DvTrsR:=true; Exit
             end;
      If T>A[S]
        then N:=S+1
        else K:=S-1;
    until N>K;
  End;
Begin
  Write('Брой на елементите: '); Readln(BrEl);
  Writeln('Стойности на елементите:');
  For i:=1 to BrEl do begin Write('  ',i,'  '); Readln(B[i]) end;
  Writeln;Write('Задайте търсена стойност X= '); Readln(X);
  If DvTrsR(BrEl,B,X,NomNamEl)
    then writeln('Зададената стойност съвпада с ',NomNamEl,'-ия елемент')
    else writeln('Няма такава стойност в масива.');
  Readln
End.
```

7.3.2.2. Двоично търсене на елемент, равен или по-голям от зададена стойност

В някои задачи се търси равен или най-близък по-голям елемент в масив. Алгоритъмът за намиране на такъв елемент чрез двоично търсене не се различава много от предходния. Функцията DvTrsPGR е от логически тип и връща стойност False, само когато търсената стойност е по-малка от първата или по-голяма от последната стойност на масива. При обръщане към функцията NamSt се замества с променливата, която трябва да върне намерената равна или най-близката по-голяма стойност в масива.

Програма 7.10. Двоично търсене в подреден масив на елемент, които е най-близък по-голям или равен на зададена стойност.

```
Type
  Danni=integer;
  Msv1=array[0..15] of danni;
Var
  i,M:integer; B,C:Msv1; X,Y:danni;
Function DvTrsPGR(m:integer;Var A:Msv1;B:Danni;Var NamSt:Danni):boolean;
  Var
    N,K,S:integer;
  Begin
    If B>A[m]
      then DvTrsPGR:=false
      else begin if B<=A[1]
                   then S:=1
                   else begin
                        N:=1;K:=m;
                      Repeat
                        S:=(N+K) div 2;
                        If B>A[S]
                        then N:=S
```

```
                    else K:=S;
                  until (B=A[S]) or (K-N=1);
                  If B<>A[S] then S:=K
                end;
              NamSt:=A[S]; DvTrsPGR:=true
    end
  End;
Begin
  Write('Брой на елементите: '); Read(M);
  Writeln('Стойности на елементите:');
  For i:=1 to M do begin Write('  ',i,'  '); Readln(B[i]) end;
  Writeln;Write('Задайте търсена стойност X= '); readln(X);
  If DvTrsPGR(M,B,X,Y)
    then Writeln('Намерената най-близка по-голяма или равна стойност е ',Y)
    else Writeln('Няма по-голяма или равна стойност.');
  Readln
End.
```