9. МНОЖЕСТВА

9.1. Множество в Паскал

Понятието множество ни е познато както от ежедневието, така и от математиката. Можем да говорим за множеството на студентите от една група, множеството от изпитите за дадена сесия, множеството на дните от седмицата, множеството от работните дни в седмицата. Съществената е, че всяко от споменатите множества е съвкупност от обекти от един и същ тип.

В програмите на Паскал също могат да се дефинират и използват типове множества, константи от тип множество и променливи от тип множество. Множеството в Паскал може да съдържа еднотипни компоненти. Допустими типове на компонентите са всички дискретни типове в Паскал, т.е. множество в Паскал може да бъде с компоненти от целочислен тип, символен тип, логически тип, изброен тип или ограничен тип.

9.2. Дефиниране на тип множество

Типът множество в Паскал е съставен тип с компоненти от определен базов тип, който може да съдържа за Турбо Паскал до 255 дискретни стойности. Типовете **char**, **byte** и **shortint** могат да бъдат базови типове, защото са дискретни и съдържат до 255 дискретни стойности. Типът **integer** не може да бъде базов тип, защото включва 65 535 дискретни стойности, но всеки ограничен тип, който има за базов тип типа **integer** и включва до 255 стойности, може да бъде базов тип за типа множество. Такъв е например ограниченият тип 1001..1100.

Описанието на тип множество има следния общ вид:

Set of *базов тип*

Примери:

```
Type
   Den = (Pon,Vtr,Srd,Ctv,Ptk,Sbt,Ned);  {Изброен тип}
   OgrInt = 2001..2100;                  {Ограничен тип}
   IntSet = Set of OgrInt;
   Dni = Set of Den;
   CharSet = Set of char;
```

9.3. Константи от тип множество

Константите от тип множество се задават с указване списъкът на компонентите, затворен в скоби от вида **[ ]**. Списъкът може да се зададе чрез изброяване на компонентите или чрез указване на интервала, когато компонентите са поредни. Редът на компонентите в списъка е без значение. Една и съща компонента не може да присъства в множеството повече от един път.

Съществува и празно множество. То се записва така [].

Примери:

```
[Pon, Ptk, Sbt, Srd]
['A','M','B','Ф']
[Pon..Ned]
['A'..'Г']
[2001, 2007, 2012, 2021..2030, 2050]
['A','B','D','P'..'W']
```

В раздела **Const** могат да се дефинират и именувани константи, например

```
Const
  RabDni=[Pon..Ptk];
```

9.4. Операции с множества

Множествата, които участват в дадена операция, трябва да са от един и същ тип.

9.4.1. Обединение на две множества (+)

Обединението **A+B** на множествата A и B представлява новото множество C, което получаваме като към първото множество A добавим онези елементи от второто множество B, които не се съдържат в първото множество.

[5,8,15,20] + [3,8,12] → [5,8,15,20,3,12]

9.4.2. Разлика на две множества (-)

Разликата **A-B** на множествата A и B е ново множество C, което получаваме като от първото множество A отстраним онези елементи, които се съдържат и във второто множество.

[5,8,15,20] - [3,8,12] → [5,15,20]

9.4.3. Сечение на две множества (*)

Сечението **A*B** на множествата A и B е ново множество C, което включва онези елементи от първото множество, които са елементи и на второто множество.

[5,8,15,20] * [3,8,12] → [8]

9.5. Логически операции с множества

Множествата, които участват в дадена логическа операция, трябва да са от един и същ тип.

9.5.1. Еквивалентност на две множества (=)

Две множества A и B са еквивалентни, когато са съставени от едни и същи елементи. Логическият израз **A=B** има стойност true, когато множествата са еквивалентни, и false, когато не са еквивалентни.

9.5.2. Нееквивалентност на две множества (<>)

Две множества A и B са нееквивалентни, когато не са съставени от едни и същи елементи. Логическият израз A <> B има стойност true, когато множествата не са еквивалентни, и false, когато са еквивалентни.

9.5.3. Подмножество (<=)

Множеството A е подмножество на множеството B, когато всички елементи на A са елементи и на B. Логическият израз **A<=B** има стойност true, когато A е подмножество на B, и false, когато A не е подмножество на B.

9.5.4. Надмножество (>=)

Множеството A е надмножество по отношение на множеството B, когато то съдържа всички елементи на множеството B. Логическият израз **A>=B** има стойност true, когато A съдържа B, и false, когато A не съдържа B.

9.5.5. Принадлежност на множество (in)

Логическият израз

*X **in** A*

има стойност:

- true, т.е. X принадлежи на множеството A, когато има компонента на A, която е равна на стойността на X;
- false, т.е. X не принадлежи на A, когато няма компонента на A, която е равна на стойността на X;

9.6. Изрази от тип множество

Един израз е от тип множество, когато резултатът от изпълнението му е множество. Такъв израз съдържа константи и променливи от тип множество, знаци за операции (обединение, разлика и сечение), знаци за логически операции и скоби.

Операцията сечение (*) е с по-висок приоритет от операциите обединение (+) и разлика (-), които са с еднакъв приоритет. Например след изпълнение на изразите:

а) [5,7,3,12] * [2..6] + [10..15] - [10,13,25,40]
б) ([5,7,3,12] + [8,9,12]) * ([1..10] - [8..12])

се получават множествата:

а) [3,5,11,12,14,15];
б) [3,5,7].

9.7. Оператор за присвояване

Операторът за присвояване

Име на променлива := израз

се използва и при множества. Разбира се остава в сила изискването променливата и изразът да са от един и същи тип.

9.8. Въвеждане и извеждане на множества

Имената на променливи от тип множество не могат да присъстват във входните списъци на операторите Read и Readln и изходните списъци на операторите Write и Writeln. Поради това те се въвеждат и извеждат компонента по компонента, като се организират цикли. Когато и компонентите на множеството са от такъв тип, че също не могат да присъстват в операторите Read, Readln, Write и Writeln, те се въвеждат чрез съответни числови кодове.

Програма 9.1.Програма, която извършва следните операции с масив от множества с целочислени компоненти:

- въвежда масива;
- извежда масива;
- намира броя на компонентите във всяко множество;
- намира общите компоненти за всички множества;

- намира честотата на срещане на компонентите в масива от множества.

Програмата е създадена като съставна. Тя включва следните подпрограми:

- процедура за въвеждане на множество;
- процедура за извеждане на множество;
- функция за намиране броя на компонентите на множество;
- процедура за намиране общите компоненти за масив от множества;
- процедура за намиране честотата на срещане на компонентите в масива от множества.

От програмата се вижда как се реализират най-често срещаните операции с множества.

```
Type
  TipMno= set of 1..6;                    {Тип множество}
  TipMasMno=array[1..10] of TipMno;       {Тип масив от множества}
  TipMasChest=array[1..6] of integer;     {Тип масив с честотите на повторение на
компонентите}
Var
  i,BrMno:integer;
  Mno, MnoObComp:TipMno;      {Променливи от тип множество}
  MasMno:TipMasMno;           {Масив от множества}
  BrComp:array[1..10] of integer; {Брой на компонентите във всяко множество}
  MasChest:TipMasChest;  {Масив на честотите на повторение на компонентите}
{Процедура за въвеждане на множество}
Procedure ReadMno(Var Mno:TipMno);
  Var
    ElMno:integer;  {Елемент на множеството}
  Begin
    Mno:=[];
    Repeat
      Write('Въведете цифра от 1 до 6 или 0 за край:'); Readln(ElMno);
      If ElMno in [1..6]
        then Mno:=Mno + [ElMno]
    until ElMno=0;
End;
{Процедура за извеждане на множество}
Procedure WriteMno(Mno:TipMno);
  Var
    ElMno:integer;
  Begin
    Write('Съдържание на множеството: ');
    If Mno=[]
      then Writeln('празно')
      else For ElMno:=1 to 6 do
             if ElMno in Mno then write(ElMno,' ');
    Writeln
  End;
{Функция за намиране броя на компонентите на множество}
Function BrCompMno(Mno:TipMno):integer;
  Var
    ElMno,b:integer;
  Begin
    b:=0;
    For ElMno:=1 to 6 do if ElMno in Mno then b:=b+1;
```

```
    BrCompMno:=b
  End;
{Процедура за намиране общите компоненти за масив от множества}
Procedure ObComp(BrMn:integer; Var MasMno:TipMasMno;
                                 Var MnObComp:TipMno);
  Var
    i:integer;
  Begin
    MnObComp:=MasMno[1];
    For i:=2 to BrMn do MnObComp:=MnObComp*MasMno[i];
  End;
{Процедура за намиране честотата на срещане на компонентите в масива от
множества}
Procedure Chestota(BrMn:integer;Var MasMno:TipMasMno;
                                 Var MsChest:TipMasChest);
  Var
    ElMno,i,Br:integer;
  Begin
    For ElMno:=1 to 6 do
    begin
      Br:=0;
      For i:=1 to BrMn do if ElMno in MasMno[i] then Br:= Br+1;
      MsChest[ElMno]:=Br;
    end;
  End;
Begin {Главна програма}
{Въвеждане на множествата}
  Write('Въведете броя на множествата:'); Readln(BrMno);
  For i:=1 to BrMno do
  begin
    Writeln('Въвеждате ',i,'-то множество:');
    ReadMno(MasMno[i])
  end;
{Извеждане на множествата}
  Writeln('Въведени са следните множества:');
  For i:=1 to BrMno do WriteMno(MasMno[i]);
{Намиране броя на компонентите във всяко множество от масива}
  Writeln('Брой компоненти във всяко множество:');
  For i:=1 to BrMno do
    begin
      BrComp[i]:=BrCompMno(MasMno[i]);{Намиране бр. на комп. в i-то
                                                          множество}
      Writeln(i,'  ',BrComp[i])          {Извеждане бр. на комп. в i-то множество}
    end;
{Намиране общи компоненти за всички множества}
Writeln('Множество на общите компоненти за всички елементи от масива:');
ObComp(BrMno,MasMno,MnoObComp);{Намиране множ. на общите
                                                          компоненти}
WriteMno(MnoObComp);   {Извеждане множеството на общите компоненти}
{Намиране честотите на срещане на компонентите в масива от множества}
Writeln('Честота на срещане на компонентите:');
Chestota(BrMno,MasMno,MasChest);
For i:=1 to 6 do Writeln(i,'  ',MasChest[i]);
```

```
  Readln
End.
```

Програма 9.2. Програма, която въвежда от клавиатурата символи докато срещне символа @ и брои отделно буквите от латиницата, цифрите и препинателните знаци.

В програмата са дефинирани три множества - от букви (Buc), от цифри (Cif) и от препинателни знаци (PrZn). Проверката за принадлежност на прочетения символ X към всяко множество става с оператора **in**.

```
Program Scan;
Var
  Buc,Cif,PrZn : Set of char;
  BrBuc,BrCif,BrPrZn : integer;
  Ch : char;
Begin
  Buc := ['A'..'Z','a'..'z'];
  Cif := ['0'..'9'];
  PrZn := ['!', '?', '.', ',', ';', ':'];
  BrBuc := 0;
  BrCif := 0;
  BrPrZn := 0;
  Repeat
    Read(Ch);
    If Ch in Buc
      then BrBuc := BrBuc + 1
      else If Ch in Cif
              then BrCif := BrCif + 1
              else If Ch in PrZn
                      then BrPrZn := BrPrZn+1
  until Ch = '@';
  Writeln('Букви -',BrBuc);
  Writeln('Цифри -',BrCif);
  Writeln('Препинателни знаци -',BrPrZn)
End.
```