11. ФАЙЛОВЕ

Нека да си припомним програма 10.1. Там дефинирахме масив от записи, като всеки запис - елемент на масива съдържа данните за един студент, а целият масив - данните за всички студенти от една група. След това въведохме данните за студентите. По-нататък с тези данни могат да се изготвят и извеждат различни справки. След приключване изпълнението на програмата обаче, въведените данни не се запазват в компютъра. Ако стартираме програмата отново, трябва да въведем същите данни отново.

Естествено възниква въпросът, не може ли въведените данни да се запазят в компютъра, за да не се налага да ги въвеждаме при всяко стартиране на програмата. Отговорът е положителен - данните могат да се запазят в компютъра, но в неговата външна памет, т.е. във вид на файл на твърдия диск или на дискета.

Файлът е именуван набор от данни, намиращ се във външната памет. Той може да съдържа всякакъв вид информация: от писмо, създадено с текстообработваща програма, до изходен код за управление на металообработваща машина или робот в машиностроително предприятие. Различни приложни програми създават различни файлове. Текстообработващите програми създават файлове, съдържащи текстови документи (писма, отчети, доклади и др.). Програмите, известни като електронни таблици, създават файлове, съдържащи таблично представени документи. Графичните компютърни системи създават файлове, съдържащи различни машиностроителни, архитектурни и други чертежи.

Извеждането и въвеждането на информация във/от файлове на външни носители става посредством външни запомнящи устройства (ВЗУ).

Чрез програми на Паскал могат да се създават и използват два вида файлове: *файлове от компоненти и текстови файлове*. Файловете от компоненти се наричат още структурирани файлове, форматирани файлове и двоични файлове.

11.1. Структурирани файлове

Структурираният файл е съвкупност от произволен брой еднотипни компоненти, последователно наредени на външен носител (твърд диск или дискета). Поради това казваме, че структурираният файл е с последователна организация или с последователен достъп. Компонентите на файла са наредени във външния носител по реда на тяхното извеждане (записване). В стандартния Паскал такъв файл може да бъде отворен за четене и тогава от него могат само да се четат компоненти и то само последователно. Файлът може да бъде отворен и за извеждане на компоненти в него, но извеждането може да става само последователно. И в Турбо Паскал структурираните файлове са с последователна организация, но последните версии допускат и пряк достъп, т.е. те позволяват да прочетем направо третата, седмата или коя да е друга компонента, без да сме прочели предходните, или да подменим третата, седмата или коя да е друга компонента с нова. Тук ще разгледаме възможностите на Турбо Паскал 7.0.

11.1.1. Дефиниране на тип структуриран файл и променлива от тип структуриран файл

Типът на компонентите може да бъде целочислен, реален, символен,

логически или дефиниран от програмиста. Тип структуриран файл се дефинира в раздел **Type**, като описанието на файла има вида

file of *Тип на компонентите на файла;*

Например

```
Type
  TipIntFl = file of integer; {Тип файл с име TipIntFl и целочислени компоненти}
  TipRlFl = file of real;     {Тип файл с име TipRlFl и реални компоненти}
  TipRlMasiw = array[1..100] of real; {Тип масив с име TipRlMasiw и реални елементи}
  TipZapis = record           {Тип запис с име TipZapis }
       Ime:string[30];
       Zaplata;integer
       end;
  TipFlRlMas = file of TipRlMasiw;        {Тип файл с компоненти от тип TipRlMasiw }
  TipFlIntMas = file of array[1..50] of integer; {Тип файл компоненти от анонимен тип}
  TipFlZapis = file of TipZapis;    {Тип файл с компоненти от тип TipZapis}
```

На всеки файл програмистът дава име, с което операционната система го регистрира в каталога на външния носител. Очевидно името трябва да съответства на изискванията на операционната система за имена на файлове. В програма на Паскал файлът се представя от променлива от тип файл, избрана от програмиста.

Променлива от тип структуриран файл се дефинира в раздела Var, например

```
Var
  IntFl : TipIntFl;    {Променлива от тип файл с компоненти от тип TipIntFl}
  RlFl : TipRlFl;      {Променлива от тип файл с компоненти от тип TipRlFl}
  FlRlMas : TipFlRlMas;{Променлива от тип файл с компоненти
                                                от тип TipFlRlMas}
  FlIntMas : TipFlIntMas; {Променлива от тип файл с компоненти
                                                от тип TipFlIntMas}
  FlZapis : TipFlZapis; {Променлива от тип файл с компоненти
                                                от тип TipFlZapis }
  FlMasInt : file of array[1..20] of integer;{Променлива от анонимен
тип файл с компоненти от анонимен тип масив с целочислени елементи}
  FlZapis1 : file of TipZapis; {Пром. от анонимен тип файл с компоненти
                                                от тип TipZapis}
  FlZapis2 : file of record {Пром. от тип файл с компоненти
                                                от анонимен тип запис}
                Ime : string[30];
                EGN : string[10]
             end;
```

Както ще видим малко по-късно, в програмата променливата от тип файл се свързва с конкретен файл, намиращ се на външен носител, и тя го представя докато трае изпълнението на програмата или докато тя бъде свързана с друг файл. Поради това е логично променливата от тип файл да се нарича и променлива, свързвана с файл. За по-кратко, тя се наричана и променлива-файл или дори само файл.

При различни изпълнения на програмата, променливата от тип файл може да представя различни файлове от външния носител, но те трябва да са от един и същ тип.

11.1.2. Номер на компонента. Файлов указател. Край на файла

Компонентите на файла се разглеждат като номерирана последователност, като номерирането започва от нула, т.е. компонентата, записана първа, получава номер нула, следващата - номер едно и т.н.

При работа с файла системата поддържа специален указател, наречен файлов указател. При записване на компоненти във файл, той сочи във всеки момент края на последната записана компонента. При четене на компоненти от файл, той сочи разделителя между последната прочетена компонента и следващата след нея още непрочетена компонента.

Краят на файла се открива посредством функцията:

EOF*(Променлива-файл).*

При отварянето на файла, стойността на тази функция е False (освен ако файлът е празен) и остава такава до прочитане на последната компонента. Когато се прочете последната компонента, функцията EOF придобива стойност True.

11.1.3. Операции с файлове, изпълнявани от процедури и функции

За улесняване работата с файлове в Турбо Паскал са включени голям брой процедури и функции. По-надолу са разгледани по-важните от тях.

а) Свързване на променлива от тип файл с конкретен файл

Тази операция се изпълнява от процедурата:

Assign(VarFile, ImeNaFile);

където VarFile е променлива от тип файл (променлива-файл), а ImeNaFile е символен низ, представляващ спесификация на файла (път, име), който трябва да се свърже с VarFile. Свързването може да се оприличи с операцията присвояване, защото след нейното изпълнение променлива-файл вече има конкретно значение, което означава конкретен файл от външната памет. Променливата VarFile ще представя в програмата файла ImeNaFile до свързването ѝ с нов файл чрез нов оператор **Assign** или до края на програмата.

Примери:

```
Assign(F1,'a:\Students.Dat');
Assign(F2, 'c:\Work\Pas\DanStud.Dat');
Assign(F3, FileName);
```

В първите два примера името на файла представлява константа, а в последния - променлива от тип символен низ.

б) Създаване на нов файл

Нов файл се създава ("отваря") с процедурата

Rewrite (Променлива-файл).

След нейното изпълнение файлът вече съществува във външната памет и там той е регистриран под името, което е посочено за свързване с променливата-файл в оператора **Assign**.Файлът е празен, т.е. с нулева дължина, стойността на функцията **EOF** е True. Създаденият файл е отворен и в него могат да се извеждат компоненти. При отваряне на съществуващ файл с тази процедура, съдържанието му се загубва и в него може да се пише като в нов файл.

в) Отваряне на съществуващ файл

Съществуващ файл се отваря посредством процедурата:

Reset (Променлива-файл);

След изпълнението на тази процедура може да се четат и записват компоненти във файла, свързан с посочената променлива-файл. Стойността на функцията **EOF** е false, освен ако файлът е празен. Обикновено след прочитане и на последната компонента, т.е. когато стойността на **EOF** стане true, започва извеждане на компоненти в същия файл. Извеждането може да започне и по-рано, но тогава новите компоненти ще заличат част от старите.

г) Записване (извеждане) на компоненти във файл

Стойността на променливата A може да бъде записана (изведена) като поредна компонента на файла, свързан с променлива-файл F, посредством следния оператор:

Write(F,A)

Типът на A трябва да съвпада с типа на компонентите на файла. След записването файловият указатело се премества след записаната компонента.

д) Четене (въвеждане) на компоненти от файл

Поредната компонента на файла, свързан с променлива-файл F, може да се прочете като стойност на B посредством следния оператор:

Read(F,B)

След прочитане на последната компонента стойността на **EOF** става true. При опит за четене от този момент нататък "прочетената" стойност е НЕОПРЕДЕЛЕНА. Използването на такава стойност по-нататък в програмата е недопустимо. За да избегне такава ситуация, програмистът трябва да проверява стойността на **EOF** преди всяко четене на компонента от файл. След прочитането файловият указатело се премества след прочетената компонента.

е) Приключване на работата с файл

Когато програмата престане да ползва даден файл, тя трябва да го затвори. Това тя може да направи чрез стандартната процедура:

Close (Променлива-файл);

"Затварянето" на файла е задължително, освен когато от него само са четени компоненти.

ж) Намиране броя на компонентите във файла

Това става с помощта на функцията:

FileSize(Променлива-файл),

която е от целочисления тип **Longint**.

з) Намиране позицията на файловия указател

Това става с помощта на функцията

FilePos(Променлива-файл),

която е от целочисления тип **Longint**. Стойност на тази функция е номерът на компонентата, пред която се намира файловият указател, който съвпада с броя на компонентите пред файловия указател.

и) Позициониране указателя на файл на зададена компонента

Нека да предположим, че се интересуваме от компонента номер 4 на файл с файлова променлива FS. Един начин да достигнем до нея и да я прочетем е следният:

```
For i := 0 to 4 do Read(FS,FR);
```

Последователно ще бъдат прочетени компонентите с номера 0, 1, 2, 3 и 4 и стойността на последната ще остане стойност на FR.

В някои "диалекти" на Паскал, в това число и Турбо Паскал, е възможен и "*пряк*" достъп до отделните компоненти. За достъп до компонента от файла с посочен номер в Турбо Паскал се използва процедурата

Seek(Променлива-файл, Номер на компонента),

където *Номер на компонента* е израз от целочислен тип.

Например, за да прочетем направо компонента номер 4 от файла 'STUD.DAT', намиращ се на текущата директория, трябва да използваме следните два оператора:

```
Seek(FS,4);          Read(FS,FR);
```

Първият оператор ще позиционира файловия указател пред компонента номер 4, а вторият ще я прочете като стойност на FR.

к) Изтриване на файл

Става с процедурата:

Erase(Променлива-файл).

л) Преименуване на файл

Става с процедурата

Rename(Променлива-файл, Ново име на файла),

където *Ново име на файла* е новото име, с което искаме да се запише файлът в каталога на дисковото устройство, и може да бъде константа или променлива от тип символен низ.

Програма 11.1 Програма, която:

- създава файл с компоненти от тип Real, намиращ се в текущата директория;
- прочита числата от създадения файл и изчислява сумата от онези от тях, които имат стойности, по-големи от предварително зададена стойност.

```
Var
  Fl : file of real;    {Fl-променлива от тип файл}
  ImeNaFail:string;
  E, X, S : real; Ch:char;
Begin
  Write('Задайте името на файла: ');Readln(ImeNaFail);
  Assign(Fl,ImeNaFail);
  Rewrite(Fl);
  Repeat
    Write('Въведете число: ');Readln(X);Write(Fl,X);
    Write('Ще продължите ли въвеждането? ');Readln(Ch);
  until Ch in ['N','n','Н','н'];
  Close(Fl);
  Write(' Задайте E: '); ReadLn(E); {E-предварително зададена стойност}
  S := 0;
  Reset(Fl);
  While not EOF (Fl) do
```

```
  begin
    Read(Fl,X);
    If X > E then S := S + X;
  end;
  Close(Fl);
  WriteLn('Сумата на числата по-големи от ',E:7:3,' е ',S:9:3);
  Readln
End.
```

Програма 11.2. В таблица се съдържат данни за група лица. Програма, която извършва следните операции с данните от таблицата:

- създава нов празен файл;
- добавя към файла компоненти с данни за лица;
- изтрива компоненти от файла;
- заменя компоненти от файла с нови
- извежда списък на лицата, родени през посочена година;
- извежда данните на всички лица, включени във файла.

```
Type
  TipLice=record {Тип запис с данните за едно лице}
            Ime:string[20]; EGN:string[10];
            end;
Var
  PromFl:file of TipLice;
  ImeFl:string[20];
  Lice:TipLice;
  C:byte;
{Процедура за въвеждане данните за едно лице}
Procedure ReadLice(Var Lc:TipLice);
Begin
  With Lc do
  begin
    Write(' ':20,'Въведете името: ');Readln(Ime);
    Write(' ':20,'Въведете ЕГН: ');Readln(EGN)
  end
End;
{Процедура за извеждане данните за едно лице}
Procedure WriteLice(Lc:TipLice);
Begin
  With Lc do Writeln('Име: ',Ime,' ':21-length(Ime),'EGN: ',EGN)
End;
{Процедура за създаване на празен файл}
Procedure SzdPrz;
Begin
  Rewrite(PromFl); Close(PromFl);
  Writeln;Writeln('       Файлът е създаден.')
End;
{Процедура за добавяне на компоненти към файла }
Procedure Dobaviane;
  Var
    Ch:char;
  Begin
    Reset(PromFl); Seek(PromFl,FileSize(PromFl));
    Repeat
```

```
    ReadLice(Lice); Write(PromFl,Lice);
    Write('Ще продължите ли въвеждането? ');readln(Ch)
  until Ch in ['N','n','Н','н'];;
  Close(PromFl);
End;
```

{Процедура за изтриване на компонента}

```
Procedure Iztrivane;
  Var
    DelFl:file of TipLice;     {Помощен файл, нужен за изтриването}
    EGN:string[10];            { ЕГН за изтриване}
  Begin
    Write('Въведете ЕГН: ');Readln(EGN);
    Reset(PromFl); Assign(DelFl,'Del'); Rewrite(DelFl);
    While not EOF(PromFl) do
    begin
      Read(PromFl,Lice);
      If Lice.EGN<>EGN
      then Write(DelFl,Lice)
      else Writeln('Данните за лице с ЕГН ',EGN,' са изтрити.');
    end;
    Close(PromFl); Close(DelFl); Erase(PromFl); Rename(DelFl,ImeFl);
    Writeln;
  End;
```

{Процедура за заменяне на компонента}

```
Procedure Zamiana;
  Var
    n:integer;
    EGN:string[10];{ ЕГН за заменяне }
  Begin
    Write('Въведете ЕГН: ');Readln(EGN);
    Reset(PromFl);
    While not EOF(PromFl) do
    begin
      Read(PromFl,Lice);
      If Lice.EGN=EGN
      then begin
           n:=FilePos(PromFl);
           ReadLice(Lice);
           Seek(PromFl,n);
           Write(PromFl,Lice);
           Writeln('Данните за лице с ЕГН ',EGN,' са заменени.');
           Break
           end;
    end;
  Close(PromFl);
  Writeln;
  End;
```

{Процедура за извеждане списък на родените през посочена година}

```
Procedure SpisGod;
  Var
    k,GodTrs,GodLice,Gr:integer;
  Begin
    Write('Задайте годината на раждане: ');Readln(GodTrs);
    Reset(PromFl); k:=0;
```

```
    While not EOF(PromFl) do
      with Lice do
      begin
        Read(PromFl,Lice);
        Val(Copy(EGN,1,2),GodLice,Gr);
        If Gr=0
        then Godlice:=GodLice+1900
        else begin
             Writeln('Некоректно ЕГН=',EGN);
             Exit
             end;
        If GodLice=GodTrs then begin k:=k+1;WriteLice(Lice) end
      end;
    If k=0 then Writeln('Няма лица, родени през ',GodTrs,' година.');
    Close(PromFl)
  End;
{Процедура за извеждане данните на всички регистрирани лица}
Procedure ObSpis;
  Var
    k:byte;
  Begin
    Reset(PromFl); k:=0;
    Writeln('Списък на всички лица:');
    While not EOF(PromFl) do
    begin
      Read(PromFl,Lice); k:=k+1; WriteLice(Lice)
    end;
    If k=0 then Writeln('Файлът е празен');readln;
    Close(PromFl)
  End;
{Главна програма}
BEGIN
  Write('Задайтето името на файла: ');Readln(ImeFl);
  Assign(PromFl,ImeFl);
  Repeat
    Writeln(' ':20,'МЕНЮ НА ИЗПЪЛНЯВАНИТЕ ОПЕРАЦИИ:');
    Writeln(' ':33,'1 - за създаване на нов файл');
    Writeln(' ':33,'2 - за добавяне на компоненти');
    Writeln(' ':33,'3 - за изтриване на компонента');
    Writeln(' ':33,'4 - за замяна на компонента');
    Writeln(' ':33,'5 - за списък на родените през посочена година');
    Writeln(' ':33,'6 - за извеждане данните на всички лица');
    Write(' ':20,'Посочете операция или 0 за край: ');Readln(C);
    Case C of
      1:SzdPrz;
      2:Dobaviane;
      3:Iztrivane;
      4:Zamiana;
      5:SpisGod;
      6:ObSpis
    end;
  until C=0
End.
```

С цел програмата да е по-кратка предположихме, че данните включват само име и ЕГН. Препоръчваме на читателя да разшири съдържанието на таблицата (записите), например както е в програма 10.1, и да допълни тази програма с подпрограми за извършване на следните дейности:

- за допълване записа за всеки студент със средния успех и броя слаби оценки на студента;
- извеждане на данните на студентите от посочена група;
- извеждане на данните на студентите от посочен курс, като се знае, че номерът на групата е трицифрено число и първата цифра указва курса;
- извеждане на данните на студент с посочено име;
- извеждане на данните на студент с посочен факултетен номер;
- извеждане списък на студентите със слаби оценки.

11.2. Текстови файлове

11.2.1. Определение

Текстовият файл представлява линейна последователност от символи, разделени на редове. Текстови файлове се създават с помощта на специализирани програми, наречени текстови редактори. Програмната среда Турбо Паскал притежава вграден текстов редактор и посредством него ние въвеждаме и коригираме текстовете на програмите на Паскал.

Данните, които въвеждаме от клавиатурата за операторите Read и Readln, също се разглеждат, като част от текстов файл и това е стандартният входен текстов файл със стандартното име **Input**, наречен клавиатурен файл. Данните, които извеждаме на екрана с операторите Write и Writeln също се разглеждат, като част от текстов файл и това е стандартния изходен текстов файл със стандартно име **Output**, наречен екранен файл.

Поради това, че в програмите често се работи с текстови файлове, в Турбо Паскал е въведен стандартният файлов тип **Text**. Всеки текстов файл се състои от редове от символи. Всеки ред завършва със специален символ (символът с код 13) - маркер за край на ред, евентуално следван от символа с код 10.

11.2.2. Операции с текстови файлове

От разгледаните по-горе процедури и функции за изпълняване на операции със структурирани файлове при текстовите файлове не могат да се използват само процедурата Seek и функциите FileSize и FilePos, т.е. променливата от тип текстов файл също трябва да се свърже с конкретен файл, текстовият файл също се създава като нов или като съществуващ, от него се чете, в него се пише и т.н.

По-особени обаче тук са операторите за въвеждане (четене) и извеждане (писане) на данни.

При четене на данни от текстов файл се използват операторите:

Read(Променлива от тип текстов файл, Входен списък);
Readln(Променлива от тип текстов файл, Входен списък);
Readln(Променлива от тип текстов файл);

Входният списък може да бъде такъв, какъвто се допуска при четене от клавиатурата, т.е. може да съдържа променливи от целочислен, реален и символен тип и от тип символен низ.

Очевидно операторите Read и Readln могат да различат целочислени, реални и символни стойности и такива от тип символен низ сред

последователността от символи в реда. Правилата, по които те се различават, бяха разгледани по-рано в глава 2.

Операторът Readln се отличава от Read по това, че при неговото изпълнение се четат от текущия ред толкова стойности, колкото изисква входният списък и се преминава на нов ред, като остатъкът от реда се прескача. Когато операторът е без входен списък, се прескача цял ред от файла.

При писане на данни в текстов файл се използват операторите:

Write(Променлива от тип текстов файл, Изходен списък);
Writeln(Променлива от тип текстов файл, Изходен списък);
Writeln(Променлива от тип текстов файл);

Изходният списък може да бъде такъв, какъвто се допуска при писане върху екрана, т.е. може да съдържа изрази от целочислен, реален, логически и символен тип и от тип символе низ. И тук важат правилата за извеждане на стойности на екрана, разгледани в глава 2.

Операторът Writeln се отличава от Write по това, че изпълнението му завършва с извеждане на символа край на ред. Когато операторът е без изходен списък, се извежда само символът край на ред.

Чрез операторите Write и Writeln може да се извежда текстов файл и на хартия. Необходимо е обаче в процедурата **Assign** да се посочи системното име на печатащото устройство (Lpt1 или PRN) като име на файла.

Ако в операторите Read и ReadLn първият параметър в списъка не е име на файл, подразбира се четене от клавиатурния файл **Input**. Ако в операторите Write и WriteLn първият параметър в списъка не е име на файл, подразбира се извеждане в екранния файл **Output**.

При текстовите файлове не съществува възможност за пряк достъп.

Допълнителни процедури и функции за операции само с текстови файлове са:

а) За отваряне на файла с цел добавяне на редове

За целта се използва процедурата:

Append(Променлива от тип текстов файл).

Тя отваря файла и позиционира файловия указател в края на файла.

б) За определяне достигнат ли е краят на реда

Използва се функцията от логически тип

EOLn(Променлива от тип текстов файл),

която връща стойност False, докато не е достигнат края на реда и стойност True, когато той е достигнат.

Програма 11.3. Програма за копиране на текст от файл InFile.txt във файл OutFile.txt, намиращи се в текущата директория.

```
Var
   FI, FO : text;
   C : string;
Begin
  Assign (FI,'InFile.txt');
  Assign (FO,'OutFile.txt');
  Reset(FI);
  Rewrite(FO);
  While not EOF (FI) do
    Begin
      Readln(FI,C); Writeln(FO,C)
```

```
    end;
  Close(FI); Close(FO);
End.
```

Програма 11.4. Програма, която преброява колко пъти се среща всяка главна буква и всяка малка буква в разказ, който се съдържа на български езиик във файл Razcaz.doc, като освен това го извежда на екрана. Масивът MasBrGlav е с индекс Simvol - от символен тип. Когато прочетеният като стойност на Simvol пореден символ е например Д, на елемента MasBrGlav['Д'] се увеличава стойността с 1, поради това, че е прочетена още една буква Д. Подобен е и масивът MasBrMalk, но той регистрира срещаните малки букви.

```
Const
  MnoGlavni : set of char =['А'..'Я']; {Множество на главните букви}
  MnoMalki  : set of char = ['а'..'я']; {Множество на малките букви}
Var
  FI : text;
  Simvol : char;
  MasBrGlav : array ['А'..'Я'] of integer;
  MasBrMalk  : array ['а'..'я'] of integer;
Begin
  Assign(FI,'Razcaz.doc');
  Reset(FI);
  For Simvol := 'А' to 'Я' do MasBrGlav[Simvol] := 0;
  For Simvol := 'а' to 'я' do MasBrMalk[Simvol] := 0;
  While not EOF (FI) do
  Begin
    While not EoLn(FI) do
    Begin
      Read(FI,Simvol); Write(Simvol);
      If Simvol in MnoGlavni
        then
          MasBrGlav[Simvol] :=MasBrGlav[Simvol] + 1;
      If Simvol in MnoMalki
        then
          MasBrMalk[Simvol] := MasBrMalk[Simvol] + 1
    end;
    ReadLn(FI); WriteLn;
  end;
  Close(FI);
  For Simvol := 'А' to 'Я' do Write(Simvol:4,' - ',MasBrGlav[Simvol],' пъти');
  Writeln;
  For Simvol := 'а'  to 'я' do Write(Simvol:4,' - ',MasBrMalk[Simvol],' пъти');
  Readln
End.
```

При тестване на нова програма е удобно данните, с които става тестването да се четат от текстов файл, подготвен предварително с текстовия редактор. Това спестява повторното им въвеждане от клавиатурата след всяка грешка в тестваната програма.

Програма 11.5. Програма за четене на данни за студенти от текстов файл и извеждане на прочетените данни на екрана.

В текстовия файл на един ред се съдържат данните за един студент в следния ред:

- име - символен низ - 20 позиции;
- ден, месец, година на раждане - цели числа с по една празна след всяко;

- фак. номер - символен низ - 6 позиции;
- курс - цяло число с 1 празна позиция след него;
- оценки - 10 цели числа с по 1 празна позиция след всяко;
- кодове на спортове - до 6 цели числа, разделени с празна позиция.

В оператора Read(InTxtFl,Ime,D,M,G,F,FakNom,Kurs) присъства символната променлива F, за да се прочете празната позиция пред символния низ факултетен номер.

```
Type
  TipStudent=record
               Ime:string[20];
               Rodata:record D,M,G:integer end;
               FakNom:string[6];
               Kurs:integer;
               Ocenki:array[1..10] of integer;
               MnoSport:set of 1..6;
               end;
Var
  Stud:tipStudent;
  InTxtFl:text;
Procedure ReadStud(Var Student:TipStudent);
Var
  i,KodSport:byte; F:char;
Begin
  With Student, Student.RoData do
  Begin
    Read(InTxtFl,Ime,D,M,G,F,FakNom,Kurs);
    For i:=1 to 10 do read(InTxtFl,Ocenky[i]);
    MnoSport:=[];
    While not EOLn(InTxtFl) do
    Begin
      Read(InTxtFl,KodSport);
      If KodSport in [1..6]
        then MnoSport:=MnoSport+[KodSport]
    end;
    Readln(InTxtFl)
  end
End;
Procedure WriteStud(Student:TipStudent);
Var
  i,KodSport:byte;
Begin
  With Student, Student.RoData do
  begin
    Write(Ime:10,D:3,M:3,G:5,FakNom:8,Kurs:2);write(' ');
    For i:=1 to 10 do write(' ',Ocenki[i]);write(' ');
    For KodSport:=1 to 6 do
      If KodSport in MnoSport then write(' ',KodSport);writeln;
  end
End;
Begin
  Assign(InTxtFl,'Studenti.pas');
  Reset(InTxtFl);
  While not EOF(InTxtFl) do
  begin
```

```
    ReadStud(Stud);
    WriteStud(Stud)
  end;
  Close(InTxtFl);
  Readln
End.
```

Програма 11.6. Програма, която с данните за група точки и група окръжности извършва следните операции:

- създава един файл с данните за окръжностите;
- създава втори файл с данните за точките;
- създава текстов файл, като всеки ред от текстовия файл съдържа номер на окръжност и номерата на точките, които лежат в тази окръжност;
- извежда на екрана данните от текстовия файл.

```
Type
  TipOkr=record
           x,y,r:real
         end;
  TipToc=record
           x,y:real
         end;
Var
  FlOkr:file of TipOkr;ImeFlOkr:string;
  FlToc:file of TipToc;ImeFlToc:string;
  TxtFl:text;ImeTxtFl:string;
  Okr:TipOkr; Toc:TipToc;Ch,C:char;
  NomOkr,NomToc:integer;
{1. Създаване на файл с данните за окръжностите}
Begin
  Write('Име на файла с данни за окръжности: ');Readln(ImeFlOkr);
  Assign(FlOkr,ImeFlOkr);Rewrite(FlOkr);
  Repeat
    with Okr do
    begin
      Write('x=');Readln(x);
      Write('y=');Readln(y);
      Write('r=');Readln(r);
    end;
    Write(FlOkr,Okr);
  Write('Ще продължите ли да въвеждате? ');Readln(Ch);
  until Ch in ['N','n','Н','н'];
  Close(FlOkr);
{2. Създаване на файл с данните за точките}
  Write('Име на файла с данни за точки: ');Readln(ImeFlToc);
  Assign(FlToc,ImeFlToc);Rewrite(FlToc);
  Repeat
    with Toc do
    begin
      Write('x=');Readln(x);
      Write('y=');Readln(y);
    end;
    Write(FlToc,Toc);
     Write('Ще продължите ли да въвеждате? ');Readln(Ch);
  until Ch in ['N','n','Н','н'];
```

```
  Close(FlToc);
{3. Създаване на текстов файл, съдържащ номерата на точките}
  Write('Име на текстовиа файл: ');Readln(ImeTxtFl);
  Reset(FlOkr);
  Assign(TxtFl,ImeTxtFl);Rewrite(TxtFl);
  NomOkr:=0;
  While not EOF(FlOkr) do
  begin
    Read(FlOkr,Okr);NomOkr:=NomOkr+1;Write(TxtFl,NomOkr,'   ');
    Reset(FlToc); NomToc:=0;
    While not EOF(FlToc) do
    begin
      Read(FlToc,Toc);
      NomToc:=NomToc+1;
      If sqr(Toc.x-Okr.x)+sqr(Toc.y-Okr.y)<=sqr(Okr.r)
      then Write(TxtFl,NomToc,' ');
    end;
    Writeln(TxtFl);Close(FlToc)
  end;
  Close(FlOkr);Close(TxtFl);
{Извеждане данните от текстовия файл на екрана}
  Reset(TxtFl);
  While not EOF(TxtFl) do
  begin
    Read(TxtFl,NomOkr,C,C,C); Writeln(NomOkr,'- окръжност');
    While not EOLn(TxtFl) do
    begin
      Read(TxtFl,NomToc,C);Write(NomToc,'  ')
    end;
    Readln(TxtFl);Writeln;
  end;
  Close(TxtFl);
  Readln
End.
```