10. ЗАПИСИ

10.1. Понятие за запис

Вече се запознахме със съставните типове данни масив и множество. Видяхме, че те създават редица улеснения при правене на програми. Те са съвкупности от еднотипни елементи. В редица случаи обаче за програмиста е удобно да съществува възможност да се обединяват и разнотипни елементи в съставни променливи. Например една информационна система за следене успеха и спортните интереси на студентите може да съдържа за всеки студент следните данни: име, факултетен номер, дата на раждане, курс, оценките от 10-те изпита през последната учебна година и множеството на любимите спортове измежду упражняваните в университета спортове: футбол, волейбол, баскетбол, хандбал, атлетика и плуване, зададени с техните кодове от 1 до 6.

Тези данни всъщност включват пет компоненти (име, дата на раждане, курс, оценки и интереси), които не са от еднакъв тип.

Езикът Паскал разрешава да разглеждаме данните за един студент като една съставна променлива от тип "**запис**" с избрано от нас име. Отделните компоненти на записа се наричат полета и имат свои имена. В разглеждания пример полетата са следните шест:

- име - променлива от тип **string**[25];
- факултетен номер - променлива от тип **string**[6];
- дата на раждане - запис с три полета (ден, месец, година) от тип integer;
- курс - променлива от тип integer;
- оценки - едномерен масив с 10 елемента от тип integer;
- интереси - множество с компоненти от тип integer.

10.2. Дефиниране на тип запис

Тип запис се декларира по описания в глава 7 начин за деклариране на нестандартни типове. Описанието на типа обаче сега има вида:

Record *Списък от компоненти* ***end****;*

Компонентите са разделени със символа "**;**". Всяка компонента има вида:

Списък от имена на полета : тип на полетата;

Имената на полетата в списъка са разделени със запетая.

За разглеждания пример ще имаме:

```
Type
  TipStudent = record
                 Ime : string[25];
                 FakNom : string[6];
                 RoData : record D, M, G : integer end;
                 Kurs : integer;
                 Ocenky : array [1..10] of integer;
                 MnoSport : set of 1..6
               end;
```

Типът на полетата RoData и Ocenky и MnoSport бихме могли да декларираме

предварително. Тогава по-горе щяха да присъстват имената на декларираните типове вместо описанията им.

Всяко поле може да бъде от кой да е от разглежданите в Паскал типове освен типа **File**, разглеждан по-нататък.

10.3. Константи от тип запис

Константите от тип запис се описват в раздела **Const** по начин, който може да се види от следния пример:

```
Const
  Stud1: TipStudent=(Ime:'Иван Иванов'; FakNom:983155;
                     RoData:(D:5;M:3;G:1977);Kurs:3;
                     Ocenky:(5,4,4,3,6,4,2,4,5,6); MnoSport:[1,2,3,4,5]);
```

От примера се вижда, че за всяко поле се дава името, двоеточие и стойността. Когато стойността е съставна (RoData, Ocenky и MnoSport), тя се поставя в скоби. Стойността на полето Ocenky, трябва да съдържа 10 цели числа, тъй като то е декларирано като масив с 10 целочислени елементи.

10.4. Променливи от тип запис

Променливи от тип запис се декларират в раздела **Var** по един от двата начина: чрез името на деклариран тип или чрез описание на типа, т.е.

```
Var
  A,B, Student : TipStudent;
```

или

```
  P,Q,R: record
           Ime : string [25];
           RoData : record D, M, G: integer end;
           FakNom:string[6];
           Kurs : integer;
           Ocenky : array [1..10] of integer;
           MnoSport: set of 1..6
         end;
```

В операторите на програмата полетата на записите могат да присъстват и чрез съставните си имена, които имат вида:

Име на променлива от тип запис.Име на поле

или

Име на променлива.Име на поле.Име на поле,

когато полето е поле на поле от тип запис.

Полетата на променливата A от тип TipStudent са със следните съставни имена:

A.Ime	име на студента;
A.FakNom	факултетен номер;
A.Data	име на записа, съдържащ датата на раждане;
A.Data.D	ден на раждане;
A.Data.M	месец на раждане;
A.Data.G	година на раждане;

A.Kurs	курс;
A.Ocenky	име на масива с оценките на студента;
A.Ocenky[1]	оценка по първата дисциплина;
A.Ocenky[2]	оценка по втората дисциплина и т.н.;
A.MnoSport	множество на любимите спортове на студента.

10.5. Оператор With

Съставното име в повечето случаи е твърде дълго. Многократното му писане в операторите обикновено отегчава програмистите, разсейва ги и в крайна сметка става причина за досадни грешки. Като изход от това, езикът Паскал предлага оператора **With**.

Общият вид на оператора е следния:

With *Списък от променливи от тип запис* ***do*** *Оператор;*

Променливите в списъка са разделени със запетаи.

Оператор най-често е съставен оператор. В границите на този оператор вместо съставните имена на полетата можем да използваме простите имена. Например, ако в списъка се съдържа променливата A, то вместо съставните имена A.Ime, A.Spec, A.Kurs, A.Ocenky можем да използваме простите Ime, Spec, Kurs, Ocenky и вместо по-сложното съставно име A.Data.D по-простото Data.D. Тук заслужава да отбележим, че променливите от един и същ тип, например A и B, имат полета с еднакви прости имена, и съвместното им използване в списък на оператор **With** може да доведе до объркване.

Пример:

```
With  A, A.RoData do
  Begin
    Ime := 'Иван Иванов';
    D := 1; M := 5; G := 1979;
    FakNom := '983155';
    Kurs := 1;
    Ocenky[1] := 6;
    Ocenky[2] := 5;
    Ocenky[3] := 2;
    . . . . . .
  end;
```

10.6. Операции със записи

10.6.1. Операции с константи и променливи от тип запис

Променливите от тип запис могат да участват само в оператора за присвояване, т.е. на променливата B от тип запис можем да присвоим стойността на променливата A от тип запис, но само когато двете са от един и същ тип. Например операторът

B := A;

е коректен, защото и A и B са от един и същ тип TipStudent.

10.6.2. Операции с полета на записи

Полетата на константите и променливите от тип запис могат да участват във всички операции, разрешени за техния тип, но със съставните си имена. (Нека си припомним, че и името на елемент на масив също е съставно - X[3]).

Примери за операции с полета на записи:

```
If A.Kurs = B.Kurs
    then Writeln(A.Ime,' и ',B.Ime,' са студенти от един и същ курс');
If A.Ocenky[1] = 6
    then Writeln(A.Ime,' ',A.FakNom,' ',A.Kurs);
```

10.6.3. Въвеждане и извеждане на променливи от тип запис

- Променливите от тип запис се въвеждат и извеждат поле по поле и подполе по подполе, тъй като, както знаем, от стандартното входно устройство могат да се въвеждат и на стандартното изходно устройство могат да се извеждат само целочислени, реални и символни стойности и стойности от тип символен низ. Обикновено е по-удобно програмистът да си подготви отделни процедури за въвеждане и извеждане на записи. По-долу е дадена примерна програма с процедури за въвеждане и извеждане на записи.

Програма 10.1. В таблица се съдържат данни за група лица, които включват.

- име;
- факултетен номер;
- дата на раждане;
- административна група;
- оценки по 10 дисциплини;
- множество от любими спортове (до 6 спорта).

Програмата по-долу въвежда данните за група студенти, изчислява средния успех на всеки студент и извежда данните за студентите на екрана.

В програмата данните за студентите са представени като масив от записи, т.е. всеки елемент от масива е запис, който съдържа данните за един студент, включително и неговия среден успех.

```
Const
  Sports:array[1..6]of string[9]=          {Масив с имената на спортовете}
  ('футбол','волейбол','баскетбол','хандбал','атлетика','плуване');
Type
  TipStudent=record       {Тип запис с данните за един студент}
             Ime:string[20];
             FakNom:string[6];
             RoData:record D,M,G:integer end;
             Grupa:byte;
             Ocenki:array[1..10] of byte;
             MnoSport:set of 1..6;
             Uspeh:real
       end;
Var
  i,j,BrStud,s:byte;
  MasStud:array[1..25] of TipStudent;     {Масив с данните за всички студенти}
{Процедура за въвеждане на данните за един студент}
```

```
Procedure ReadRec(Var Student:TipStudent);
  Var
    i,KodSport:byte;
  Begin
    With Student, Student .RoData do
    begin
      Write('Въведете името: ');Readln(Ime);
      Writeln('Задайте дата на раждане');
      Write('Ден:');Readln(D);
      Write('Месец:');Readln(M);
      Write('Година:');Readln(G);
      Write('Задайте фак. номер:');Readln(FakNom);
      Write('Задайте групата:');Readln(Grupa);
      Writeln('Задайте оценките по 10-те дисциплини!');
      For i:=1 to 10 do
      begin
        Write(' По ',i,'-а дисциплина:');Readln(Ocenki[i])
      end;
      MnoSport:=[];
      Writeln('Задайте множеството на любимите спортове!');
      Repeat
        Writeln(' ':20,'Кодове на любимите спортове:');
        Writeln(' ':25,'1-футбол;');
        Writeln(' ':25,'2-волейбол;');
        Writeln(' ':25,'3-баскетбол;');
        Writeln(' ':25,'4-хандбал;');
        Writeln(' ':25,'5-атлетика;');
        Writeln(' ':25,'6-плуване.');
        Write(' ':15,'Посочете спорт или 0 за край:');Readln(KodSport);
        If KodSport in [1..6] then MnoSport:=MnoSport+[KodSport]
      until KodSport =0
    end
End;
{Процедура за извеждане на данните за един студент}
Procedure WriteRec(Var Student:TipStudent);
  Var
    i,KodSport:byte;
  Begin
    With Student, Student .RoData do
    begin
      Writeln('Име: ',Ime);
      Writeln('Дата на раждане:',D,'.',M,'.',G);
      Writeln('Факултетен номер:',FakNom);
      Writeln('Група:',Grupa);
      Write('Оценки: ');
      For i:=1 to 10 do write(Ocenki[i],' ');
      Writeln;
      Write('Любими спортове: ');
      For KodSport:=1 to 6 do
      if KodSport in MnoSport then Write(Sports[KodSport],' ');
      Writeln;
```

```
      Writeln('Среден успех: ',Uspeh:4:2);
    end;
  End;
{Главна програма}
Begin
  Write('Задайте броя на студентите:');Readln(BrStud);
  For i:=1 to BrStud do ReadRec(MasStud[i]);
  For i:=1 to BrStud do with MasStud[i] do
  begin
    s:=0;
    For j:=1 to 10 do s:=s+Ocenki[j];
    Uspeh:=s/10
  end;
  For i:=1 to BrStud do begin Writeln;WriteRec(MasStud[i]); Readln end;
End.
```

Препоръчваме на читателя 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, чиято първа цифра указва курса;
- извеждане на данните на студент с посочено име;
- извеждане на данните на студент с посочен факултетен номер.
- сортиране на масива по успеха на студентите;
- извеждане списък на любителите на определен спорт;
- извеждане името на спорта с най-много почитатели и др.

Програма 10.2. Програма, която извършва следните операции с данните за група окръжности и група точки:

- въвежда данните за окръжностите;
- въвежда данните данните за точките;
- намира броя на точките, които съдържа всяка окръжност;
- намира номерата на окръжностите, в които лежат макс. брой точки;
- сортира окръжностите по броя на съдържащите се в тях точки.

```
Type
  TipOkr=record x,y,r:real;Br:integer end;
  TipToc=record x,y:real end;
Var
  Okr:array[1..20] of TipOkr;
  T:TipOkr;
  Toc:array[1..25] of TipToc;
  BrOkr,BrToc,i,j,k:integer;
Begin
{Въвеждане данните за окръжностите}
  Write('Задайте броя на окръжностите: ');Readln(BrOkr);
  For i:=1 to BrOkr do with Okr[i] do
  begin
    Write('x=');Readln(x);
```

```
    Write('y=');Readln(y);
    Write('r=');Readln(r);
  end;
{Въвеждане данните за точките}
  Write('Задайте броя на точките: ');Readln(BrToc);
  For j:=1 to BrToc do with Toc[j] do
  begin
    Write('x=');Readln(x);
    Write('y=');Readln(y);
  end;
{Намиране броя на точките лежащи във всяка окръжност}
  For i:=1 to BrOkr do with Okr[i] do
  begin
    Br:=0;
    For j:=1 to BrToc do
    if sqr(Toc[j].x-x)+sqr(Toc[j].y-y)<=sqr(r)
    then Br:=Br+1
  end;
{Извеждане броя на точките, лежащи във всяка окръжност}
  Writeln('Брой на точките, лежащи в окръжностите:');
  For i:=1 to BrOkr do with Okr[i] do
    writeln(i,'  ',Br);
{Намиране номера на първата окръжност, съдържаща максимален брой точки}
  k:=1;
  For i:=2 to BrOkr do
  if Okr[i].Br > Okr[k].Br then k:=i;
{Извеждане номерата на окръжностите, съдържащи максималния брой точки}
  Writeln('Максимален брой точки (',Okr[k].Br,') съдържат следните окръжности:');
  For i:=k to BrOkr do
    if Okr[i].Br = Okr[k].Br then writeln(i);
{Сортиране окръжностите по броя на съдържащите се в тях точки}
  For i:=2 to BrOkr do
  begin
    j:=i; T:=Okr[i];
    While (j>1) and (T.Br<Okr[j-1].Br) do
    begin
      Okr[j]:=Okr[j-1]; j:=j-1
    end;
    Okr[j]:=T;
  end;
  Writeln('Списък на окръжностите след сортиране');
  For i:=1 to BrOkr do with Okr[i] do
    writeln(i,'  ',x:7:2,y:7:2,Br:7);
  Readln
End.
```

Програма 10.3. Даден е масив Mas, съдържащ имена на лица, някои от които се повтарят по няколко пъти. По-долу е дадена програма, която с помощта на специална процедура определя по колко пъти се среща всяко име в дадения масив с имена. Процедурата създава нов масив MasZap от тип запис, като всеки запис има

две полета Ime и Br. Първото поле съдържа име от дадения масив, а второто поле показва колко пъти се среща това име в дадения масив.

```
Type
  TipMas=array[1..10] of string[10];
  TipZapis=record
                    Ime:string[10];
                    Br:byte
           end;
  TipMasZap=array[1..10] of TipZapis;
Var
  i,k,n:byte;
  Mas:TipMas;
  MasZap:TipMasZap;
Procedure BrEdnElem(n:byte;Mas:TipMas; Var MasZap:TipMasZap;
                                                    Var k:byte);
  Var
    i,j:byte;
  Begin
    i:=1;
    k:=1;
    MasZap[k].Ime:=Mas[i];
    MasZap[k].Br:=1;
    For i:=2 to n do
    begin
      j:=1;
      While (j<=k) and (Mas[i]<>MasZap[j].Ime) do j:=j+1;
      If j>k
        then begin
               k:=k+1;
               MasZap[k].Ime:=Mas[i];
               MasZap[k].Br:=1
             end
        else MasZap[j].Br:=MasZap[j].Br+1
  end
End;
Begin
  Write('n=');Readln(n);
  For i:=1 to n do readln(Mas[i]);
  BrEdnElem(n,Mas,MasZap,k);
  For i:=1 to k do Writeln(MasZap[i].Ime:10,' ',MasZap[i].Br);
  Readln
End.
```

10.7. Вариантни записи

Типът запис Student, деклариран по-горе, е инвариантен, защото всички променливи от този тип имат:

- еднакъв брой полета;
- еднакви имена и типове на съответните полета.

Практиката налага, а и езикът Паскал разрешава, да се работи и със записи, които нарушават това правило. Такива записи се наричат *вариантни*.

Един пример е литературната справка по дадена тема, която с оглед на компютърна обработка искаме да представим като един масив от записи. Тя представлява списък от книги и публикации в специализирани списания по темата. Всяка книга е представена в списъка с името на автора, заглавието, годината на издаването, името на издателя и името на града, в който е издадена. Всяка публикация е представена с името на автора, заглавието, годината на публикуването, името на списанието, номера на книжката. Както се вижда, три от показателите (автор, заглавие, година) са общи за двата вида обекти, а останалите два са различни. Такива обекти се описват с вариантни записи.

Вариантният запис се състои от две части - *инвариантна част и вариантна част*. Общият вид на описанието му е:

Record

Списък на компонентите на инвариантна част;

Case Селектор ***of***

конст1 : (Списък на компонентите на Продължение1);

конст2 : (Списък на компонентите на Продължение2);

. . . . . .

констN : (Списък на компонентите на ПродължениеN)

End*;*

10.7.1. Инвариантна част

Тя съдържа описанието на общите полета за всичките вариантни записи. Такива в примера са полетата автор, заглавие и година. Тя се оформя като инвариантен запис, но без **end**, защото има продължение.

10.7.2. Вариантна част

Тя описва вариантните продължения и се оформя като оператор **Case**.

Типът на селектора може да бъде **char**, **boolean** или предварително дефиниран нестандартен дискретен тип с не повече от 256 елемента.

Заглавията *(конст1, конст2, . . ., констN)* на вариантните продължения, са "*етикети*" в оператора **Case** и следователно трябва да са допустими стойности за *Селектора*. Селекторът може да бъде зададен в два варианта:

Име на селектора : Тип на селектора

В този случай инвариантната част се допълва автоматично с едно поле, което има името и типа на Селектора. При създаването на всеки запис, то следва да се запълни с една от допустимите за Селектора стойности (Kniga или Spisanie). Съдържанието на това поле се използва за разпознаване на записите при обработката им.

Тип на селектора

В този случай инвариантната част не се допълва автоматично с поле, което има името и типа на Селектора. Ако е необходимо, програмистът сам може да я допълни с такова поле, но с избрано от него име.

За всяка променлива от даден вариантен запис се отделя в оперативната памет толкова място, колкото е необходимо за инвариантната част и на най-дългото инвариантно продължение на този тип.

По-удобно е останалите разяснения да се дадат посредством следния пример:

```
Type
  VidLitIzt = (Kniga,Spisanie);
  LitIzt =  record
      Avtor, Zaglavie : string;
      God : 1800..1999;
      Case UkVida : VidLitIzt of
      Kniga : (Izdatel, Grad : string);
      Spisanie : (Ime : string; Nomer : integer)
    end;
```

Инвариантната част се състои от три полета.

Вариантните продължения са две - Kniga и Spisanie. В оператора **Case** те се използват като “етикети” на операторите, декларащи полетата на вариантните продължения.

Селекторът UkVida : VidLitIzt е причина за автоматичното деклариране на поле с име UkVida. Съдържанието на това указващо поле е грижа на програмиста и указва каква е вариантната част на записа. Например, ако списъкът от литературни източници е изграден като масив, деклариран както следва

```
Var Lit = array [1..500] of LitIzt;
```

а елементът Lit[I] описва книга, то програмистът трябва да се погрижи полето Lit[I].UkVida да получи стойност Kniga. Ако обаче елементът Lit[I] описва списание, то тогава същото поле трябва да има стойност Spisanie.

Съдържанието на указващото поле се използва както е показано в следния оператор:

```
With Lit[I] do
  Begin
    Write(Avtor,' ',Zaglavie,' ',God,' ');
    If UkVida = Kniga
      then Writeln(Izdanie,' ',Grad)
      else Writeln(Ime,' ',Nomer);
  end;
```

Допуска се селекторът да бъде от вида VidLitIzt, т.е. без указателно поле. В този случай програмистът може да декларира явно в инвариантната част указателно поле например със същото или с друго име. Ако обаче не направи това, трябва да си осигури други външни средства за разпознаване на записите - например нечетните са книги, а четните списания и т.н.

Всички полета, както в инвариантната част, така и във вариантната част, трябва да имат различни имена.

Програма 10.4. Програма, която извършва следните операции:

1. Въвежда следните данни за преподаватели и студенти:

- за преподаватели: собствено име, фамилно име, факултет, катедра, трудов стаж;

- за студенти: собствено име, фамилно име, факултет, факултетен номер, група, общ успех

2. Извежда поотделно списък на студентите и преподавателите.

Данните за едно лице се разглеждат като вариантен запис, защото са различни за студентите и преподавателите. Съвкупността от данните за всички лица е представена в програмата като масив от записи. Един елемент от масива съдържа данните за един студент или един преподавател.

```
Type
  Lice=record
         Ime,Fam:string[10];
         Fk:string[6];
         Case Vid:char of
         'S','s','C','c': (Fn:string[6];Gr:byte;Usp:real);
         'P','p','П','п': (Kat:string[10];St:byte)
       end;
Const
  Stud:set of char=['S','s','C','c'];
  Prep:set of char=['P','p','П','п'];
Var
    i,K,n:integer;
    S:string;Ch:char;
    Lica:array[1..30] of Lice;
Begin
    i:=0;
    Repeat
      i:=i+1;
      With Lica[i] do
        begin
          Write('Име: '); Readln(Ime);
          Write('Фамилия: '); Readln(Fam);
          Write('Факултет: '); Readln(Fk);
          Repeat
            Write('Студент или преподавател?' ); Readln(Vid);
          until Vid in Stud + Prep;
          Case Vid of
          'S','s','C','c':begin
                             Write('Фак. ном.: '); Readln(Fn);
                             Write('Група: '); Readln(Gr);
                             Write('Ср. успех: '); Readln(Usp);
                          end;
       'P','p','П','п':begin
                             Write('Катедра: '); Readln(Kat);
                             Write('Тр. стаж: '); Readln(St)
                          end
        end;
        Write('Ще продължите ли въвеждането?');Readln(Ch);
      end
    until Ch in ['N','n','Н','н'];
    n:=i; Writeln;Writeln;
    Writeln; Writeln(' ':11,'Списък на студентите и преподавателите');
    Writeln; Writeln(' ':20,'1. Студенти:');
    Writeln;
```

```
        Writeln('    Име и фамилия        Факултет  Фак.ном.   Група  Успех');
        K:=0;
        For i:=1 to n do With Lica[i] do
          if Vid in Stud
          then begin
                 K:=K+1; S:='.  '+Ime+' '+Fam;
                 Writeln(K:2,S,' ':26-Length(S),Fk,' ':10-Length(Fk),Fn,' ':4,Gr:3,Usp:9:2)
               end;
        Writeln;Writeln; Writeln(' ':20,'2. Преподаватели:');
        Writeln; Writeln('    Име и фамилия        Факултет   Катедра     Стаж');
        K:=0;
        For i:=1 to n do With Lica[i] do
          if Vid in Prep
          then begin
                 K:=K+1;
                 S:='.  '+Ime+' '+Fam;
                 Writeln(K:2,S,' ':26-Length(S),Fk,' ':11-Length(Fk),Kat,'
                                                    ':11-Length(Kat),St:3);
               end;
        Writeln;
        Readln
End.
```