13. Записи (Records).

Как мы уже знаем, массив представляет совокупность элементов одного типа. *Запись (Record)* — это также структурированный тип данных, который позволяет представить группу элементов различных типов. В частном случае, запись может быть также совокупностью однотипных элементов.

Рассмотрим пример. Пусть мы хотим создать запись данных, которая бы хранила имя и оценку студента. Имя студента можно представить строкой, а оценку — целым числом. Мы могли бы использовать два раздельных массива для представления этой информации. Но в общем случае может понадобиться более сложная информативная структура о студентах, включающая данные о них: год рождения, номер группы, пол и др. Определять для каждой характеристики новый массив — не самый удачный и удобный способ. Запись позволяет это сделать более легким путем. Для этого надо определить группу данных и объявить новый тип (запись), затем создать рабочую переменную данного типа. Такое описание может иметь вид:

```
Type studentname = String[20];
     studentinfo = Record
                     name: studentname;
                     mark : integer
     End;
Var student: studentinfo;
```

В разделе Type объявляется тип *studentinfo,* который состоит из двух частей, т.н. *полей.* Первое поле записи — это строка, определяемая идентификатором *name.* Второе поле, определяемое идентификатором *mark,* может представлять целое число.

Объявление типа—записи начинается с ключевого слова *Record* и завершается словом *End.*

В разделе *Var* объявляется рабочая переменная *student,* которая имеет структуру типа *studentinfo.*

К каждому индивидуальному полю можно обращаться по формату:

имя – записи.имя – поля

Например:

```
studentname := 'Kate Smith'; { максимум 20 символов }
student.mark := 4;
```

Можно определить тип для представления даты. В этом случае запись будет объединять группу однотипных элементов:

```
Type date = Record
    day : integer;
    month : integer;
    year : integer
End;
```

или же:

```
Type date = Record day, month, year: Integer

    End;
```

Это объявление определяет новый тип *date,* состоящий из трех базовых элементов целого типа. Теперь объявим рабочие переменные для представления даты, которые будут иметь структуру типа данных *date:*

```
Var toDay, anyDay : date;
```

Следующий фрагмент определяет значения отдельных составляющих элементов переменной — записи *today:*

```
today .day :=21;
today.month := 07;
today .year := 1998;
```

Общий формат присвоения значений элементам переменной—записи, т.е. полям записи, имеет вид:

имя – записи . имя – поля: = значение

Можно присвоить значение переменной в целом, т.е. допускается оператор:

```
anyday := today;
```

13.1. Записи и процедуры.

Когда процедуре передается запись в качестве параметра, как и в случае массивов, его тип должен быть предварительно объявлен в разделе Type.

Процедуре *distance* задаются координаты двух точек, а она выдает расстояние между ними:

```
Program distance;
Uses Crt;
Type
{Для представления координат точек}
point = Record
      x,y:Real;
      End;
Var a,b. point;
      z:Real;
      c:Char;

Procedure dist(x,y:point; Var z:Real);
Begin
      z:=Sqrt( Sqr(x.b-x.a) + Sqr(y.b-y.a) );
End;
{Главная программа}
Begin
      Writeln('Enter (X,Y) coordinates of the points A and B:');
      Read(x.a, x.b, y.a, y.b);
      dist(x,y,z);
      Writeln('The distanse is:', z:8:3);
      c:=Readkey;
End.
```

Следующий пример также демонстрирует использование записей в качестве параметров процедуры. Программа запрашивает два мнимых числа и выдает их сумму, разность, произведение или разность по выбору:

```
Program Imagine_math;
Uses Crt;
Type imagine = Record
     a,b:Real;
End;
Var x,y,z:Imagine;
     c:Char;

Procedure add_im(x,y:Imagine; Var z:Imagine);
Begin
     z.a := x.a + y.a;
     z.b := x.b + y.b;
End; { add_im }

Procedure subt_im(x,y:Imagine; Var z:Imagine);
Begin
     z.a := x.a - y.a;
     z.b := x.b - y.b;
End; {subt_im}

Procedure mult_im(x,y:Imagine; Var z:Imagine);
Begin
     z.a := x.a * y.a + x.b * y.b;
     z.b := x.b * y.a + x.a * y.b;
End; {mult_im}



Procedure div_im(x,y:Imagine; Var z:Imagine);
Uses Crt;
Var c:Char;
     denominator:Real;
```

```
Begin
      denominator:=Sqr(y.a) - Sqr(y.b);
      If denominator<>0
      Then Begin
           z.a := (x.a * y.a + x.b * y.b) / denominator;
           z.b := (Sqr(y.a)+Sqr(y.b)) / denominator;
       End
      Else Begin
           Writeln(' Деление недопустимо!');
           Writeln(' Нажмите на любую клавишу.');
           c:=ReadKey;
       End;
End; {mult_im}

Procedure enterNN(Var x,y: imagine);
Begin
      Writeln(' Введите два компонента первого числа:');
      Read(x.a, x.b);
      Writeln(' Введите два компонента второго числа:');
      Read(y.a, y.b)
End; {enterNN}

BEGIN {MAIN}
Repeat
      ClrScr;
      Writeln('Операции на мнимых числах:');
      Writeln('+ сложение ');
      Writeln('- разность');
      WriteIn('* умножение');
      Writeln('/ деление');
      Writeln('0 выход');
      Writeln;
      Writeln('Выберите вариант ');
      ch:=ReadKey;
      Case ch of
      '+': Begin enterNN(x,y);
                 add_im(x,y,z);
```

```
    End;
    '-': Begin enterNN(x,y);
                subt im(x,y,z);
    End;
    '*': Begin enterNN(x,y);
               mult_im(x,y,z);
    End;
    '/': Begin enterNN(x,y);
          div_im(x,y,z);
    End;
    '0': Halt;
    End;
    Writeln('=',z.a,'+i',z.b,'.');
  Until ch='0'
  END. {MAIN}
```

13.2. Массивы записей.

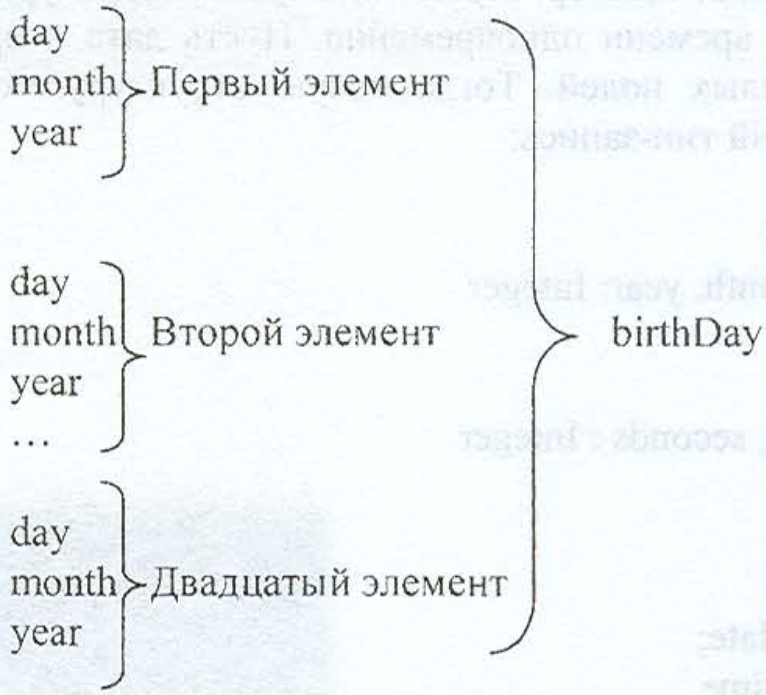
Массивы записей могут быть сформированы точно так же, как и массивы любых других типов. Пусть задана запись:

```
Type date = Record
                    day, month, year: Integer
                End;
```

Теперь определим новый тип, массив записей *date:*

```
Var birthdays : Array [1..20] of date;
```

Это объявление задает массив из 20 элементов, каждый из которых представляет дату, состоящую, в свою очередь, из трех элементов.



Формат присвоения значений элементам массива записей можно продемонстрировать на примере:

```
birthdays[l]. month := 1;
birthdays[l].day := 17;
birthdays[l].year := 1962;
birthdays[2].year := birthdays[l].year;
```

и т.д.

Вывод на экран можно организовать с помощью цикла:

```
For i:=l to 20 Do Writeln(birthdays[i].day,'.',
                          birthdays[i].month,'.',
                          birthdays [i].year);
```

13.3. Записи, содержащие записи.

Рассмотрим случай, когда запись в качестве полей содержит другие записи. Пусть задан тип *point* для представления координат точек на плоскости и с его помощью определим два новых типа для представления треугольников и прямоугольников:

```
Type point = Record x,y:Integer End
     Triangle = Record a,b,c:point End;
     Rectangle = Record a,b,c,d:point End;
```

Рассмотрим еще один пример. Пусть нам нужна структура для представления даты и времени одновременно. Пусть дата и время заданы как записи целых полей. Тогда новую структуру можно объявить как следующий тип-запись:

```
Type date = Record
              day, month, year: Integer
            End;
time = Record
         hour, minute, seconds : Integer
       End;

date_time = Record
              sdate: date;
              stime: time
            End;
Var today .tomorrow: date_time;
```

Таким образом, новый тип записи содержит два поля, оба типа записи. Переменные *today, tomorrow* являются носителями структурных данных типа *date_time.* Определение значения для переменной можно осуществить с помощью следующих присвоений:

```
today.sdate.day := 17;
today. sdate.month := 1;
today.sdate.year := 1998;
today.stime.hours := 3;
today.stime.minutes := 30;
today.stime.seconds := 00;
```

что определяет дату и время: 17 января 1998 года, три часа, тридцать минут и ноль секунд.

13.4. Записи, содержащие массивы.

Запись в качестве поля может содержать массив. Следующий тип позволяет хранить важные комментарии по месяцам (до 255 символов) для каждого года:

```
Type months(January, Fabruary, March, April, May, June, July,
            August, September, Octomber, November, December);
             year = Record
             year_number:Integer;
             comments:Array [January.. December] of String;
            End;
Var yl,y2:year;
```

Пример обращения к элементам массива:

```
yl .year_number:= 1998;
yl.monthst[June]:' Защитил диссертацию;';
yl .months[July]:='Начал работать.';
yl .months[September]:='Женился.';
yl.months[September]:=yl.months[September] +
                      ' + Ушел с работы.';
```

Естественно, приведенный способ хранения данных– не самый удобный, но он дает представление, как можно использовать записи для организации базы данных.

13.5. Оператор «With».

Чтобы обращаться к полям записи, надо полностью указывать идентификатор, представляющий переменную-родитель. Оператор *With* предоставляет более удобный способ обращения к отдельным элементам-полям записи без использования точки-разделителя. Например, чтобы присвоить переменной *today* типа *date* дату: 21 Апреля, 1998 года, надо написать:

```
With today Do
Begin
    day:=21;
    month:= 4;
```

```
        year:= 1998;
    End;
```

Теперь рассмотрим пример, где запись используется для представления и обработки данных о студентах. Сначала объявляется тип записи *student,* который может сохранить следующие данные: фамилия, имя, курс (год обучения), номер группы, специальность, возраст и пол (который, в свою очередь, описывается перечисляемым типом). Затем вводятся две переменные для представления данных о двух студентах (в общем случае можно использовать массив). Данные о первом студенте вводятся с клавиатуры, о втором студенте данные сначала берутся от первого студента прямым присвоением, а затем переопределяются некоторые поля. В конце выводятся на экран данные о двух студентах с помощью оператора *With* (как и в случае ввода данных о первом студенте).

Программный код будет иметь вид:

```
Program testWithRecords;
Type gender = (Male, Female);
    student = Record
            surname,name: String[20];
            course,group,age: Integer;
            sex: gender;
            speciality: String[ 10]
          End;
Var student 1,student2 : student;
Begin
{ Определение значений для первой записи }
            With student 1 Do
                Begin
                surname:='Joana';
                name:='Smith';
                        course :=1;
                        group:=378;
                        speciality:='CS505';
                        age:=17;
```

```
                    sex:==female;
End;
{ Определение значений для второй записи }
student2:= student1;
{ Переопределение значений некоторых элементов }
student2.name:='Iohan';
student2.surname:=' Bach';
student2.sex:=male;
{ Вывод данных на экран }
Writeln(' Данные о двух студентах:');
Writeln('Имя': 10, 'Фамилия:': 15, 'Курс':7,' Группа:':7,
        'Специальность:': 15,' Возраст: ':15,' Пол:':7);
With student1 Do
     Writeln(name:10, surname:15, course:15, group:15,
         speciality: 15, age: 15);
     Case sex of
         male: Write(' Женский ':9);
         female: Write(' Мужской ':9);
       End;
     End;
     Writeln;
     With student2 Do
              Writeln(name:10, surname: 15, course: 15, group:15,
              speciality: 15, age: 15);
              Case sex of
         male: Write(' Женский ':9);
         female: Write(' Мужской ':9);
      End;
     End;
     Writeln;
End.
```