Лекция 16. Структурированный тип данных: «**Запись**»

Записи – это один из структурированных типов в Delphi.
Раньше мы рассматривали другой структурированный тип – массивы

Массив – совокупность **однородных** элементов, хранящихся в **смежных** ячейках памяти, имеющих общее **имя** и различающихся по **индексам**.

Тип	**Массив**	**Запись**
Тип элементов	**Однотипные** элементы	Могут быть **разных** типов
Имя	Общее для всех элементов	Общее для всех элементов
Доступ/Обращение к элементам	По **индексу** элемента: **Имя_массива[индекс]**	По имени **поля:** **Имя_записи.Имя_поля**
Память	Смежные ячейки	Смежные ячейки, но (по умолчанию) с выравниванием по 4 или 8 байт (неплотно - пример на 6-7 страницах)

Таким образом,
Запись – совокупность **не**однородных элементов, хранящихся в **соседних** ячейках памяти, имеющих общее **имя** и различающихся по **именам полей**.

Запись

(по этой схеме рассматривали остальные типы в первых лекциях)

1) **(Запись –) Пользовательский тип**, а значит, как и для массива, (обычно) перед использованием надо определить тип в разделе **type**

Для использования переменной типа запись в качестве **параметра процедуры (функции)** – определение типа обязательно (как и для статического массива),
а также если надо описать переменные одного типа в разных местах программы, но чтобы работало **присваивание.**

2) **Определение типа и описание переменных**

```
Type //в разделе type
  Trec =   record  // ключевое слово record
             a, b: integer; // поля с одинаковым типом через «,» можно
             c: real; // могут быть разные типы  a,b,c – имена полей
           end; // описание записи заканчивается словом end

Var // теперь опишем переменную
  Rec1: Trec;
  Rec2: Trec;
  Rec2_2, Rec2_3: record  // похожа на Trec, но не она по имени
                    a, b: integer;
                    c: real;
                  end;
  Rec3:    record  // в некоторых случаях можно и без определения типа
                  N: integer;
                  A: array [1..20] of real; // элементом записи может быть и массив
                                            // и другая запись , и  строка
           End;
```

```
Rec4: record
        Data: record // полем записи может быть запись
                d,m,g: integer;
              End;
        N: string;
  End;
```

3) Ввод значений

Значение каждого поля вводится отдельно (как и в массиве) (Исключением будет ввод и вывод из/в типизированные файлы – это изучаем во втором семестре)

```
Writeln('Введите Rec3:');
Write('n=?'); readln(Rec3.n);
Write('Введите массив A:');
For i:=1 to n do
    Readln(Rec3.A[i]);


Writeln('Введите Rec1:');
Write('a=?'); readln(Rec1.a);
Write('b=?'); readln(Rec1.b);
Write('c=?'); readln(Rec1.c);


Writeln('Введите Rec4:');
Write('День=?');
readln(Rec4.Data.d);
Write('Месяц=?');
readln(Rec4.Data.m);
Write('Год=?');
readln(Rec4.Data.g);
Write('Имя=?'); readln(Rec4.N);
```

чтобы не упоминать каждый раз имя записи можно использовать оператор **WITH, Например**, изменим код, указанный слева для *Rec*1 и *Rec*4

```
Writeln('Введите Rec1:');
With Rec1 do
begin
        Write('a=?'); readln(a);
        Write('b=?'); readln(b);
        Write('c=?'); readln(c);
End;

Writeln('Введите Rec4:');
With Rec4.Data do
begin
    Write('День=?'); readln(d);
    Write('Месяц=?'); readln(m);
    Write('Год=?'); readln(g);
End;
Write('Имя=?'); readln(Rec4.N);
```

4) Присваивание значений

Каждому полю отдельно

```
Rec4.Data.d:=31;
```

Или запись целиком (только при **точном совпадении типа по <u>его имени</u>**, а не по размеру или структуре!)

```
Rec2:=Rec1;  {тип TRec}
Rec2_2:=Rec2_3; {Неименованный тип}
Rec2:= Rec2_2;  // НЕЛЬЗЯ! Разные типы, хоть и близнецы-братья
```

5) Особые операции

Использование конструкции **With** Имя_записи **do**

С полями операции производятся соответственно типу каждого поля

6) Вывод значений

как и ввод – *каждое поле отдельно* ((типизир. файлы – будут исключением – см 2 семестр))

```
Writeln('Rec4:');
With Rec4.Data do
begin
    Write('День= ', d);
    Write('Месяц= ', m);
    Write('Год= ', g);
End;
```

Write('Имя=?'); readln(**Rec4.N**);

7) **Примеры программ**

Пример 1.
Есть информация о 10 людях: Фамилия, имя, отчество, дата рождения, рост
Найти ФИО и рост самого старшего и самого младшего.

Используем массив из 10 записей типа **TChelovek**:

```
Const N=10;
Type
  TChelovek = record
        Fio: record
               F,I,O: string[20];
             End;
        Data: record
               d,m: byte;
               g: word;
             End;
       Rost: real;
  End;
  Massiv = array [1..N] of TChelovek;
```

```
Procedure FindOldHeight(var Mas: Massiv; const n: integer; out Old: TChelovek);
Var i: integer;
Begin
  Old:=Mas[1];
  For i:=2 to n do
     If (Mas[i].Data.g < Old.Data.g) OR
        (Mas[i].Data.g = Old.Data.g) AND (Mas[i].Data.m < Old.Data.m) OR
        (Mas[i].Data.g = Old.Data.g) AND (Mas[i].Data.m = Old.Data.m) AND (Mas[i].Data.d < Old.Data.d)
     Then
        Old:=Mas[i];
End;

Procedure FindYoungHeight(var Mas: Massiv; const n: integer; out Young: TChelovek);
Var i: integer;
Begin
  Young:=Mas[1];
  For i:=2 to n do
    With Mas[i].Data do
     If (g > Young.Data.g) OR
        (g = Young.Data.g) AND ((m > Young.Data.m) OR
                                (m = Young.Data.m) AND (d > Young.Data.d))
     Then
        Young:=Mas[i];
End;
```

```
Var
  Mas: Massiv;
  Max, Min: TChelovek;
  i: integer;

Begin
```

```
  For i:=1 to N do
  Begin
    Writeln(I,'- Введите Фамилию'); readln(Mas[i].Fio.F);
    Writeln('Введите Имя'); readln(Mas[i].Fio.I);
    Writeln('Введите Отчество'); readln(Mas[i].Fio.O);

    Writeln('Введите день месяц и год через пробел');
    with mas[i].data do readln(d, m, g);

    Writeln('Введите рост'); readln(Mas[i].Rost);
  End;

  FindOldHeight( mas, n, Max);
  With Max.Fio do WriteLN('Самого старшего зовут ', F, ' ', I, ' ', O);
  Writeln('Рост самого старшего = ', Max.Rost:4:2);

  FindYoungHeight( mas, n, Min);
  With Min.Fio do WriteLN('Самого младшего зовут ', F, ' ', I, ' ', O);
  Writeln('Рост самого младшего = ', Min.Rost:4:2);

  Readln;
End.
```

Тот же Пример 1, но с процедурным типом (изменения в ту же программу):

```
Type  // процедурный тип описываем:
  TSravneniaFunc = function (chel1, chel2: TChelovek): boolean;

Procedure FindHeight(var Mas: Massiv; const n: integer; out Extrem: TChelovek; Sravn: TSravneniaFunc);
Var i: integer;
Begin
 Extrem:=Mas[1];
 For i:=2 to n do
    If Sravn (Mas[i] , Extrem)  Then
        Extrem:=Mas[i];
End;
```

```
function Younger (chel1, chel2: TChelovek): boolean;
begin
    Result:= (chel1.Data.g > chel2.Data.g) OR
      (chel1.Data.g = chel2.Data.g) AND ((Mas[i].Data.m > chel2.Data.m) OR
                                  (chel1.Data.m = chel2.Data.m) AND (chel1.Data.d > chel2.Data.d))
End;
```

```
function Older (chel1, chel2: TChelovek): boolean;
begin
    Result:= (chel1.Data.g < chel2.Data.g) OR
      (chel1.Data.g = chel2.Data.g) AND ((Mas[i].Data.m < chel2.Data.m) OR
                                  (chel1.Data.m = chel2.Data.m) AND (chel1.Data.d < chel2.Data.d))
End;
```

```
Var
 Mas: Massiv;
 Max, Min: TChelovek;
 i: integer;
```

```
Begin
  For i:=1 to N do
  Begin
    Writeln(I,'- Введите Фамилию'); readln(Mas[i].Fio.F);
    Writeln('Введите Имя'); readln(Mas[i].Fio.I);
    Writeln('Введите Отчество'); readln(Mas[i].Fio.O);

    Writeln('Введите день месяц и год через пробел');
    with Mas[i].data do readln(d, m, g);

    Writeln('Введите рост'); readln(Mas[i].Rost);
  End;

  FindHeight( mas, n, Max, Older);
  With Max.Fio do WriteLN('Самого старшего зовут ', F, ' ', I, ' ', O);
  Writeln('Рост самого старшего = ', Max.Rost:4:2);

  FindHeight( mas, n, Min, Younger);
  With Min.Fio do WriteLN('Самого младшего зовут ', F, ' ', I, ' ', O);
  Writeln('Рост самого младшего = ', Min.Rost:4:2);

  Readln;
End.
```

Выделение памяти.

При описании переменной типа запись память выделяется для совокупности всех ее полей.

Обычные записи (по умолчанию) – *оптимизация по времени* для работы (доступа) с полями размеров 4 и 8 байт. **Разумно располагая поля в запись можно сэкономить память (в полтора-два раза), не ухудшая время выполнения программы!** См пример ниже.

Упакованные записи используются для *экономии памяти* (а не времени) и описываются ключевым словом - **Packed** record

Пример 2: Программа с подсчетами размеров записей:

```
program PrRecord;
{$APPTYPE CONSOLE}

type                          // оптимизация по памяти
  t0= packed record  //6     // меньше памяти но дольше
        a: byte; //1             | a |
        b: single; //4           | b   b   b   b |
        c: char; //1             | c |
  end;

  t1= record  //12            // оптимизация по времени под тип 4 и 8 байт
        a: byte; //1             | a |   |   |   |
        b: single; //4           | b   b   b   b |
        c: char;  //1            | c |   |   |   |
  end;
                           // >>переменой мест<< - доп. оптимизация по памяти
  t2= record  //8             // при авто оптимизации по времени
        b: single; //4           | b | b | b | b |
        a: byte; //1             | a | c |   |   |
        c: char;  //1
```

```
end;

t3= record  //24
      a: byte;  //1    | a |   |   |   |   |   |   |   |
      b: double; //8   | b | b | b | b | b | b | b | b |
      c: char; //1     | c |   |   |   |   |   |   |   |
end;

t4= record  //16           // >>перемена мест<<
      b: double;  //8 | b | b | b | b | b | b | b | b |
      a: byte; //1    | a | c |   |   |   |   |   |   |
      c: char; //1
end;

t5= record   //24
      a: byte;  //1        | a |   |   |   |   |   |   |   |
      b: Extended; //10>8 | b | b | b | b | b | b | b | b |
      c: char;  //1        | b | b | c |   |   |   |   |   |
end;

t6= record  //16
      b: Extended; //10  | b | b | b | b | b | b | b | b |
      a: byte;  //1      | b | b | a | c |   |   |   |   |
      c: char;  //1
end;

t7= record  //32
      a: single; //4      | a | a | a | a |   |   |   |   |
      b: real;  //8       | b | b | b | b | b | b | b | b |
      c: char;  //1       | c | d | d | d | d | d | d | d |
      d: string[10]; //11 | d | d | d | d |   |   |   |   |
end;


t8= record //24
      b: real;  //8       | b | b | b | b | b | b | b | b |
      a: single;  //4     | a | a | a | a | c | d | d | d |
      c: char;  //1       | d | d | d | d | d | d | d | d |
      d: string[10]; //11
end;

t9= record //72
   a: byte; //1
   b: double; //8
   c: byte; //1
   d: double; //8
   e: single; //4
   f: real; //8
   g: char; //1
   h: double; //8
   i: byte; //1
end;

t10= record //40
   a: byte;
   c: byte;
```

```
      g: char;
      i: byte;
      e: single;
      b: double;
      d: double;
      f: real;
      h: double;
  end;

var t: t4;
begin
  writeln('1+4+1 = packed ', SizeOf(t0));  //6 (6)
  writeln('1+4+1 =', sizeof(t1)); //12 (6)
  writeln('4+1+1 =', sizeof(t2)); //8 (6)
  writeln('1+8+1 =', sizeof(t3)); //24 (10)
  writeln('8+1+1 =', sizeof(t4)); //16 (10)
  writeln('1+10+1 =', sizeof(t5)); //24 (12)
  writeln('10+1+1 =', sizeof(t6)); //16 (12)
  writeln('4+8+1+11 =', sizeof(t7)); // 32 (24)
  writeln('8+4+1+11 =', sizeof(t8)); // 24 (24)
  writeln('1+8+1+8+4+8+1+8+1 =', sizeof(t9)); // 72 (40)
  writeln('1+1+1+1+4+8+8+8+8 =', sizeof(t10)); // 40 (40)
  readln
end.
-----------------
```

Записи с вариантами

Одну и ту же память можно использовать для хранения нескольких наборов полей – память выделяется по размеру *максимальной* вариантной части.

Пример

```
Type
   TData= record
              d,m: byte;
              g: word;
    End;
  TChelovek = record
        Fio: record
              F,I,O: string[20];
            End;
        DataRozhdenia: TData;
        Zhiv: Boolean;  // жив? Да/Нет  True/False
        Case Boolean of  // в конце записи вариантная часть
         True: (  // для живых
              Rost, Ves: single;
              Davlenie1, Davlenie2: byte;
              );
         False: (  // для умерших
              DataSmerti: TData;
              );
   End;
```

Или

```
  TChelovek = record
        Fio: record
```

```
            F,I,O: string[20];
          End;
      DataRozhdenia: TData;
~~    // Zhiv: Boolean;~~
      Case Zhiv:Boolean of  // то же самое покороче
       True: (
            Rost, Ves: single;
            Davlenie1, Davlenie2: byte;
            );
       False: (
            DataSmerti: TData;
            );
  End;
```

//----------Пример 3. Записи с вариантами----------

Пример 3: **Средний рост живущих?**

```
Const N=10;
Type Massiv = array [1..N] of TChelovek;  // тип описан выше
Var mas: Massiv;    i, k, kol: Integer;   SredRost: real;
Begin
  For i:=1 to N do
  Begin
    Writeln(I,'- Введите Фамилию'); readln(Mas[i].Fio.F);
    Writeln('Введите Имя'); readln(Mas[i].Fio.I);
    Writeln('Введите Отчество'); readln(Mas[i].Fio.O);

    Writeln('Дата рождения?');
    Writeln('Введите день месяц и год через пробел');
    with mas[i].DataRozhdenia do readln(d, m, g);

    Writeln('Жив (0-нет, 1-да)?');  readln(k); Mas[i].Zhiv := k<>0;
    If Mas[i].Zhiv then
    begin
      Writeln('Рост?'); ReadLN(Mas[i].Rost);
      Writeln('Вес?'); ReadLN(Mas[i].Ves);
      Writeln('Верх.давление?'); ReadLN(Mas[i].Davlenie1);
      Writeln('Ниж.давление?'); ReadLN(Mas[i].Davlenie2);
    end
    else
    begin
      Writeln('Дата смерти?');
      Writeln('Введите день месяц и год через пробел');
      with mas[i].DataSmerti do readln(d, m, g);
    end;
    Writeln('Введите рост'); readln(Mas[i].Rost);
  End;

 SredRost:=0; kol:=0;
  For i:=1 to N do
    If Mas[i].Zhiv then
    Begin
      SredRost:=SredRost + Mas[i].Rost;
      Inc(kol);
    End;
```

```
   If Kol>0 then
   Begin
      SredRost:=SredRost / Kol;
      Writeln('Средний рост = ', SredRost:4:2);
   End
   Else
      Writeln('Живых нет. Средний рост невозможно вычислить ');

   Write('Нажмите ENTER…'); Readln;
End.
```

//----------Пример 4. Записи с вариантами----------
//----------Массив из разного вида фигур ----------

```
type
  TFigVid = (treug, krug, kvadr, pryam);   // значения 0..3

  TFig = record
    fig: TFigVid;
    case TFigVid of
      treug: (a,b,c: integer;); // три стороны
      krug : (r: real;);  // радиус
      kvadr: (aa: real;);  // длина стороны
      pryam: (d,s: real;);  // длины двух сторон
  end;

  Tmas = array [1..20] of TFig; // в массиве разные фигуры, но они одного типа!! – это обход
                                  // требования однотипности элементов в массиве

function sum_perimetrov(var mas: Tmas; const n: integer): real;
var i:integer; p, pp: real;
begin
  p:=0;
  for i:=1 to n do
    case mas[i].fig of
      treug:
        p:=p+ mas[i].a+mas[i].b+mas[i].c;
      krug:
        p:=p+ 2*pi*mas[i].r;
      kvadr:
        p:=p+ mas[i].aa*4;
      pryam:
        p:=p+ 2*(mas[i].d+mas[i].s);
    end;

  sum_perimetrov := p;
end;

var
  mas: Tmas;
  i,n,k: integer;
  sump: real;

begin
….// ← ввод массива добавить  или  сгенерировать (см. дальше)

  sump:=sum_perimetrov( mas, n);
```

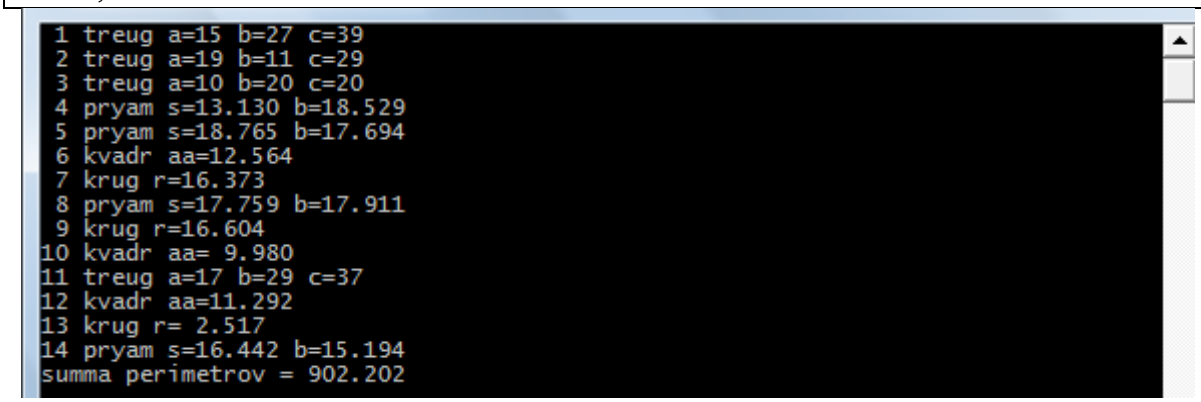
```
  writeln('summa perimetrov = ', sump:7:3);
  readln;
end.
```

Псевдослучайные числа

Вместо ввода иногда можно создать числа псевдослучайным образом. Но тестирование, а значит, и проверка правильности будет затруднена, ведь тесты это не только исходные данные, но и ожидаемые результаты. К тому же ***подходящего по смыслу теста можно не дождаться***…

Но ***для демонстрации работы*** уже проверенного алгоритма с большим объемом данных такой способ вполне подходит.

```
  randomize;  //  инициализация генератора псевдослучайных чисел системным временем
  n:=random(17)+4;  //  4+ число от 0 до 16  = длина не меньше 4, но не больше 20
  for i:=1 to n do
  begin
    k:=random(4);       // от 0 до 3       целое 0<=random(4)<4
    mas[i].fig:= TfigVid(k);  // приведение типа
    case mas[i].fig of
      treug:
        begin
          mas[i].a:=random(20)+10;  // от 10 до 39
          mas[i].b:=random(20)+10;
          mas[i].c:=mas[i].a+mas[i].b -1 - random(10);  // меньше суммы двух других сторон на 1-10
        end;
      krug:
        mas[i].r:=random*20+1;    // от 1 до 20,999999          вещ. число  0<=random<1
      kvadr:
        mas[i].aa:=random*10+5;  // от 5 до 14,999999
      pryam:
        begin
          mas[i].d:=random*15+5;  // от 5 до 19,999999
          mas[i].s:=random*10+10;  // от 10 до 19,999999
        end;
    end;
  end;
  for i:=1 to n do                              // вывод:
    case mas[i].fig of
      treug: writeln(i: 2,' treug a=', mas[i].a,' b=', mas[i].b,' c=', mas[i].c);
      krug:  writeln(i: 2,' krug r=', mas[i].r:6:3);
      kvadr: writeln(i: 2,' kvadr aa=', mas[i].aa:6:3);
      pryam: writeln(i: 2,' pryam s=', mas[i].d:6:3,' b=', mas[i].s:6:3);
    end;
```



А также с помощью генератора псевдослучайных чисел можно создать ***исходные последовательности*** для проверки алгоритмов сортировки, что особенно актуально при большом количестве элементов. Например,
Массив со значения ***по возрастанию*** (неубыванию, если равные попадутся) из диапазона от -2500 до максимум -2500+100*100=7500:

```
Var A: array[1..100] of integer;  i: integer;
Begin
  Randomize;
  A[1]:=-2500 + random(101); // первый элемент массива от -2500 до -2400
  For i:=2 to 100 do A[i]:= A[i-1] + random(101); // каждый следующий больше на 0-100
  Writeln('Array:');
  For i:=1 to 100 do Write(A[i]:8); // вывод по 10 чисел в строке (экран шириной 80)
  Readln
End.
```