



Лекция №9

# Комбинированный тип данных. Записи.

# План лекции

1. Комбинированный тип данных
2. Записи
3. Примеры записей
4. Поле
5. Обращение к полю
6. Оператор with
7. Операции над записями
8. Массивы записей
9. Записи с вариантами

# Комбинированный тип данных

Часто оправданным является представление некоторых элементов в качестве составных частей другой, более крупной логической единицы. Можно, например, сгруппировать информацию о номере дома, названии улицы и городе в единое целое и назвать адресом, а объединенную информацию о дне, месяце и годе рождения – датой.

В языке Паскаль для представления совокупности разнородных данных служит **комбинированный тип данных**.

# Пример комбинированных типов данных

Дата  
День

- Месяц
- Год
- Человек
  - Фамилия
  - Имя
  - Отчество
  - Пол
  - Дата (рождения)
- Студент
  - Человек
  - Группа
  - Курс

# Записи

**Запись** – это структура данных, состоящая из фиксированного числа компонентов, называемых *ПОЛЯМИ*.

Запись и массив схожи в том, что обе эти структуры составлены из ряда отдельных компонент.

В то же время, если компоненты массива должны быть одного типа, записи могут содержать компоненты разных типов.

# Пример записи

Приведем пример описания переменной, имеющей структуру записи:

```
Type Date = Record
```

```
    Day : 1..31;
```

```
    Month : String[20];
```

```
    Year : integer;
```

```
End;
```

```
Var
```

```
    User : Record
```

```
        FirstName : String[20];
```

```
        LastName : String[20];
```

```
        Sex : ('m','f');
```

```
        BirthDay : Date;
```

```
End;
```



# Примеры записей

Type

User = Record

    FirstName : String[20];

    LastName : String[20];

    Sex : ('m','f');

    BirthDay : Date;

End;

Отметим, что поля FirstName и LastName имеют одинаковый тип: String[20].

Поскольку в описании эти поля могут располагаться в любом порядке, то можно сократить описание записи с полями одинакового типа. Сокращенное описание записи User выглядит следующим образом:

Type

User = Record

    FirstName, LastName : String[20];

    Sex : ('m','f');

    BirthDay : Date;

End;

# Поля. Обращение к полю

Var

User : Record

    FirstName : String[20];

    LastName : String[20];

    Sex : ('m','f');

    BirthDay : Date;

End;

Каждая компонента записи называется **полем**.

Для того чтобы обратиться к полю записи, следует написать имя переменной и имя поля. Эти два идентификатора должны разделяться точкой.

Оператор, который присваивает полю LastName значение Alex, выглядит так:

```
User.LastName:= 'Alex';
```



# Поля

```
Type Date = Record
    Day : 1..31;
    Month : String[20];
    Year : integer;
End;

User = Record
    FirstName : String[20];
    LastName : String[20];
    Sex : ('m','f');
    BirthDay : Date;
End;

Var
    Student : User;
Begin
    Student.FirsName:= 'Ivanov';
    Student.LastName:= 'Ivan';
    Student.Sex:= 'm';
    Student.BirthDay.Day:=13;
    Student.BirthDay.Month:= 'January';
    Student.BirthDay.Year:=1992;
```

...

# Оператор with

...

```
Student.BirthDay.Day:=13;
```

```
Student.BirthDay.Month:='January';
```

```
Student.BirthDay.Year:=1992;
```

...

Соблюдение всех правил перечисления индексов и имен полей при составлении ссылок является довольно утомительным занятием, часто приводящим к ошибкам. В некоторых программах, содержащих большое количество обращений к одному и тому же полю, такое положение приводит к однообразному повторению. Чтобы облегчить выполнение многократных ссылок для описанных структур вводится оператор With (в переводе с английского – предлог "с").

Общая форма записи:

```
with <имя переменной> do <оператор>
```

# Оператор with

В рамках оператора, определяемого внутри оператора with, к полям определяемой переменной можно обращаться просто по имени.  
Например,

```
with Student.BirthDay do
  Day:=22;
```

```
With Student.BirthDay do
  begin
  if (Year>1989) and (Year<2000) then
    inc (k);
  if Day<16 then
    inc(x);
  end;
```

# Оператор with

Операторы with могут быть вложенными. Приведенные ниже три оператора эквивалентны друг другу:

```
Student.BirthDay.Day:=13;
```

```
with Student.BirthDay do  
  Day := 13;
```

```
with Student do  
  with BirthDay do  
    Day := 13;
```

Однако недопустимым является использование вложенных операторов With, в которых указываются поля одного типа, поскольку возникает неоднозначность конструкции.

# Операции над записями

```
Type Date = Record
    Day : 1..31;
    Month : String[20];
    Year : integer;
```

```
End;
```

```
Var
```

```
    d1, d2 : Date;
```

После приведенного описания переменные d1 и d2 имеют тип записи Date. Помимо действий над отдельными полями записей d1 и d2 можно выполнять операции над всей записью. Следующий оператор присваивания устанавливает равенство значений записей d1 и d2 :

```
    d2:=d1;
```

Это присваивание эквивалентно следующей последовательности операторов:

```
d2.Day := d1.Day;
```

```
d2.Month := d1.Month;
```

```
d2.Year := d1.Year;
```



# Операции над записями

Для переменных одного типа можно проверить выполнение отношения равенства или неравенства ("=", "<>").

```
Type Date = Record
    Day : 1..31;
    Month : String[20];
    Year : integer;
```

```
End;
```

```
Var
```

```
    d1, d2 : Date;
```

```
Begin
```

```
d1.Day := 1;
```

```
d1.Month := 'September';
```

```
d1.Year := 2009;
```

```
d2:=d1;
```

После выполнения приведенных выше присваиваний следующее булево выражение будет иметь значение True:

```
d1 = d2
```



# Массивы записей

Так как на тип компонент массива не накладывается ограничений, то можно использовать массив, компонентами которого являются записи.

Посмотрите описание такого массива:

```
Var  
    Birthdays : Array [1..100] of Date;
```

Чтобы обратиться к некоторому полю определенной записи массива, следует определить имя массива, индекс интересующей записи и имя необходимого поля.

Например, следующий оператор печатает содержимое поля Year записи Birthdays[3]:

```
Write(Birthdays[3].Year);
```

Поля записи в свою очередь тоже могут быть массивами, множествами, записями.

# Задача

В массиве хранятся данные об учениках класса: школа, фамилия, класс. Вывести список учеников, которые учатся в восьмом классе.

```
Program Z;
  Type
  Uchenik=record
    Shkola : integer;
    Fam : string[15];
    Klass : integer;
  end;
  Var
  I,n,a,j : integer;
  Begin
  writeln('Введите число учеников ');
  read(n);
  for i:=1 to n do
  begin
  writeln('Введите через пробел номер школы и фамилию ученика ');
  with massiv[i] do
  begin
  readln(Shkola,Fam);
  write('Введите класс ученика');
  read(Klass);
  end;
  end;
  end;
```

# Задача

```
writeln('Ученики 8-ых классов:');
  writeln('Школа Фамилия Класс');
  writeln('-----');
for i:=1 to n do
  if massiv[i].klass=8
  then
    with massiv[i] do
      writeln(Shkola:4,' ',Fam:15,' ',klass);
End.
```

# Записи с вариантами

Записи, рассмотренные выше – это записи с фиксированными частями. Они имеют в различных ситуациях строго определенную структуру. Соответственно записи с вариантами в различных ситуациях могут иметь различную структуру.

Предположим, что написана программа для введения списка библиографических ссылок. Если известно, что все входы в этом списке – ссылки на книги, то можно использовать следующее описание:

```
Const
    Kol = 1000;
Type
    Entry = Record
        Autor, Title, Publisher, City : String;
        Year : 1..2010;
    End;
Var
    List : Array[1..Kol] of Entry;
```

# Записи с вариантами

Использование записей с вариантами позволяет образовать структуру, каждый вход которой соответствует содержанию записи. Опишем новый тип, в котором перечислены различные входы:

```
Type  
  EntryType = (Book, Magazine);
```

Теперь можно привести скорректированное описание Entry

```
Type  
  Entry = Record  
    Autor, Title : String;  
    Year : 1..2010;  
    Case EntryType of  
      Book : (Publisher, City : String);  
      Magazine : (MagName : String,  
        Volume, Issue : Integer)  
    End;
```

Это описание делится на две части: фиксированную и вариантную. Поля Autor, Title, Year составляют фиксированную часть. Оставшаяся часть описания Entry образует вариантную часть, структура которой, может меняться в пределах двух альтернативных определений.



# Записи с вариантами

Каким образом можно узнать , что List[3] содержит ссылку на книгу, а List[4] – ссылку на журнал?

Естественное решение этой проблемы заключается в добавлении в каждой записи нового поля, называемого полем тега. Язык Паскаль позволяет за счет совмещения задать описание поля тега в сокращенной форме:

Type

```
Entry = Record
  Autor, Title : String;
  Year : 1..2000;
  Case TAG : EntryType of
    Book : (Publisher, City : String);
    Magazine : (MagName : String,
      Volume, Issue : Integer)
  End;
```

Поле, названное TAG, является переменной типа EntryType. Когда запись содержит ссылку на книгу, TAG следует присвоить значение Book. Когда запись содержит ссылку на журнал, TAG следует присвоить значение Magazine.



# Записи с вариантами

Вариантная часть может содержать произвольное число альтернатив. Хотя перечисляемые типы предпочтительнее, так как они более понятны, тем не менее для именованых альтернатив записи с вариантами могут использоваться идентификаторы произвольного порядкового типа.

Очевидно, что один и тот же идентификатор поля не может дважды использоваться при описании записи, даже если он применяется в определении различных альтернатив записи с вариантами. Если же это условие не выполняется, то обращение к такому идентификатору приведет к непредсказуемому результату.

