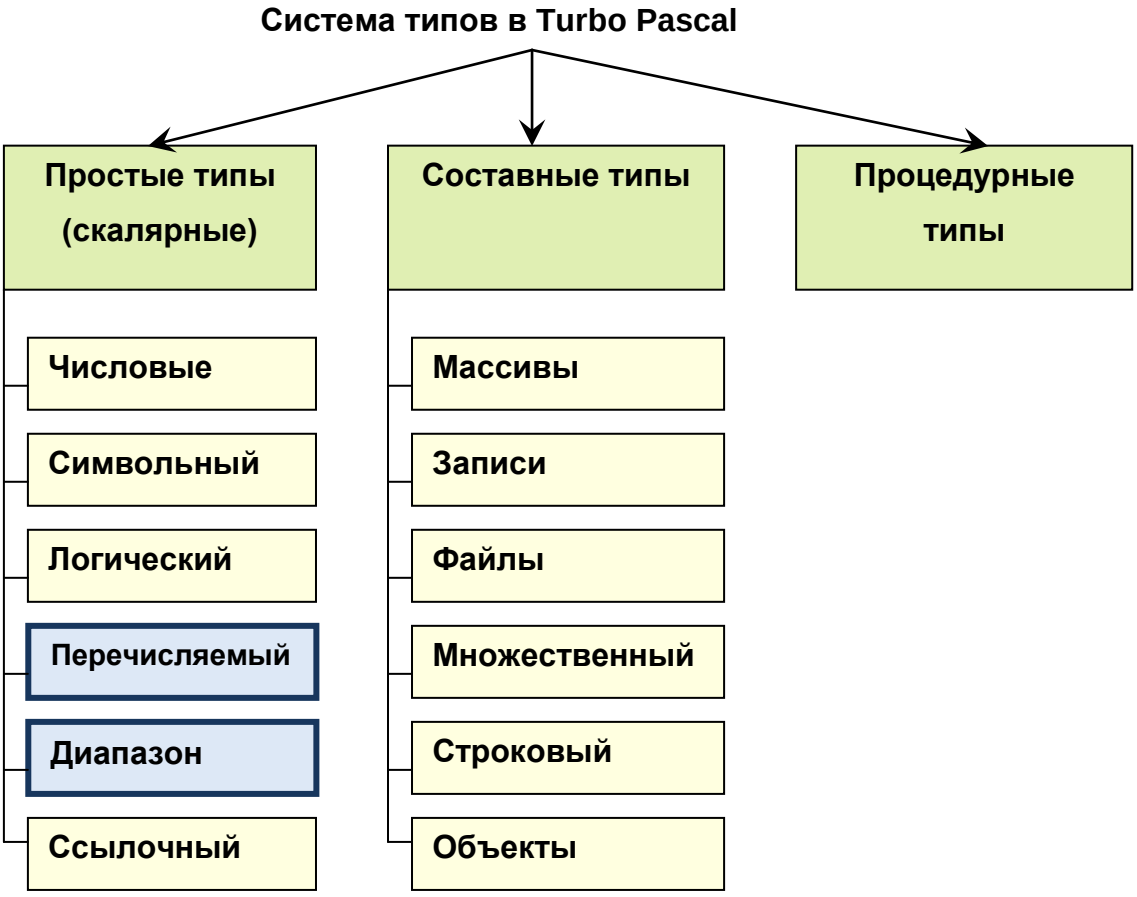
Конспект лекций №2



Пользовательские типы

Применение пользовательских типов обеспечивает семантический контроль вводимых данных.

Перечисляемый тип

Перечисляемый тип (enumerated type). Тип данных определяется набором идентификаторов, с которыми могут совпадать значения параметра.
Список идентификаторов указывается в круглых скобках (без апострофов), который разделен запятыми.

```
Type
<имя типа>=(<значение1>, <значение 2>, …);

Var
<переменная>: <имя типа>;
```

Значения перечисляемого типа **нельзя** вводить с клавиатуры и выводить на печать (экран).
При необходимости программист сам должен организовать ввод-вывод таких данных.

Пример

```
Type
     Gaz       = (Ge, C, O, N);
     Metall    = (Na, K, Li, Cu, Zn);
Var
     G1, G2, G3    : Gaz;
     Met1          : Metall;
     Palette       : (Red, Green, Blue);
```

Какие записи правильные, а какие нет?

```
Palette:= 'Red';
Palette:= Ge;
Palette:= Green;
Palette:= Black;
```

У любого перечислимого типа существует внутренняя нумерация, причем первый элемент имеет номер 0, следующий 1 и т.д. Порядок нумерации элементов соответствует порядку их перечисления.

```
Type
                    0    1    2    3    4
     Metall   = (Na, K, Li, Cu, Zn);
```

Стандартные функции при работе с перечислимыми данными

Функция	Значение функции	Пример
`Ord(x)`	Номер перечисляемого типа **x** в общем списке.	`Ord(Li)` (результат 2)
`Succ(x)`	Перечисляемый тип, следующий в списке за **x**.	`Succ(Na)` (результат K)
`Pred(x)`	Перечисляемый тип, следующий в списке перед **x**.	`Pred(Zn)` (результат Cu)

Тип Boolean фактически является стандартным предопределенным перечисляемым типом:

```
Type Boolean = (False, True);
```

Элементы перечислимых типов упорядочены, и значит, их можно сравнивать. Считается большим то значение, чей порядковый номер в типе больше.

```
Na < K
Zn > Li
```

Интервальный тип (диапазон)

Интервальный тип позволяет задавать две константы, определяющие границы интервала (диапазона) значений для данной переменной.

Обе константы должны принадлежать одному из стандартных типов (кроме Real). Значение первой константы должно быть обязательно меньше значения второй.

Компилятор при каждой операции с переменной интервального типа генерирует подпрограммы проверки, определяющие, остается ли значение переменной внутри установленного для нее диапазона.

```
Type
    <имя типа> = <константа1>..<константа2>;
Var
    <переменная> : <имя типа>;
```

Пример

```
Type
    Days = 1 .. 31;
    Year =  1920 .. 1999;
Var
    D1   : Days;
    Y1   : Year;
    NN   : 1 .. 7;
```

Оператор выбора (Case)

```
Program TRIAL_Case;
Var M: Byte;
begin
   Writeln('Введите оценку:');
   ReadLn (M);

   Case M of
      1..2: Writeln('Плохо');
      3   : Writeln('Удовлетворительно');
      4, 5: Writeln('Хорошо')
      else  Writeln('Ошибка');
   end;

end.
```

Выражение любого порядкового типа

Ошибки при использовании оператора

Ошибка	Исправлено

case a+5 of
2: a:=b;d:=0;
4: a:=c;
end;
	case a+5 of
	2: begin
	a:=b;d:=0;end;
	4: a:=c;
	end;

case a+5 of 2..5: a := b; 4: a := c; end;	case a+5 of 2..3: a := b; 4: a := c; 5: a := b; end;
case a+5 of 2: a:=b; 2: a:=c; end;	

Примеры программ

```
Program TRIAL_1;
Var
     Lastmonth  : (jan, feb, mar, apr, may, jun,
jul, aug, sep, oct, nov, dec);
Begin
  For Lastmonth  := jan to dec do
    Case Lastmonth  of
jan,feb,dec:Writeln(Ord(Lastmonth)+1,'-Зима');
mar,apr may:Writeln(Ord(Lastmonth)+1,'-Весна');
jun,jul,aug:Writeln(Ord(Lastmonth)+1,'-Лето');
sep,oct,nov:Writeln(Ord(Lastmonth)+1,'-Осень');

     End
End.
```

```
Program TRIAL_2;
{$R+}
{Ключ компиляции – режим проверки границ}
Var I,K : 1..10;
Begin
     Writeln ('Введите число');
     Readln (I);
     I := 13; {Ошибка при компиляции}
     I := 4;
     K := 10 * I; {Ошибка при выполнении}
End.
```