Информационные технологии

Лекция №5

Язык программирования Pascal

Основные сведения

- Pascal разрабатывался с 1968 по 1970 г. Николаусом Виртом.
- Цель заключалась в том, чтобы создать язык, лишенный многочисленных недостатков ALGOL.
- Pascal был назван в честь французского математика Блеза Паскаля, который еще в 1642 г. изобрел цифровой калькулятор.
- С конца 70-х до конца 80-х гг. этот язык доминировал среди языков, используемых на начальном этапе обучения программированию; позже его заменили C и C++, а затем Java
- Обучение Pascal - отличный пример изучения структурного программирования

Пример простейшей программы на Pascal

```
program Hello;
begin
  writeln('Hello, world!');
  readln;
end.
```

Структура программы

Структура программы на Pascal

Любая программа на Pascal состоит из трех блоков: блока объявлений, блока описания процедур и функций и блока операторов (основной блок программы)

Блок объявлений:

```
program ... (название программы)
const ... (подраздел описания констант)
type ... (подраздел объявления типов)
var ... (подраздел объявления переменных)
```

Блок описания процедур и функций:

```
procedure (function)
begin
...
end;
...
```

Блок основной программы:

```
begin
(операторы основной программы) ...
end.
```

Раздел program

- состоит из зарезервированного слова PROGRAM и имени программы.

- Имя, или идентификатор, строится по следующим правилам: оно может начинаться с большой или малой буквы латинского алфавита или знака "_", далее могут следовать буквы, цифры или знак "_"; внутри идентификатора не может стоять пробел.
- После имени программы следует поставить ";", этот знак служит в Паскале для разделения последовательных инструкций.

- Имя программы может не совпадать с именем соответствующего файла на диске.

```
Program baza_dannyh;
```

Раздел описания констант
Описание констант позволяет использовать имена как синонимы констант, их необходимо определить в разделе описаний констант:

```
const
  Year=2007;
  Month='november';
  Day='Saturday';
```

При присвоении значений константам вместо оператора присвоения “:=” используется просто знак равенства “=”.
Тип константы определяется автоматически по виду значения, присваемового константе и не может быть сложным.

Раздел описания типов

Раздел описания типов type позволяет определить новый тип в программе.
Определение типа записывается как:

```
<имя нового типа> = <описание нового типа>;
```

```
type
  T1 = Integer;
  T2 = T1;
  T3 = T2;
```

Раздел описания переменных
Здесь содержится список используемых в программе переменных и определяется их тип.

Объявления переменных записываются в следующей форме:
```
<переменная> : <тип>;
```

Если описываются несколько переменных одного типа, то достаточно записать их имена через запятую, а после двоеточия поставить общий тип.
```
var
  a,b,c: integer;
  x,y: real;
```

Переменные могут хранить данные различной природы: числа, строки текста, отдельные символы и т. п.

Блок операторов
Основной блок программы, ограниченный операторами
`begin`
и
`end.`
Как уже говорилось, оператор `end.` указывает компилятору, что программа закончена, в отличие от операторов end;, которые завершают блоки, процедуры, модули и т.п.

Текст, следующий за оператором `end.`, игнорируется транслятором.

Структура программы

Символ “;”
Этот символ завершает каждый оператор
Наличие точки с запятой обязательно, т.к. этот символ показывает компилятору, где заканчивается один оператор и начинается следующий.
Благодаря тому, что Паскаль - язык со "свободной формой записи", можно без опасений "растянуть" оператор на несколько строк.
С помощью символа точка с запятой Вы сообщаете компилятору, какую часть текста программы следует рассматривать как цельный, неделимый фрагмент.

Зарезервированное слово begin, с которого начинаются блоки программы, не требует после себя символа точка с запятой

```
;
```

Типы данных

Тип переменной задает вид того значения, которое ей присваивается и правила, по которым операторы языка действуют с переменной
Если переменные A и B целочисленного типа, то программа:

```
x:=3;
y:=2;
writeln(x,' ',y,' ',x+y);
```

Выведет на экран строку:

```
3 2 5
```

Если же они строкового типа, то программа:

```
x:='3';
y:='2';
writeln(x,' ',y,' ',x+y);
```

выведет:

```
3 2 32
```

так как оператор сложения просто добавит строку y в конец строки x.

Тип константы определяется способом записи ее значения:

```
const
  c1=17;
  c2=3.14;
  c3='a';
  c4=false;
  c5=c2+c1;
```

При определении констант можно использовать выражения.
Выражения должны в качестве операторов содержать только константы, в том числе ранее объявленные, а так же знаки математических операций, скобки и стандартные функции.

Целочисленные типы данных

`byte`	целое число от 0 до 255, занимает одну ячейку памяти (байт).
`word`	целое число от 0 до 65535, занимает два байта
`integer`	целое число от –32768 до 32767, занимает два байта.
`shortint`	целое число от -128 до 127, занимает 1 байт
`longint`	целое число от –2147483648 до 2147483647, занимает четыре байта.

Вещественные типы данных

`real`	число с дробной частью от 2.9*10-39.до 1.7*1038, может принимать и отрицательные значения, на экран выводится с точностью до 12-го знака после запятой, если результат какой либо операции с real меньше, чем 2.9*10-39, он трактуется как ноль. Переменная типа real занимает шесть байт.
`double`	число с дробной частью от 5.0*10-324.до.1.7*10308, может принимать и отрицательные значения, на экран выводится с точностью до 16-го знака после запятой ,если результат какой либо операции с double меньше, чем 5.0*10-324, он трактуется как ноль. Переменная типа double занимает восемь байт

Типы данных

`char`	символ, буква, при отображении на экран выводится тот символ, код которого хранится в выводимой переменной типа char, переменная занимает один байт. Каждому символу приписывается целое число в диапазоне от 0 до 255. Для кодировки используется код ASCII.
`string`	строка символов, на экран выводится как строка символов, коды которых хранятся в последовательности байт, занимаемой выводимой переменной типа STRING; в памяти занимает от 1 до 256 байт – по количеству символов в строке, плюс один байт, в котором хранится длина самой строки.
`boolean`	логическое значение (байт, заполненный единицами, или нулями), true, или false

При обьявлении переменной строкового типа можно заранее указать ее длину в байтах – X:

```
MyString: string[100];
```

При присвоении этой переменной строки длиннее X, присваиваемая строка будет обрезана с конца после X-того символа.
Размер переменной типа STRING в памяти можно узнать следующим способом:

```
Size := SizeOf(MyString);
```

Функция `SizeOf()` возвращает размер, занимаемый переменной, служащей параметром.
Кроме того, можно узнать, сколько символов в строке (индекс последнего непустого символа в строке):

```
Size:=Ord(MyString[0]);
```

Используется ибращение к нулевому элементу (символу) строки, в котором хранится ее длина, но MyString[0] – значение типа char, то есть символ, код которого равен длине строки, нужный нам код – число возвращает функция Ord()Таким же образом можно обратиться к любому N – тому элементу строки:

```
MyChar:=MyString[N];
```

Приводимость типов

В Pascal существуют ограничения на присвоение значений одних переменных другим. Если переменные которую и которой присваивают одного типа, то никаких проблем не возникнет.
Но если они разных типов, присвоение не всегда может быть произведено.

```
x:=y; {x:integer; y:real} Ошибка!
a:=b; {a:char; b:string} Ошибка!
```

В то же время, такие присвоения будут выполнены вполне корректно:

```
y:=x;
b:=a;
```

При этом переменная y примет значение с нулевой дробной частью, а b – станет строкой, содержащей один символ – из a.
В первом же случае, можно поизвести следующие операции:

```
x:=trunc(y); {функция trunc() возвращает целую часть аргумента}
x:=round(y); {round() – округляет аргумент стандартным способом}
```

Кроме рассмотренного случая может существовать множество других, но наиболее общее правило таково: следить за однозначностью присвоения с потерями информации и не удивляться, а экспериментировать переделывать программу, если компилятор выдает сообщение о невозможности присвоения.

Арифметические операции

`+`	сложение
`-`	вычитание
`*`	умножение
`/`	деление
`div`	Целочисленное деление `{ 5 div 2 = 2 }`
`mod`	Остаток от деления `{ 5 mod 2 = 1 }`

Операции отношения

=	равно
<>	Не равно
>	больше
<	меньше
>=	Больше или равно
<=	Меньше или равно

Стандартные математические функции

`abs(x)`	`random(x)`
`arctan(x)`	`round(x)`
`cos(x)`	`sin(x)`
`exp(x)`	`sqr(x)`
`int(x)`	`sqrt(x)`
`ln(x)`	`trunc(x)`

Процедуры ввода/вывода

```
write(p1,p2,... pn);
```
выводит на экран значения выражений p1,p2,... pn.
Выражения могут быть числовые, строковые, символьные и логические.
Возможен форматный вывод, т. е. явное указание того, сколько выделять позиций на экране для вывода значения.
Например, для того, чтобы вывести значение выражения a+b с выделением для этого 10 позиций, из них 5 - после запятой
```
write(a+b:10:5);
```
Или например, вывести значение выражения p любого другого типа , выделив под него 10 позиций
```
write(p:10);
```
```
writeln(p1,p2,... pn);
```
аналогично write, выводит значения p1,p2,... pn, после чего переводит курсор на новую строку.
```
writeln;
```
означает лишь перевод курсора на начало новой строки.
```
readln(v1,v2,...vn);
```
ввод с клавиатуры значений переменных v1,...vn.
Переменные могут иметь строковый, символьный или числовой тип.
```
read(v1,v2,...vn);
```
аналогочно readln;