# Программирование на языке Паскаль лекция 1

# Структура простейшей программы:

```
program имя_программы;
begin
   {основная программа}
end.
```

## Оператор вывода текста на экран:

```
write ('Hello!');
writeln ('Hello, world!');
```

Оператор `writeln` после вывода текста осуществляет переход на следующую строку.

Пустой оператор `writeln`; делает только переход на новую строку.

# Алгоритм работы в среде Паскаль.

1. Вызвать «К → Служебные → Terminal»;
2. В терминале набрать «fp» и нажать «Enter»;
3. В интерфейсе FreePascal выбрать пункт меню File → New
4. Набрать программу.

   Например,

```
program first;
begin
    write ('Hello, ');
    write ('friends!');
    writeln ('This is my first program!');
end.
```

5. Сохранить программу в своей личной папке, например, под именем «first.pas».
6. Выбрать меню Compile, команду Compile (или сочетание клавиш ALT+F9).

   Если компилятор не обнаружил ошибок, то на экране появляется окно с сообщением: «Compilation successful: press any key» (компиляция прошла успешно: нажмите любую клавишу).

   Если обнаружена ошибка, появляется окно с сообщением: «Compile failed». А ниже окно сообщений компилярора «Compiler Messages». В нём содержится описание ошибок в следующем виде: «<имя_программы> (X,Y) Error: ...», где X — порядковый номер строки с ошибкой, а Y — порядковый символ в этой строке, с которого начинается неправильная запись.

   ESC — возврат в окно редактирования.
7. После успешной компиляции запустить программу на исполнение: выбрать меню Run, команду Run (или сочетание клавиш CTRL+F9).
8. Посмотреть на результат выполнения (в конце будет написано «Press any key to return to IDE») и нажать любую клавишу для возврата в окно редактирования.

# Оформление текста на экране

Для реализации дополнительных возможностей, не входящих в стандарт языка, создаются специальные модули или библиотеки. Эти модули содержат набор программ (процедур), которые при подключении модуля можно использовать.

Рассмотрим подключение модуля **CRT**, в котором содержаться процедуры, позволяющие задавать цвет символам, очищать экран, устанавливать курсов в любую позицию экрана и выполнять множество других полезных действий.

Паскаль работает в текстовом режиме. Это означает, что информация на экран выводится в виде символов, каждый из которых отображается на экране в определённой позиции. Экран при этом можно себе представить как таблицу из 25 строк и 80 столбцов. Каждая ячейка этой таблицы имеет 2 координаты — `x` и `y`, где `x` — номер столбца `y` — номер строки. Строки нумеруются сверху вниз от 1 до 25, столбцы — слева направо от 1 до 80. Т.е. левый верхний угол экрана имеет координаты (1, 1), правый верхний — (80, 1), а левый нижний — (1, 25). Символы можно выводить на экран 16 различными цветами, которые кодируются числами от 0 до 15.

Рассмотрим некоторые процедуры модуля **CRT**.

- `TextBackGround (N);` - процедура для выбора фонового цвета, в скобках указывается номер выбранного цвета. Вместо номера можно писать название, например, black, red, green.
- `ClrScr;` - процедура очистки экрана.
- `TextColor (N);` - процедура выбора цвета символов, в скобках указывается номер выбранного цвета. Эта команда не меняет цвет символов, уже имеющихся на экране, она лишь устанавливает цвет, которым будут выведены следующие символы.
- `GoToXY (X,Y);` — процедура установки курсора в точку экрана с координатами (X,Y).
- `Delay (N);` - процедура временной задержки на N мкс.

## Пример программы:

```
program second_v1;
uses Crt;
begin
   TextBackGround (3);
   ClrScr;
   TextColor (14);
   GoToXY (40,10);
   Writeln ('Hello, world!');
   Delay (2000);
end.
```

## Задание №2.

1. Набрать пример программы, сохранить в личной папке под именем `second_v1.pas`.
2. Изменить цвета фона и текста.
3. Поменять местами процедуры вызова цвета фона и очистки экрана. Оценить результат.
4. Написать программу, выводящую два любых сообщения в левом верхнем и правом нижнем углах экрана. Каждое сообщение выводить своим цветом. Сохранить в личной папке под именем `second_v2.pas`.
5. Написать программу, которая очищает экран и выводит слова red, green, blue, yellow каждое своим цветом в центр четвертей экрана (зелёный — левая верхняя четверть, жёлтый — левая нижняя, синий — правая верхняя, красный — правая нижняя). Сохранить в личной папке под именем `second_v3.pas`.

# Работа с числовыми данными

Данные программы принято называть ***величинами***. Величины, которые меняются, называются ***переменными***, а те, которые не меняются — ***постоянными*** (***константами***).

Величину (число), хранящуюся в ячейке, называют ***значением ячейки***. Программа работает с адресами и значениями ячеек памяти. Но для удобства работы ячейкам принято давать ***имя*** (***идентификатор***). В специальной таблице программа-компилятор будет запоминать, какому имени какой адрес ячейки памяти соответствует.

## Имена могут включать:

- латинские буквы (A-Z), причём заглавные и строчные буквы не различаются;
- цифры, причём имя не может начинаться с цифры;
- знак подчеркивания «_»

## Имена НЕ могут включать:

- русские буквы;
- пробелы;
- скобки, знаки +, =, !, ? и др.

Переменные объявляются в разделе описания переменных **VAR**.

```
var имя_переменной: тип_переменной;
```

Константы объявляются в разделе описания констант **CONST**. Он ставится перед разделом переменных.

```
const имя_константы = значение_константы;
```

# Типы переменных

- `byte` { целое число от 0 до 255 }
- `integer` { целое число от $-2^{15}$ до $2^{15}-1$}
- `real` { вещественное число }
- `char` { символ }
- `string` { символьная строка }
- `boolean` { логическое значение }

# Оператор присваивания

***Оператор*** – это команда языка программирования высокого уровня.

***Оператор присваивания*** служит для изменения значения переменной.

В результате выполнения оператора присваивания в ячейку помещается новое число. Старое содержимое ячейки при этом пропадает.

```
<имя переменной> := <выражение>;
```

Справа от оператора присваивания может стоять число или любое выражение. Слева может стоять только имя переменной. Выражения слева быть не может — иначе Паскаль не будет знать, в какую ячейку памяти поместить результат. Тип результата выражения справа от оператора присваивания должен быть таким, чтобы помещаться в переменную слева от «:=»

## Арифметическое выражение может включать:

- константы
- имена переменных
- знаки арифметических операций:

| | |
|---|---|
| + | (сложения) |
| - | (вычитание) |
| * | (умножение) |
| / | (деление) |
| **div** | (деление нацело) |
| **mod** | (остаток от деления) |

- вызовы функций
- круглые скобки ( )

## Пример записи арифметического выражения:

$$z = \frac{5ac + 3(c - d)}{ab}(b - c)$$

```
z := (5*a*c+3*(c-d))/a*(b-c)/ b;
```

## Пример программы:

```
program third_v1;
const P=3.14;
var A, B: Integer;
    C: Real;
begin
   A:=17;
   B:=3;
// Операция сложения
   C:=A+B;
   writeln('17+3=',C);
// Операция вычитания
   C:=A-B;
   writeln('17-3=',C);
// Операция умножения
   C:=A*B;
   writeln('17*3=',C);
// Операция деления нацело
   C:=A div B;
   writeln('17 div 3=',C);
// Вычисление остатка от деления
   C:=A mod B;
   writeln('17 mod 3=',C);
// Действие с константой
   C:=(A+B)/P;
   writeln('(17+3)/3.14=',C);
end.
```

## Задание №3.

1. Набрать пример программы без комментариев, сохранить в личной папке под именем `third_v1.pas`.
2. Изменить значения переменных, оценить результат.
3. Написать программу, которая вычисляет следующее выражение при исходных данных a=2, b=7, c=6, d=8 и сохранить в личной папке под именем `third_v2.pas`.

$$x = \frac{a^2 + 5c^2 - d(a + b)}{(c + d)(d - 2a)}$$

# Оператор ввода

Заносить данные в ячейки памяти можно не только оператором ввода, но и путём непосредственного ввода с клавиатуры. Это удобно тем, что в программу при каждом запуске можно вводить разные начальные значения, что добавляет ей универсальности.

`read (a);` — ввод значения переменной a
`read (a, b)` — ввод значений переменных a и b

Вводить несколько переменных можно через пробел или <Enter>. Так же можно использовать оператор `readln`.

Рассмотрим в качестве примера программу красивого вывода на экран текста.
Для этого подключим опять модуль Crt. И будем использовать ещё один, входящий в него оператор определения на экране текстового окна — `Window(X1,Y1,X2,Y2)`, где `(X1,Y1)` — координаты верхнего левого угла, а `(X2,Y2)` — координаты нижнего правого угла.

Задача: задать с клавиатуры цвет фона (экрана), символов и координат для вывода текста, а затем вывести текст в окно с заданными координатами.

Назовём переменную для хранения цвета экрана C11, а для цвета символов — C12. Переменные X и Y будут использоваться для хранения координат.

В конце программы будем использовать оператор `readln;` без параметров. Этот «пустой» оператор задержит на экране изображение до нажатия <Enter>.

```
program Inp_color;
uses Crt;
var Cl1, Cl2: 0..15;   // 0..15 — этот тип для переменных называется
                       // интервальным. В данном случае переменные
                       // могут принимать значения от 0 до 15. Здесь
                       // мы имеем отрезок (интервал) базового типа
                       // integer. Палитра цветов лежит именно в
                       // этом интервале.
    X, Y:integer;
begin
 ClrScr;
 Window(20,5,60,20);
 writeln('Input color of the screen:');
 readln(Cl1);
 writeln('Input color of symbols');
 readln(Cl2);
 TextBackGround(Cl1);
 TextColor(Cl2);
 writeln('Input coordination X and Y:');
 readln(X,Y);
 ClrScr;
 GoToXY(X,Y);
 writeln('Very well!');
 readln;
end.
```