

# Язык программирования Паскаль

```
jsr register
lda #(<normstart
ldy #>normstart
sta p2
sty p2+1
jmp input

pla ;dem aufruf
sta p1 ;den text
pla ;brk=ende
sta p1+1
```

# Паскаль



- **Язык назван в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля, который создал первую в мире механическую машину, складывающую два числа**

# Разработчик



- **Язык Паскаль  
был создан  
Никлаусом Виртом  
в 1968—1969 годах**

**Он был опубликован в 1970 году Виртом как  
небольшой и эффективный язык, чтобы  
способствовать хорошему стилю  
программирования, использовать структурное  
программирование и структурированные данные**

# Программирование на Паскале:

- Заголовок программы      **Program** <имя программы>;

---
- Раздел описания  
**Const**<раздел констант>;  
**Type**<раздел типов>;  
**Var**<раздел переменных>;

---
- Основной блок программы  
**Begin**  
    <раздел операторов>;  
**End.**

# Алфавит языка

- Латинские буквы (A-Z, a-z)
- Цифры 0-9
- Специальные символы:  
+ - \* / = < > { } [ ] . , ( ) : ; ^ @ \$ #
- Служебные слова
- Идентификаторы- символическое имя определенного программного объекта

# Служебные слова

**Служебные слова** – предназначены для написания команд. В Паскале есть несколько десятков служебных слов, которые программисту нельзя использовать в качестве имен переменных.

Таковыми словами являются:

**and**

**file**

**not**

**string**

**else**

**begin**

**for**

**of**

**then**

**case**

**function**

**or**

**type**

**const**

**goto**

**to**

**mod**

**div**

**procedure**

**until**

**do**

**if**

**program**

**var**

**downto**

**in**

**while**

# Идентификаторы

## Правила создания идентификаторов

1. Состоит из строчных или прописных латинских букв, цифр и знака подчеркивания «\_».
2. Начинается с буквы или знака подчеркивания «\_».
3. Не может быть служебным словом.
4. Длина не должна превышать 127 символов
5. Желательно, чтобы идентификатор отображал смысл переменной.

Правильные идентификаторы:

Temp\_ x1 \_33name  
\_1\_2\_3 My\_Variable

Неправильные идентификаторы:

Temp- 1x 33name1\_2\_3  
My Variable

# Команда присваивания

***Переменная := Выражение;***

**Примеры:**

**A:=3\*4.8;**

**Su:=X+X\*4.78;**

**C:=C+1;**

**Выражение должно быть записано в виде линейной цепочки символов!**

**Между всеми элементами выражения должны быть знаки операций**

**3x ⇨ 3\*x**

**Аргументы функций должны быть заключены в ():**

**sinx ⇨ sin(x)**

# Задачи:

- Определите значение переменной после выполнения следующего фрагмента программы:

1)  $a := 7$  ;  
 $a := a - 4$  ;  
 $b := -a$  ;  
 $c := -a + 2 * b$  ;

2)  $a := 2$  ;  
 $b := 2 + 4$  ;  
 $b := 1 - b$  ;  
 $c := -b + 3 * b$  ;

3)  $a := -3$  ;  
 $b := a + 3$  ;  
 $b := 1 - b$  ;  
 $c := -b + 3 * a$  ;

# Стандартные процедуры и функции:

- **Abs (x)** – абсолютное значение аргумента  $x$ ;
- **ArcTan (x)** – арктангенс  $x$ , выраженный в радианах;
- **Cos (x)** – косинус  $x$ ,  $x$  задается в радианах;
- **Sin (x)** – синус  $x$ ,  $x$  задается в радианах;
- **Sqr (x)** – квадрат  $x$ ;
- **Sqrt (x)** – квадратный корень из  $x$ ;
- **Exp (x)** –  $e^x$  (экспонента);
- **Ln (x)** – натуральный логарифм  $x$ ;
- **Pi** – число  $\pi$  (3,141592653589793285...);
- **Frac (x)** – дробная часть  $x$ ;
- **Int (x)** – выделяет целую часть  $x$ ;
- **Random (x)** – генерирует случайное число в пределах  $[0; x)$ ;

# Правила записи арифметических выражений:

- Не допускаются два следующих подряд знака операций;
- Приоритет операций:  $*$ ,  $/$ ,  $\text{div}$ ,  $\text{mod}$ ,  $+$ ,  $-$ .
- Несколько записанных подряд операций одинакового приоритета выполняются последовательно слева направо.
- Часть выражения заключенного в скобки выполняется в первую очередь.

# Арифметические операции:

**+     A+B**

**-     A-B**

**/     A/B**

**\*     A\*B**

**Div   A div B**

**Mod   A mod B**





# Типы данных:

Типы данных определяются по свойствам величин:

- 1) Форма внутреннего представления.
- 2) Множество принимаемых значений.
- 3) Множество допустимых операций.

# Целые типы:

Идентификатор	Длина(байт)	Множество значений
<b>Integer</b>	<b>2</b>	<b>-32768-32767</b>
<b>byte</b>	<b>1</b>	<b>0-255</b>
<b>word</b>	<b>2</b>	<b>0...65535</b>
<b>shortint</b>	<b>1</b>	<b>-128...127</b>
<b>longint</b>	<b>4</b>	<b>-2147483648...</b>

# Вещественный тип

Идентификатор	Длина(байт)	Диапазон
<b>real</b>	<b>6</b>	<b><math>2.9 \cdot 10^{-39} \dots 1.7 \cdot 10^{38}</math></b>
<b>single</b>	<b>4</b>	<b><math>1.5 \cdot 10^{-45} \dots 3.4 \cdot 10^{38}</math></b>
<b>double</b>	<b>8</b>	<b><math>5 \cdot 10^{324} \dots 1.1 \cdot 10^{4932}</math></b>

# Описание переменных

**Var** m,n,k: **Integer**;

x,y,z: **Real**;

**Symbol**: **Char**;

# Описание констант

## Const

**Max=1000;**

**G=9.8;**

**Cod='ОШИБКА';**

# Ввод данных.

- Это передача информации от внешних устройств в оперативную память. Вводятся исходные данные решаемой задачи.

Read (<список ввода>)

Пример: Read(a,b,c,d)

**Var T: Real;**

**J: Integer;**

**K: Char;**

**Begin**

**Read (T,J,K);**

Набираем на клавиатуре: 253.98 100 G (Enter)

# Ввод потоком.

```
Var A,B: Integer;  
    C,D: Real;
```

```
Begin
```

```
    Read(A,B);
```

```
    Read(C,D);
```

Набираем на клавиатуре:

18758 34 (Enter) 2.62E-02 1.54E+01 (Enter)

# Ввод строками.

**ReadLn(<список ввода>)**

**ReadLn(A,B);**

**ReadLn(C,D);**

3758 34 (Enter)

2.62E-02 1.54E+01 (Enter)

# Вывод на экран.

**Write (<список вывода>)**

**Пример:**

**Write (234); {выводится целая константа}**

**Write(A+B-2);{результат вычисления}**

**Write(X, Summa, Arg1,Arg2);{выводятся значения переменных}**

# Вывод строкой.

**WriteLn(<список вывода>)**

**WriteLn (I,' ',J,' ',K);**

# Форматы вывода.

**Формат определяет представление выводимого значения на экране. Он отделяется от соответствующего ему значения двоеточием.**

# Примеры:

**I=134**

**Write (I)**

134

**R=715.432**

**Write(R)**

7.1543200000E+02

**R=511.04**

**Write(R:8:4)**

511.0400

**I=287**

**Write(I,I,I)**

287287287

**R=46.78**

**Write(-R:12)**

-4.67800E+01

**R=-46.78**

**Write(R:7:2)**

-\_ 46.78

**I=134**

**Write(I:6)**

\_\_ \_ 134

# Управление символьным выводом на экран.

Дополнительные возможности управления выводом на экран дают процедуры и функции модуля CRT.

Формат команды:

Uses CRT

Для работы с модулем необходимо знать:

- Режимы экрана;
- Координаты на экране;
- Текстовое окно;
- Цвет фона и цвет символа.

# Режимы экрана.

В модуле CRT каждый режим имеет определенный номер, за которым закреплено символическое имя (описанная константа).

Для установки режима экрана используется процедура:

**TextMode (<номер режима>)**

Пример:

**TextMode(1);**

**TextMode(CO40);**

По умолчанию устанавливается режим CO80.

# Координаты позиции.

Каждая символьная позиция на текстовом экране определена двумя координатами (X, Y).

X – позиция в строке;

Y – номер строки, в которой находится символ.

Строки нумеруются сверху вниз.

Для установления курсора на экране в позицию с координатами (X, Y) в модуле CRT существует процедура: GoToXY(X, Y)

Координаты курсора задаются переменными типа Byte.

# Пример программы.

Программа очищает экран и  
выставляет в центре символ «\*»:

**Uses CRT;**

**Begin**

**ClrScr;**

**GoToXY(40,13);**

**Write('\*')**

**End.**

# Текстовое окно.

Прямоугольное пространство на экране, в котором производится вывод символов, называется текстовым окном.

## Процедура:

`Window(X1, Y1, X2, Y2)` – определяет положение и размер окна.

`X1, Y1, X2, Y2` – переменные типа `Byte`, координаты верхнего левого и правого нижнего угла окна.

# Управление цветом.

В модуле CRT объявлены константы, имена которых представляют собой английские названия цветов, а соответствующие им значения – порядковые номера этих цветов.

Процедура назначения цвета фона:

**TextBackGround(Color)**

Аргумент величина типа Byte, задающая номер цвета.

Процедура назначения цвета символа:

**TextColor(Color).**

# Пример программы:

По очереди откроются четыре окна, и каждое из них будет залито своим фоновым цветом:

**Uses CRT;**

**Begin**

**Window(1,1,40,12);**

**TextBackGround(White); ClrScr;**

**Window(41,1,80,12);**

**TextBackGround(Red); ClrScr;**

**Window(1,13,40,25);**

**TextBackGround(LightRed); ClrScr;**

**Window(41,13,80,25);**

**TextBackGround(Green); ClrScr;**

**End.**

# Пример программы:

На белом фоне в середине экрана будут выделены номера первых 15 цветов. Каждый номер будет того цвета, который он обозначает:

```
Uses CRT;  
Var I: Byte;  
Begin  
  TextBackGround(White); ClrScr;  
  GoToXY(1,12);  
  For I=0 To 14 Do  
    Begin  
      TextColor(I);  
      Write(I:5);  
    End;  
End.
```

# Процедуры управления текстовым экраном из модуля CRT.

ClrEOL –стирает часть строки от текущей позиции курсора до конца этой строки в окне. Положение курсора не меняется.

DelLine – уничтожает всю строку с курсором. Нижние строки сдвигаются на одну вверх.

InsLine – вставляет пустую строку перед строкой, в которой стоит курсор.

LowVideo, NormVideo, HighVideo – устанавливает режимы пониженной, нормальной и повышенной яркости символов соответственно.

KeyPressed –часто используют для организации задержки окна результатов на экране.

Repeat Until KeyPressed; - пустой цикл, который крутится на месте до нажатия какой-либо клавиши. Ставится в конце программы.

# Пример:

**В приведенной выше программе перед  
концом добавим:**

```
Repeat Until KeyPressed;  
Window(1,1,80,25);  
TextBackGround(Black);  
ClrScr;
```

**Логические величины, операции, выражения. Логический оператор присваивания.**

**Логические значения обозначаются служебными словами `false` (ложь) и `true` (истина), а идентификатор логического типа – `boolean`.**

**Кроме величин (констант и переменных) типа `boolean` логическое значение `false` и `true` принимают результаты операций отношения.**

# Операции отношения.

Операции отношения осуществляют сравнение двух операндов и определяют, истинно или ложно соответствующее отношение между ними.

**<знак отношения>:=**

**=(равно)**

**<>(не равно)**

**<(меньше)**

**>(больше)**

**<=, >= (меньше(больше) или равно)**

# Логические операции.

- **Not** – отрицание
- **And** – логическое умножение.
- **Or** – логическое сложение.
- **Xor** – исключаящая или.

**Операции отношения имеют самый низкий приоритет, поэтому заключаются в скобки.**

# Логическое выражение.

- Логическая формула, записанная на языке программирования.
- Состоит из логических операндов, связанных логическими операциями и круглыми скобками.
- Результатом вычисления логического выражения является булева величина (false или true).
- Логическими операндами могут быть логические константы, переменные, функции, операции отношения.

# Примеры логических выражений.

**A, b, c – логические переменные.**

**X, y – вещественные переменные,**

**K – целая переменная.**

**1)  $x < 2 * y$ ; 2) true; 3) d;**

**4) odd(k); 5) not not d; 6) not ( $x > y / 2$ );**

**7) D and ( $x < > y$ ) and b; 8) (c or d) and ( $x = y$ ) or not b.**

**Если d=true; b=false; c=true; x=3; y=0.5; k=5, то  
результаты вычислений:**

**1) false; 2) true; 3) true; 4) true; 5) true; 6) false;**

**7) false; 8) true.**

# Логический оператор присваивания.

- **<Логическая переменная> :=  
<логическое выражение>**

**Примеры:**

- 1) **d:=true;**
- 2) **B:=(x>y) and (k<>0);**
- 3) **C:=d or b and not(odd(k) and d)/**

**Odd(k) – функция целого аргумента k, принимает значение true, если k нечетное и false, если k – четное.**

# Функции связывающие различные ТИПЫ ДАННЫХ.

Обращение	Тип аргумента	Тип Результата	Действие
Ord(x)	Любой Порядковый	Integer	Порядковый номер значения x в его типе
Pred(x)	Любой Порядковый	Тот же, что для x	Предыдущее относительно x значение в его типе
Succ(x)	Любой Порядковый	Тот же, что для x	Следующее относительно x значение в его типе
Chr(x)	Byte	Char	Символ с порядковым номером x
Odd(x)	Integer	Boolean	true, если x неч. False, если x четн.