ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ.

Язык программирования *Pascal*

Язык программирования Паскаль был разработан профессором, директором Института информатики Швейцарской высшей политехнической школы Николаусом Виртом в *1968-1970* гг. как язык обучения студентов программированию.

Николаус Вирт

Но думать, что Паскаль- язык исключительно для обучения, было бы неверно. Вот что говорил об этом Н. Вирт *(1984* г.*)*:

«Утверждалось, что Паскаль был разработан в качестве языка для обучения. Хотя это утверждение справедливо, но его использование при обучении не являлось единственной целью. На самом деле я не верю в успешность применения во время обучения таких инструментов и методик, которые нельзя использовать при решении каких-то практических задач».*

* http://pascal.sources.ru

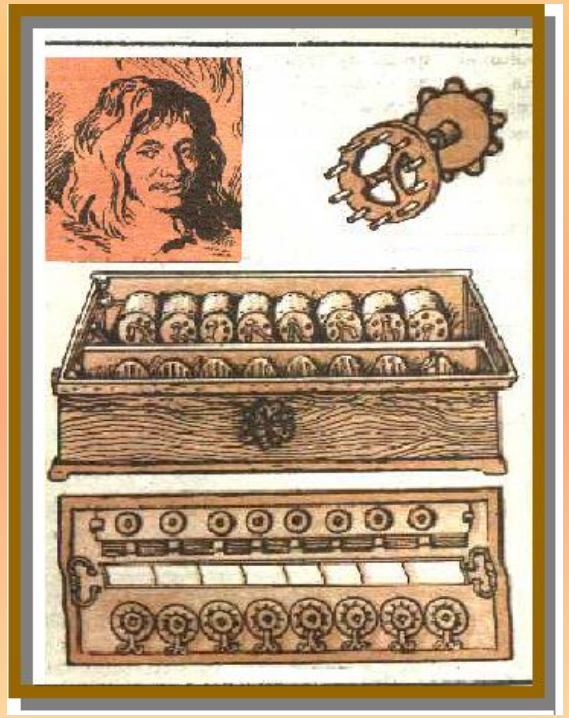
Язык программирования *Pascal*

С тех пор Паскаль становился всё более и более популярным, причем не только, как язык для обучения принципам программирования, но и как средство создания достаточно сложного программного обеспечения.

В своем первоначальном виде Паскаль имел довольно ограниченные возможности, но расширенный вариант этого языка – *Turbo Pascal* является мощным языком программирования.

Почему *PASCAL?*

Язык программирования *Pascal* был назван в честь французского учёного Блеза Паскаля, который еще в *1642* г. изобрел первую механическую счётную машину. Она представляла собой систему взаимодействующих зубчатых колёсиков, каждое из которых соответствовало одному разряду десятичного числа и содержало цифры от *0* до *9*. Когда колёсико совершало полный оборот, следующее сдвигалось на одну цифру. Машина Паскаля была суммирующей машиной.



Основные сведения о языках программирования

Язык – система знаков.

Язык ЭВМ (машинный язык) – двоичная знаковая система.

Поэтому, чтобы компьютер мог понять написанную программу, она должна быть переведена на язык, понятный компьютеру. Этот процесс перевода называется трансляцией.

Интерпретаторы и компиляторы

Существует два различных подхода к трансляции – **интерпретация и компиляция**:

- Интерпретатор переводит и выполняет программу строка за строкой.
- Компилятор переводит программу целиком, а затем выполняет её.

Интегрированная среда *Turbo Pascal-7.0*

Огромную роль в массовом распространении Паскаля сыграла компания *Borland International*. Она сумела создать знаменитую *Turbo*-среду разработки. Это был огромный шаг вперед в облегчении процесса программирования.

Почему *Turbo*? *Turbo* в переводе с английского сленга означает ускорение. Компилятор, входящий в состав *Turbo Pascal* очень быстро переводит программу с языка программирования в машинные коды.

Интегрированная среда *Turbo Pascal-7.0*

В состав интегрированной среды входят:

- Текстовый редактор
- Компилятор
- Отладчик
- Справочная система
- Среда выполнения программы

Основные средства языка

Символы языка – это элементарные знаки, используемые при составлении текстов.

Алфавит языка – набор таких символов.

Алфавит языка *Turbo Pascal 7.0* включает:

- все латинские прописные и строчные буквы
- арабские цифры *(0 – 9)*
- символы + - * / = < > , . ; : ‘ _ ( ) { } и др.
- служебные *(*зарезервированные*)* слова

Основные средства языка

Для записи команд, имен функций, поясняющих терминов *Turbo Pascal 7.0* существует набор строго определенных слов, которые называются **служебными** или **зарезервированными** (это английские мнемонические сокращения).

Служебные слова делятся на три группы:

- операторы (*READ, WRITELN* и др.)
- имена функций (*SIN, COS* и др.)
- ключевые слова (*VAR, BEGIN, END* и др.)

Именем величины **называется её обозначение, это слово из букв, цифр и знаков подчёркивания, начинающееся с буквы.**

Числа*: целые,*

вещественные: **с фиксированной точкой (-1.23; 654.2),**

с плавающей точкой (2,473 10^{3}, 437,8 10^{-2}).

Шесть операций: **+ сложение, - вычитание, / деление,**
*** умножение, mod нахождение остатка, div деление нацело.**

Арифметическое выражение **конструируется из имён, чисел, знаков арифметических действий, математических функций.**

Для указания *порядка действий* используются только круглые скобки.

Для *возведения в квадрат* используется обозначение sqr(x).

Для *извлечения квадратного корня* используется обозначение sqrt(x).

Модуль **обозначается abs(x).**

Оператор присваивания

Имя переменной := арифметическое выражение.

x:= 3.24 или x:= x+4.

Тип переменной

1. **Если переменная слева вещественного типа, то арифметическое выражение может быть как целого, так и вещественного типа.**
2. **Если переменная слева целого типа, то арифметическое выражение только целочисленное.**

Структура программы в Паскале

1 часть – описание данных и операторов.

2 часть – программный блок.

Общий вид программы:

Program (имя программы)

label (список меток)

const (список постоянных значений)

type (описания сложных типов данных)

var (описания данных программы)

begin (начало программного блока)

(*алгоритм)*

end. (конец программы)

<u>Имя программы</u>: не более 8 знаков, начинается с буквы.

end с точкой.

Описательная часть состоит из 4 разделов: *меток, констант, имён и типов переменных.*

var имя и тип переменной: **integer** (целый), **real** (вещественный).
Например: var i, j: integer; x: real;

Описание каждого типа заканчивается точкой с запятой.

<u>Программный блок</u> содержит операторы, описывающие алгоритм решения задачи.

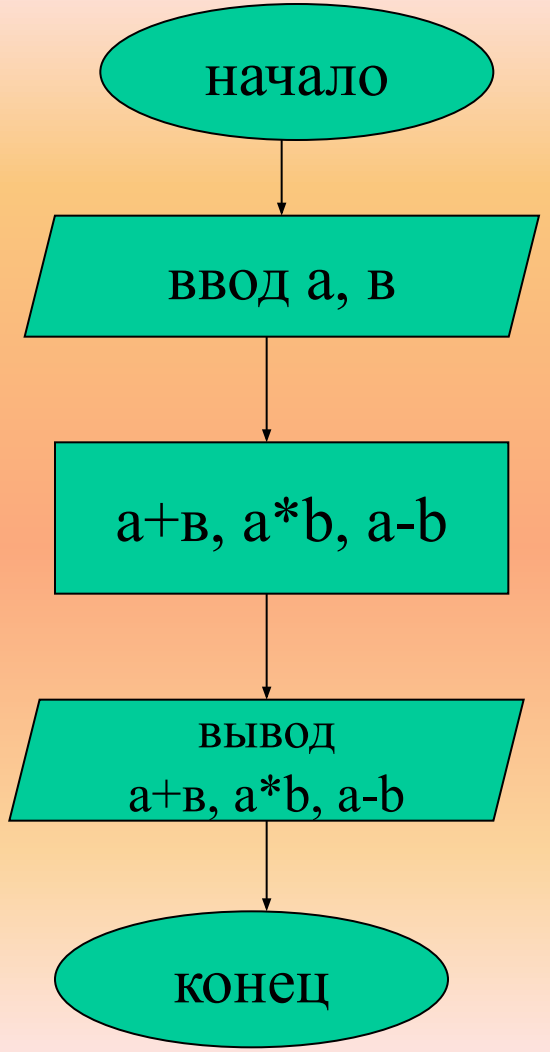
Операторы ввода и вывода:

read (список имён) – останавливает работу программы и ждёт, пока пользователь наберёт на клавиатуре числа (через пробел) и нажмёт ENTER. **Например: read (i, j);** После работы этого оператора курсор располагается за последним числом но не переводится на новую строку. Для перевода курсора на новую строку после ввода данных надо использовать оператор **readln**(список имён).

write (список вывода) – выводит данные на экран. Список вывода - перечисленные через запятую имена результатов вычисления и пояснительные тексты в апострофах. **Например: write('x=', x);** На экране напечатается число с фиксированной точкой. **Например: write('x=', x:6:2);** на экране будет выдано число из 6 знаков из них два после запятой, (x = -23.57 **).**

Перевод курсора на новую строку осуществляется оператором пустого вывода **writeln.** Оператор пустого ввода **readln**

Задача: для двух вещественных чисел найдите сумму, произведение и разность



```
Program E1;
var a,b: real;
begin
    write (‘введите два числа через пробел, затем нажмите <ENTER>’);
   readln (a,b);
   write (‘a + b = ‘ , a + b , ‘ a * b = ‘ , a * b , ‘a - b = ‘ , a – b);
   readln
end.
```

Работа в системе Турбо Паскаль

Alt + F10 – меню

File – New – создать новый файл

Enter – следует поставить в конце каждой строки

Ctrl + Y – удалить строку

Enter – вставить строку

Для выполнения программы надо в меню выбрать Run

Транслятор – переводит программу с языка Паскаль на машинный язык и ищет синтаксические ошибки.

1. **Если ошибки найдены, то произойдёт возврат в редактор, курсор укажет на ошибку.**
2. **Если ошибки не найдены, программа выполняется**

Для выхода из программы выберите файл – exit

или нажмите Alt+x

Для сохранения программы наберите путь и наберите имя файла

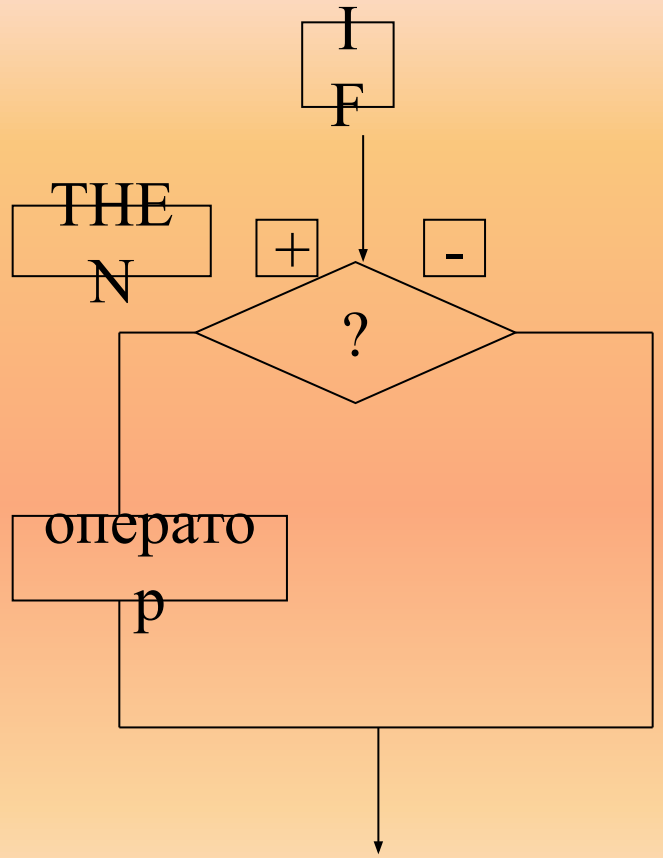
```
Program E2;
var a,b,x,y,z: real;
begin
    write (‘введите два числа через пробел, затем нажмите <ENTER>’);
    readln (a,b);
    x := a + b
    y := a * b
    z := a - b
    write (‘a + b = ‘ , x , ‘ a * b = ‘ , y , ‘a - b = ‘ , z);
    readln
end.
```

Команда ветвления

неполная форма условного оператора

IF условие

THEN оператор;



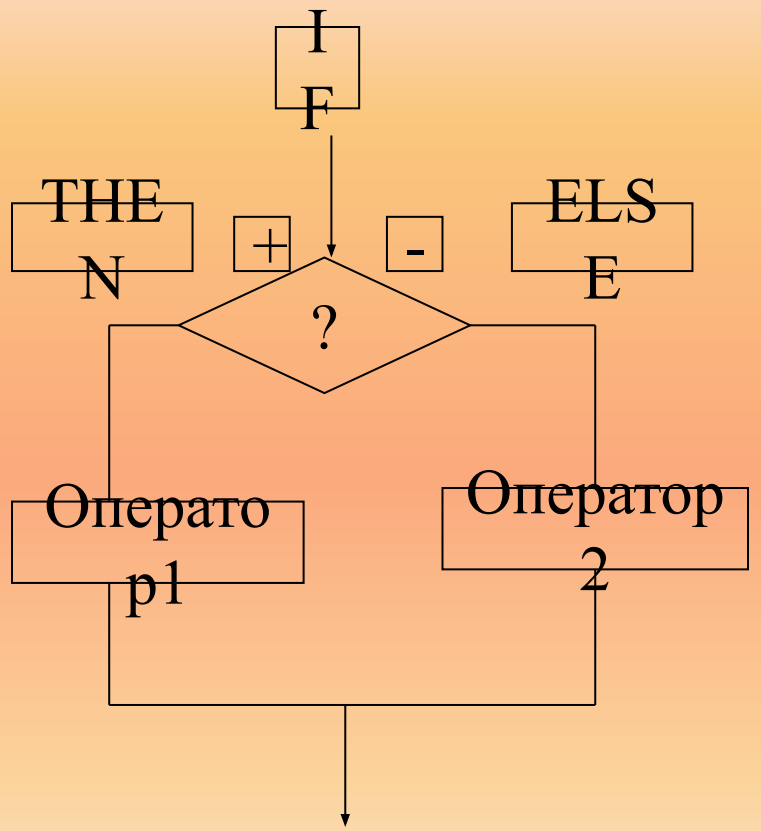
Команда ветвления

полная форма условного оператор

IF условие

THEN оператор 1

ELSE оператор 2**;**



Иногда после слов THEN и ELSE надо выполнить не один, а **несколько** операторов. Тогда эти операторы заключаются в **операторные скобки.**
Открывающая скобка – слово **BEGIN**,
закрывающая скобка – слово **END**.
Перед словом ELSE точка с запятой не ставится.
Рекомендуется каждую пару BEGIN – END записывать в одном столбце.

IF условие
THEN
begin
оператор 1;
оператор 2;
оператор N
end
ELSE
begin
оператор 1;
оператор 2;
оператор M
end ;

Команда ветвления

IF условие **THEN** оператор; - неполный условный оператор,

IF условие **THEN** оператор 1 **ELSE** оператор 2; - полный условный оператор.

Задание

Для двух чисел определите являются ли они чётными?

```
Program E3;
var a,b: integer;
begin
    write (‘введите два целых числа через пробел, затем нажмите    <ENTER>’);
   readln (a,b);
   if a mod 2 = 0 then writeln (‘a – чётное’)
                          else writeln (‘a – нечётное’);
   if b mod 2 = 0 then writeln (‘b – чётное’)
                          else writeln (‘b – нечётное’);
readln
end.
```

Составьте программу для вычисления Y и назовите её E31

$$Y=\begin{cases} 3, \text{ если } a>0 \\ 6, \text{ если } a<=0 \end{cases}$$

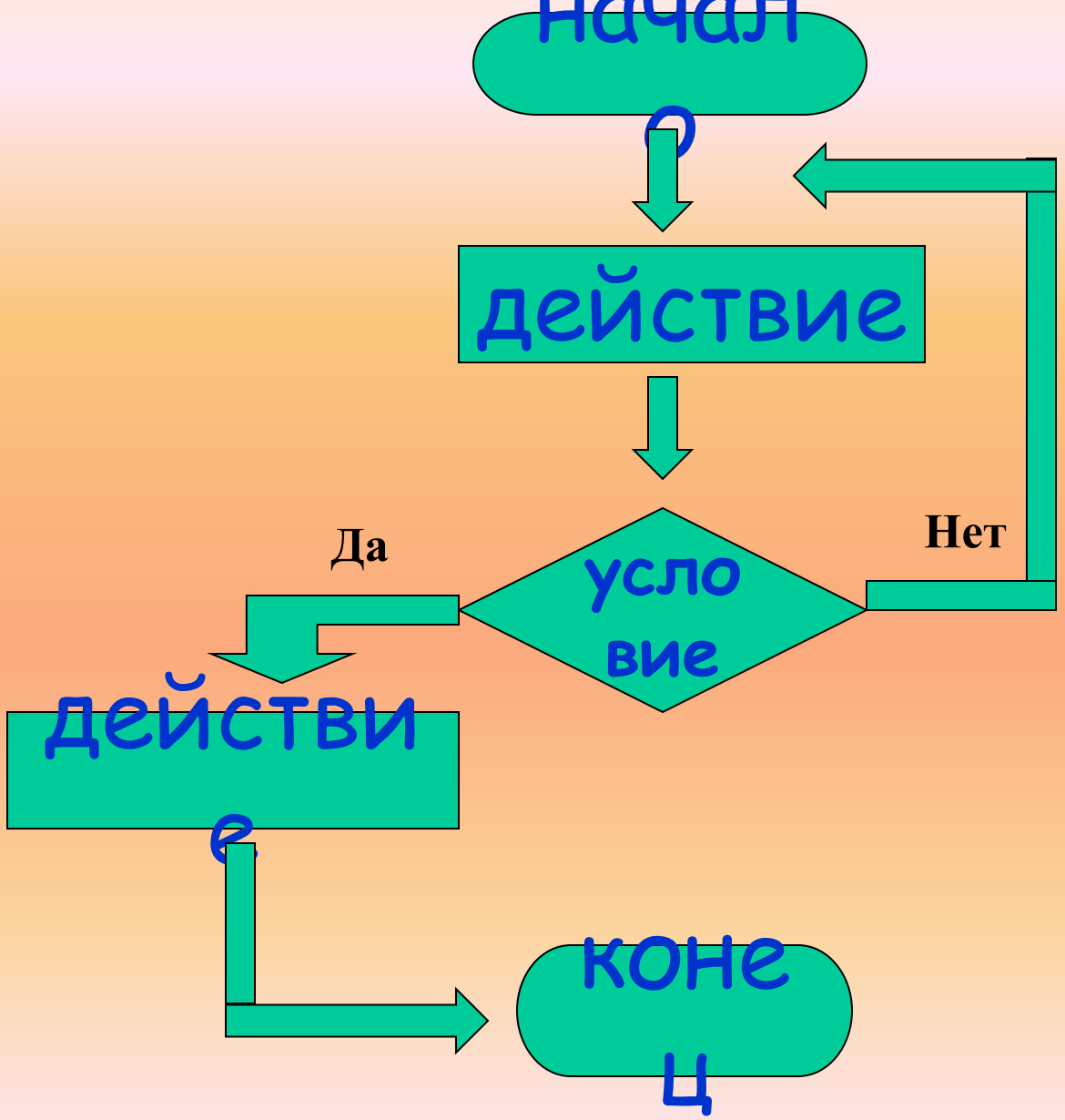
```
Program E31;
var a,y: real;
begin
    write (‘введите число затем нажмите    <ENTER>’);
   readln (a);
   if a >0 then y:=3  else y:=6;
 write (‘y=‘,y);
readln
end.
```

Составьте программу для вычисления X и назовите её E32, a и b –вещественные числа.

$$X=\begin{cases} a+b, \text{ если } a>b \\ a*b, \text{ если } a<=b \end{cases}$$

```
Program E32;
var a,b,x: real;
begin
    write (‘введите два числа через пробел затем нажмите <ENTER>’);
   readln (a);
   if a >b then x:=a+b  else x:=a*b;
 write (‘x=‘,x);
readln
end.
```

Повторение



Организация циклов

Повторяющиеся действия в программировании называются **циклом**

Оператор безусловного перехода

goto n, n - целое число (не более 4-х символов), метка.

Метка повторяется 3 раза:

1. В разделе Label;
2. В операторе goto;
3. Перед оператором на который осуществляется безусловный переход.

Организация циклов с помощью операторов условного и безусловного переходов.

Задача.

Найти сумму первых двадцати чисел. (1+2+3….+20).

a		1	2	3	4	5……20
S	0	1	3	6	10	15……

```
Program E4;
 label 2,1;
 var a, b, s: integer;
begin
write (‘введите два целых числа через пробел, затем нажмите  <ENTER>’);
readln (a,b);
    s:=0;
1:  if a<b+1  then  s:=s+a
                else goto 2;
   a:=a+1;
   goto  1;
 2:  write (‘s=‘,s);
readln;       end.
```

Оператор цикла пока

While *условие* **do** *оператор*

Оператор повторяется пока выполняется условие

Оператор может быть простым и составным

begin…… end – операторные скобки

Задача.

Вычислить наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида:

будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба числа не станут равными.

Например:

Исходные	*1 шаг*	*2 шаг*	*3 шаг*	*НОД(a,b)=5*
a= 25	10	10	5	
b=15	15	5	5	

```
Program E5;
var a, b: integer;
begin
write (‘Введите два натуральных числа через пробел’);
readln (a,b);
 while a<>b do
                    if a>b then a:=a-b
                           else   b:=b-a;
write (‘НОД=‘,a);
readln;
end.
```

Оператор цикла до

repeat *оператор* **until** *условие*

Между словами **repeat** *и* **until** *может быть любое количество операторов без использования операторных скобок.*

Задача.

Вычислить наибольший общий делитель двух натуральных чисел a и b.

```
Program E6;
var a, b: integer;
begin
write (‘Введите два натуральных числа через пробел’);
readln (a,b);
 repeat
                    if a>b then a:=a-b;
                    if b>a then b:=b-a
 until a=b;
write (‘НОД=‘,a);
readln;
end.
```

Операторы циклов пересчёт

for *i:=n1* **to** *n2* **do** *оператор;*

«Для i начиная с n1 до n2 выполнить оператор»

i – переменная цикла;

Если i+1<= n1, *то оператор выполняется, если нет, то происходит выход из цикла и выполнение следующего по порядку оператора.*

Оператор может быть простым и составным

begin…… end – *операторные скобки*

Задача.

Вычислите P = n! *(n факториал).*

*По определению n! = 1*2*3*4*5…..n.*

i	1	2	3	4	5
P	1	2	6	24	120

5! = 120

```
Program E7;
var n, p, i : integer;
begin
    write (‘Введите целое n =’);
    readln (n);
    p: = 1;
    for i: = 1 to n do
        p: = p*i ;
write (n , ‘!=‘ , p);
readln;
end.
```