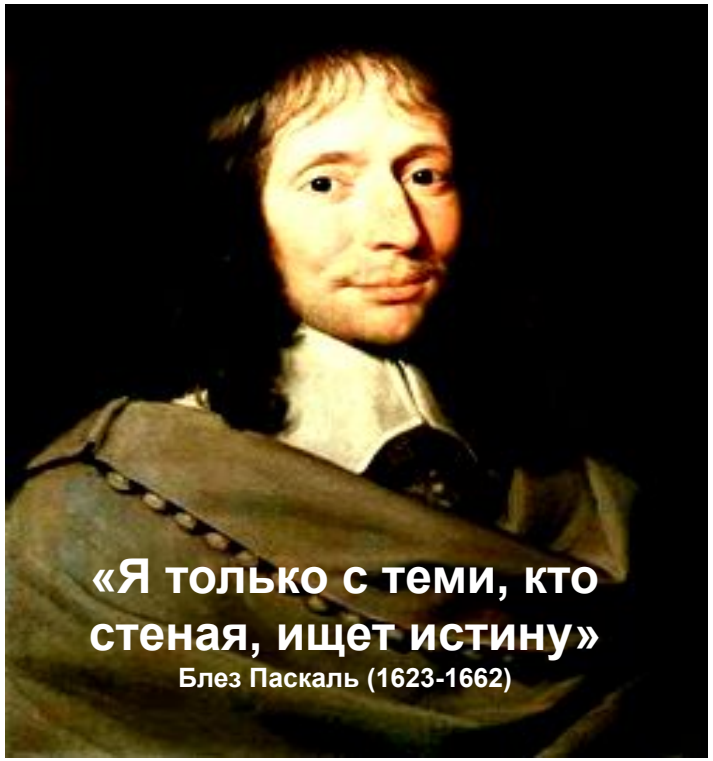


PASCAL

ВВЕДЕНИЕ В ОСНОВЫ ПРОГРАММИРОВАНИЯ

Основные этапы решения задач на ЭВМ

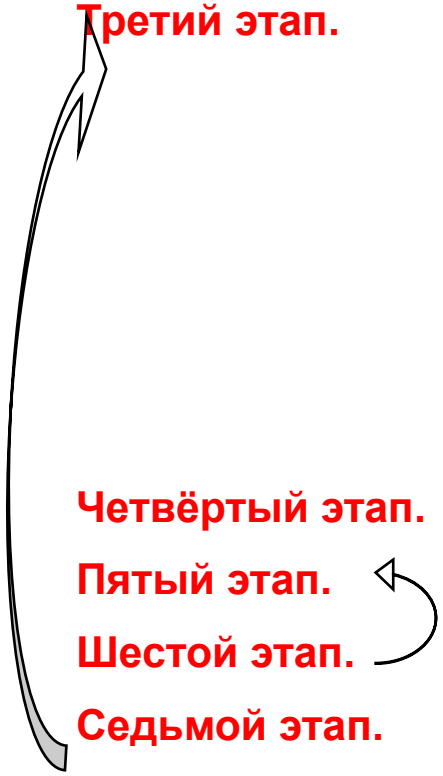
Первый этап. **Постановка задачи.**

Второй этап. **Математическое или информационное моделирование.**

Третий этап. **Алгоритмизация задачи.**

Свойства алгоритма:

1. Понятность.
2. Дискретность.
3. Определённость.
4. Результативность.
5. Массовость.



Четвёртый этап. **Программирование.**

Пятый этап. **Ввод программы и исходных данных в ЭВМ.**

Шестой этап. **Тестирование и отладка программы.**

Седьмой этап. **Исполнение отлаженной программы и анализ результатов.**

Основные достоинства языка Pascal

1. Отвечает требованиям структурного программирования

Позволяет строить программу из отдельных блоков.

- *применяются три управляющие конструкции: следование, выбор, повторение;*
- *структура программы отражает структуру данных;*
- *на первом этапе проводится проектирование программы, а на втором её написание.*

2. Строго типизированный язык

Содержит полный набор структурных типов данных, а также развитые средства построения из них новых типов данных.

Работа с данными

ОПРЕДЕЛЕНИЕ ТИПА ДАННЫХ

Однозначность операций над данными

ИДЕНТИФИКАЦИЯ ПЕРЕМЕННЫХ

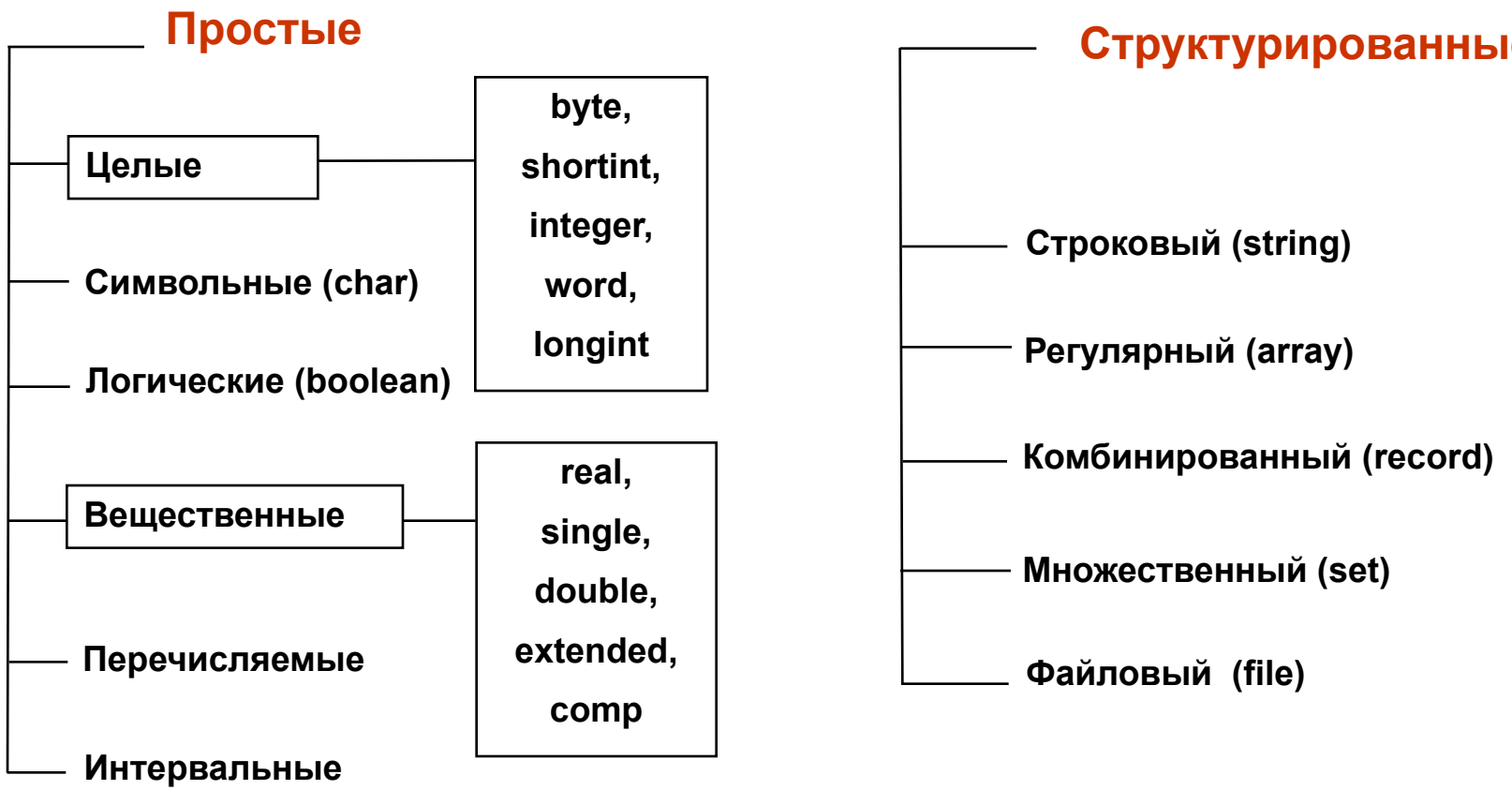
Объявление идентификаторов

ЗАДАНИЕ ЗНАЧЕНИЙ

Операции с данными

Типы данных

Тип данных — это характеристика идентификатора, определяющая множество значений, которые он может принимать (целые или дробные числа, строки и т. д.).



Простые типы: одна переменная — одно значение.
Структурированные типы: одна переменная — несколько значений.

Целый и вещественный типы данных

Конечный набор возможных значений

Тип	Диапазон значений
byte	0…255
shortint	− 128…127
word	0…65535
integer	− 32768…32767
longint	− 2147483648…2147483647

Тип	Диапазон десятичного порядка
real	−39…+38
single	−45…+45
double	−324…+308
extended	−4932…+4932
comp	−263+1…263 −1

Выход за пределы диапазона приводит к ошибке

Синтаксис:

```
Var <имя переменной>:<тип переменной>;
```

Резервирует место в памяти компьютера под переменные: **a**, **x**, **y**.

```
Program My_program;
uses CRT; {подключение модуля CRT}

var
    a:integer;
    x,y:real;
begin

ClrScr;{процедура очистки экрана}


Readkey;{ожидание нажатия клавиши}

end.
```

Операции с целыми переменными

Арифметические операции: `Sqr` ,`+`, `–`, `*`, `/`

Нельзя использовать с целыми типами

Стандартные функции:

div — вычисляет целую часть от частного, дробная откидывается.	`10 div 3=3;` `2 div 3=0;`
mod — вычисляет остаток, полученный при делении.	`11 mod 5 = 1;` `14 mod 5 = 4;`

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:integer;

begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('y='); read(y);
writeln(x ,' div ',y,'=',x div y );
writeln(x ,' mod ',y,'=',x mod y );
readkey;  {ожидание нажатия клавиши}
END.
```

Работа функций используется в операторе вывода.

```
введите два числа
x=10
y=3
10 div 3=3
10 mod 3=1
```

Операции с вещественными переменными

Арифметические операции: `Sqr`, `+`, `-`, `*`, `/`

Стандартные функции: `Pi`, `Sqrt`, `Sin`, `Cos`, `Abs`, `Exp`, `Ln`.

вещественный → вещественный: **`Frac`**, **`Int`**;	✔ вычисление дробной части числа **`Frac(5.67)=0.67`**
	✔ вычисление целой части числа **`Int(5.67)=5.0E+00`**
вещественный → целый: **`Round`**, **`Trunc`**.	✔ округление вещественного числа до ближайшего целого **`Round(5.67)=6`**
	✔ отбрасывание дробной части числа **`Trunc(5.67)=5`**

```
uses crt;    {подключение модуля CRT}
var x:real;
begin
ClrScr; {очистка экрана}
writeln('введите любое дробное число');
write ('x='); read(x);

writeln('Frac(' ,x, ')=', frac(x),' форматируем и получаем - ', frac(x):6:2);
writeln('Int(' ,x, ')=', int(x),' форматируем и получаем - ', int(x):2:0);
writeln('Round(' ,x, ')=', round(x));
writeln('Trunc(' ,x, ')=', trunc(x));
readkey;
end.
```

```
Output
введите любое дробное число
x=5.67
Frac( 5.6700000000E+00)= 6.7000000000E-01 форматируем и получаем -   0.67
Int( 5.6700000000E+00)= 5.0000000000E+00 форматируем и получаем -  5
Round( 5.6700000000E+00)=6
Trunc( 5.6700000000E+00)=5
```

Запрос данных с клавиатуры

Ввод информации с клавиатуры обеспечивает процедура ввода:
Read или **ReadLn.**

Синтаксис:

Read (N1, N2, … Nn);

Где **N1, N2, … Nn** — переменные (целые, вещественные, строковые).

Read(Ln) — курсор устанавливается на следующую строку.

В переменную ***X*** и ***A*** заносится значение, введённое с клавиатуры.

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:real;
   a:integer;
begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('a='); read(a);
readkey;  {ожидание нажатия клавиши}
END.
```

- После ввода значения, необходи-мо нажать клавишу **Enter**.
- Если переменных в операторе указано несколько, то они вводятся через **пробел**, либо через нажатия клавиши **Enter**.

```
введите два числа
x=2.5
a=78
```

Ввод данных

Ввод данных с клавиатуры в текстовом режиме:

1. Через функцию **ReadKey** для чтения первого байта из очереди нажатий на клавишу.

2. Через процедуру ввода **Read(Ln)**

```
uses crt;
var c:char;
begin
ClrScr;
writeln('нажмите  клавишу') ;
c:= readkey; {ожидания нажатия
пользователем клавиши и занесение
символа, соответствующего нажатой
клавише, в переменную C}
write('c=' ,c);{ ожидание нажатия
пользователем клавиши}
readkey;
end.
```

```
3:20
Output -2
нажмите  клавишу
c=s
```

Ввод данных с клавиатуры непосредственно в программе:

3. Через оператор присваивания **:=**.

Тип переменной должен совпадать с **типом вводимых значений** для этой переменной.

Операторы присваивания

Для задания значения переменной необходимо воспользоваться оператором присваивания **:=**

Синтаксис:

```
<Переменная>:=<Значение>;
```

```
Program My_program;

Var A:Integer;
    X,Y:Longint;

Begin

A := 3;

End.
```

В переменную (целочисленную) с именем A заносится значение 3.

Вывод информации на экран

Вывод информации на монитор обеспечивает процедура вывода: **Write** или **WriteLn**.

Синтаксис:

```
Write (N1, N2, … Nn);
```

N1, **N2**, … **Nn** — переменные (целые, вещественные, строковые).

WriteLn — перемещает курсор на следующую строку.

```
Program My_Program;

Begin

 Write('Message 1');
 Write('Message 2');
 Write('Message 3');

End.
```

```
Message 1Message 2Message 3
```

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:real;
   a:integer;
begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('a='); read(a);
write(x);
writeln;
write(a);
readkey;  {ожидание нажатия клавиши}
END.
```

```
введите два числа
x=4
a=56
 4.0000000000E+00
56
```

«Пустой» оператор **WriteLn** добавляет пустую строку.