



# Basics of Programming. Introduction

Course Basics of Programming Semester 1, FIIT

Mayer Svetlana Fyodorovna

# Resources & Timetable of our classes

- **Internet resources**

- Course forum topic [forum.mmcs.sfedu.ru](http://forum.mmcs.sfedu.ru)
- Moodle environment for practice [pascalabc.net](http://pascalabc.net) [edu.mmcs.sfedu.ru](http://edu.mmcs.sfedu.ru)
- Site of PascalABC.NET Development Environment <http://pascalabc.net>

- **Timetable of classes**

- *Lectures :*

Tuesday

**11.55 – 13.30**

**13.45 – 15.00 (from 13 of September, every other week)**

Friday

**8.00 – 9.35**

**9.50 – 11.25**

# Programming as a human activity

- Programming is a part of the human activity of every person
- In everyday life, we take actions, make decisions using some algorithms
- Many tasks today require automation in the form of a computer programming
- The number of such tasks is growing
- Every year new areas of human life are covered by the computer programming. The areas where programming is involved are: machine learning, images recognition, robot programming, etc.
- We need to learn **how to write computer programs** using some programming language and learning some of **standard computer algorithms**

# Algorithms

- The main concept in the field of CS is the **Algorithm**
- An **algorithm** is a set of instructions for solving a problem

Example. How to register in a moodle system?

- \* Step 1. Open browser and type the address [edu.mmcs.sfedu.ru](http://edu.mmcs.sfedu.ru)
- \* Step 2. Press the button "Register" (Register)
- \* Step 3. Enter your email address (@sfedu), name and password
- \* Step 4. Go to your e-mail and confirm the registration
- Many other algorithms surround us in real life
- Algorithm in computer science (CS) is adapted to solve a problem using computer
- Our task is to study the process of algorithms creation and the process of computer programs writing

# Main definitions

- An **algorithm** is a set of commands to solve a problem
- A **program** is an algorithm written in some **Programming Language**
- **Algorithm steps** are called **commands**
- The **simplest steps of a program** are called **statements** (or operators)

Some Programming Languages:

Java	JavaScript	Scratch
Kotlin	Go	Perl
C	Swift	Assembler
C++	<b>Delphi Pascal</b>	<b>PascalABC.NET</b>
Python	Basic	
<b>C#</b>	Haskell	

# We will use PascalABC.NET. Why?

- **Traditional Pascal:**

- Was developed for teaching schoolchildren and students in the 70s of the XX century. Was created by the scientist Niklaus Wirth

- **PascalABC.NET :**

- is created in our institute
- is oriented on modern programming
- makes it possible to write code compactly and clearly
- includes baseline Pascal, Delphi Pascal, .NET extensions (similar to C#)

- **As a result, after 1 semester:**

- 1. We will study all basic constructions of modern programming languages
- 2. We will move to the industrial C# language



Let's start programming !

# Variables

## Some preliminary concepts

**Variable** – memory cell that has a **name**, **type** and **value**

5
---

a

2.5
-----

r

Hello
-------

s

**Type of a variable** – defines set of values that a variable can take

### Main types:

integer      5   -3   2019

real          2.5   3.33

string        'Hello'   'Student'   '2019'

**Comment.** Every variable in a moment can have only one value

5
---

a

6
---

a



# Variable Definition

Every variable must be defined before using it in a program. When we define a variable, we specify its **name** and **type**:

```
var a: integer;  
var r: real;  
var s: string;
```

semicolon  
after every definition

**var** – keyword (boldface)

What could be a name? **Name of a variable** is any sequence of letters & digits, started with letter:

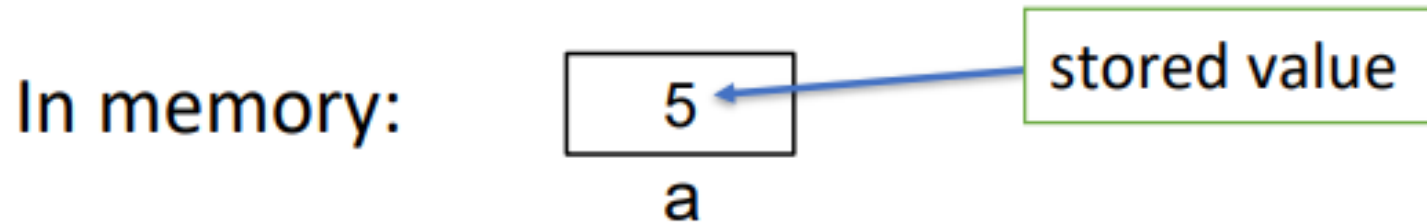
```
i r s a123 Count0 NumberOfPupil _System
```

Bad name:

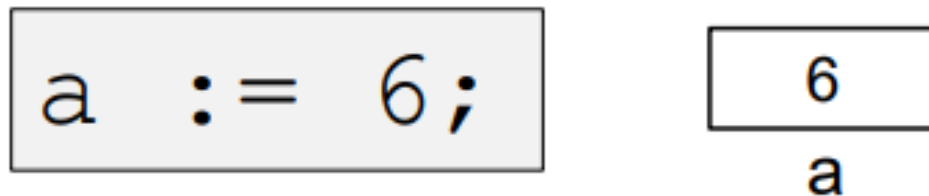
```
123a
```

Underline symbol is  
a letter too!

# Assigning value to a variable



Memory cell can store only one value! Old value is erased!



**Statement** is a minimal action in a programming language

# Calculations

How to calculate an expression?

Simply write it and assign to a variable!

```
a := 3 + 4;
```

expression

**Step 1.** Expression is calculated

**Step 2.** The result is assigned to variable

7

a

# Variables in expressions

- How to calculate an expression with variables?

```
x := 3;  
y := 4;  
a := x + y;
```

- **Step 1.** If the expression contains variables, then the values are substituted into place of variables:
- **Step 2.** Expression is calculated
- **Step 3.** The result is assigned to a variable

```
a := 3 + 4;
```

# First program in PascalABC.NET

```
var x,y: integer;  
var a: integer;
```

```
begin
```

```
  x := 3;  
  y := 4;  
  a := x + y;  
  Write(a);
```

```
end.
```

definition  
section

statements  
section

Write statement  
outputs the results to  
an output window  
(console window)

output window  
(console)


We can move  
variable definitions  
into the statements  
section

```
begin
```

```
  var x,y: integer;  
  var a: integer;  
  x := 3;  
  y := 4;  
  a := x + y;  
  Write(a);
```

```
end.
```

# How the program is executed?

Press  on a toolbar

The special program – PascalABC.NET **Compiler** – will start

Compiler verifies program and if it has no errors, compiler generates so called **machine codes**

Machine codes are executed by Central Microprocessor (CPU)

```
begin
  var x,y: integer;
  var a: integer;
  x := 3;
  y := 4;
  a := x + y;
  Write(a);
end.
```

7



Compiler

```
5:      x := 3;
mov dword [rip-0x11d212], 0x3
6:      y := 4;
mov dword [rip-0x11d218], 0x4
7:      a := x + y;
mov ecx, [rip-0x11d222]
add ecx, [rip-0x11d224]
mov [rip-0x11d226], ecx
8:      Write(a);
mov rcx, 0x7fff94d9c158
call 0x5f5e0aa0
mov [rbp-0x10], rax
mov rcx, [rbp-0x10]
mov eax, [rip-0x11d243]
mov [rcx+0x8], eax
mov rcx, [rbp-0x10]
call 0xffffffffffffe758
```

are executed by  
Central  
Microprocessor: its  
performance is  
**3 000 000 000** of  
commands per  
second

Machine codes



# Modifying program (1)

```
begin  
  var x,y: integer;  
  var a: integer;  
  x := 3;  
  y := 4;  
  a := x + y;  
  Write(x,y,a);  
end.
```

347

Oops! Output is bad! No  
spaces between elements

# Modifying program (2)

```
begin
  var x,y: integer;
  var a: integer;
  x := 3;
  y := 4;
  a := x + y;
  print(x,y,a);
end.
```

3 4 7

Use **Print** instead **Write** to output spaces between elements



# Modifying program (3)


```
begin
  var x,y: integer;
  var a: integer;
  x := 3;
  y := 4;
  a := x + y;
  print('${x} + {y} = {a}');
end.
```

**Interpolated string** (with \$)  
all the expressions and variables within the braces are replaced by its values

3 + 4 = 7

# Modifying program (4)

```
begin
  var x,y: integer;
  var a: integer;
  x := 3;
  y := 4;
  a := x + y;
  writelnFormat('{0} + {1} = {2}', x, y, a);
end.
```



3 + 4 = 7

## Formatted output

all the numbers within the braces are replaced by values of variables

# Thus, about output:

1.

```
begin
  var n:integer;
  n := 5;
  n:= n * n;
  print('n =',n); // n = 25
end.
```

```
begin
  write('2+');      { без перехода }
  writeln('2=?');  { переход на новую строку}
  writeln('Ответ: 4');
end.
```

2.

```
begin
  var a:=1.2;
  var b:=4;
  var c:=a+b;
  WritelnFormat ('f ({0}, {1}) = {2}', a, b, c);
end.
```

3.

```
begin
  var x := 5;
  var y := 6;
  var res := x + y;
  Print('$'Cymma {x} + {y} = {res}');
end.
```

# Arithmetic operations and expressions

## Common method:

**begin**

```
var a := 6; // Assigning value 6
a := a + 2; // Increasing by 2
a := a - 2; // Reduction of 2
a := a * 3; // Multiplication by 3
a := a / 2; // division
print(a**2); // a is a base number, 2 is an exponent
```

**end.**

## Short method:

**begin**

```
var a := 6; // Assigning value 6
a += 2; // Increasing by 2
a -= 2; // Reduction of 2
a *= 3; // Multiplication by 3
a /= 2; // division
```

**end.**

# Data input

```
begin
```

```
  var n:integer; // n is a variable of integer type  
  read(n); // input some value to store it in n variable
```

```
end.
```

```
begin
```

```
  var n:real; // n is a variable of real type - floating point number  
  read(n); // input some value to store it in n variable
```

```
end.
```

```
begin
```

```
  // x1 is a variable of integer type & we input some value to store it in x1:
```

```
  var x1:=ReadInteger('please, enter x1');
```

```
  // x2 is a variable of real type & we input some value to store it in x2:
```

```
  var x2:=ReadReal('please, enter x2');
```

```
  var (y1,y2):=ReadInteger2('please, enter two numbers'); ;
```

```
  var (z1,z2,z3):=ReadInteger3;
```

```
end.
```

# Tasks

- To do: tasks 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Follow the rules to make the tasks

1. Save your files with names as it is given in tasks (e.g. `task-04.pas`).
2. Give meaningful names to your variables.
3. Use comments to make the program clear.
4. Give the task of the program as a comment before the program code. Use curly braces for comments:

```
•Program1.pas*  
{Find the distance between two points on the plane;  
coordinates (x1,y1) and (x2,y2) are given. }  
  
begin  
  Writeln('coordinates of the first point:');
```

copy & paste the text  
of the task in the form  
of comment before the code

# Example



## Sample 1:

**To do:** Calculate the expression. The values of `x`, `y` and `z` are entered.

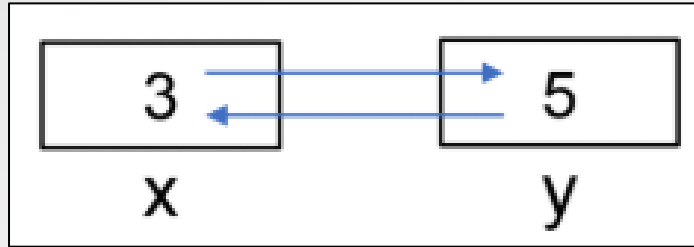
$$\frac{x + \sin x}{y - \sin z} + \ln(x + \sin x)$$

## The resulting example:

```
Input x
3
Input y
4
Input z
5
The result = 1.77800712886037
```

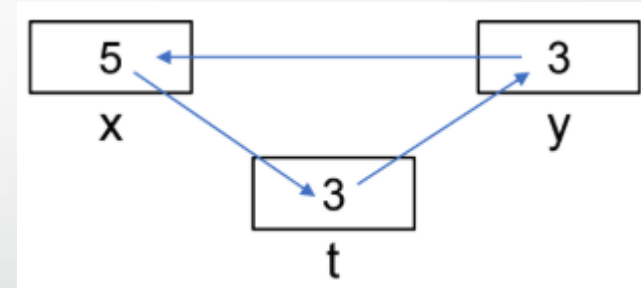
[Program name: L1sample1.pas]

# Interchange x and y



**Solution 1.** Using temporary variable:

```
var t := x;  
x := y;  
y := t;
```



**Solution 2.** Using multiple assignment:

```
(x, y) := (y, x);
```





Q & A