

Санкт-Петербургский государственный университет  
Экономический факультет  
Кафедра Информационных систем в экономике

---

# АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Программирование

**Лекция 1**

*Основы языка  
программирования*

# *План лекции*

---

- 1. Структура программы на алгоритмическом языке.*
- 2. Основные понятия языка программирования. Алфавит языка.*
- 3. Типы данных и объявление переменных.*

# 1. Структура программы на АЯ

Лекция 1(определение2) =>

**Программа** – это набор операторов (команд), представленный как единое целое в некоторой вычислительной системе и который используется для управления поведением этой системы

(некоторая инструкция для ЭВМ, составленная на некотором «языке» по определенным «правилам»)

# 1. Структура программы на АЯ

«правила» математики:

$$\underline{y=f(z(x),x), f(z(x),x)=z(x)^2+z(x)+x, z(x)=2x+3}$$

Чтобы получить значение  $y$  нужно:

**I.** вычислить (**вызвать для выполнения**) функцию  $f$ ;

1. Чтобы получить значение функции  $f$  нужно вычислить (**вызвать для выполнения**) функцию  $z$ ;

a. Чтобы получить значение функции  $z$  нужно ее вычислить, подставив в формулу значение **формального аргумента  $x$** .

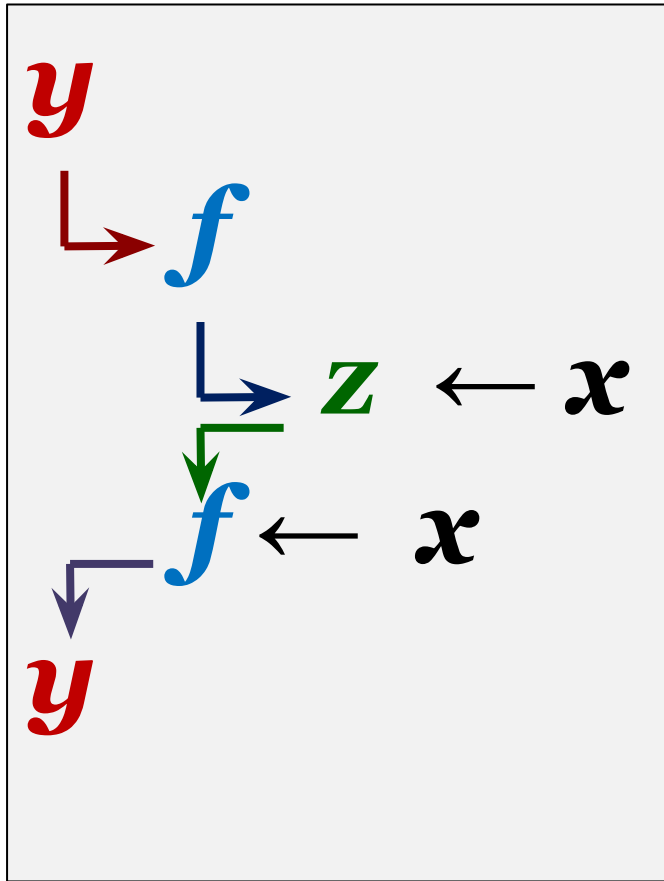
b. Функция  $z$  подставит (**вернёт** в точку вызова) своё значение в формулу для вычисления функции  $f$ .

2. Функция  $f$  подставит (**вернёт** в точку вызова) своё значение на место правой части равенства  $y=f(z,x)$ ;

**II.**  $y$  получит **значение результата  $f$** .

# 1. Структура программы на АЯ

$$y = f(z(x), x), \quad f(z(x), x) = z(x)^2 + z(x) + x, \quad z(x) = 2x + 3$$
$$\Rightarrow \quad y = (2x + 3)^2 + (2x + 3) + x$$



Итак. В математике: **выражение** – есть суперпозиция **функций**, каждая из которых:

имеет **имя** ( $f$  или  $z$ ) и каждая из которых может **зависеть** ( $f$  от  $z$ ) или **не зависеть** ( $z$ ) одна от другой.

Каждая функция в результате своего вычисления получает значение

(**возвращает результат**

**выполнения**). Каждая функция

зависит от аргументов – значений

**переменных**

# 1. Структура программы на АЯ

«правила» программирования (Си, С++, С#, ...):

**Функция** (программа/подпрограмма)

**Предопределённая функция**

(встроенный набор: `sin( )...` )

**Пользовательская функция**

( разработанная пользователем )

**Главная функция**

( точка входа в программу, начало ее работы )

**Параметры функции**

( аргументы: переменные, константы и др.)

**Возвращаемое значение**

функции (результат вычислений)

**Имя функции** ( точка вызова подпрограммы)

**Область действия** функции (Начало/конец функции )

# 1. Структура программы на АЯ

«скелет» условной программы

Задание директив препроцессора

```
#include <stdio.h>  
#include <iostream>
```

Описание внешних переменных,  
констант, заголовков  
(прототипов) внешних функций  
(подпрограмм)

```
using namespace std;
```

Описание главной функции

```
{  
    Тело главной функции;  
}
```

```
int main (void)
```

```
{  
    printf ("Hello\n");  
    cout << "Word!";  
    return 0;
```

Описание внешней функции

```
{  
    Тело внешней функции;  
}
```

```
}
```

# 1. Структура программы на АЯ

директивы препроцессора

**#include <stdio.h>** - заголовочный файл библиотеки стандартного ввода-вывода (функции: **scanf()**, **printf()**, ... )

**#include <iostream>** - заголовочный файл библиотеки потокового ввода-вывода (операторы **cin>>**, **cout<<**, ...)

**#include <conio.h>** - заголовочный файл библиотеки консольного ввода-вывода (функция **getch()**, ....)

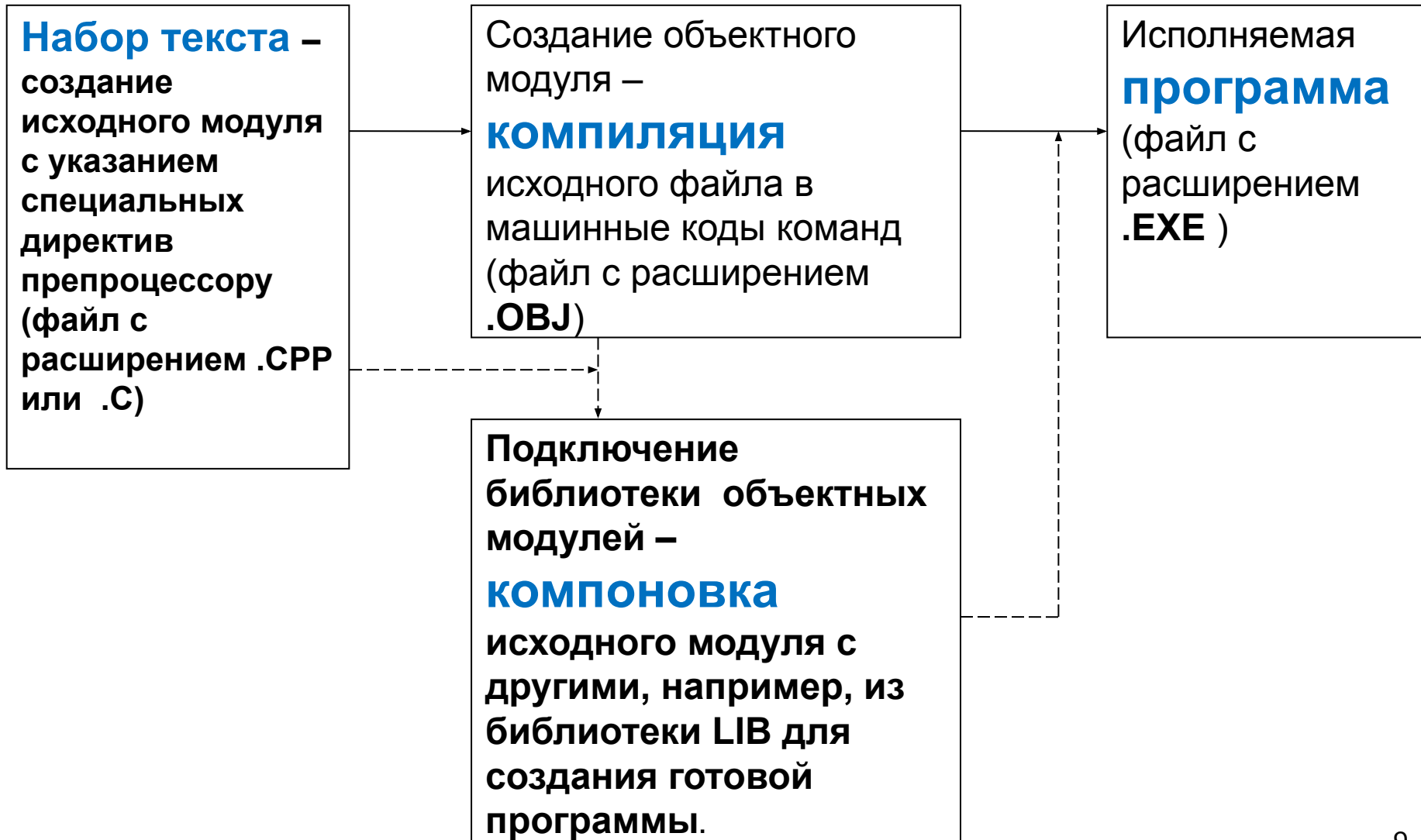
**#include <math.h>** - заголовочный файл библиотеки математических функций (функции: **sqrt()**, **pow()**,...)

...



# 1. Структура программы на АЯ

## процесс создания исполняемого файла программы (.exe)



## **1. Структура программы на АЯ. Выводы**

**1. Программа** в языке Си **состоит из** одной или нескольких **функций**.

**2.** Функции могут быть **пользовательскими** и **встроенными** (предопределёнными) функциями.

**3.** Встроенные функции подключаются к исходному коду программы на этапе **препроцессорной обработки**.

**4.** Из полного текста программы **компилятором** формируется **объектный код**.

**5.** После обработки объектного кода **компоновщиком** получается **исполняемый код программы**.

## 2. Язык программирования. Алфавит

### Определение 9.

**Алфавит языка** – это **символы**, которые можно использовать для записи программы на данном языке.

- Множестве символов таблицы кодов **ASCII**
- Пять типов лексем:
  - ◆ **идентификаторы,**
  - ◆ **ключевые (служебные) слова,**
  - ◆ **знаки (символы) операций,**
  - ◆ **литералы,**
  - ◆ **разделители**

## 2. Язык программирования. Алфавит

**Идентификаторы** – это **имена**, которые присваиваются переменным, функциям, операторам и другим функциональным объектам программы (любая последовательность символов).

Рекомендации при объявлении

правильно	ошибочно	причина
Name_1	Name 1	Содержит пробел
_1Name	1Name	Начинается с цифры
MyName	ЯName	Содержит символ кириллицы
My_int	int	Содержит служебное слово

## 2. Язык программирования. Алфавит

### Символы операций и разделителей

,	!	!=		=	%	%=	&
&&	&=	()	*	*=	+	++	+=
-	--	-=	->	->*	.	.*	/
/=	::	<	<<	<=	<<=	>	>>
>=	>>=	==	?:	[]	^	^=	~
	#	##	<u>sizeof</u>	<u>new</u>	<u>delete</u>	<u>typeid</u>	<u>throw</u>

### Символы-разделители

...	;	}
-----	---	---

## 2. Язык программирования. Алфавит

### Литералы

#### ✓ Целочисленный литерал

последовательностью цифр (возможно со знаком '-'):

122<sub>10</sub>, 02<sub>8</sub>, 0X<sub>16</sub>

#### ✓ Вещественный литерал

десятичная или научная нотации: 2.345, 2345E-3

#### ✓ Символьный литерал

последовательность из одной или нескольких литер, заключённых в одинарные кавычки: 'A', 'abc'

#### ✓ Строковый литерал

последовательность (возможно, пустых) литер, заключённых в двойные кавычки: "my programm 1"

## *2. Язык программирования. Алфавит*

### **Служебные литералы**

<b>\0</b>	<b>пустая литера</b>
<b>\a</b>	<b>сигнал</b>
<b>\b</b>	<b>возврат на шаг</b>
<b>\f</b>	<b>перевод страницы</b>
<b>\n</b>	<b>перевод строки</b>
<b>\r</b>	<b>возврат каретки</b>
<b>\t</b>	<b>горизонтальная табуляция</b>
<b>\v</b>	<b>вертикальная табуляция</b>
<b>\\</b>	<b>обратная косая черта</b>
<b>\'</b>	<b>апостроф</b>
<b>\"</b>	<b>двойная кавычка</b>
<b>\?</b>	<b>Вопросительный знак</b>

## *2. Язык программирования. Алфавит*

### **Выводы**

- 1. Файл с программой на Си состоит из последовательности объявлений различных синтаксических единиц языка программирования.**
- 2. Тело функции представляет собой блок, состоящий из различных синтаксических конструкций, построенный на основе алфавита языка, заключаемый в фигурные скобки.**



# 3. Типы данных и объявление переменных



## 3. Типы данных и объявление переменных

### Простые типы данных

[unsigned] **char** – [без знака] символ (**1 байт**);

[unsigned] **int** – [без знака] целое число (**2 байта**);

[unsigned] **short** – [без знака] короткое целое (**2 байта**)

[unsigned] **long** – [без знака] длинное целое (**4 байта**)

**float** – дробное (вещественное) число (**4 байта**)

**double** – вещественное с двойной точностью (**8 байт**),

**long double** – длинное дробное двойной точности (**10 байт**)

**bool** – логический тип данных (**true, false**)

### 3. Типы данных и объявление переменных

#### Определение 9.

**Переменная** – это именованная область памяти, в которой хранятся данные определённого типа.

```
#include<iostream>
```

```
int main( )
```

```
{ float a1; // объявлена переменная a1 для работы с  
// дробными числами
```

```
int two; // объявлена целочисленная переменная two
```

```
char s; // объявлена символьная переменная s.
```

```
int a,b,c; //объявлены три целочисленные переменные
```

```
return 0;
```

```
}
```

### 3. Типы данных и объявление переменных



1. Когда речь идёт об описании какой-либо переменной, то имеют в виду задание этой переменной ее **уникального имени**, а также **типа**, соответствующего тем данным, с которыми эта переменная будет оперировать.
2. Машинное представление программы предполагает, что **каждой переменной отводится некоторый участок памяти**. Размер этого участка и интерпретация его содержимого **определяется типом**.
3. **Основные характеристики** переменной: **имя**, **тип**, **класс памяти**, **время жизни**

### 3. Типы данных и объявление переменных

#### Определение 10.

**Выражение** – это формула для вычисления переменных.

Выражение состоит из одного или более числа **операндов** (переменные, константы и др. конструкции языка), соединённых знаком **операций**.

#### Определение 11.

**Операция** – это конструкция языка Си, состоящая из одного или более арифметических или логических символов и имеющих приоритет (ранг) выполнения.

## 3. Типы данных и объявление переменных

Ранг	Операции	Группа
1.	() [] -> .	Выбор компонентов
2.	! (НЕ-логическое отрицание) ~ (побитовое отрицание) + - ++ (инкремент) -- (декремент) & (получение адреса) * (обращение по адресу) (тип) (преобразования типа) <u>sizeof</u> (вычисление размера в байтах)	Унарные
3.	* / % (остаток от деления)	Мультипликативные - бинарные
4.	+ -	Аддитивные - бинарные
5.	<< >>	Поразрядного сдвига- бинарные
6.	< <= >= > (отношения)	Бинарные
7.	= = != (отношения)	Бинарные
8.	& (поразрядная конъюнкция «И»)	Бинарные
9.	^ (поразрядная исключающее «ИЛИ»)	Бинарные
10.	(поразрядная дизъюнкция «ИЛИ»)	Бинарные
11.	&& (конъюнкция «И»)	Бинарные
12.	(дизъюнкция «ИЛИ»)	Бинарные
13.	?: (условная операция)	Трехместная
14.	= *= /= %= += -= &= ^=  = <<= >>=	Присваивания - бинарные
15.	, (операция запятая)	

# Примеры операций

Сложение	+	<code>c=a+b;</code>
Вычитание	-	<code>d=a-b;</code>
Умножение	*	<code>s=a*b;</code>
Деление	/	<code>p=a/b;</code>
Деление по модулю (остаток от деления)	%	<code>int a=19; int b=8; D=a % b; // D=3</code>

```
#include<iostream>
int main()
{
    int x,y,z; // объявление целочисленных переменных x,y,z
    x=20;
    y=5;
    z=x/y; // выполнение операции деления
    cout<< z <<'\n'; // вывод на экран результата деления
    cout<<x*y; // вывод на экран результата умножения
    return 0;
}
```

# Логические операторы.

Результатом логических операций может быть либо «ложь» (0), либо «истина» (1). Логическими операторами являются:

**логическое отрицание** (инверсия): **!**

**логическое «или»** (дизъюнкция): **||**

**логическое «и»** (конъюнкция): **&&**.

Оператор	Название
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
!=	не равно

---



## Логические операторы. Пример

```
#include<iostream >
int main( )
{
    int a,b,c,d;
    a=34; b=12; c=12;
    d = c == b;    // сравнение на равенство
                    // переменных c и b (d=1)
    cout<<d<<'\n'; // вывод результата сравнения
    d = b > a;    // b больше a ? (d=0)
    cout << d <<'\n'; // вывод результата сравнения
return 0;
}
```

---

### 3. Типы данных и объявление переменных

#### Пример программы линейной структуры

##### **Задача** (лекция 1):

Даны три числа  $a$ ,  $b$ ,  $c$ . Разработать алгоритм программы для **нахождения среднего арифметического и среднего геометрического** этих чисел.

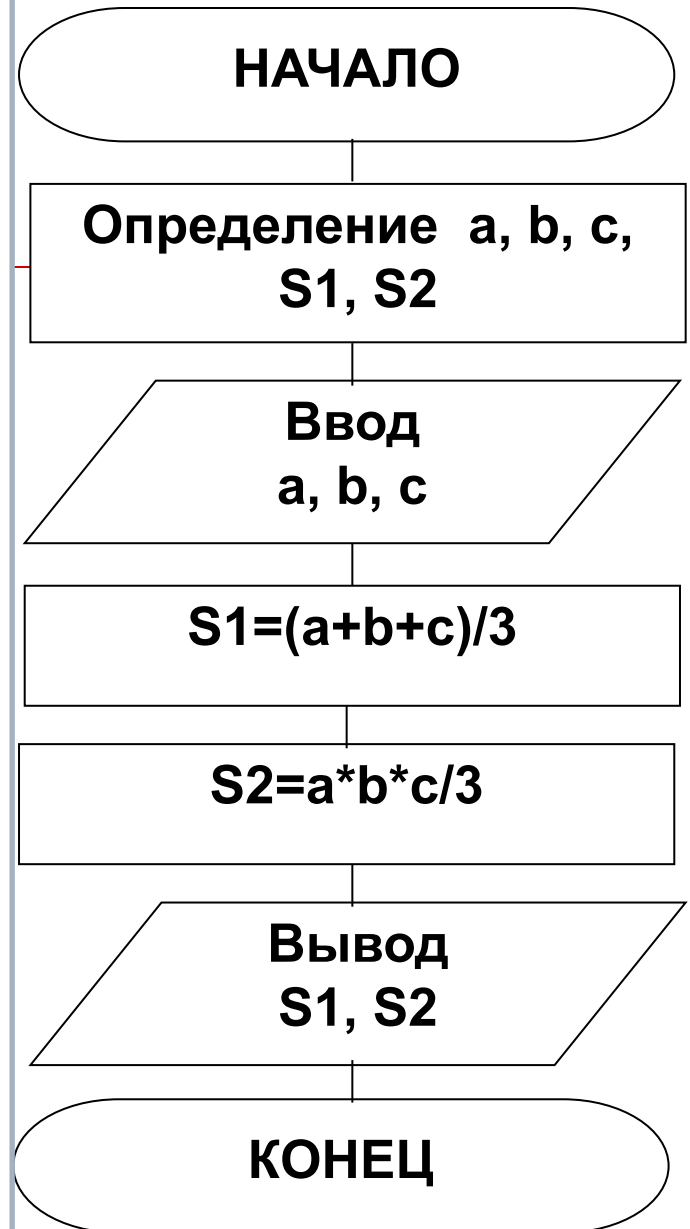
По данному алгоритму **разработать программу** так, чтобы при выводе на экран этих значений курсор вывода был переведён на новую строку экрана, и произошла выдача звукового сигнала.

---

```
#include<stdio.h>
#include<iostream>
```

```
int main(void)
{
    int a,b; float c,S1,S2;
    // printf("Введите три числа \n");*
    cin>>a>>b>>c);
    S1=(a+b+c)/3;
    S2=a*b*c/3;
    printf("Среднее
    арифметическое, S1=%f\n\la
    S2=%f\n\la ",S1,S2);
    return 0;
}
```

\*- необязательная строка программы



# Контрольные вопросы

- 1.** *Дайте определение программы и ее структуры*
- 2.** *Что понимается под алфавитом языка программирования и из каких лексических единиц он состоит?*
- 3.** *Что означает тип данных и какие типы данных используются в программах?*
- 4.** *Дайте определение переменным и правилам их определения в программах*
- 5.** *Что понимается под терминами «выражение», «операция» и «операнд» и как они связаны друг с другом ?*

**БЛАГОДАРИЮ ЗА ВНИМАНИЕ !**

---

**ВОПРОСЫ ?**

