Начальное программирование на C++. Графики. Технические приложения

(презентация – практикум)

Часть 1

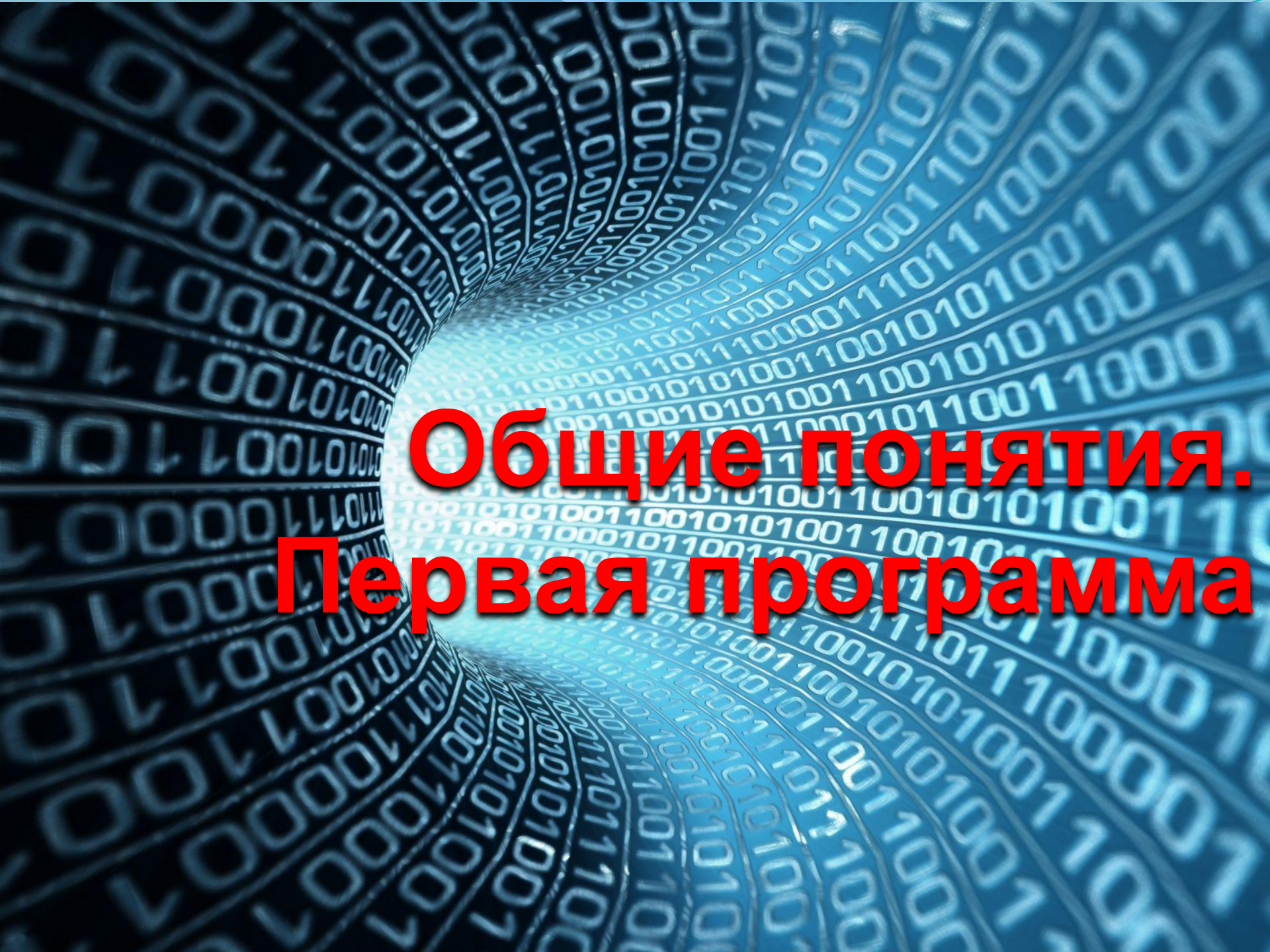
Иванов Виктор Никитович,
преподаватель высшей категории, к.т.н.

2016

СОДЕРЖАНИЕ

Слайды

1. Общие понятия. Первая программа 3
2. Запись математических выражений 27
3. Условный оператор if ... else 41
4. Циклы. Конструкция циклов 60
5. Отладка программы 83
6. Массивы 90
7. Оператор switch 101
8. Функция (подпрограмма) 107
9. Файловый ввод/вывод 125
10. Построение графиков (C++ & Excel) 130
11. Задания для самостоятельной работы 147
12. Технические приложения 162
13. Литература 167


Общие понятия.
Первая программа

Этапы создания программы

Программа – это перечень действий (инструкций), которые должна выполнить вычислительная машина.

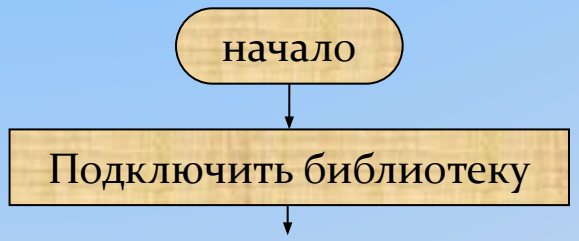
Алгоритм → Текст программы на каком-либо языке программирования → Машинный код

программист **транслятор**

Подключить библиотеку
Ввести переменную a
Вычислить площадь
Вывести результат

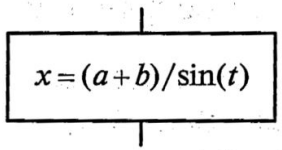
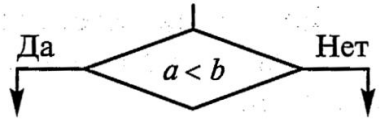
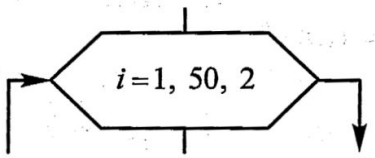
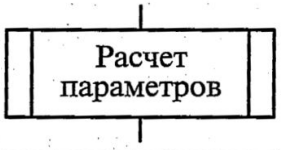
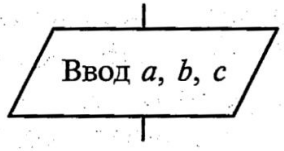
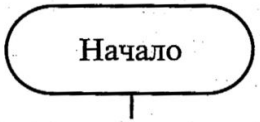
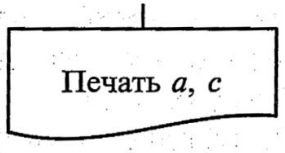
```
#include <iostream>
a=10;
s=a*a;
cout << s;
```

001011101010
010101100101
010111010110
101101111010



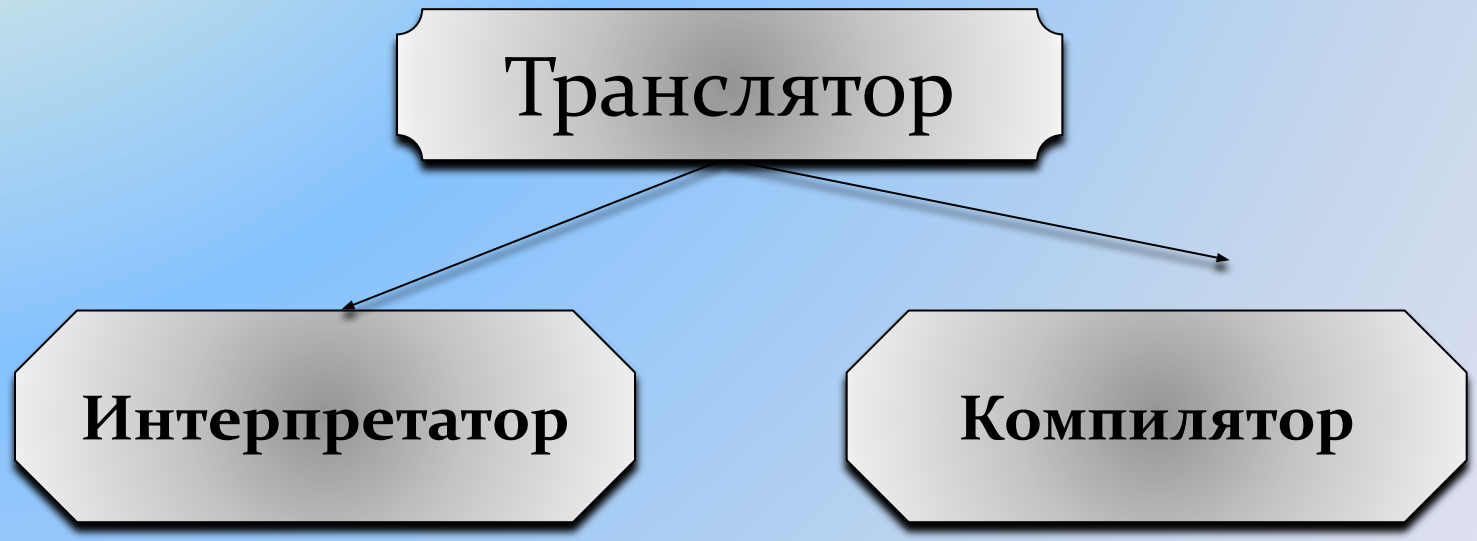
Последовательность инструкций, которые должна выполнить вычислительная машина, называется **алгоритмом.** Графически алгоритм можно изобразить в виде блок схемы, состоящей из условных обозначений.

Обозначения, используемые в блок-схемах

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисление по подпрограмме или вспомогательному алгоритму
Ввод-вывод		Ввод-вывод в общем виде
Пуск-остановка		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

Транслятор

Транслятор *(translator)* - это программа переводчик. Она преобразует программу, написанную на языке высокого уровня, в программу, состоящую из машинных команд.



последовательно анализирует и исполняет каждую строку программы

Формирует машинный код, готовый к исполнению вычислительной машиной

Текст (код) программы состоит из переменных и операторов.

Переменная - это буквенно-цифровое обозначение, составленное по определенным правилам, которое используется для написания кода программы, и которому отводится определенное количество оперативной памяти. В языке программирования C++ создание переменных называется также **объявлением переменных.**

Оператор — это инструкция, описание действия, которое необходимо выполнить над переменными. В состав операторов могут входить служебные слова, данные, выражения и другие операторы, например, арифметические операторы, условные операторы, операторы цикла, операторы ввода/вывода данных и др.

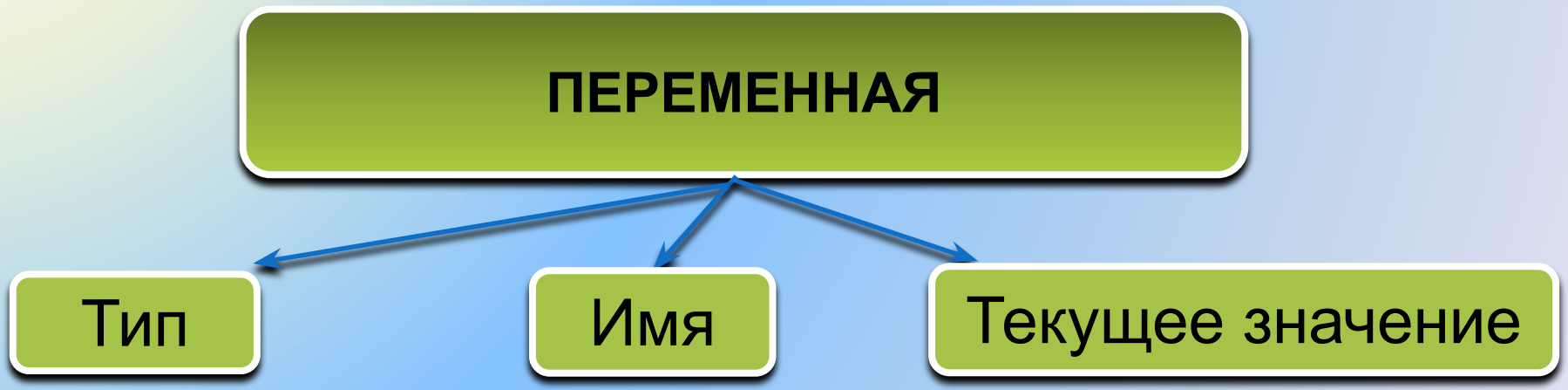
Правила записи **переменной**

ПЕРЕМЕННАЯ

Тип

Имя

Текущее значение



```
unsigned  int   abc          12134
int          A          -31745
float          b          0.56
double        a_21         -8.75
```

Правила записи **имени переменной**

МОЖНО использовать

- латинские буквы (A-Z, a-z)
- Цифры
- знак подчеркивания _

НЕЛЬЗЯ использовать

- ~~руские буквы~~
- ~~скобки~~
- ~~знаки арифметических действий, знаки препинания и др.~~
- ~~имя не может начинаться с цифры~~
- заглавные и строчные буквы различаются

Объявление переменных

Что означает объявить переменную? Это значит:

1. **Задать тип.**
2. **Задать имя.**
3. **Задать значение, которое можно присвоить переменной сразу или в процессе выполнения программы.**

Под заданную таким образом переменную будет выделена некоторая область оперативной памяти.

Глобальные и локальные переменные

Переменные могут быть глобальные, действующие во всей программе, и локальные, действующие в пределах определенного блока программы. Глобальные переменные объявляются в начале программы, а локальные внутри блока программы.
Глобальные переменные могут менять свое значение в процессе выполнения программы. Если переменная не должна менять свое значение, то она задается, например, следующим образом:

```
const int a=10;
```

Тип переменной и диапазон ее допустимых значений

Идентификатор	Размер, бит	Диапазон (множество) значений
unsigned char	8	0...255
char	8	-128..127
enum	16	-32768..32767
unsigned int	16	0..65535
short int	16	-32768..32767
int	16	-32768..32767
unsingned long	32	0..4294967295
long	32	-2147483648..2147483647
float	32	3.4E-38..3.4E+38
double	64	1.7E-308..1.7E+308
long double	80	3.4E-4932..1.1E+4932

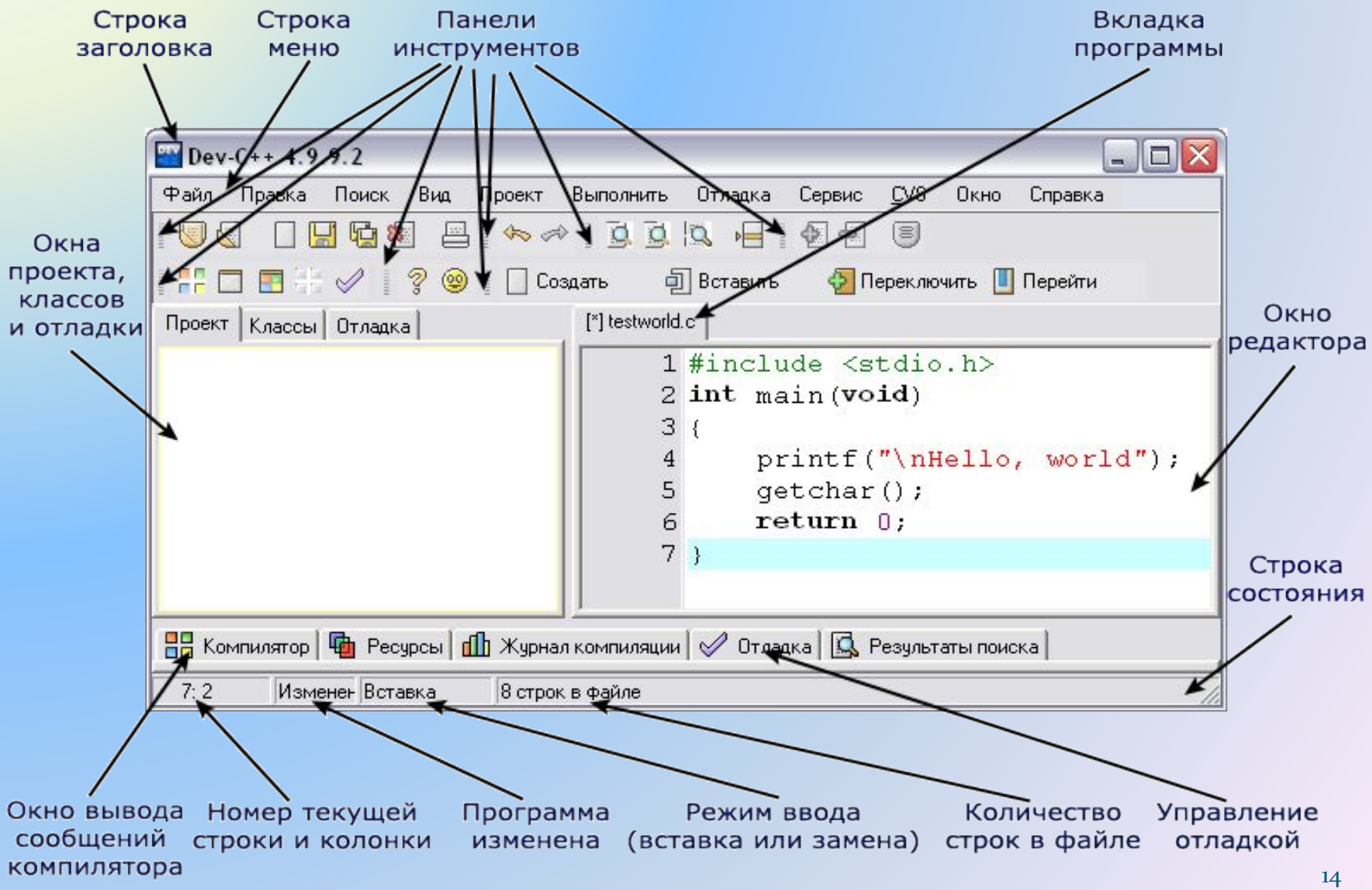
Программное обеспечение для изучения C++

Для первоначального изучения языка программирования **C++** удобно использовать небольшую по объему бесплатную интегрированную среду **Dev-C++** . В настоящее время официально проект Dev-C++ не поддерживается и сайт проекта - http://www.bloodshed.net/ закрыт, но на сайте **http://orwelldevcpp.blogspot.ru/** имеются обновления Dev-C++, разрабатываемые энтузиастами. Последняя версия, представленная на сайте - Version 5.11 - 27 April 2015. Интерфейсное окно другой популярной версии Dev-C++4.9.9.2 представлено на следующем слайде.

В продолжение проекта Dev-C++ сейчас разрабатывается также бесплатный русифицированный проект **wxDev-C++**, который включает все свойства Dev-C++ и дополнительно имеет возможность создания виджетов, официальный сайт этого проекта - **http://wxdsgn.sourceforge.net/**

Интерфейсное окно программы Dev-C++ версии 4.9.9.2

(заимствовано из **http://www.studfiles.ru/preview/3740997/**)



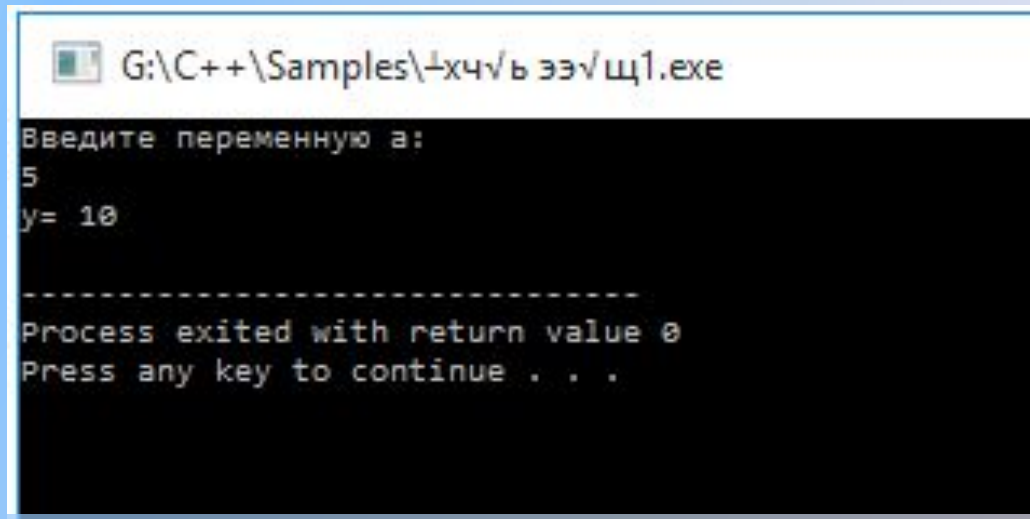
Порядок работы с программой Dev-C++

1. Создать новую папку на рабочем столе и дать ей имя.
2. Запустить программу Dev-C++
3. Настроить интерфейс. Для этого выбрать ***Сервис > Параметры редактора > Синтаксис > Цветовые схемы > Classic.***
4. Выбрать пункты меню: ***Файл > Создать > Исходный файл.***
5. Набрать текст программы.
6. Нажать кнопку ***Скомпилировать (F9 или Ctrl+F9).***
7. Задать имя файла и сохранить в созданную папку.
8. Нажать кнопку ***Выполнить (F10 или Ctrl+F10).***

Пример 1 (первая программа)

```
#include <iostream>
using namespace std;
double a, y;
int main()
{
setlocale (0, "");
cout << "Введите переменную a: " << endl;
cin >> a;
y=a+a;
cout << "y= " << y << endl;
return 0;
}
```

Результат работы программы



Директива **#include** используется для подключения библиотечных подпрограмм.
Библиотечная подпрограмма **<iostream>** отвечает за ввод/вывод данных.
Строка — **using namespace std** указывает на то, что используется подпространство имен с названием «std».
double a, y – объявление вещественных переменных.
int main() {...} – главная программа. В круглых скобках указывается либо возвращаемое значение, либо ничего не указывается. Далее в фигурных скобках идет тело главной программы. Все, что находится внутри фигурных скобок, будет автоматически выполняться после запуска программы. Количество открывающих и закрывающих скобок должно быть одинаковым.
Оператор **setlocale (0, "")** позволяет распознавать русский текст в программе. Эквивалентная форма записи этого оператора **setlocale (LC_ALL, "Russian").**

Оператор cin >> a задает ввод данных в программу с клавиатуры. После оператора cin (Console INput) ставятся двойные угловые кавычки, направленные от оператора cin (данные **уходят** от пользователя).
Оператор cout << "y= " << y << endl задает вывод данных из программы на экран монитора. После оператора cout (Console OUTput) ставятся две угловые кавычки, направленные к оператору cout (данные **приходят** к пользователю). Если надо вывести текст (но не значение переменной), то он заключается в верхние кавычки.
Оператор endl (end line) переводит курсор на следующую строку. Вместо endl можно применять "\n".
Команда return 0 (пишется со строчной буквы) сообщает о завершении главной программы
В конце каждого оператора (строки) ставится точка с запятой.

Пространство имен

Пространство имён — это группа идентификаторов, внутри которой все идентификаторы уникальны (не повторяются). Если отсутствует необходимость в использовании разных пространств имён в рамках одной программы, то можно однажды задать пространство и далее обращаться ко всем именам без его указания.

using namespace std;

std — пространство имён, определённое для всей стандартной библиотеки C++, а «::» — это оператор разрешения области видимости, который указывает, из какого пространства имён должен браться следующий за ним идентификатор.

Ввод данных с клавиатуры

Ввод переменной a

```
cin >> a;
```

Ввод переменных a, b

```
cin >> a >> b;
```

Фрагмент программы

```
cout <<"Введите первое число: " << endl;
cin >> a;
cout << "Введите второе число: " << endl;
cin >> b;
```

Варианты вывода данных на монитор

Вывод значения переменной «a»

```
cout << a;
```

Вывод значения переменной «a» и переход на новую строку

```
cout << a << endl;
```

Вывод текста

```
cout << "Привет!";
```

Вывод текста и значения переменной «c»

```
cout << "Ответ  " << c;
```

Важно! Число пробелов между словом Ответ и последними кавычками будет равно числу пробелов между словом Ответ и значением переменной c на экране.

Форматы вывода вещественных чисел

```
float x = 1238.4767;
```

```
cout.precision(7);
cout << x << endl;
```

Значащих цифр – 7:
x = 1238.477

```
cout.precision(5);
cout << x << endl;
```

Значащих цифр – 5:
x = 1238.5

```
cout.precision(2);
cout << x << endl;
```

Значащих цифр – 2:
x= 1.2e+003
(или $1{,}2 \cdot 10^3$, или 1200)

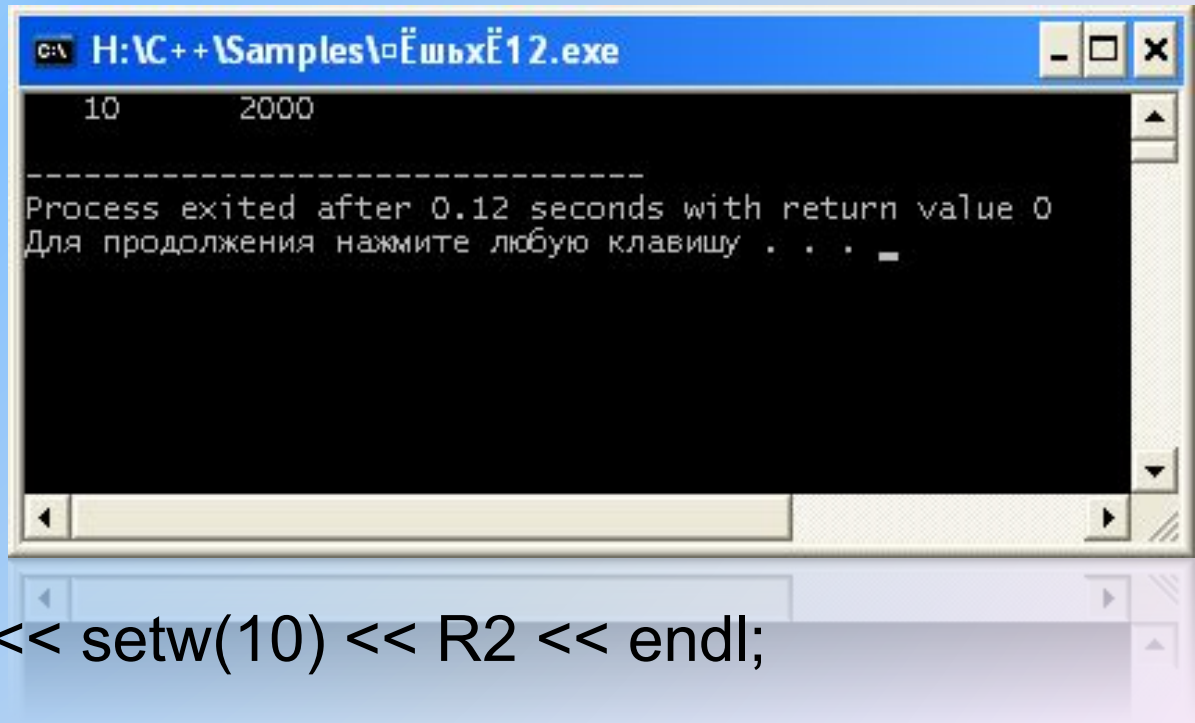
При записи результатов в виде x= 1.2e+003, число 1.2 называется мантиссой, 003 – показатель степени при основании 10. В данном примере при переходе к записи числа в обычной форме точка должна быть передвинута на три знака вправо, т. е. x = 1200. Если показатель степени отрицательный (например, -003), точка передвигается на три знака влево.

В некоторых вариантах компилятора для того, чтобы задержать результаты на экране, используется оператор **_getch()** (начинается с символа подчеркивания). При этом в начале программы должна быть подключена библиотека <conio.h>. В некоторых вариантах компилятора в этом операторе нет необходимости. Достаточно поставить галочку около опции: ***Сервис > Параметры среды > Общие > Pause console programs after return***

Вывод с заданной шириной поля

Если необходимо зарезервировать под выводимое число определенное количество позиций, то можно применить библиотеку **#include <iomanip>** и оператор **setw(n)**, где n – кол-во зарезервированных под число позиций, например,

```
#include <iostream>
#include <iomanip>
using namespace std;
int R1=10;
int R2=2000;
int main()
{
cout << setw(5) << R1 << setw(10) << R2 << endl;
return 0;
}
```

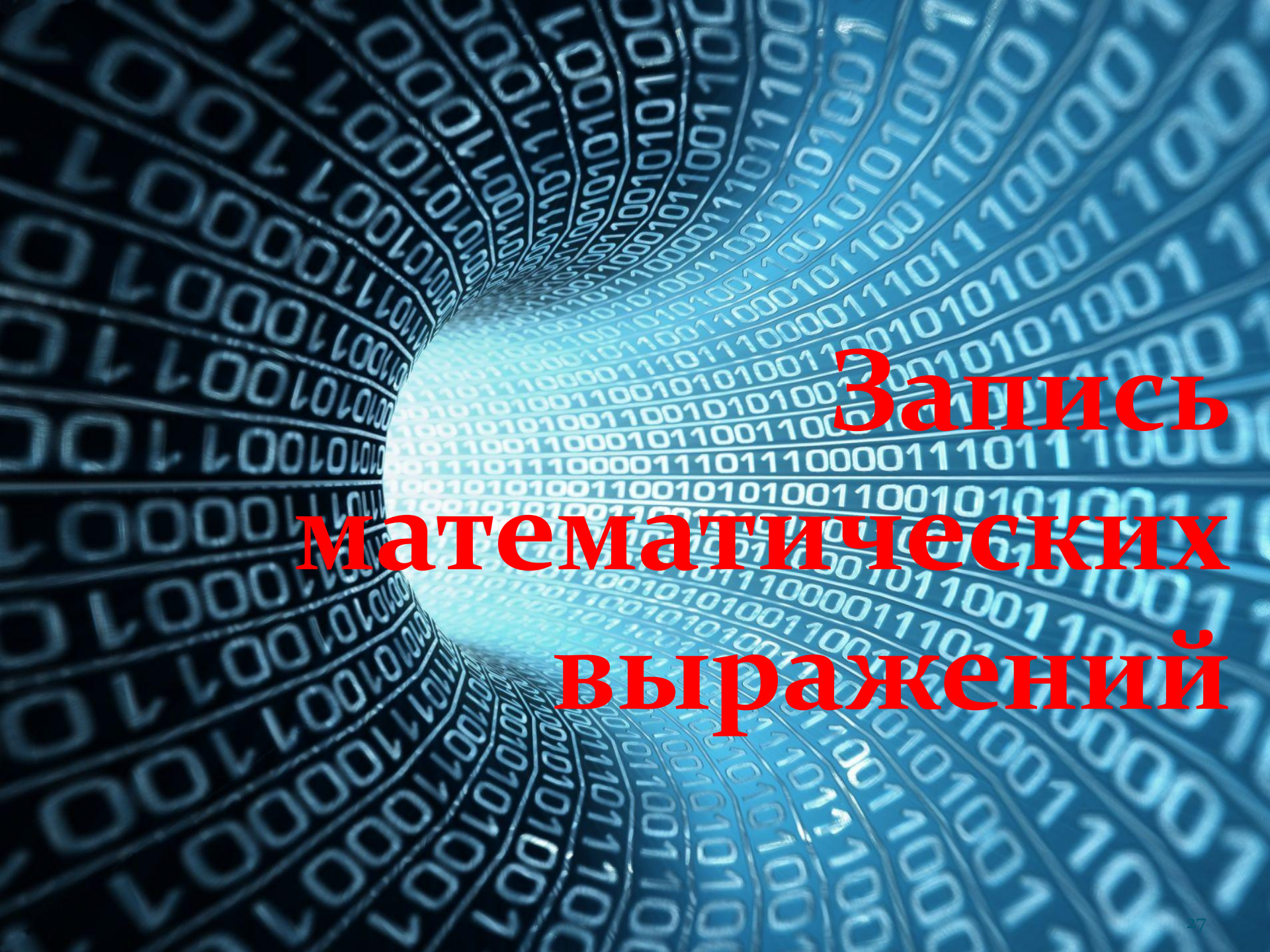


СПЕЦИАЛЬНЫЕ СИМВОЛЫ ДЛЯ ИСПОЛЬЗОВАНИЯ С ОПЕРАТОРОМ COUT.	
СИМВОЛ	**НАЗНАЧЕНИЕ**
\a	Сигнал бипера компьютера
\n	Переход на новую строку
\r	Возврат каретки в начало строки
\t	горизонтальная табуляция
\v	Вертикальная табуляция
\\	Обратный слеш
\?	Знак вопроса
\'	Одинарные кавычки
\"	Двойные кавычки
\0	Нулевой символ
\000	Восьмеричное значение, например \007
\xhhhh	Шестнадцатеричное значение, например \xffff

Пример 2.

Пример программы, выводящей на экран квадрат числа, введённого пользователем с клавиатуры и звукового сигнала:

```
#include <iostream>
using namespace std;
double s;
int main()
   {setlocale (0,  "");
   cout << "Введите число: ";
   cin >> s;
   cout << "Квадрат числа: ";
   cout << s*s << endl;
   cout << "\a"; // Это вывод звукового сигнала
   return 0;
   }
```


Запись
математических
выражений

Библиотека <cmath>

Чтобы воспользоваться сложными математическими функциями, **нужно подключить** библиотеку, в которой содержатся эти функции, а именно:

#include <cmath> (или <math.h>)

Запись математических функций в С++

Математическая запись	Запись на С++	Назначение
cos *x*	cos(x)	Косинус *x* радиан
sin *x*	sin(x)	Синус *x* радиан
tg *x*	tan(x)	Тангенс *x* радиан
Arccos *x*	acos(x)	Арккосинус числа *x*
Arcsin *x*	asin(x)	Арксинус числа *x*
arctg *x*	atan(x)	Арктангенс числа *x*
e^x	exp(x)	Значение e в степени *x*
x^n	pow(x,n)	Число x в степени n
$\|x\|$	fabs(x)	Модуль числа *x*
$\sqrt{}$	sqrt(x)	Квадратный корень из *x*
ln *x*	log(x)	Натуральный логарифм *x*
$\log_{10} x$	log10(x)	Десятичный логарифм *x*

Примеры записи математических выражений

$x^2 - 7x + 6$ → pow(x,2)-7*x+6

$\frac{|x| - |y|}{1 + |xy|}$ → (fabs(x)-fabs(y))/(1+fabs(x*y))

$ln\left(\left(y - \sqrt{|x|}\right)\left(x - \frac{y}{z + x^2/4}\right)\right)$ →

log((y-sqrt(fabs(x)))*(x-y/(z+pow(x,2)/4)))

Допустимая сокращенная запись простых математических операций

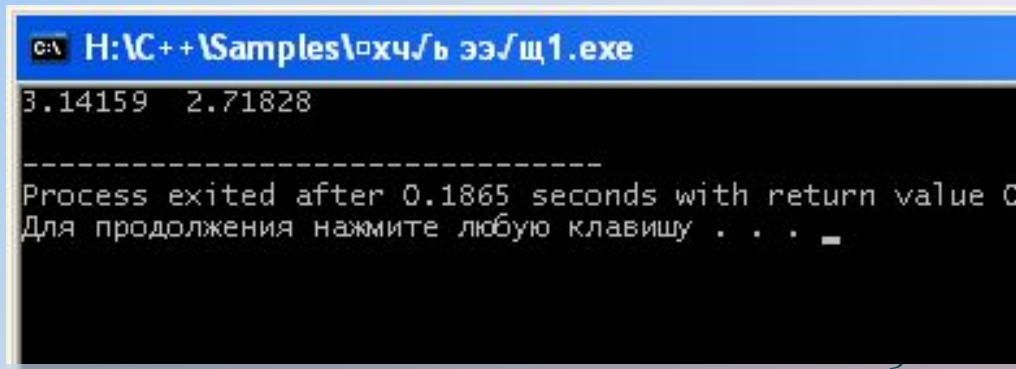
```
int a, b;
a ++;        // a = a + 1;
a --;        // a = a – 1;
a += b;      // a = a + b;
a -= b;      // a = a - b;
a *= b;      // a = a * b;
a /= b;      // a = a / b;
a %= b;      // a = a % b;
```

Важно. В C++ деление целого числа на целое дает в результате целое число, т.е. 1/2 = 0. Чтобы при делении получить 0.5 необходимо, чтобы хотя одно из чисел было вещественным числом, т.е. необходимо записать либо 1.0/4, либо 1/4.0

Операция **%** означает **остаток** от целочисленного деления. Например, 10 **%** 3 = 1

В C++ доступны две константы: число «пи» и число «е» (основание натурального логарифма). Их можно получить с помощью констант M_PI и M_E, например,

```
double a, b;
a = M_PI;
b = M_E;
cout<<a<< " " <<b<<endl;
```



Пример 3 (расчет по формуле)

```
#include <iostream>
#include <cmath>
using namespace std;
int x=4578;
int y;
int main()
{
y=pow(x,2)-7*x+6;
cout << "y= "  << y << endl;
return 0;
}
```

Пример 4 (расчет по формуле)

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int y(10);
    int x(-4);
    int z(3);
    double c=fabs(x)+sqrt(y)+pow(z,2);
    cout << "c= " << c << endl;
    return 0;
}
```

Важно: присвоение значения переменной, например, **int y=10;** можно записать, как **int y(10);**

Написать программы на C++ для вычисления значения y математических выражений 1…8 при следующих исходных данных: a = 10, b = 20

1) $y = a + b$

2) $y = (a + b)^2$

3) $y = \sqrt{a + b}$

4) $y = \frac{(a + b)(a - b)}{a + b}$

5) $y = a^2 + b^2$

6) $y = \sqrt{a} + \sqrt{b}$

7) $y = \frac{a + b}{\sqrt{a + b}}$

8) $y = \frac{a + b}{a^2 + b^2}$

Написать программы на С++ для вычисления значения функций 9...18 при следующих исходных данных: x = π/4

9) $y = sinx + tgx$

10) $y = \sin(x^2)$

11) $y = \sqrt{sinx}$

12) $y = sinx + tgx^5$

13) $y = (tgx)^3 + \frac{sinx}{cosx}$

14) $y = sin^2 x$

15) $y = (sinx)^2$

16) $y = \frac{tgx + 1/tgx}{sinx + cosx}$

17) $y = (sinx + cosx)^2$

18) $y = \sqrt[4]{sinx}$

Ответы:

1) 30	7) 5,477	13) 2
2) 900	8) 0,06	14) 0,5
3) 5,477	9) 1,707	15) 0,5
4) -10	10) 0,578	16) 1,414
5) 500	11) 0,841	17) 2
6) 7,634	12) 1,015	18) 0,917

Пример 5. Расчет по формуле с вводом исходных данных с помощью клавиатуры.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    setlocale (0, "");
    double a, b, y;
    cout <<"Введите первое число:";
    cin >> a;
    cout << "Введите второе число:";
    cin >> b;
    y=pow(a,2)+pow(b,3)+pow((pow(a,2)+pow(b,3)),2);
    cout << "y=" << y << endl;
    return 0;
}
```

Пример 6. Вычисление площади круга

```
#include <iostream>
#include <math.h>
using namespace std;
double S, R;
double pi=3.1416;
int main()
{
    setlocale(0, "");
    cout << "Введите радиус окружности в метрах: ";
    cin >> R;
    S=pi*pow(R,2);
    cout << "S=" << S << " кв.м" << endl;
return 0;
}
```

Пример 7.

Напишите программу для вычисления длины окружности:

$C = 2\pi R$

Пример 8.

Напишите программу для вычисления объема прямого цилиндра:

$V = Sh$,

где S – площадь основания,

h – высота.

Радиус основания цилиндра и высота должны вводиться с клавиатуры.

Условный
оператор
if...else

Оператор if…else (если…иначе).

При выполнении условия **if** выполняется последующий код программы, расположенный после условия. Если условие не выполняется, происходит переход к дальнейшему выполнению программы или переход к коду программы, расположенному после оператора **else.**

Программный код после операторов if и else заключается в фигурные скобки.

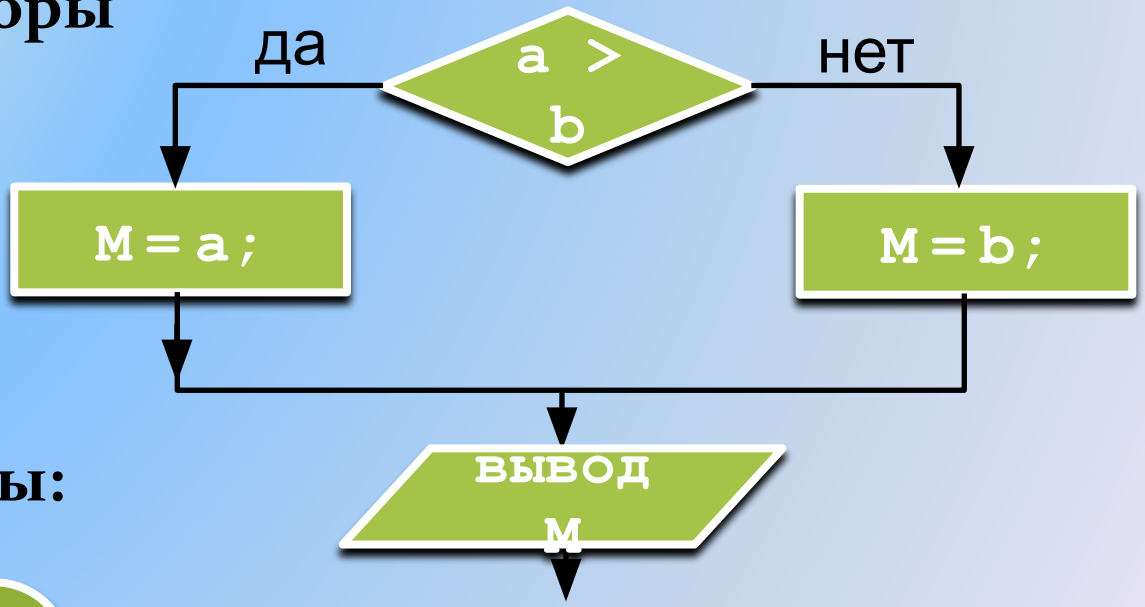
Условие if всегда записывается в круглых скобках, точка с запятой после условия не ставится. Если при выполнении условия требуется выполнить только одну команду, то фигурные скобки ставить не обязательно.

Полная форма условного оператора

Полная форма ветвления, используются операторы

if … else

Блок – схема:



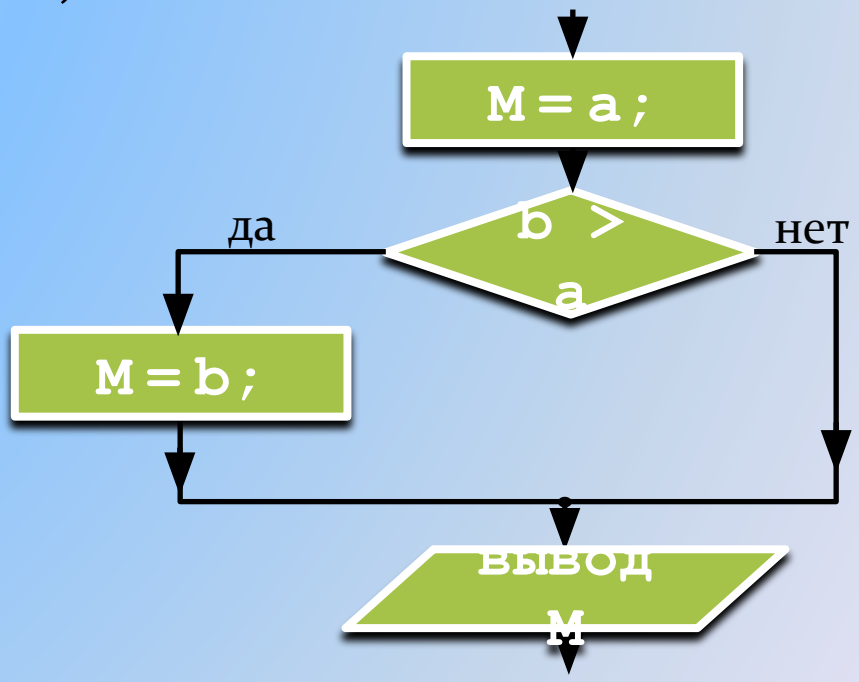
Фрагмент программы:

```
if (a > b)
   M = a;
else
   M = b;
```

Неполная форма условного оператора

Неполная форма ветвления, используется только if

Блок-схема:

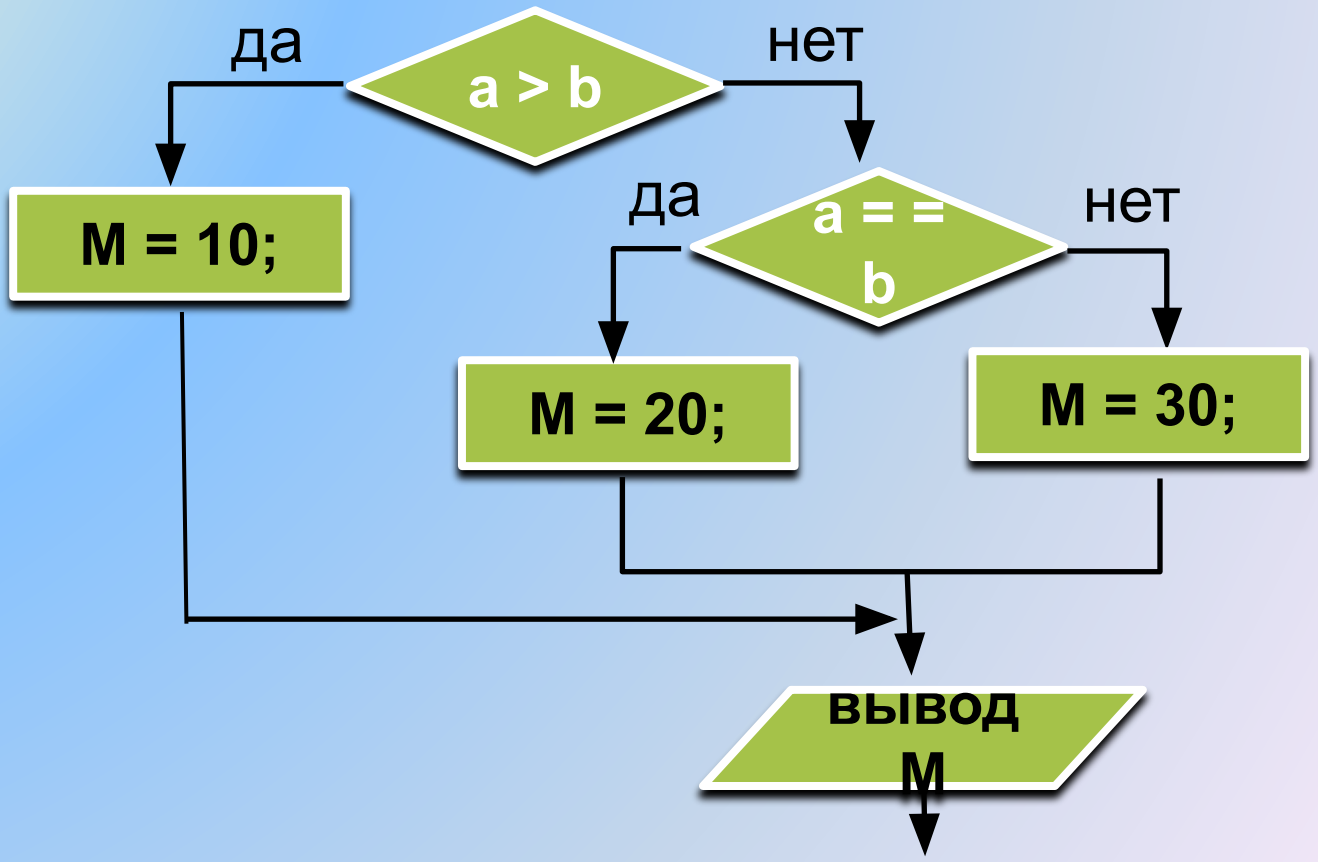


Фрагмент программы:

```
M = a;
if (b > a)
M = b;
```

Вложенный условный оператор

```
if (a>b)
M=10;
else {
if (a==b)

M=20;
else
M=30;
}
```



Знаки отношений в условном операторе if

> <	больше, меньше
>=	больше или равно
<=	меньше или равно
==	равно
! =	не равно
!	НЕ
&&	И
||	ИЛИ

Двойной знак равенства является не оператором присваивания, а оператором сравнения. Например, if (a==5) означает, что мы не присваиваем переменной «а» значение 5, а сравниваем текущее значение «а» со значением 5. И в случае равенства условие выполняется.

Пример 9

Написать программу с условным оператором if.

Кредит, который может выдать банк физическому лицу, зависит от суммы зарплаты.
Если сумма зарплаты Sz<=20000 руб, то кредит недоступен.
Если сумма зарплаты 20000 < Sz <= 40000 руб, то банк может выдать кредит Kr=0.7*Sz.
Если сумма зарплаты Sz>40000 руб, то банк может выдать кредит Kr=1.2*Sz.

Сумму зарплаты вводить с клавиатуры.
Вывести на монитор доступную сумму кредита.

Пример 9

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
int Sz,Kr; // Sz – сумма зарплаты, Kr – сумма кредита
cout << "Введите сумму зарплаты: ";
cin >> Sz;
if (Sz<=20000)
cout << "Кредит недоступен" << endl;
if (Sz>20000 && Sz<=40000)
{
Kr=0.7*Sz;
cout << "Доступен кредит: "<< Kr << "руб" << endl;
}
if (Sz>40000)
{
Kr=1.2*Sz;
cout << "Доступен кредит: "<< Kr << "руб" << endl;
}
return 0;
}
```

Пример 10. Фрагмент программы с условным оператором

```
int a=6;
if (a==5) {
c=5;
y=a+c;
}
else {
c=4;
y=a-c;
}
```

Дополнить приведенный фрагмент обязательными инструкциями и написать полностью программу. В конце программы вывести на монитор переменную y.

Пример 11.

Написать программу, в которой с клавиатуры вводятся два числа: a, b. Если выполняется условие a > b, то вычислить

$$M = a + b$$

Если условие не выполняется, то вычислить

$$M = a - b$$

Вывести на монитор значение M.

Пример 12.

Взять за основу пример 11, но ввести в него не два, а три условия: если a > b то переменной M присвоить значение

M = 10;

если a = b то присвоить

M = 20;

если a < b то присвоить

M = 30;

Вывести на монитор значение M.

Пример 13. Написать программу для нахождения случайного числа (в диапазоне 0…99), которое присваивается переменной num.

```
#include <iostream>
#include <windows.h>
#include <cmath>
using namespace std;
int main() {
setlocale(LC_ALL, "Russian");
int num = rand() % 100; /*остаток от деления
случайного числа на 100 */
cout << "Случайное число num=" << num << endl;
return 0;
}
```

Пример 14

Дополнить программу примера 13 следующим условным оператором:

```
int a, b;
if (num >=50)
{
a = 1;
cout << "a= " << a << endl;
}
else
{
b = 2;
cout << "b= " << b << endl;
}
```

Пример 15. Дополнить программу примера 13 следующим условным оператором:

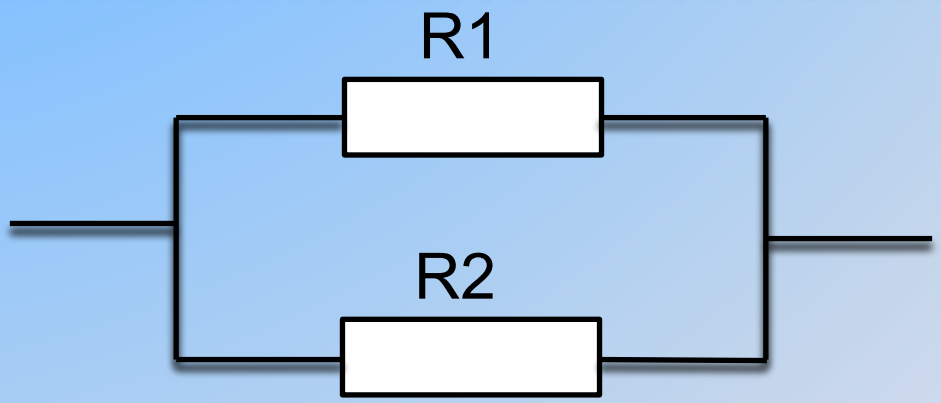
```
double a, b;
double x=3.1416/4;
if (num >=50)
 {
 a = (1 − cos^2 x) / cos x ;

cout << "a = " << a << endl;
 }
else
 {
  b = (1+tgx+cosx+sinx)^2+5sinx;
cout << "b = " << b << endl;
 }
```

Пример 16. Написать программу вычисления сопротивления последовательной или параллельной цепи в зависимости от варианта. Значения сопротивлений должны вводиться с клавиатуры. Варианты 1 или 2 также должны вводиться с клавиатуры.

Вар. 1

$$R11 = R1 + R2$$

Вар. 2



$$R22 = \frac{R1 \cdot R2}{R1 + R2}$$

Пример 16 (начало)

```
#include <iostream>
#include <math.h>
using namespace std;
double R1, R2, R11, R22;
int num;
int main()
{
    setlocale(0, "");
    cout << "Введите R1: ";
    cin >> R1;
```

Пример 16 (окончание)

```
cout << "Введите R2: ";
 cin >> R2;
cout << "Введите 1, если последовательная
цепь, введите 2, если параллельная цепь: ";
cin >> num;
 if (num==1)
{
    R11=R1+R2;
    cout << "R11=" << R11 <<  endl;
} else {
    R22=R1*R2/(R1+R2);
    cout << "R22=" << R22 << endl;
}
   return 0;
}
```

Пример 17. Написать программу вычисления сопротивления последовательной или параллельной цепи аналогично предыдущему примеру, но последовательная и параллельная электрическая цепи должны состоять уже из трех резисторов. Значения сопротивлений R1, R2, R3 должны вводиться с клавиатуры. Варианты 1 или 2 также должны вводиться с клавиатуры.

В программе использовать следующие формулы

Вар 1 (последовательная цепь):

$$R11 = R1 + R2 + R3$$

Вар.2 (параллельная цепь):

$$R33 = \frac{R22 \cdot R3}{R22 + R3}$$

где :

$$R22 = \frac{R1 \cdot R2}{R1 + R2}$$

Пример 18. После абсолютно **упругого** соударении двух тел массой m1 и m2 они будут двигаться со скоростями

$$V_{11} = \frac{(m_1 - m_2)V_1 + 2m_2V_2}{m_1 + m_2}$$

$$V_{22} = \frac{(m_2 - m_1)V_2 + 2m_1V_1}{m_1 + m_2}$$

После абсолютно **неупругого** соударении двух тел массой m1 и m2 они будут двигаться с общей скоростью

$$V = \frac{m_1V_1 + m_2V_2}{m_1 + m_2}$$

Написать программу расчета скоростей при абсолютно упругом и абсолютно неупругом соударении тел. Массы и начальные скорости двух тел вводить с клавиатуры.

Циклы.
Конструкция
циклов

Циклические алгоритмы в С++

Цикл – это многократное выполнение одинаковых действий.

Различают два вида циклов:

- цикл с **известным** числом шагов (например, выполнить действие 10 раз) – **цикл for**
- цикл с **неизвестным** числом шагов (делать, пока не выполнится условие останова цикла) – **цикл while**

Пример алгоритма, требующего многократных повторений.

Пусть надо найти значения функции y=2*x-5 при нескольких значениях аргумента x=1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Тогда фрагмент кода программы будет иметь вид:

```
x=1;
y=2*x-5;
cout << y << endl;
x=2;
y=2*x-5;
cout << y << endl;
```

и т.д. всего 30 строк.

Такая запись крайне нерациональна!

В этих случаях используют оператор цикла.

Цикл for

```
for (int i = 1; i <= 10; i++)   - заголовок цикла
{
x=i;
y=2*x-5;                        тело цикла
cout << y << endl;
}
```

Заголовок цикла, записывается в круглых скобках, точка с запятой после заголовка не ставится!

Внутри заголовка:

1. На первом месте располагается начальное значение изменяемого параметра цикла. В нашем примере изменяемым параметром является переменная i, она же служит счетчиком цикла. Начальное значение равно 1. Каждое новое значение переменной соответствует одному проходу цикла. Один проход цикла называется **итерацией.** После задания начального значения ставится точка с запятой.

2. На втором месте в заголовке указывается конечное значение переменной. В нашем примере конечное значение переменной i равно 10. Снова ставим точку с запятой.

3. И, наконец, на третьем месте, указывается шаг изменения переменной или шаг цикла. **Шаг цикла** — это значение, на которое будет увеличиваться или уменьшаться переменная при каждом проходе (i++ означает увеличение на единицу).

За заголовком цикла идет **тело цикла,** которое записывается **в фигурных скобках.**

В качестве изменяемого параметра цикла может выступать любая физическая величина.

Пример цикла **for** с параметром **«время».**

```
#include <iostream>
#include <math.h>
using namespace std;
int t, dt=10;
int main() {
setlocale (0, "");
for (t=0; t<=60; t=t+dt)
cout << "Время в сек: " << t << endl;
return 0;
}
```

```
Время в сек: 0
Время в сек: 10
Время в сек: 20
Время в сек: 30
Время в сек: 40
Время в сек: 50
Время в сек: 60

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Пример цикла **for** с параметром **«расстояние»**

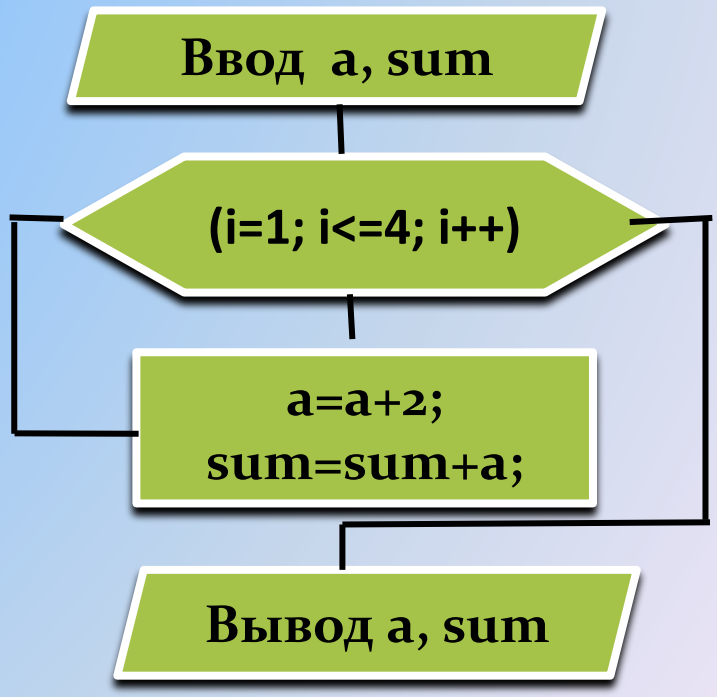
```
#include <iostream>
#include <math.h>
using namespace std;
int L, dL=5;
int main() {
setlocale (0, "");
for (L=0; L<=50; L=L+dL)
cout << "Расстояние в км: " << L << endl;
return 0;
}
```

```
Расстояние в км: 0
Расстояние в км: 5
Расстояние в км: 10
Расстояние в км: 15
Расстояние в км: 20
Расстояние в км: 25
Расстояние в км: 30
Расстояние в км: 35
Расстояние в км: 40
Расстояние в км: 45
Расстояние в км: 50

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Пример 19. **Определить значения переменных a, sum после выполнения цикла.**

```
int a = 5;
int sum=0;
for( i = 1; i <= 4; i++ )
{
a = a + 2;
sum=sum+a;
}
cout << a << endl;
cout << sum << endl;
```

i=1, a=7, sum=7
i=2, a=9, sum=16
i=3, a=11, sum=27
i=4, a=13, sum=40



Дописать программу и убедиться, что в результате выполнения цикла a=13, sum=40

Циклы while и do…while

Когда мы **не знаем**, сколько итераций должен произвести цикл, нам понадобится цикл **while** или **do...while**. Синтаксис циклов **while** и **do…while** в C++ выглядит следующим образом:

while:

```
while (условие)
{
  Тело цикла;
}
```

do…while:

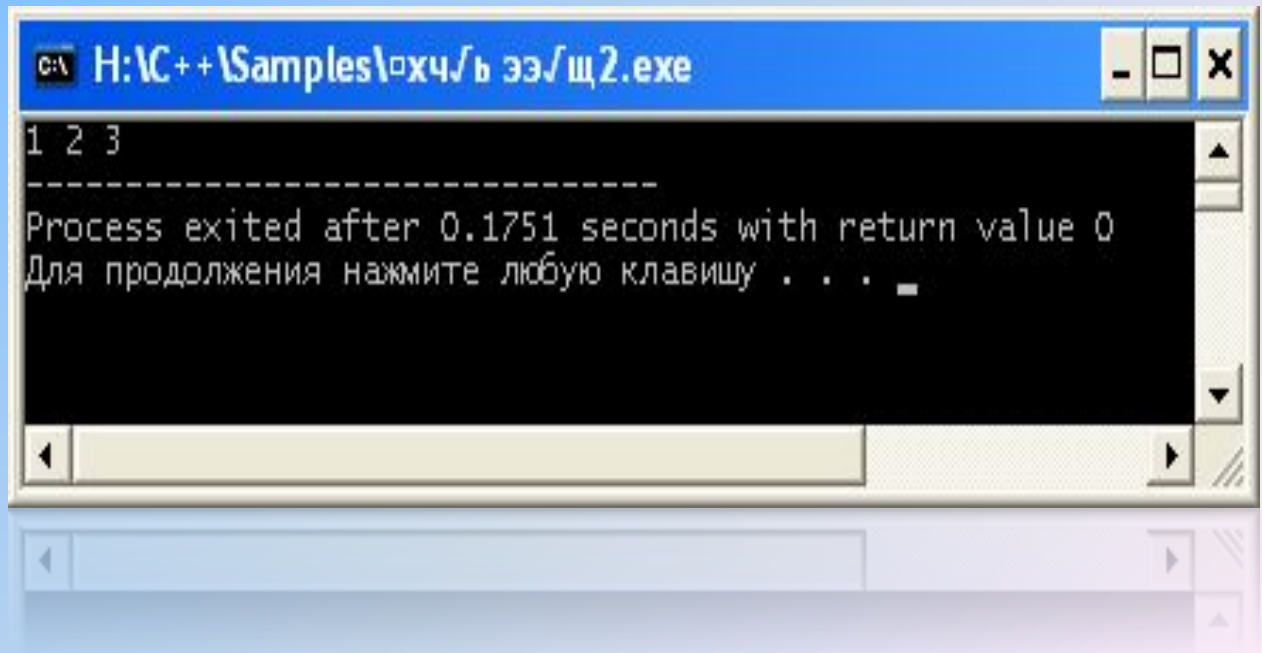
```
do
 {
Тело цикла;
 }
 while (условие);
```

Циклы будут выполняться до тех пор, пока выполняется условие, указанное в круглых скобках.

Цикл while

В цикле while условие, определяющее будет ли цикл повторяться, проверяется перед первым шагом цикла. Такой цикл называется циклом с *предпроверкой* условия.

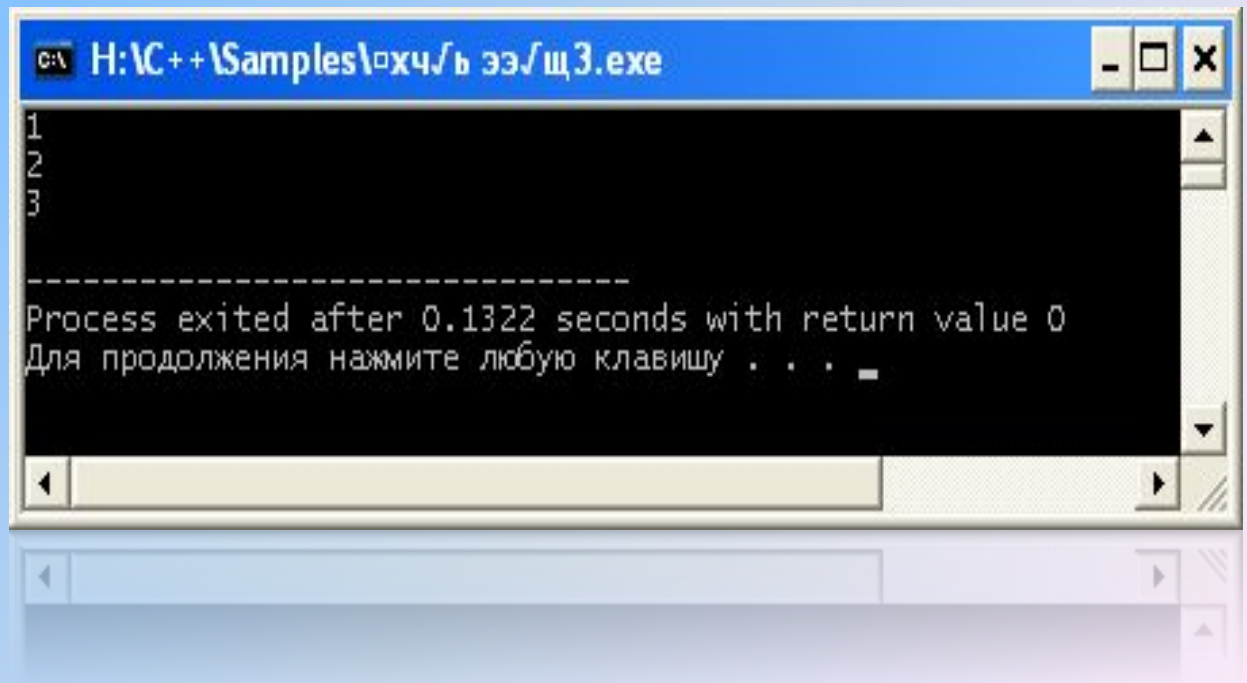
```
#include <iostream>
using namespace std;
int main()
{
int i=0;
while (i<3)
{
i++;
cout << i << " ";
}
return 0;
}
```



Цикл do…while

Если условие проверки располагается в конце цикла, то это будет цикл с *постпроверкой* условия. Для его записи используется конструкция из операторов do…while.

```
#include <iostream>
using namespace std;
int main()
{
int i=0;
do
{
i++;
cout << i << endl;
}
while (i<3);
return 0;
}
```



Важно.

В цикле **while** точка с запятой после условия цикла не ставится.

В цикле **do…while** после условия цикла надо поставить точку с запятой.

Обратить внимание. Оператор вывода в циклах while и do…while можно задавать различным образом. При этом выводимые данные будут располагаться или в строчку, или в столбик.

В чем отличие цикла while от do…while?

while

```
int i = 1;
while (i < 0) {
cout << i << " ";
i++;
}
```

do…while

```
int i = 1;
do {
cout << i << " ";
i++;
}
while (i < 0);
```

Цикл **while** не выполнится ни разу, а цикл **do…while** выполнится 1 раз, и на экран будет выведено «1».

Пример 20. Напишем с помощью цикла **for** программу, которая будет считать сумму чисел от 1 до 10.

```
#include <iostream>
using namespace std;
int main()
{
int i;
int sum = 0;
   setlocale(0, "");
   for (i = 1; i <= 10; i++)
{
      sum = sum + i;
 }
 cout << "Сумма чисел от 1 до 10 = " << sum << endl;
 return 0;
}
```

Пример 21.

Дописать программу, которая будет считать сумму чисел от 1 до 10 с помощью цикла **while**.

```
int i=0;
int sum = 0;
while (i<10)
{
i++;
sum = sum + i;
}
```

Пример 22.

Написать программу, которая будет считать сумму чисел от 1 до 10 с помощью цикла **do…while**.

Пример 23. Написать программу для расчета значений выражения при a = 5 и b = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0

$$a^2 + b^3 + a \cdot b + \frac{a}{\sqrt{b}}$$

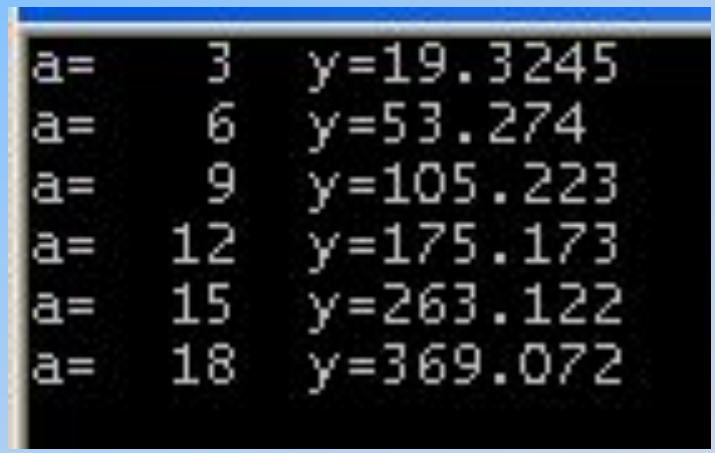
```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int a=5;
double b, y;
int main()
{
setlocale(0, "");
   int i;
   for (i = 1; i <=6; i++)
{
b=0.5*i;
   y=pow(a,2)+pow(b,3)+a*b+a/sqrt(b);
   cout << "b=" <<setw(4) << b << "  y=" << y << endl;
}
   return 0;
}
```

```
b= 0.5   y=34.6961
b=   1   y=36
b= 1.5   y=39.9575
b=   2   y=46.5355
b= 2.5   y=56.2873
b=   3   y=69.8868
```

Пример 24. Написать программу для расчета значений выражения

$$a^2 + b^3 + a \cdot b + \frac{a}{\sqrt{b}}$$

при значении b = 1.5 и при значениях
a = 3, 6, 9, 12, 15, 18



Досрочное завершение цикла (оператор break)

Как цикл **while** так и цикл **for** можно завершить досрочно, если внутри тела цикла использовать оператор **break**.

При этом произойдёт моментальный выход из цикла, не будет закончен даже текущий шаг (т. е. если после break присутствовали какие-то ещё операторы, то они не выполнятся).

Досрочное завершение цикла (оператор break)

В результате работы следующего примера на экран будут выведены только числа «1 2 3 4», хотя конечное значение счетчика цикла равно 10.

```
for (int i=1; i<=10; i++)
{
  if(i == 5) {
  break;
  }
  cout << i << " ";
}
```

Пример 25 (оператор цикла for + оператор break).

Мяч вертикально падает с высоты 2м. Высота каждого подскока составляет 0,6 от предыдущей высоты. Сколько подскоков должен сделать мяч, чтобы очередная высота подскока не превышала 0,3 м.

В программе используется оператор цикла и оператор break.

```
i=4   h=0.2592

--------------------------------
Process exited after 0.1892 seconds with return
Для продолжения нажмите любую клавишу . . .
```

Пример 25

```
#include <iostream>
#include <math.h>
using namespace std;
int ik;
double h=2.0;
int main()
{
setlocale(0, "");
for (int i = 1; i <=10; i++)
{
h=h*0.6;
if (h<=0.3)
{
ik=i;
break;
}
}
cout << "Номер подскока: " << ik << endl;
cout << "Высота подъема: " << h << endl;
return 0;
}
```

Пример 26

Мяч вертикально падает на пол со скоростью V1=6 м/с. В момент каждого отскока мяч теряет в скорости 30%, т.е. скорость отскока равна V2=0.7*V1. После отскока мяч подпрыгивает на высоту $h = V_2^2/2g$

При втором и каждом последующем падении мяча скорость падения равна скорости предыдущего отскока.

Написать программу и вывести на монитор высоту подъема после каждого отскока. Число отскоков задать равным 6.

Ускорение свободного падения $g = 9{,}81\ \text{м/с}^2$

```
Номер отскока: 1  Высота подъема: 1.83
Номер отскока: 2  Высота подъема: 0.44055
Номер отскока: 3  Высота подъема: 0.21587
Номер отскока: 4  Высота подъема: 0.105776
Номер отскока: 5  Высота подъема: 0.0518303
Номер отскока: 6  Высота подъема: 0.0253969

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Пример 26

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
setlocale(0, "");
int i=0;
double V1=6, V2, h, g=9.81;
for (i=1; i<=6; i++)
{
V2=0.7*V1;
h=pow(V2,2)/(2*g);
V1=V2;
cout<<"Номер отскока: "<< i <<"  Высота подъема: "<<h<<endl;
}
return 0;
}
```

Отладка
программы

Последовательность действий при отладке программы

1. Для демонстрации отладки выберем **пример 25** и версию программы **Dev-C++ 4.9.9.2.** Создадим, откомпилируем и выполним программу. Программа выдает результат, но, предположим, у нас есть сомнения в правильности результата.
2. Запускаем отладку (кнопка ***Отладка*** или клавиша ***F8***). При первом запуске отладки программа выдаст запрос, хотим ли мы разрешить отладку и перестроить проект. Отвечаем ***Да (Yes).***
3. Задаем точки прерывания программы. Для этого щелкаем мышью в столбце слева от операторов и строчка подсвечивается красным цветом. Программа будет выполняться до строки, предшествующей строке прерывания.

4. Далее работаем с нижней панелью интерфейсного окна. Чтобы она появилась, щелкаем **в нижней части окна** по вкладке ***Отладка.***

5. Щелкаем в нижней панели по опции ***Выполнить до курсора.*** Строчка программы (номер 12), соответствующая первой точке прерывания, становится синей, как показано на рис. 1.

6. Щелкаем в нижней панели программы по кнопке ***Добавить в наблюдаемые,*** появляется дополнительное окошко, в котором задаем интересующую нас переменную **i.** Затем повторяем процедуру и последовательно задаем переменные **h** и **ik.** Эти переменные и их значения появляются в **левой области** интерфейсного окна (рис. 1).

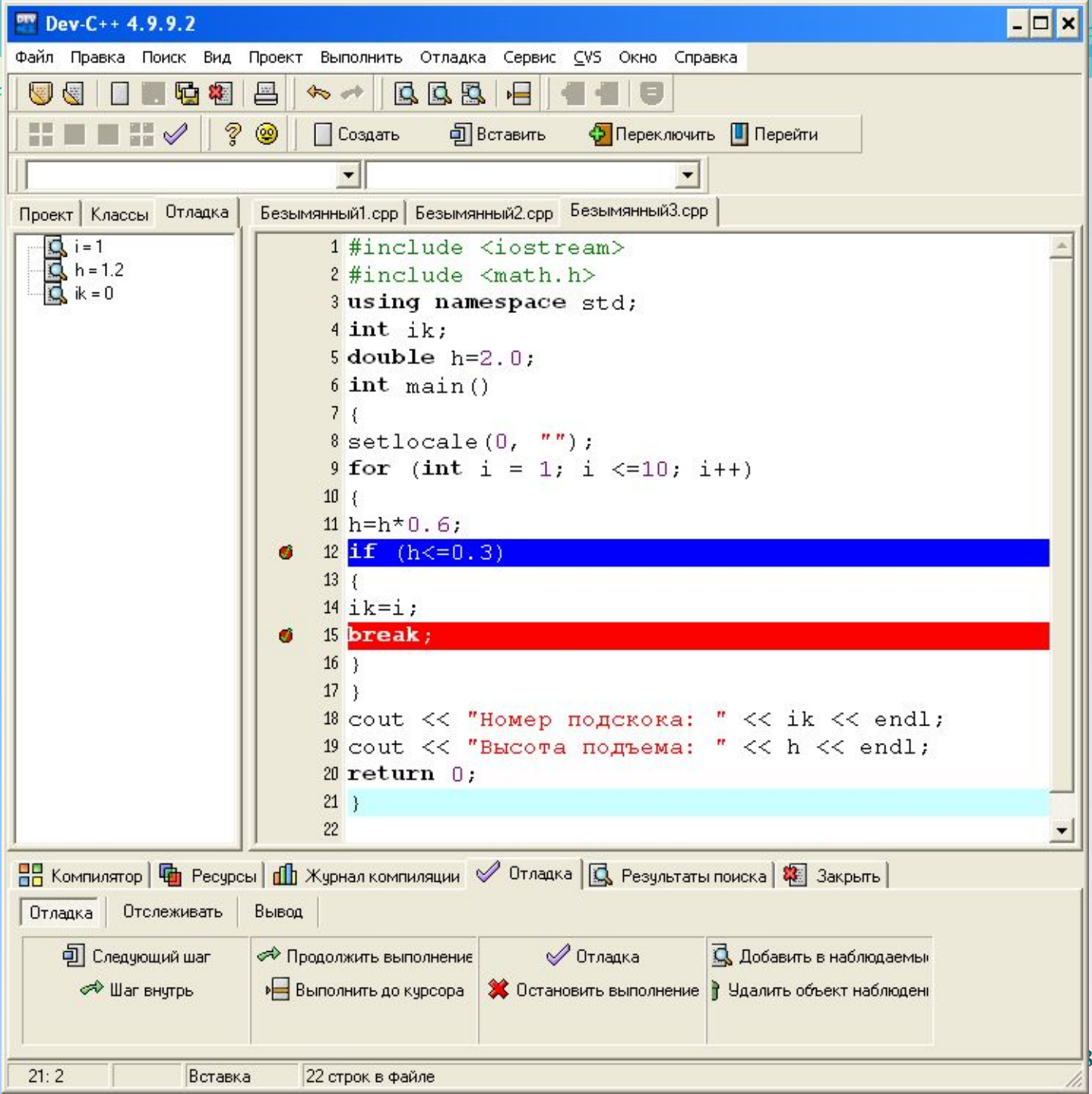

Dev-C++ 4.9.9.2
Файл Правка Поиск Вид Проект Выполнить Отладка Сервис CVS Окно Справка
Создать Вставить Переключить Перейти
Проект Классы Отладка
i = 1
h = 1.2
ik = 0
Безымянный1.cpp Безымянный2.cpp Безымянный3.cpp
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int ik;
5 double h=2.0;
6 int main()
7 {
8 setlocale(0, "");
9 for (int i = 1; i <=10; i++)
10 {
11 h=h*0.6;
12 if (h<=0.3)
13 {
14 ik=i;
15 break;
16 }
17 }
18 cout << "Номер подскока: " << ik << endl;
19 cout << "Высота подъема: " << h << endl;
20 return 0;
21 }
22
Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска Закрыть
Отладка Отслеживать Вывод
Следующий шаг
Шаг внутрь
Продолжить выполнение
Выполнить до курсора
Отладка
Остановить выполнение
Добавить в наблюдаемы
Удалить объект наблюдени
21: 2
Вставка
22 строк в файле

Рис. 1

7. Далее в нижней панели нажимаем на кнопку ***Продолжить выполнение*** и переменные i, h, ik принимают новые значения, как показано на рис. 2, рис. 3, рис. 4.

8. Когда значение i станет равным четырем, выполнится условный оператор, нижняя строчка (номер 15), соответствующая второй точке прерывания, станет синей, также выполнится оператор break и программа закончит работу (рис. 5).

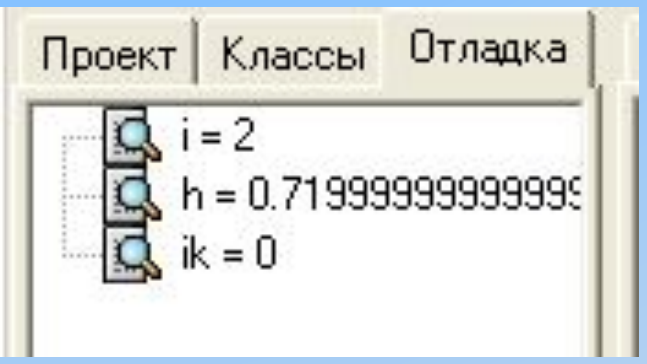


Рис. 2

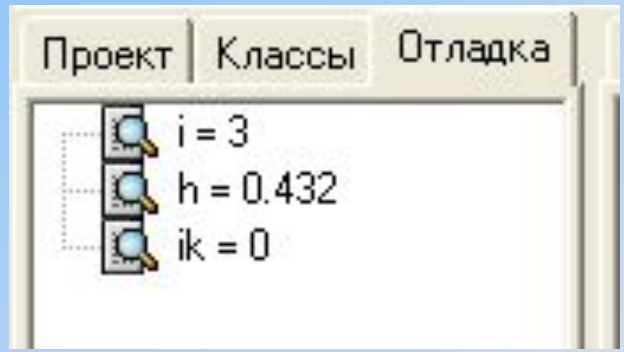


Рис. 3

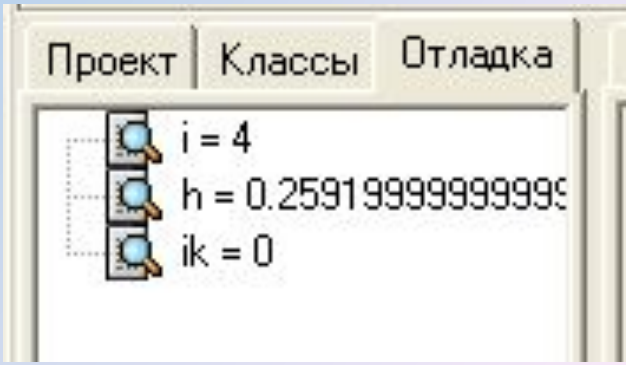


Рис. 4

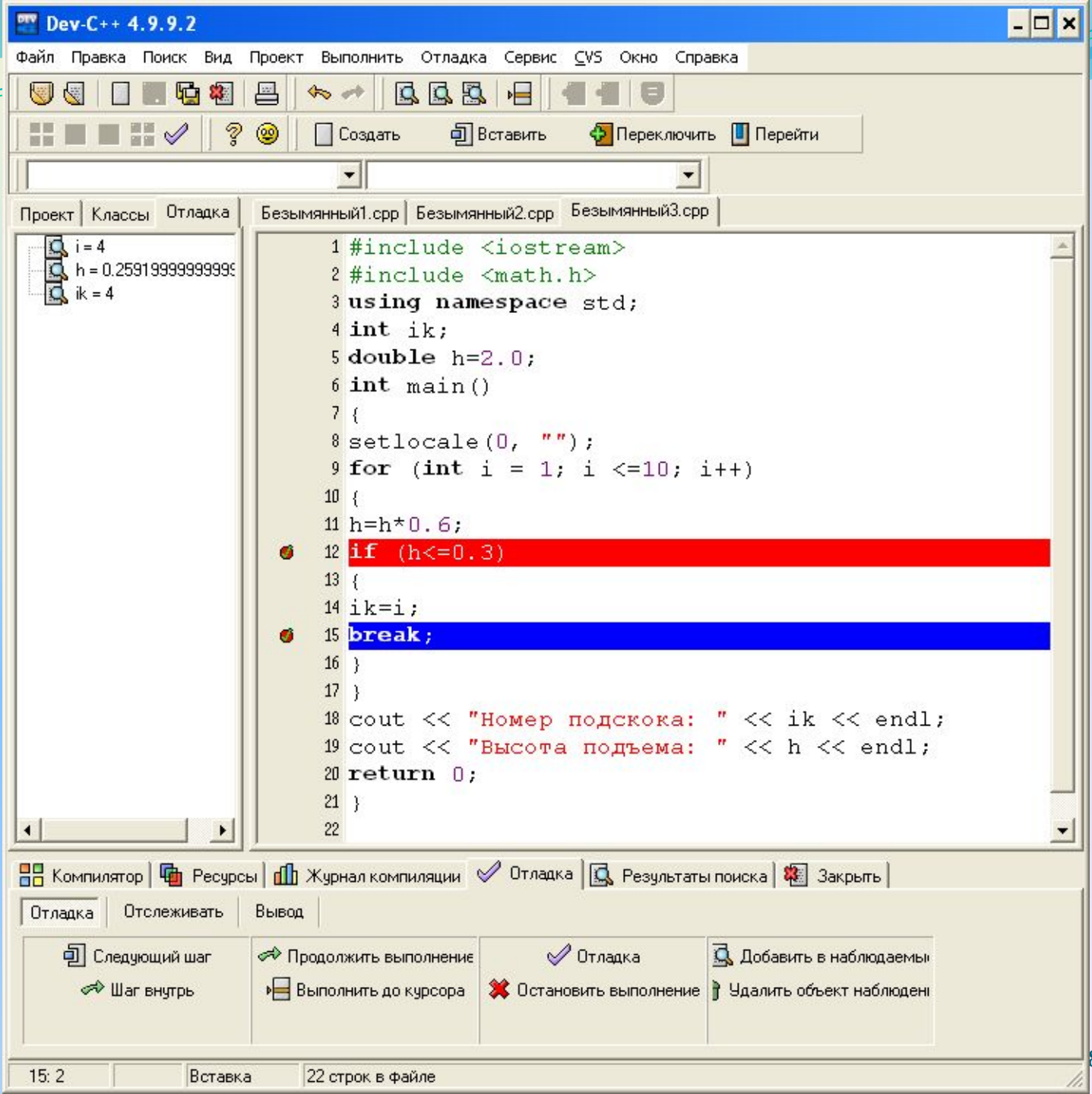

Dev-C++ 4.9.9.2
Файл Правка Поиск Вид Проект Выполнить Отладка Сервис CVS Окно Справка
Создать Вставить Переключить Перейти
Проект Классы Отладка
i = 4
h = 0.259199999999999
ik = 4
Безымянный1.cpp Безымянный2.cpp Безымянный3.cpp
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int ik;
5 double h=2.0;
6 int main()
7 {
8 setlocale(0, "");
9 for (int i = 1; i <=10; i++)
10 {
11 h=h*0.6;
12 if (h<=0.3)
13 {
14 ik=i;
15 break;
16 }
17 }
18 cout << "Номер подскока: " << ik << endl;
19 cout << "Высота подъема: " << h << endl;
20 return 0;
21 }
22
Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска Закрыть
Отладка Отслеживать Вывод
Следующий шаг
Шаг внутрь
Продолжить выполнение
Выполнить до курсора
Отладка
Остановить выполнение
Добавить в наблюдаемы
Удалить объект наблюдени
15: 2
Вставка
22 строк в файле

Рис. 5

Таким образом, с помощью режима отладки мы проконтролировали работу программы, выяснили, какие значения принимают переменные i, h, ik, убедились, что программа работает правильно.
Чтобы остановить режим отладки, надо нажать кнопку ***Остановить выполнение*** в нижней панели.
С помощью кнопки ***Следующий шаг***, можно проследить пошаговое выполнение программы.
С помощью кнопки ***Удалить объект наблюдения,*** можно удалять наблюдаемые переменные из левой области интерфейсного окна.

Литература

1. http://life-prog.ru/view_cat.php?cat=2. Язык программирования Си(C++). Обучающие уроки
2. http://code-live.ru Уроки C++ с нуля.
3. http://kpolyakov.spb.ru К.Ю. Поляков, Е.А. Еремин. Программирование на C++ (презентация).
4. http://comp-science.narod.ru/Progr/file_c.htm Шестаков А.П. Файлы в C++.
5. http://itedu.ru/courses/cpp/functions-in-cpp Обучение программированию.
6.http://www.youtube.com/playlist?list=PLbmlzoDQrXVFC13GjpPrJxl6mzTiX65gs. C++ Уроки Denis Markov (Полный курс 28 уроков)
7. Программирование Ардуино. (В книге: В.Н. Иванов, И.О. Мартынова. Электроника и микропроцессорная техника. – М.: «Академия», 2016)

Допускается использование и копирование слайдов при условии ссылки на презентацию.