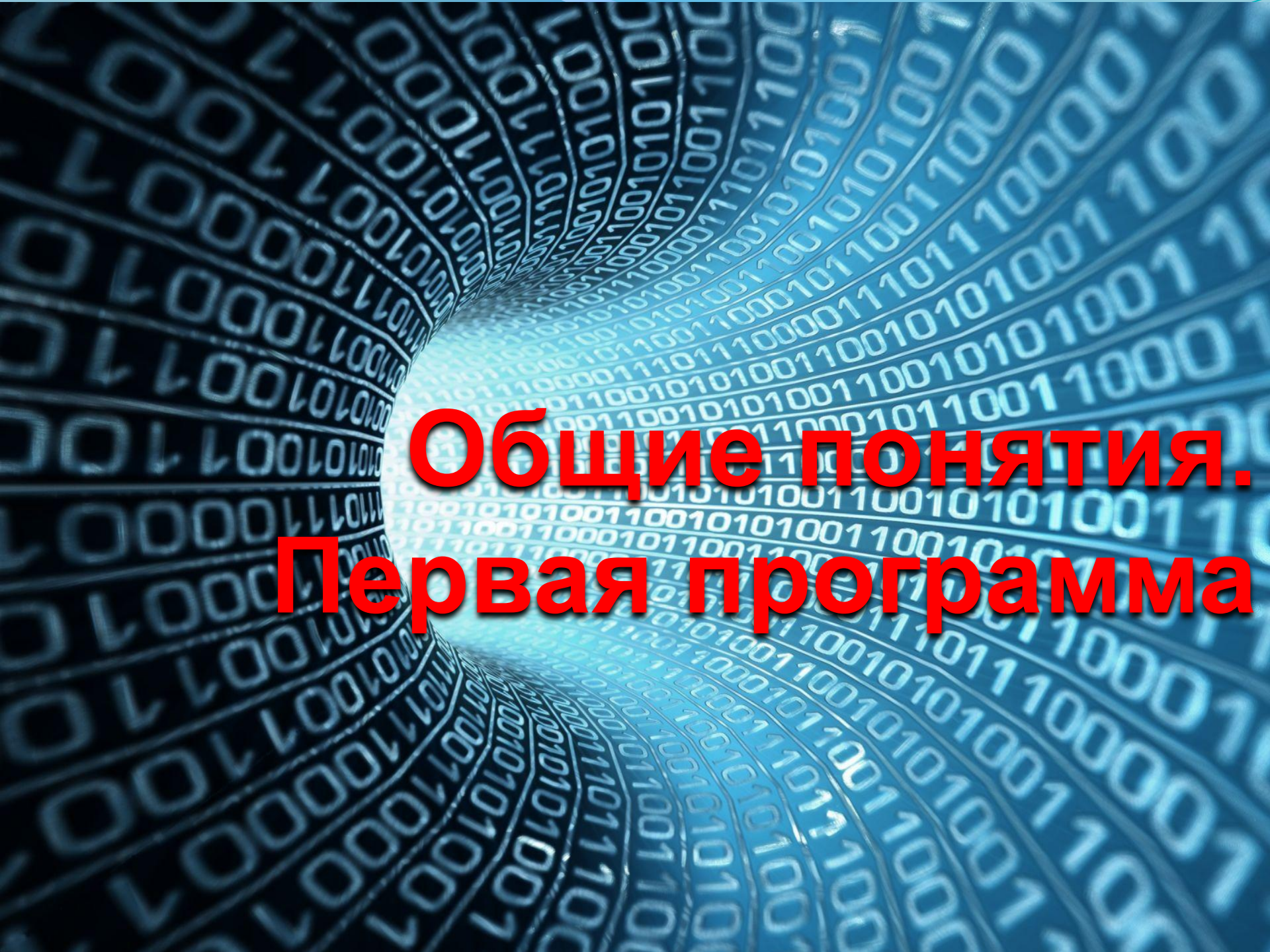
Программирование на C++ для начинающих. Построение графиков.

(презентация – практикум)

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

2016

СОДЕРЖАНИЕ


Общие понятия.
Первая программа

Этапы создания программы

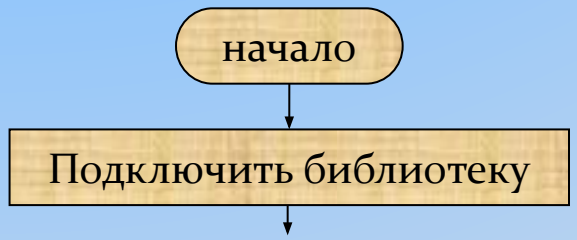
Программа – это перечень действий (инструкций), которые должна выполнить вычислительная машина.

программист **транслятор**

Подключить библиотеку
Ввести переменную a
Вычислить площадь
Вывести результат

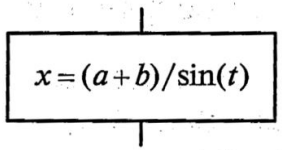
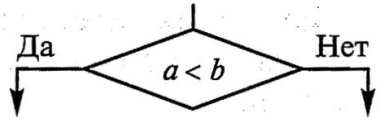
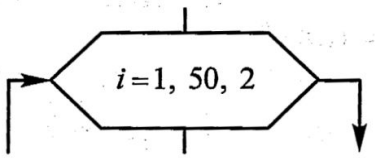
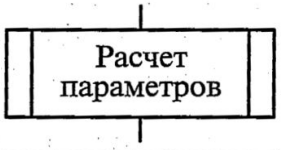
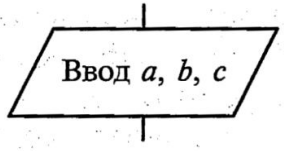
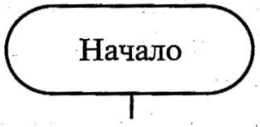
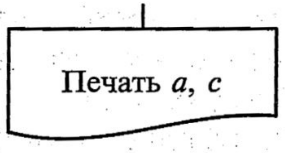
```
#include <iostream>
a=10;
s=a*a;
cout << s;
```

```
00101110101O
01010110010l
0101110101l0
1011011110l0
```



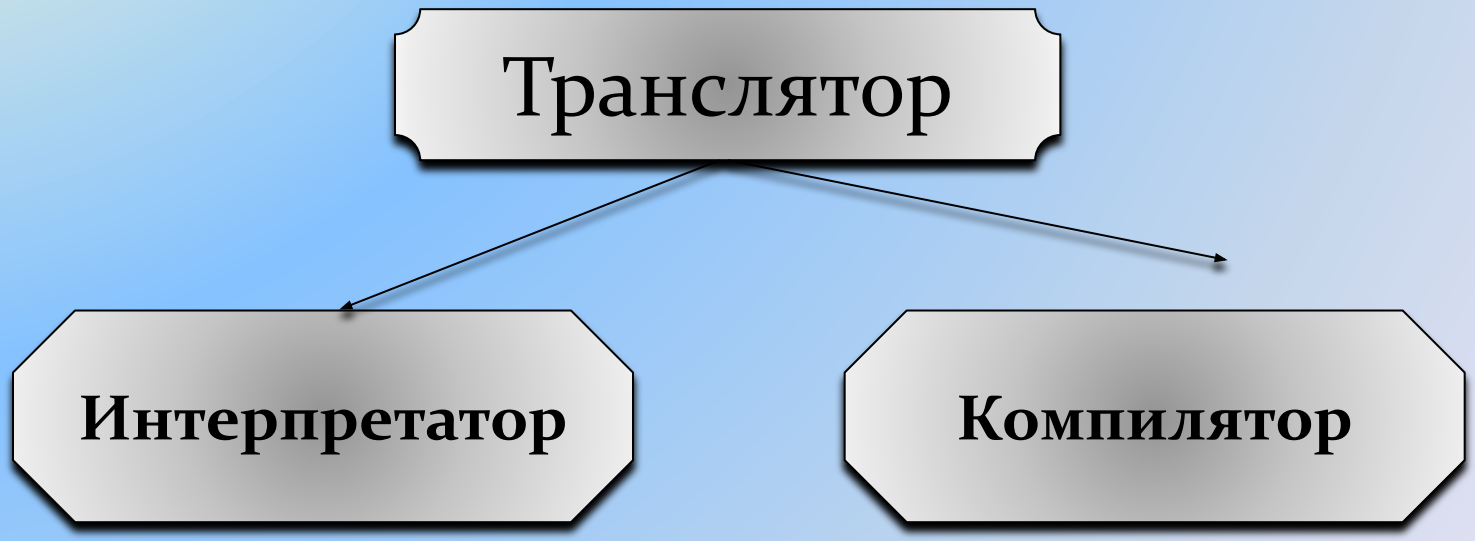
Последовательность инструкций, которые должна выполнить вычислительная машина, называется **алгоритмом.** Графически алгоритм можно изобразить в виде блок схемы, состоящей из условных обозначений.

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

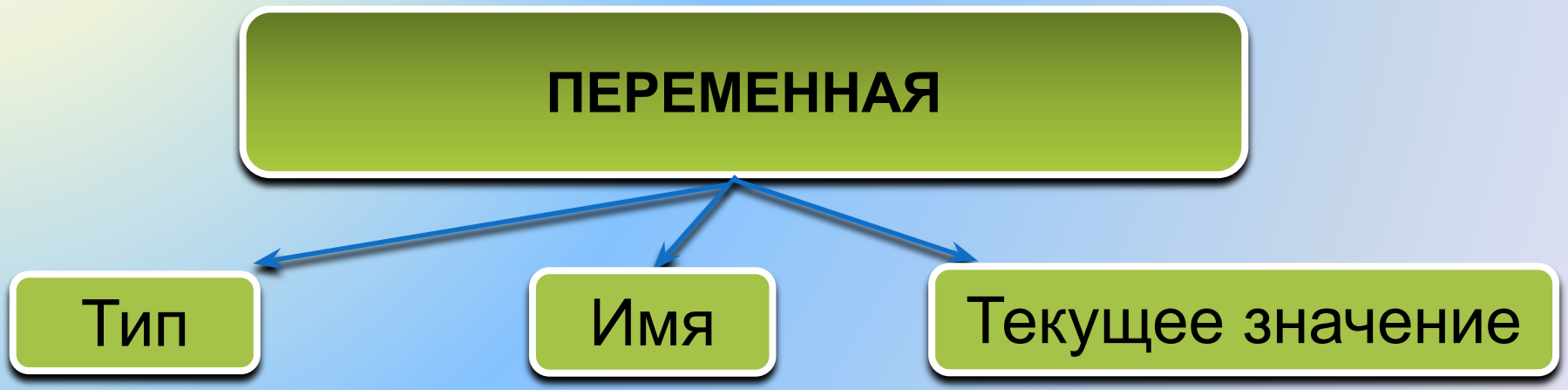
Формирует машинный код, готовый к исполнению вычислительной машиной

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**




```
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

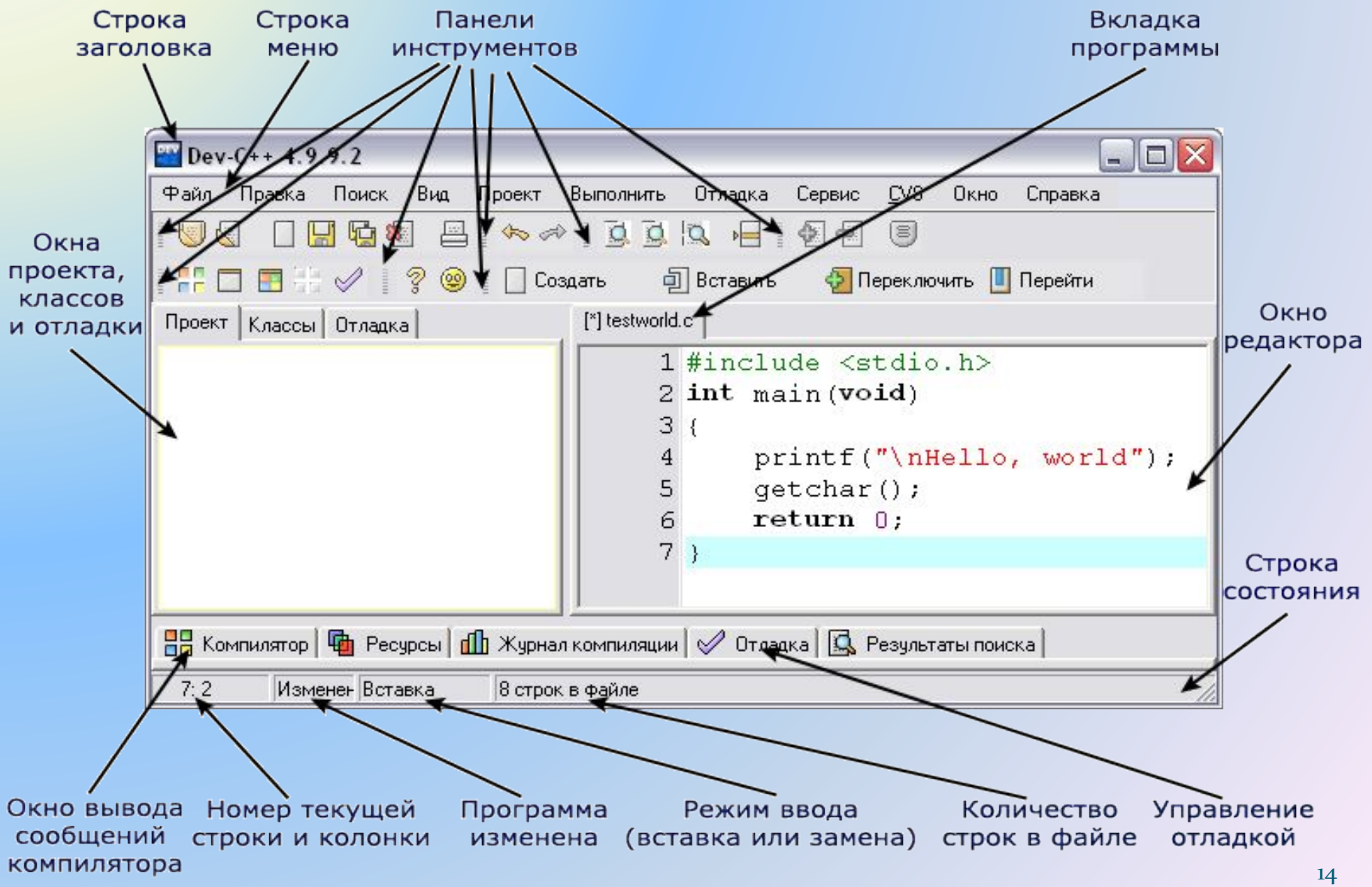
Программное обеспечение для изучения C++

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/**)



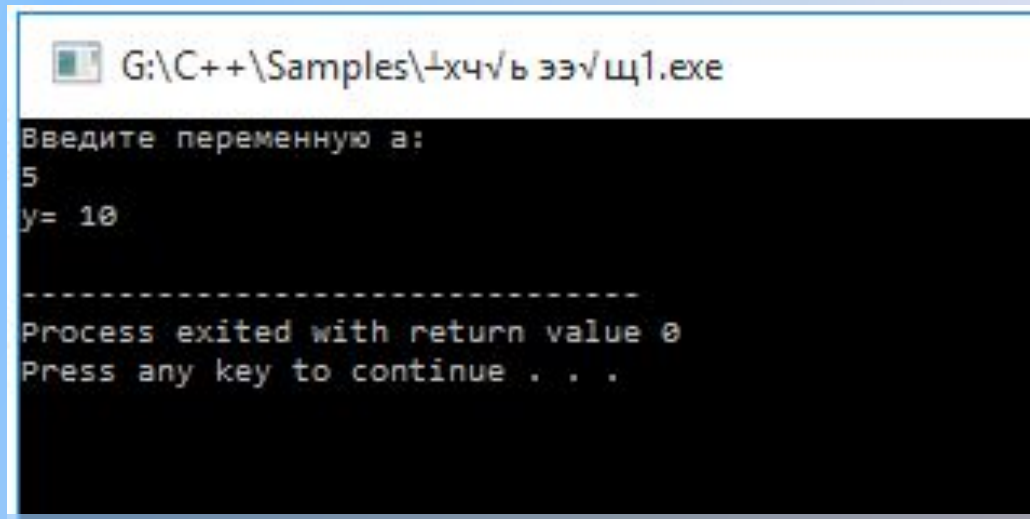
Порядок работы с программой Dev-C++

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)

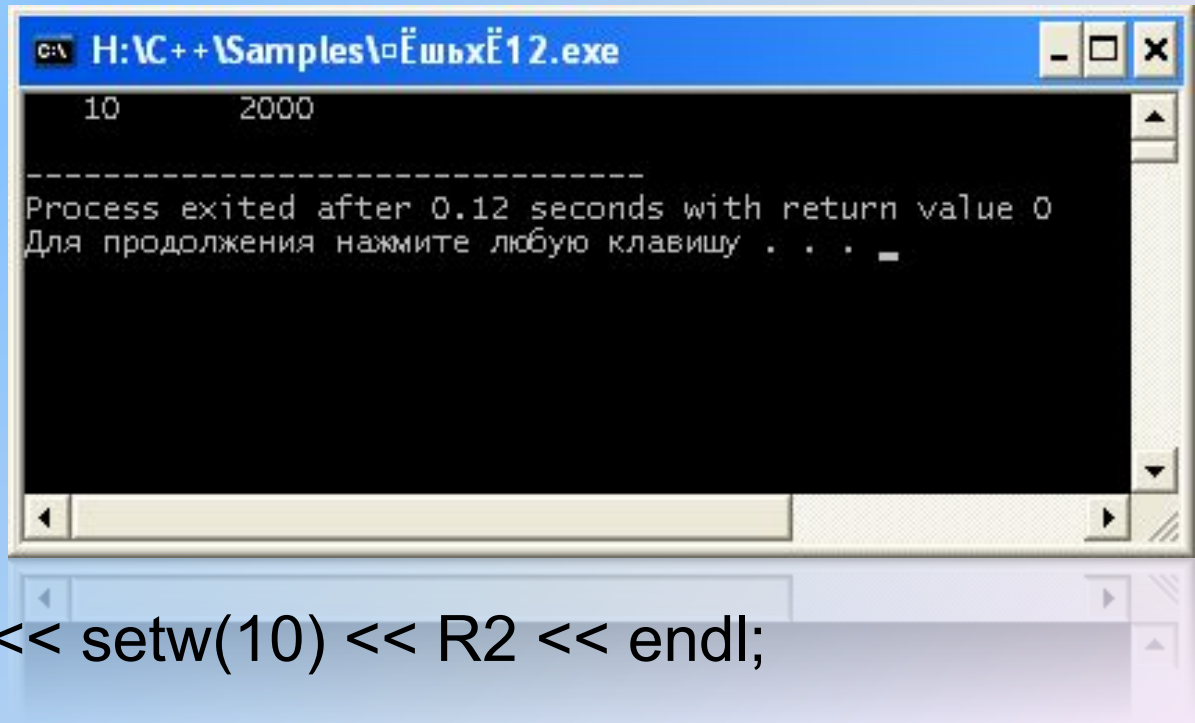
При записи результатов в виде x= 1.2e+003, число 1.2 называется мантиссой, 003 – показатель степени при основании 10. В данном примере при переходе к записи числа в обычной форме точка должна быть передвинута на три знака вправо, т. е. x = 1200. Если показатель степени отрицательный (например, -003),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

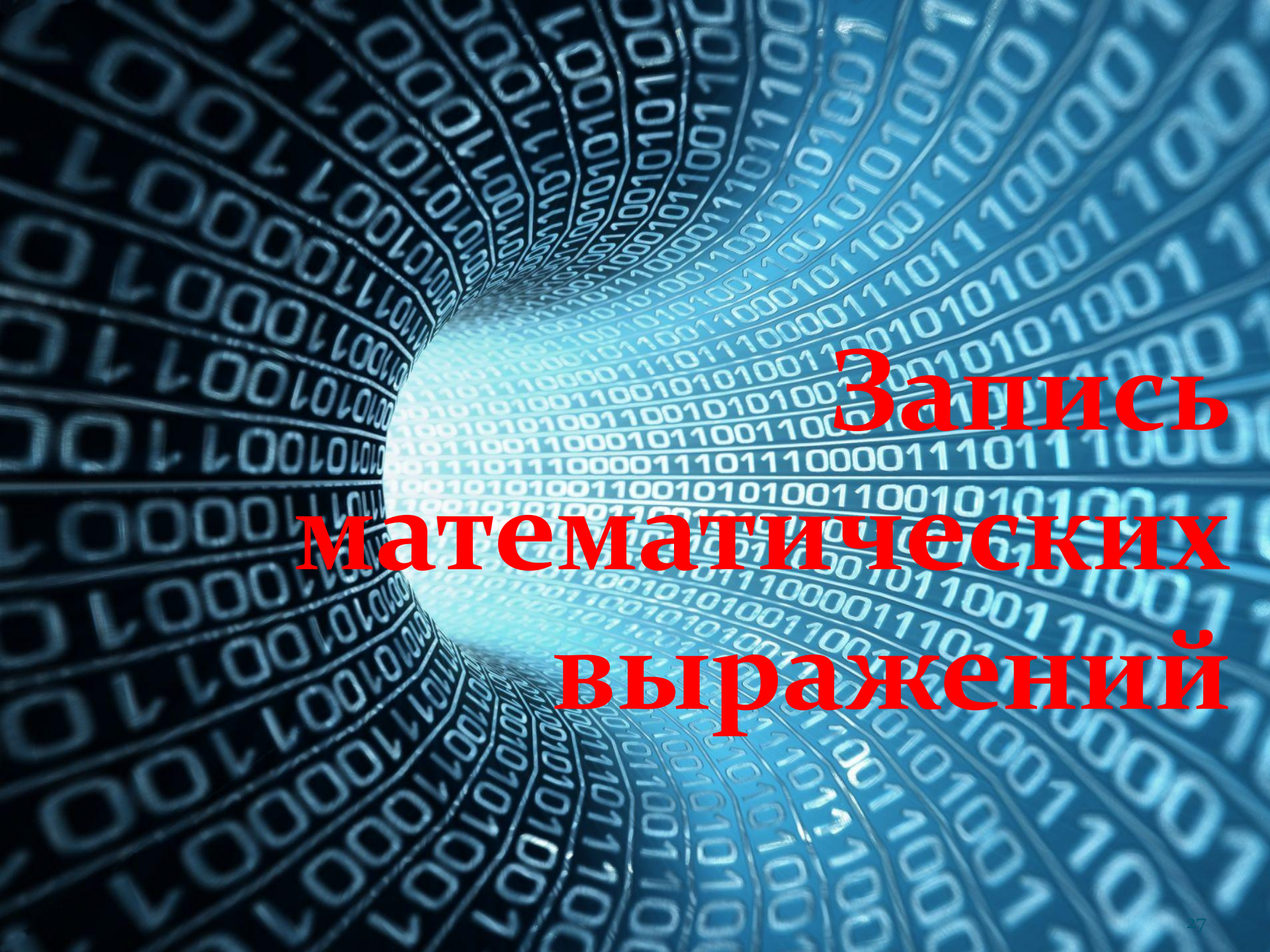


СПЕЦИАЛЬНЫЕ СИМВОЛЫ ДЛЯ ИСПОЛЬЗОВАНИЯ С ОПЕРАТОРОМ COUT.	
СИМВОЛ	**НАЗНАЧЕНИЕ**
\a	Сигнал бипера компьютера
\n	Переход на новую строку
\r	Возврат каретки в начало строки
\t	горизонтальная табуляция
\v	Вертикальная табуляция
\\	Обратный слеш
\?	Знак вопроса
\'	Одинарные кавычки
\"	Двойные кавычки
\0	Нулевой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

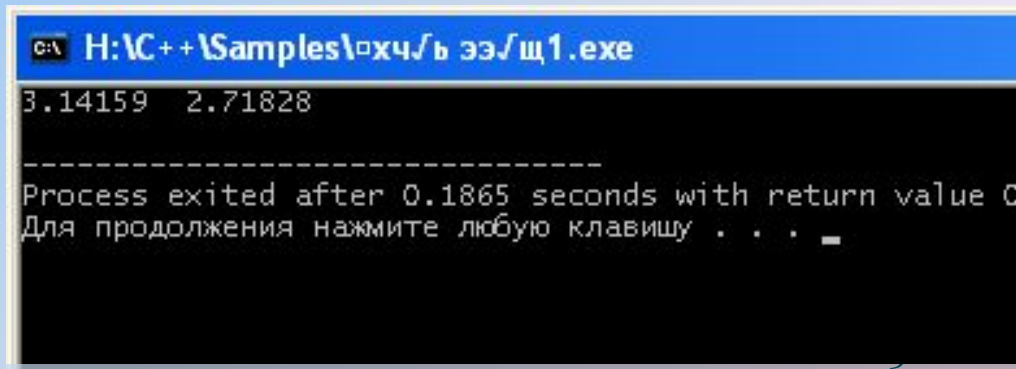
```
int a, b;
a ++;        // a = a + 1;
a --;        // a = a – 1;
a += b;      // a = a + b;
a -= b;      // a = a - b;
a *= b;      // a = a * b;
a /= b;      // a = a / b;
a %= b;      // a = a % b;
```

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.

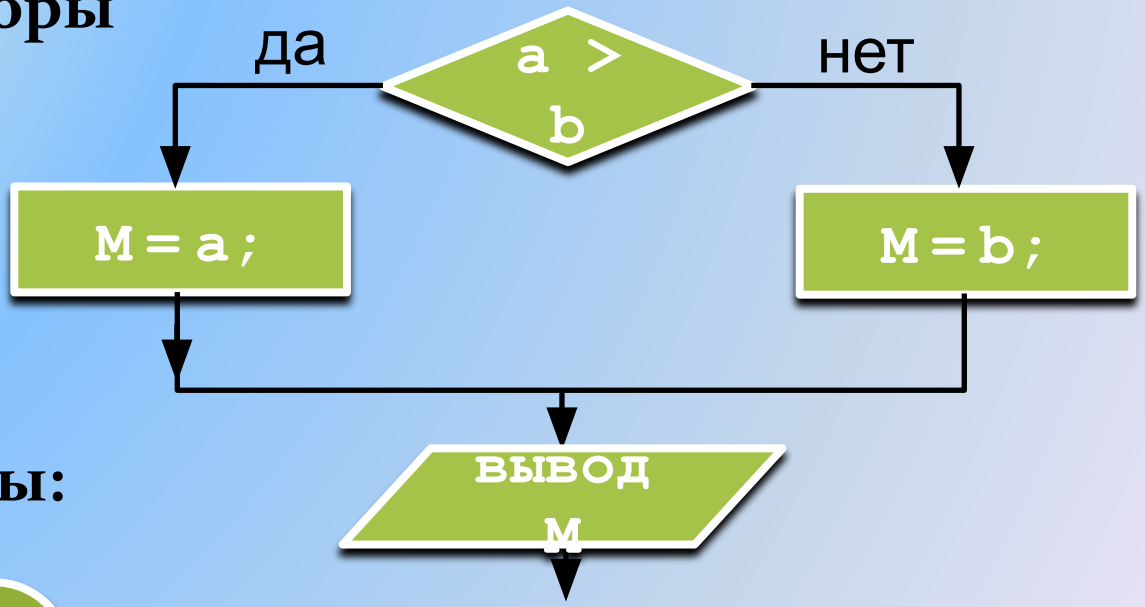
Условие if всегда записывается в круглых скобках, точка с запятой после условия не ставится. Если при выполнении условия требуется выполнить только одну команду, то фигурные скобки ставить не обязательно.

Полная форма условного оператора

Полная форма ветвления, используются операторы

if … else

Блок – схема:



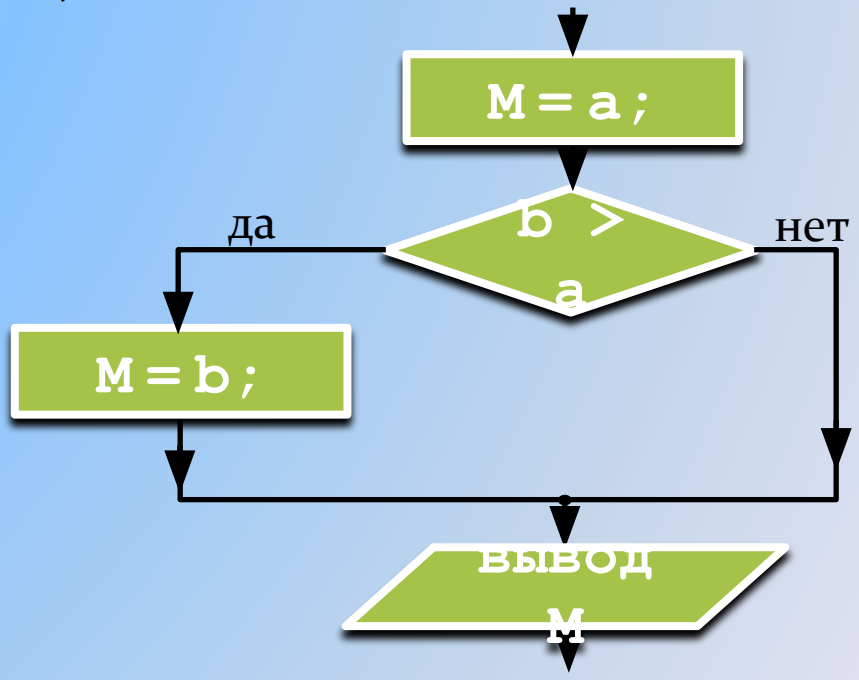
Фрагмент программы:

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

Неполная форма условного оператора

Неполная форма ветвления, используется только if

Блок-схема:

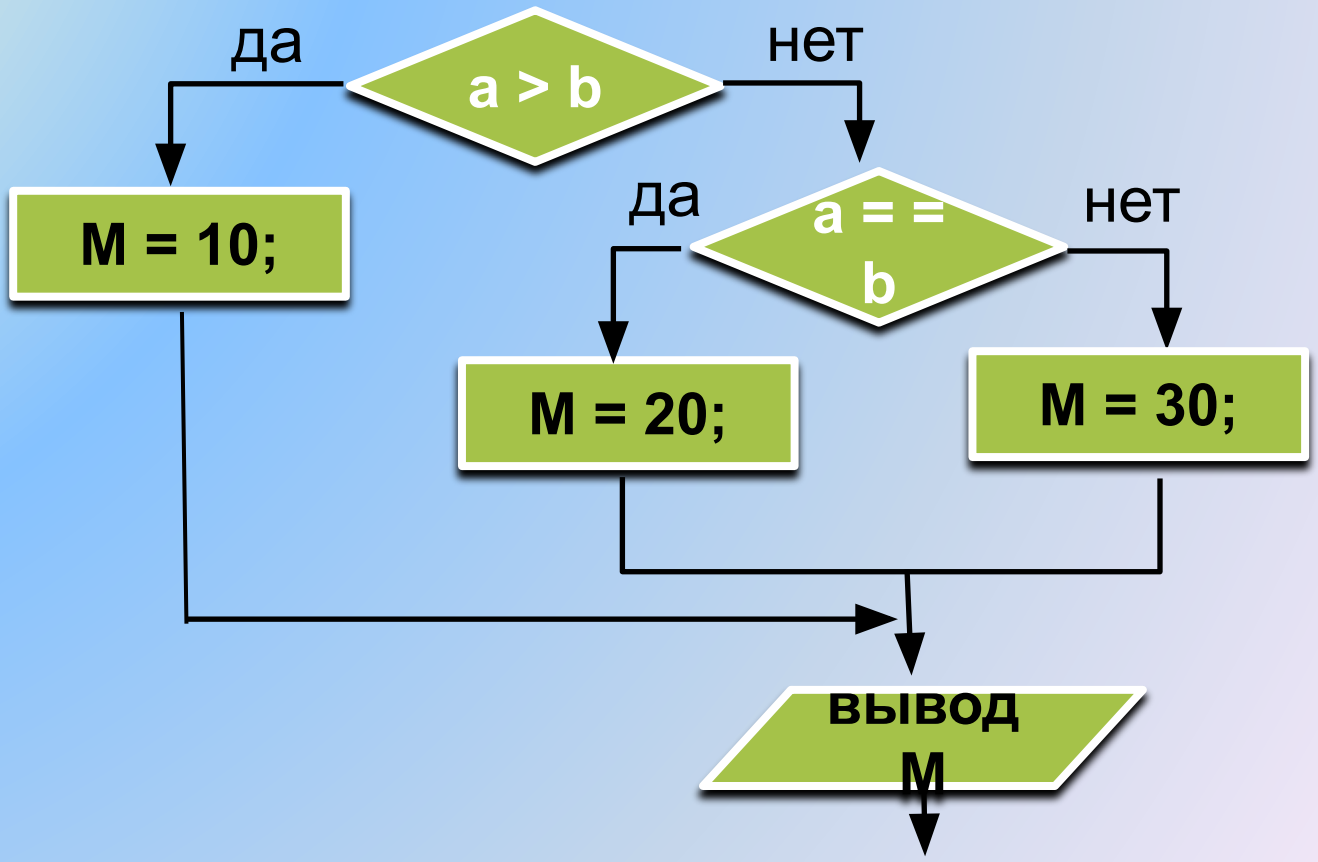


Фрагмент программы:

```
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:

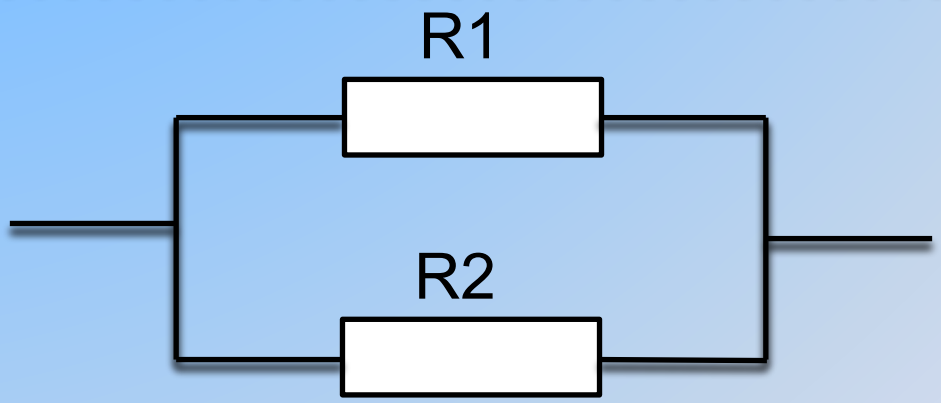
```
double a, b;
double x=3.1416/4;
if (num >=50)
 {
 a = 1 − cos²x / cos x ;

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»**

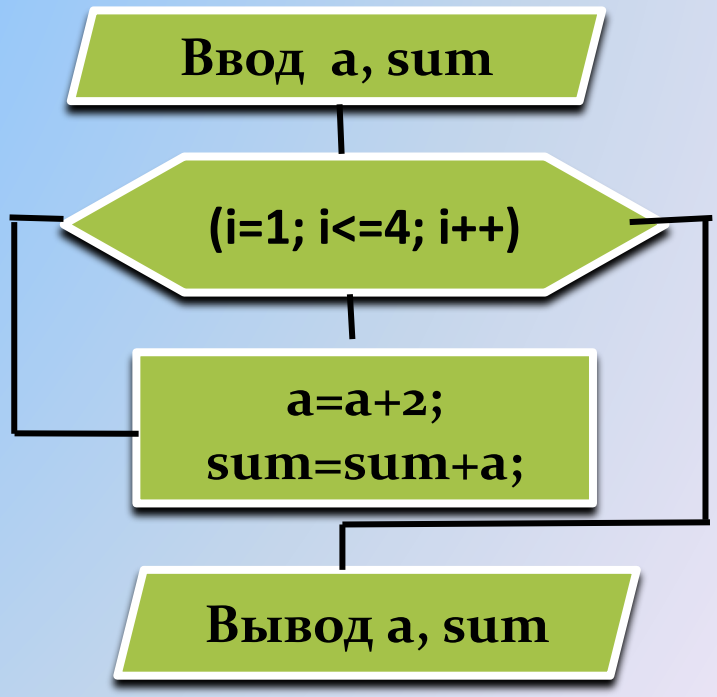
```
#include <iostream>
#include <math.h>
using namespace std;
int L, dL=5;
int main() {
setlocale (0, "");
for (L=0; L<=50; L=L+dL)
cout << "Расстояние в км: " << L << endl;
return 0;
}
```

```
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:

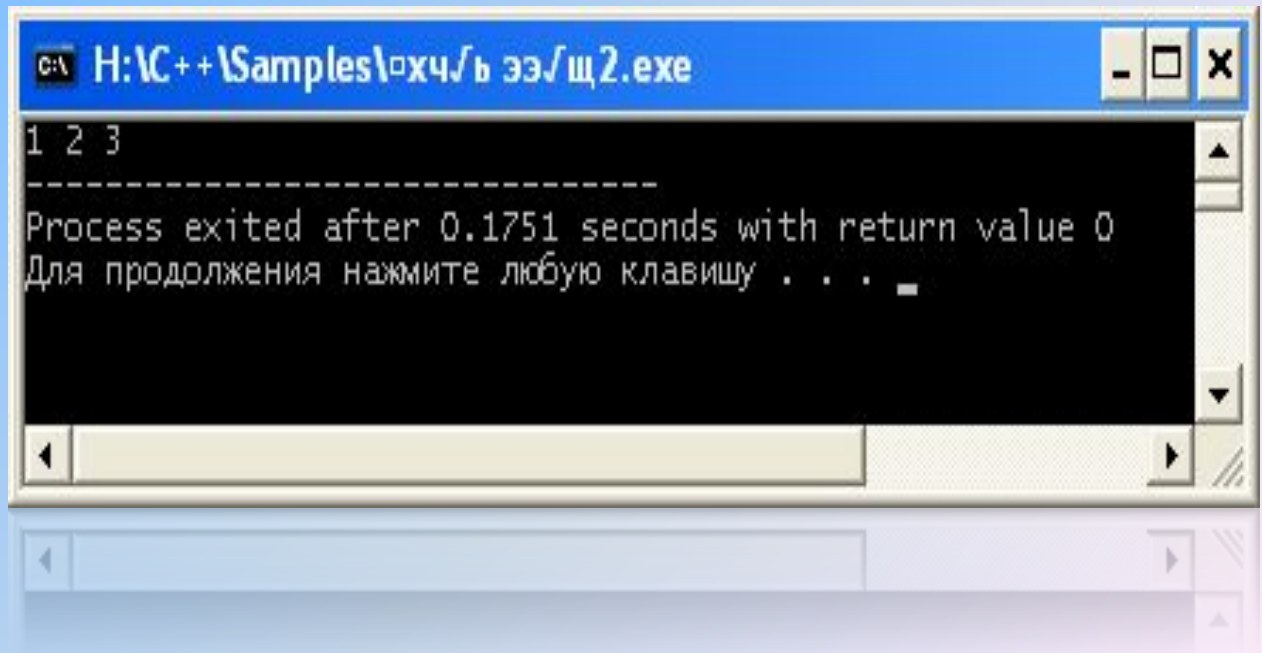
```
do
 {
Тело цикла;
 }
 while (условие);
```

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.

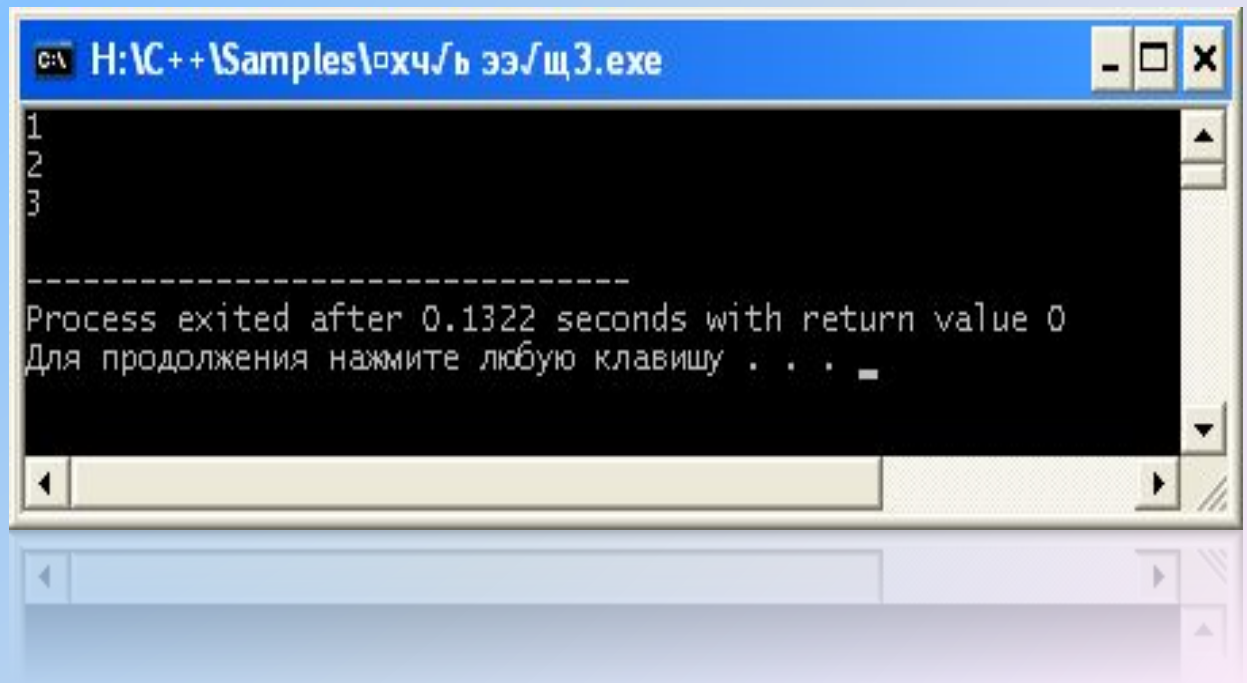
```
#include <iostream>
using namespace std;
int main()
{
int i=0;
while (i<3)
{
i++;
cout << i << " ";
}
return 0;
}
```



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

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

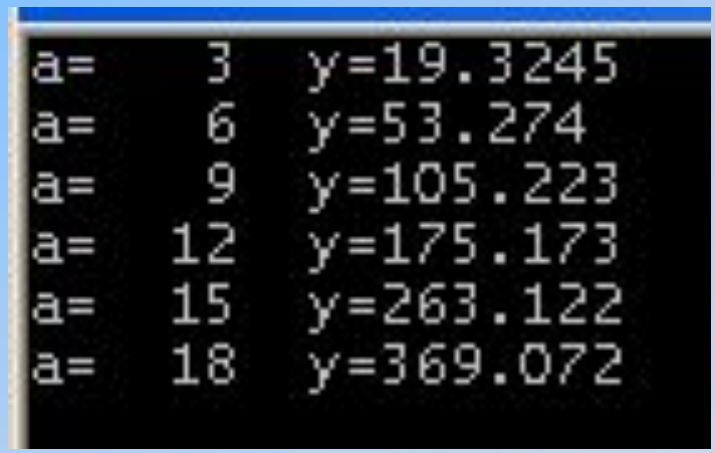
```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int a=5;
double b, y;
int main()
{
setlocale(0, "");
   int i;
   for (i = 1; i <=6; i++)
{
b=0.5*i;
   y=pow(a,2)+pow(b,3)+a*b+a/sqrt(b);
   cout << "b=" <<setw(4) << b << "  y=" << y << endl;
}
   return 0;
}
```

```
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).

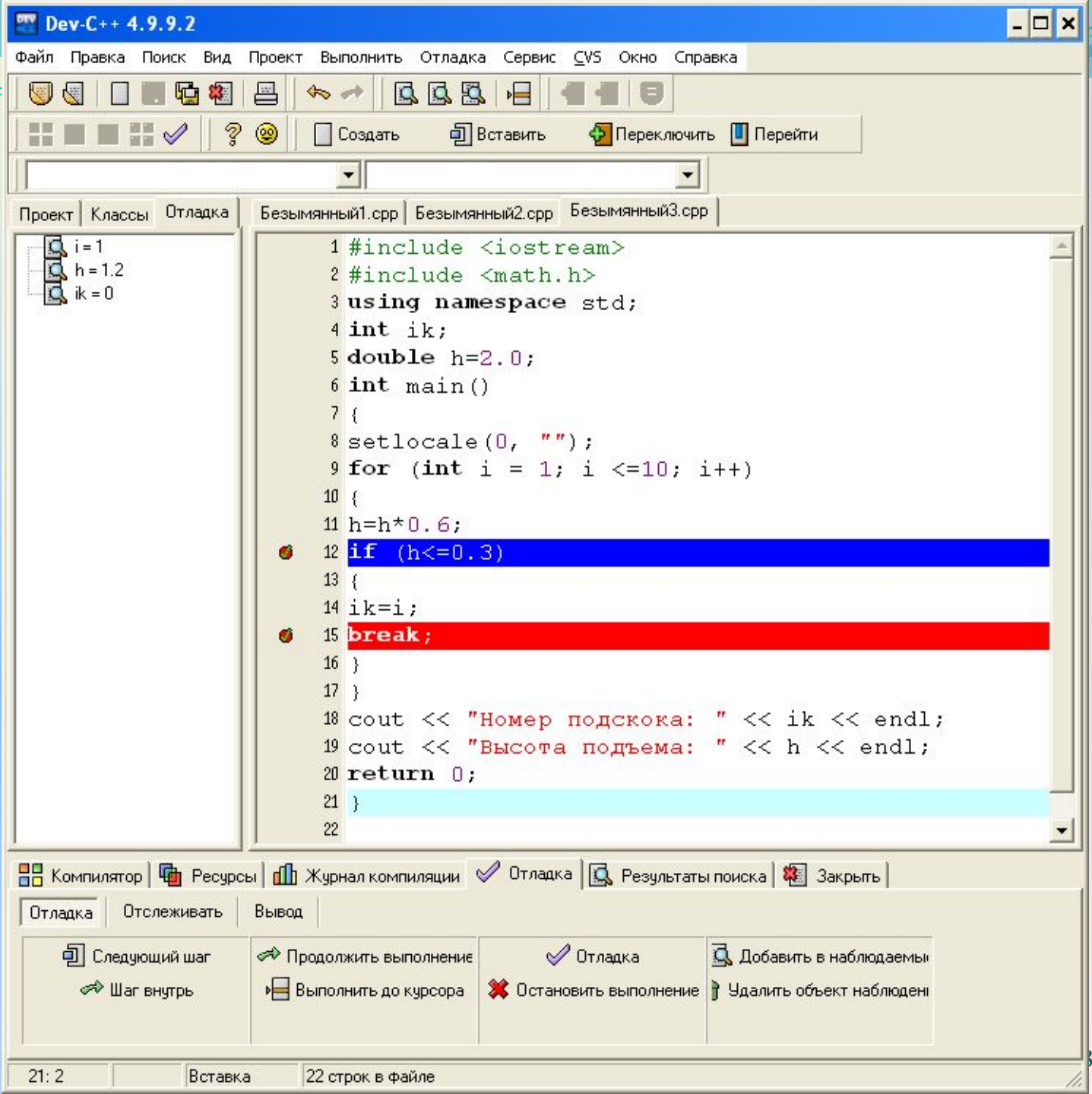

Dev-C++ 4.9.9.2
Файл Правка Поиск Вид Проект Выполнить Отладка Сервис CVS Окно Справка
Создать Вставить Переключить Перейти
Проект Классы Отладка
i = 1
h = 1.2
ik = 0
Безымянный1.cpp Безымянный2.cpp Безымянный3.cpp
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int ik;
5 double h=2.0;
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).

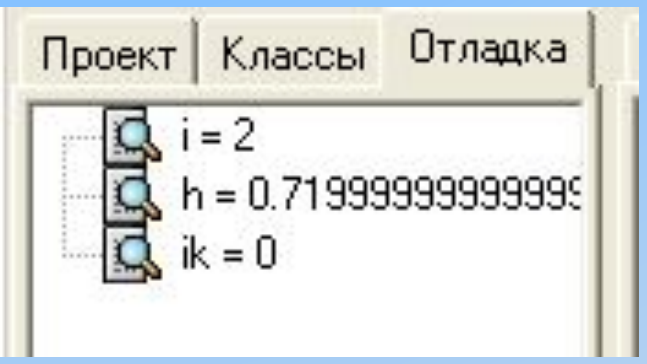


Рис. 2

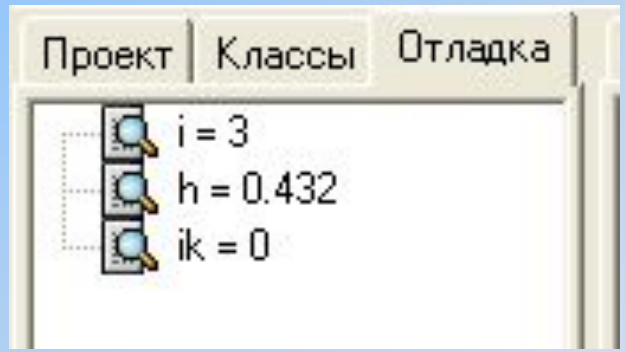


Рис. 3

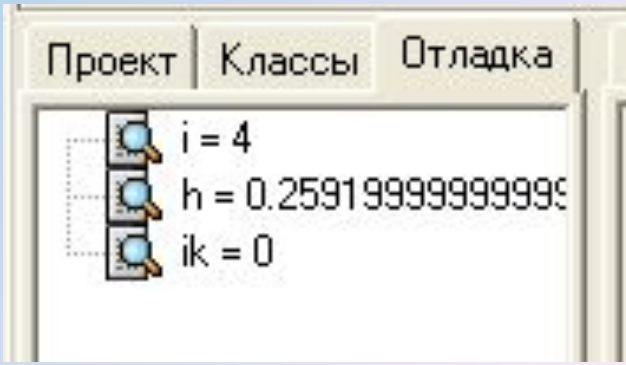


Рис. 4

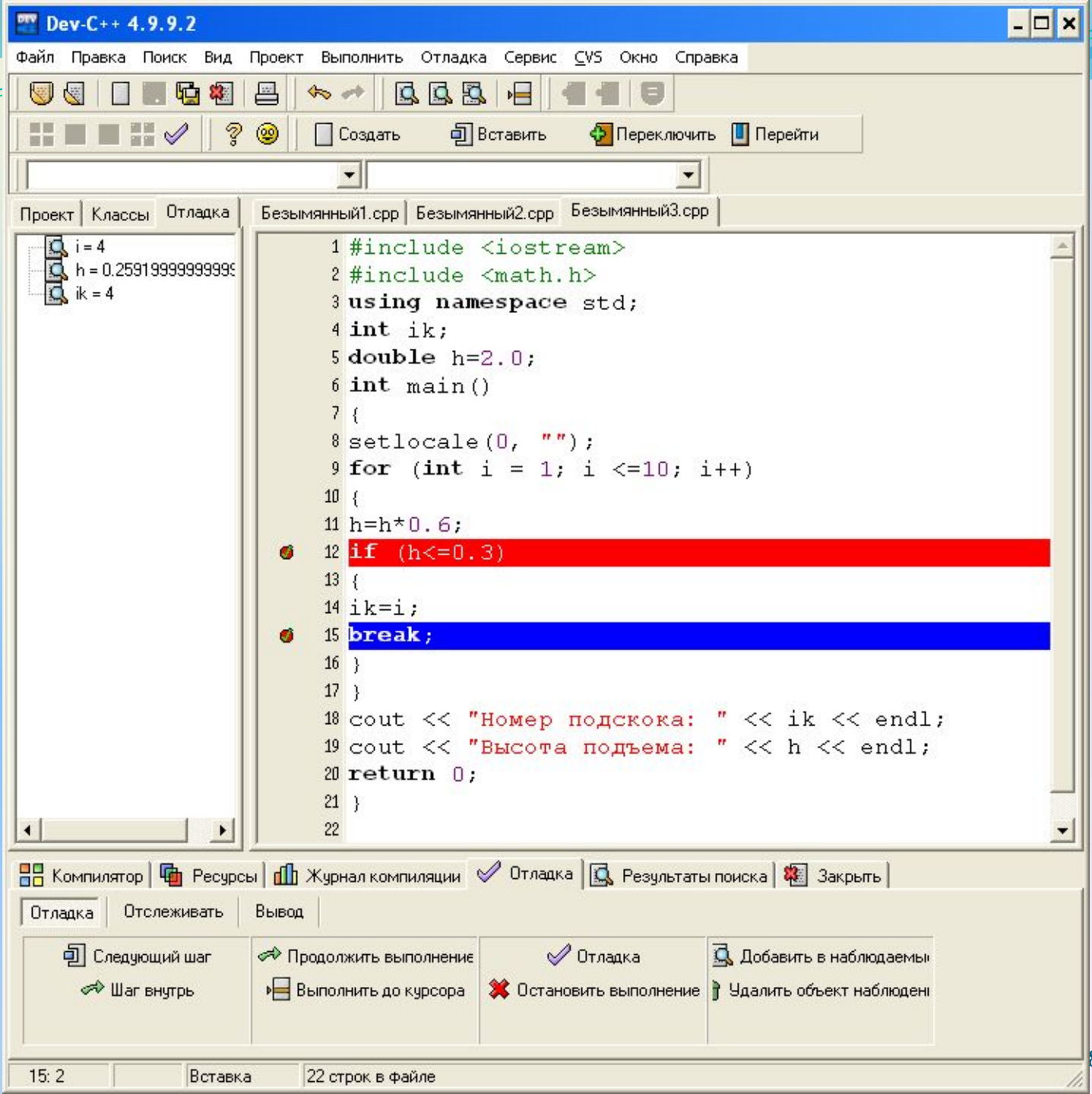

Dev-C++ 4.9.9.2
Файл Правка Поиск Вид Проект Выполнить Отладка Сервис CVS Окно Справка
Создать Вставить Переключить Перейти
Проект Классы Отладка
i = 4
h = 0.259199999999999
ik = 4
Безымянный1.cpp Безымянный2.cpp Безымянный3.cpp
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int ik;
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.


Массивы

Описание массива и вывод на печать

Массив (совокупность нескольких значений) Имя массива создается аналогично имени обычной переменной, но за именем массива **в квадратных скобках** указывается количество элементов массива, которое задается при его объявлении. Чтобы описать элементы массива сразу при его создании, можно использовать фигурные скобки. В фигурных скобках значения элементов массива перечисляются **через запятую.** В конце закрывающей фигурной скобки ставится **точка с запятой**. При **объявлении массива** нумерация элементов массива **начинается с нуля**. Массив, как и любую переменную можно не заполнять значениями при объявлении.

Пример 27 (вариант заполнения массива сразу после его объявления)

```
#include <iostream>
using namespace std;
int main()
{
setlocale (0, "");
int a[5]={5,10,15,20,25};
for (int i=0; i <= 4; i++)
 cout << a[i] << endl;
 return 0;
 }
```

Важно:

нумерация элементов массива начинается с 0:

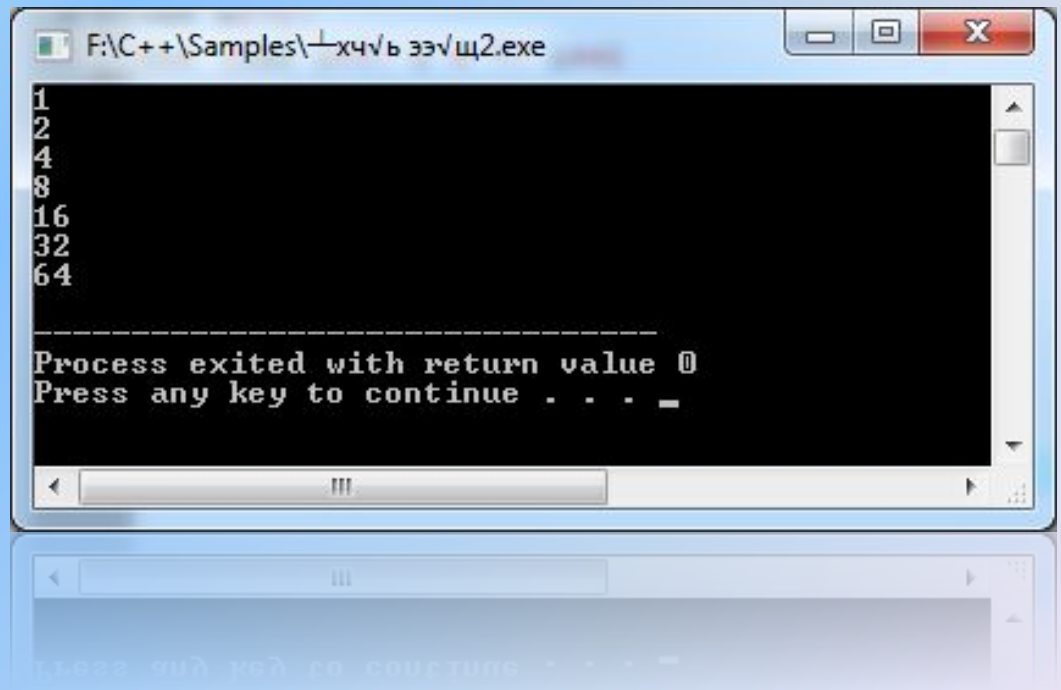
a[0] = 5
a[1] = 10
a[2] = 15
a[3] = 20
a[4] = 25

5 элементов

Элемента a[5] не существует .

Пример 28 (вариант заполнения массива с помощью оператора цикла)

```
#include <iostream>
using namespace std;
int main()
{
setlocale (0, "");
int a[7];
for (int i=0; i < 7; i++)
{
if (i==0)
a[i]=1;
else
a[i] = a[i-1]*2;
 cout << a[i] << endl;
}
 return 0;
 }
```



Написать программы, аналогичные примеру 27 на **заполнение массива при объявлении**. Вывести массивы на монитор:

P[7]=0, -1, 1, -2, 2, -3, 3;

Q[8]=1.8, 2.5, 3.7, 4.9, 2.9, 1.5, -1.7, -2.3;

R[4]=3, 33, 333, 3333;

Написать программы, аналогичные примеру 28 на **заполнение массива с помощью оператора цикла.** Вывести массивы на монитор:

S[5]=1, 10, 100, 1000, 10000;

T[5]=-1, -8, -27, -64, -125;

U[6]=1.5, 2.0, 2.5, 3.0, 3.5, 4.0;

Пример 29. Написать программу для вычисления дальности полета тела, брошенного под углом α к горизонту. Сопротивлением воздуха пренебречь. Расчет провести для следующих углов бросания: 0, 15, 30, 45, 60, 75, 90 градусов. Для хранения рассчитанных значений дальности создать массив L[7]. Задать начальную скорость бросания равной V = 20 м/сек. Ускорение свободного падения g = 9,81 м/сек^2 .

Дальность полета тела рассчитывается по формуле

$$L = (2V^2 cos\,\alpha \cdot sin\,\alpha)/g$$

Пример 29

```
#include <iostream>
#include <math.h>
using namespace std;
double v=20;
double g=9.81;
double pi=3.1416;
double alf;
double L[7];
int main()
{
    setlocale (0, "");
    alf=0;
    int i;
    for (i=0; i<=6; i++)
{
    L[i]=2*pow(v,2)*sin(alf)*cos(alf)/g;
    cout << alf << "  " << L[i] << endl;
    alf=alf+pi/12;
}
return 0;
}
```

Пример 30.

Взять за основу программу предыдущего примера и видоизменить ее так, чтобы начальная скорость и угол бросания вводились с клавиатуры. Тогда программа будет находить дальность полета при любой начальной скорости и любом угле бросания, заданных с клавиатуры. Вывести на монитор заданный угол бросания, заданную начальную скорость и рассчитанную дальность полета.

Примечание: угол вводить в градусах, а в формулу подставлять в радианах.

Пример 30

```
#include <iostream>
#include <math.h>
using namespace std;
double v, alf, L;
double g=9.81;
double pi=3.1416;
int main()
{
setlocale (0, "");
cout << "Введите скорость: ";
cin >> v;
cout << "Введите угол: ";
cin >> alf;
alf=alf*pi/180;
    L=2*pow(v,2)*sin(alf)*cos(alf)/g;
    cout << "Дальность полета: " << L << endl;
return 0;
}
```

Пример 31.

Написать программу для перевода двоичного числа в десятичное.

Использовать следующий алгоритм

$$b = a_n 2^{n-1} + a_{n-1} 2^{n-2} + \cdots + a_1 2^0$$

где ***b*** – десятичное число;

a_n – крайняя слева цифра двоичного числа;

n – количество цифр в двоичном числе.

Например, перевести двоичное число **11001011** в десятичное:

$$b = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 203$$

При написании программы представить двоичное число в виде массива **a[8] = {1,1,0,0,1,0,1,1}**

Пример 31

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int i;
int b=0;
int a[8]={1,1,0,0,1,0,1,1};
for (i=0; i<=7; i++)
{
b=b+a[i]*pow(2,(7-i));
}
cout << b << endl;
return 0;
}
```


Оператор switch

Оператор SWITCH

Конструкция оператора **switch**:

```
switch (переключатель)
{
  case значение1:
    инструкция1;
    break;
  case значение2:
    инструкция2;
    break;
  …
}
```

Пример,

```
a=2;
switch (a)
{
case 1:
b=1;
break;
case 2:
b=2;
break;
}
```

В качестве переключателя могут выступать только целочисленные переменные или выражения.

Использование оператора SWITCH

Пример 32. Вывод словесного описания оценки, основываясь на текущей оценке ученика

```
#include <iostream>
using namespace std;
int main() {
setlocale (0, "");
int grade;
cout << "Введите оценку от 2 до 5" << endl;
cin >> grade;
switch (grade)
{ case 5: cout << "У вас оценка отлично" << endl; break;
  case 4: cout << "У вас оценка хорошо" << endl; break;
  case 3: cout << "У вас всего лишь удовлетворительно" <<
endl; break;
  case 2: cout << "Плохо, у вас двойка" << endl; break;
        }
return 0;
}
```

Пример 33

Написать программу – калькулятор, которая будет выполнять для двух чисел a, b, введенных с клавиатуры, следующие математические операции:

1. a+b;
2. a-b;
3. a*b;
4. a/b;

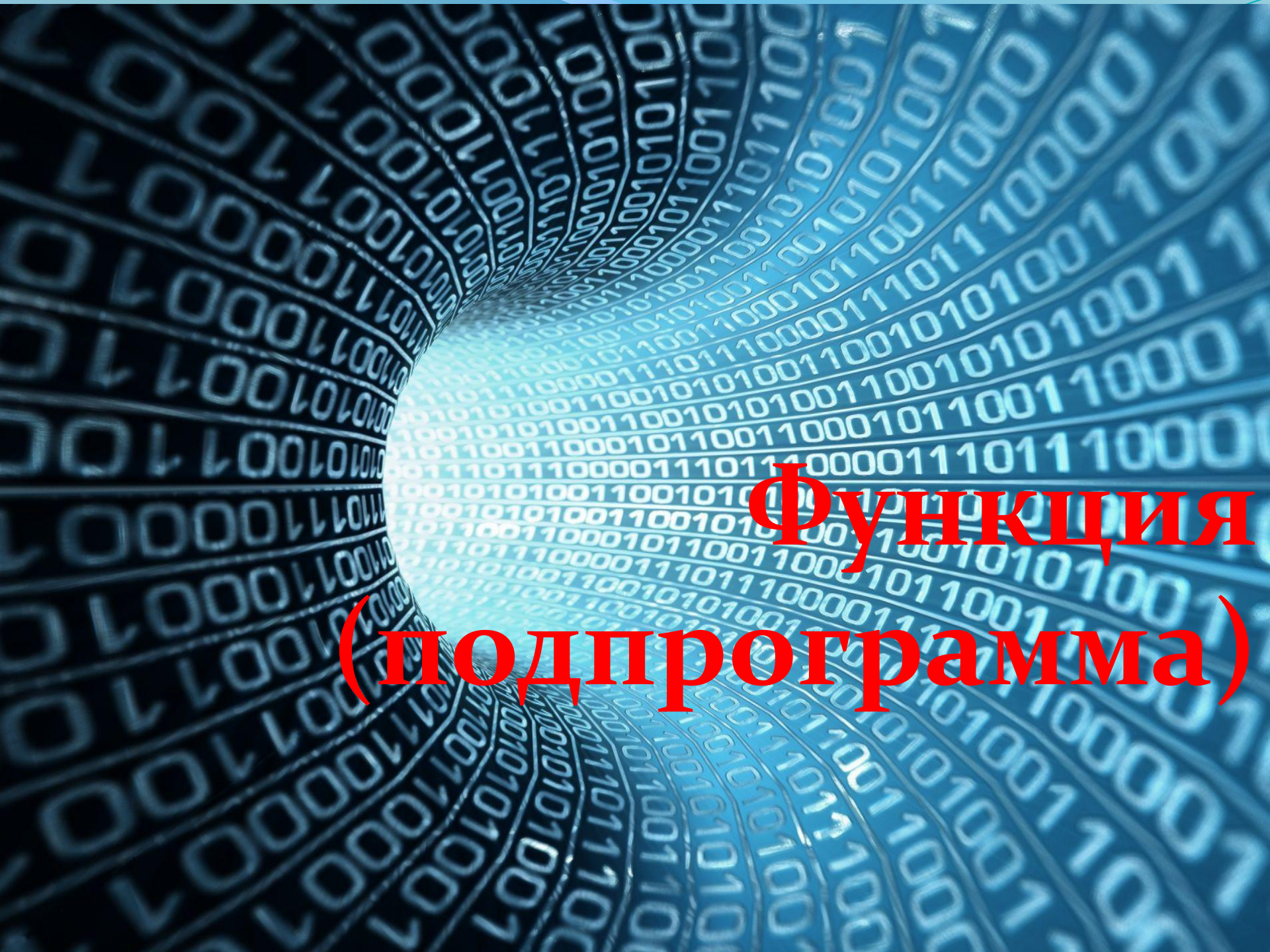
Пример 33. Дописать программу.

```
cout << "Введите число a: " << endl;
cin >> a;
cout << "Введите число b: " << endl;
cin >> b;
cout << "Выберите действие: " << endl;
cout << "1. Сложить a+b" << endl;
cout << "2. Вычесть a-b" << endl;
cout << "3. Умножить a*b" << endl;
cout << "4. Разделить a/b" << endl;
cin >> z;
switch (z)
{
case 1: cout << "a+b= " << a+b << endl;
break;
case 2: cout << "a-b= " << a-b << endl;
break;
case 3: cout << "a*b= " << a*b << endl;
break;
case 4: cout << "a/b= " << a/b << endl;
break;
}
```

Пример 34.

Дополнить пример 33 следующими математическими операциями:

5. b-a;
6. b/a;
7. sqrt (a);
8. sqrt (b);


Функция
(подпрограмма)

Функция (подпрограмма)

Функция (в Fortran – процедура) – подпрограмма , выполняющая действия, многократно повторяющиеся в основной программе. Использование функций сокращает объем программы и уменьшает вероятность появления ошибок в программе.

Функция может иметь возвращаемое значение или не иметь возвращаемого значения.

Функция может быть с параметрами или без параметров. В функции без параметров не указывается возвращаемое значение. Начинается с оператора void.

Код функции может быть размещен в начале программы или в конце программы. Во втором случае в начале программы необходимо объявить функцию с помощью оператора, который называется **прототипом функции.** При этом описание прототипа не отличается от описания заголовка функции.

Правила записи функции с возвращаемым значением.

Заголовок функции

Тип возвращаемого функцией значения

INT
LONG
DOUBLE

Имя функции (список параметров)

В списке параметров перед каждым параметром указывается тип, параметры разделяются запятыми.

Структура функции

```
Заголовок функции
{
операторы;
return выражение
(или переменная);
}
```

Пример:

```
int amp(int x, int y) – заголовок функции
{
x=pow(x,2);
y=pow(y,2);
return x+y;
}
```

тело функции

x, y – формальные параметры

Функция состоит из заголовка и тела функции. Тело функции заключается в фигурные скобки. В прототипе и теле функции задаются формальные параметры. Внутри функции они доступны как локальные переменные.

Фактические параметры существуют в основной программе. Они указываются при вызове функции на месте формальных. В момент вызова функции значения фактических параметров присваиваются формальным параметрам. Имена формальных и фактических параметров могут совпадать, это не вызовет конфликта.

Не рекомендуется создавать функции, обращающиеся к глобальным переменным, так как в этом случае функция становится привязанной к конкретной основной программе.

Соответствие фактических параметров формальным параметрам устанавливается в том порядке, в котором они были объявлены. Тип фактических параметров должен совпадать или быть совместимым с типом формальных параметров. Одна функция не может объявляться или определяться внутри другой т.е. нельзя объявлять и определять функции внутри main. Но одна функция может вызываться внутри другой. В частности, внутри своего тела функция может вызывать саму себя. Такое явление называется рекурсией.

Пример 35

Задание: записать пример 35 без прототипа функции.

```
#include <iostream>
#include <math.h>
using namespace std;
double a, b, z1, z2;
double amp(double x, double y);       // прототип функции «amp»
int main()
{
setlocale (0, "");
 cout << "Введите a: ";
 cin >> a;
 cout << "Введите b: ";
 cin >> b;
z1=a+b;
z2=pow(a,2)+pow(b,2);
cout << amp(z1,z2);   /* z1, z2 фактические параметры,
подставляемые вместо формальных параметров  в функцию amp
*/
 return 0;
}
double amp(double x, double y)       //  функция «amp»
{
x=x+1;
y=y+1;
return x/y;         // x/y - возвращаемое значение
}
```

Пример 36

Задание: записать пример 36 с прототипом функции.

```
#include <iostream>
#include <math.h>
using namespace std;
int x=4;
int y, yy, z;
int aver (int a, int b)    // функция
{
    z=(a+b)/2;
}
int main() {
    y=pow(x,2);
    aver (x,y);
    cout << z << endl;
    yy=pow(x,3);
    aver (y,yy);
    cout << z << endl;
    return 0;
}
```

Формальные параметры

Фактические параметры

Функция с параметрами

```
#include <iostream>
#include <math.h>
using namespace std;
int x=4, y;
double z;
int aver(int a, int b) {
    z=(a+b)/2;
}
int main() {
    y=pow(x,2);
    aver (x,y);
    cout << z << endl;
    return 0;
}
```

Функция без параметров

```
#include <iostream>
#include <math.h>
using namespace std;
int a, b;
double z;
void aver() {
    z=(a+b)/2;
}
int main() {
a=10, b=30;
    aver ();
    cout << z << endl;
a=20, b=40;
    aver ();
    cout << z << endl;
return 0;
}
```

Функция с возвращаемым значением

```
#include <iostream>
#include <math.h>
using namespace std;
int x=4, y;
int aver(int a, int b)
{
    return (a+b)/2;
}
int main() {
        double z;
    y=pow(x,2);
    z=aver (x,y);
    cout << z << endl;
    return 0;
}
```

Функция без возвращаемого значения

```
#include <iostream>
#include <math.h>
using namespace std;
int x=4, y;
double z;
int avrg(int a, int b)
{
    z=(a+b)/2;
}
int main() {
        y=pow(x,2);
    avrg (x,y);
    cout << z << endl;
return 0;
}
```

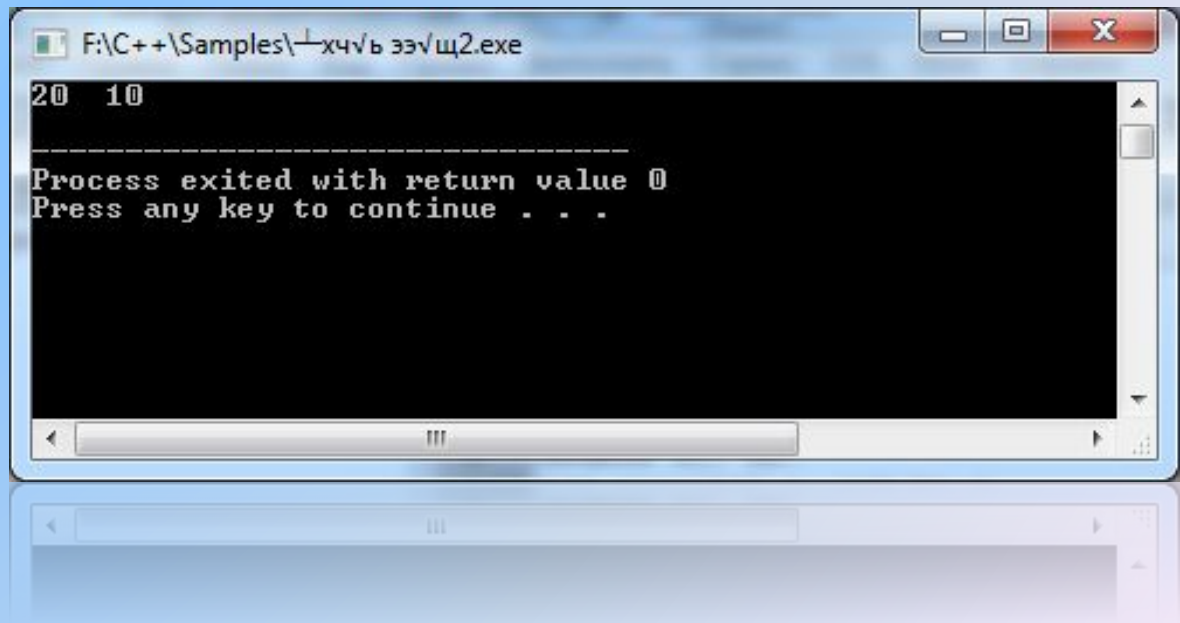
В рассмотренных примерах функция без возвращаемого значения **avrg** становится зависимой от глобальной переменной **z**, что нежелательно. В случае функции с возвращаемым значением **aver** такой зависимости нет, так как все переменные определены внутри функции. Переменная z в этом случае является локальной. В языке C++ локальными являются переменные объявленные внутри функции или блока, ограниченного фигурными скобками.

Пример 37

Задание: записать пример 37 с прототипом функции.

Возвращаемых значений в функции может быть несколько, например,

```
#include <iostream>
#include <math.h>
using namespace std;
int x (4), y;
double z1, z2;
int aver(int a, int b) {
      z1=a+b;
      z2=(a+b)/2;
      return z1, z2;
}
int main() {
      y=pow(x,2);
      aver (x,y);
      cout << z1 << "  " << z2 << endl;
      return 0;
}
```



Примечание: int x(4) означает int x=4;

Правила работы с функцией (имеющей формальные параметры и возвращаемое значение)

Итак, когда мы работаем с функцией, надо указать тип возвращаемого функцией значения, имя функции, задать перечень формальных параметров с указанием типа в круглых скобках после имени функции (точку с запятой после заголовка функции ставить не надо), записать необходимые действия в теле функции, и вернуть результат работы функции оператором return. Оператор return заканчивает выполнение функции и возвращает управление в вызывающую функцию. Если описание функции располагается после main, то перед main необходимо объявить прототип функции.

Пример 38. Необходимо найти путь и расход топлива для автомобиля, который во время поездки двигался с различной скоростью в течение различных временных отрезков.
Пусть скорость автомобиля описывается следующим массивом скоростей:
V[7] = {60,70,80,90,100,110,120} км/час. Этому массиву скоростей соответствует массив временных промежутков:
t[7] = {0.7, 0.65, 0.55, 0.4, 1.2, 1.0, 1.5} часов.
Путь будем определять по формуле $S = V \cdot t$, а расход топлива по формуле $R = 0{,}001 \cdot V^2 \cdot t$, где S[км], V[км/час], t[час], R[л]. Последняя формула учитывает зависимость расхода топлива не только от времени движения автомобиля, но и от его скорости. За исходную величину расхода топлива принимается расход в10 литров на 100 км пути при скорости 100 км/час. При уменьшении скорости расход топлива уменьшается, при увеличении скорости – увеличивается.

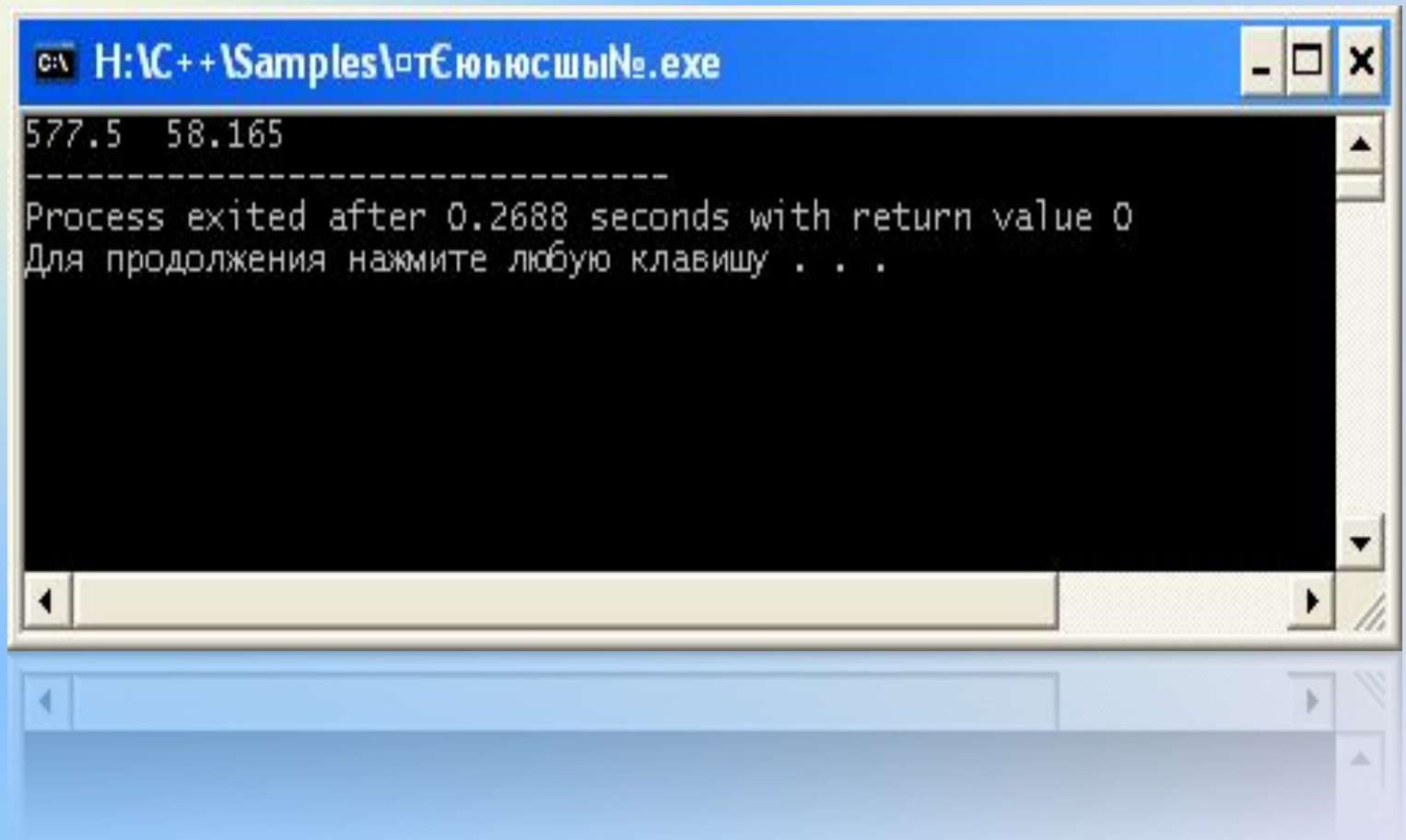
Пример 38 (начало)

```
#include <iostream>
#include <math.h>
using namespace std;
double S=0;
double R=0;
int v[7]={60,70,80,90,100,110,120};  //массив скоростей
double t[7]={0.7,0.65,0.55,0.4,1.2,1.,1.5};  /*массив временных
промежутков, соответствующих массиву скоростей*/
double F1(int a, double b);  /*прототип функции для
вычисления пути */
double F2(int a, double b);  /*прототип функции для
вычисления расхода топлива */
int main()
{
setlocale(0, "");
```

Пример 38 (продолжение)

```
    for (int i=0; i<=6; i++)
    {
    S=S+F1(v[i],t[i]); /* Вычисляем путь. Вместо формальных
параметров a, b подставляем фактические значения скорости
и времени на i-ом отрезке пути */
    R=R+F2(v[i],t[i]);  //Вычисляем расход топлива
    }
    cout << S <<"  "<< R;
    return 0;
 }
double F1(int a, double b) {  //описание функции F1
    return a*b;
}
double F2(int a, double b) {  //описание функции F2
    return 0.001*pow(a,2)*b;
}
```

Пример 38 (результаты расчета)



Задание: записать пример 38 без прототипов функций.

Пример 39. Написать программу для расчета периода колебаний математического маятника.

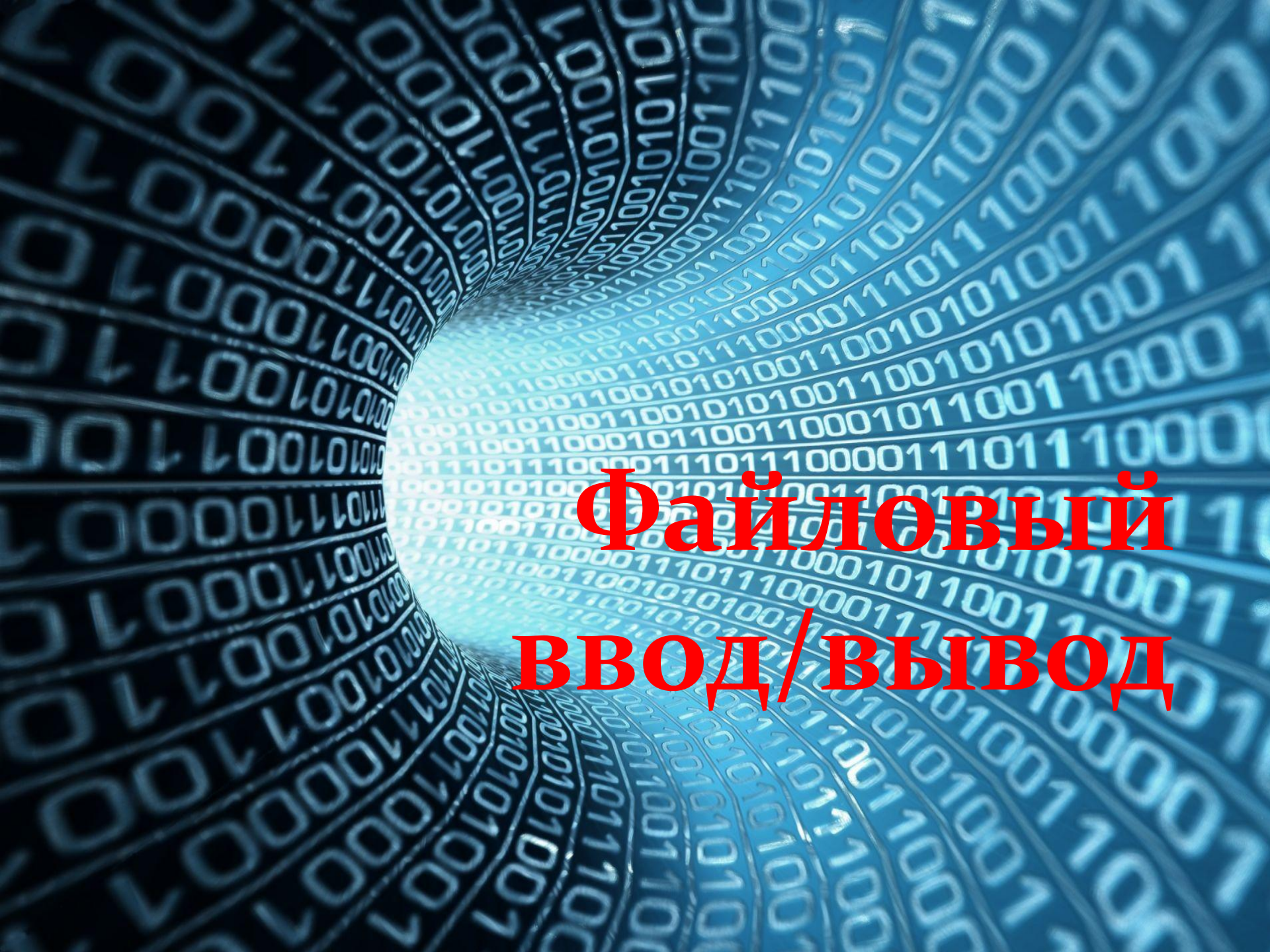
Период колебаний математического маятника определяется по формуле:

$$T = 2\pi\sqrt{\frac{L}{g}}$$

где *L* – длина маятника,
g – ускорение свободного падения.
Получить результаты для длины L1=5 м и L2=10 м.
Для вычисления ***T*** использовать подпрограмму-функцию.

Пример 39

```
#include <iostream>
#include <cmath>
using namespace std;
double PF (int LM)
{
    double pi=3.1416;
    double g=9.81;
    double TM;
    TM=2*pi*sqrt(LM/g);
    return TM;
}
int main()
{
int L=5;
double T1, T2;
T1=PF(L);
cout << "T1= " << T1 << endl;
L=10;
T2=PF(L);
cout << "T2= " << T2 << endl;
return 0;
}
```


Файловый
ввод/вывод

Пример 40. Запись результатов вычисления функции sin(x) в текстовый файл

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
double x;
double dh=3.1416/20;
int main()
{
setlocale (LC_ALL,"Russian");
ofstream f;
f.open ("snx.txt");
for (int i=0; i<=10; i++)
{
x=dh*i;
f << sin(x) << endl;
}
return 0;
}
```

Рассмотрим **пример 40** подробнее:

#include <fstream> //подключить библиотеку fstream

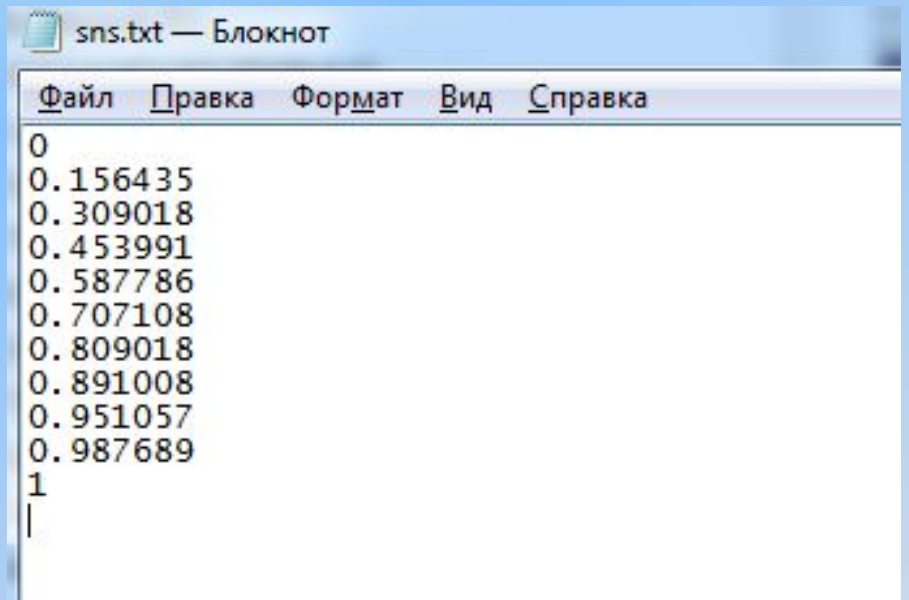
ofstream.f; //output file stream – организовать поток вывода в файл,

f.open («snx.txt"); //открыть текстовый файл snx.txt для записи,

f << sin(x) << endl; //записать значения sin(x) в файл snx.txt

Это четыре минимально необходимых строчки кода программы, чтобы организовать вывод в файл.

Результат вывода имеет вид:



Пример 41. Считывание данных из текстового файла test.txt

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
int main()
{
setlocale (LC_ALL,"Russian");
ifstream f;
f.open ("test.txt");
double a[11];
for (int i=0; i<=10; i++)
{
f >> a[i];
cout << a[i] << endl;
}
return 0;
}
```

Рассмотрим **пример 41** подробнее:

#include <fstream> //подключить библиотеку fstream

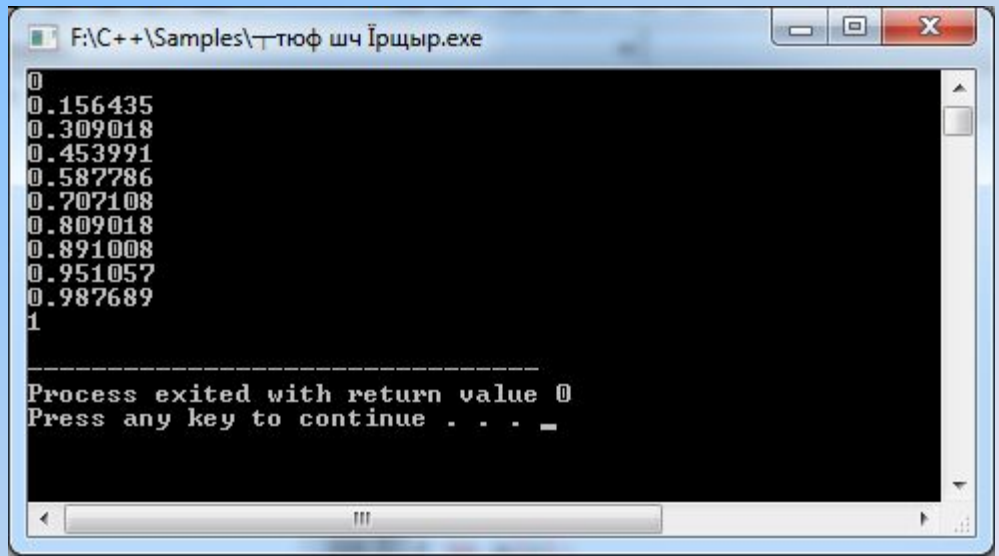
ifstream.f; // input file stream – организовать поток ввода из файла,

f.open ("test.txt"); //открыть текстовый файл test.txt,

f >> a[i]; //записать значения из файла test.txt в массив a[11]

Имеем, как и в предыдущем случае, **четыре** минимально необходимых строчки кода программы, чтобы организовать ввод из файла в программу.

Тестовый файл test.txt заполняем значениями синуса из предыдущего примера. Результат ввода из файла имеет вид:




Построение графиков (C++ & Excel)

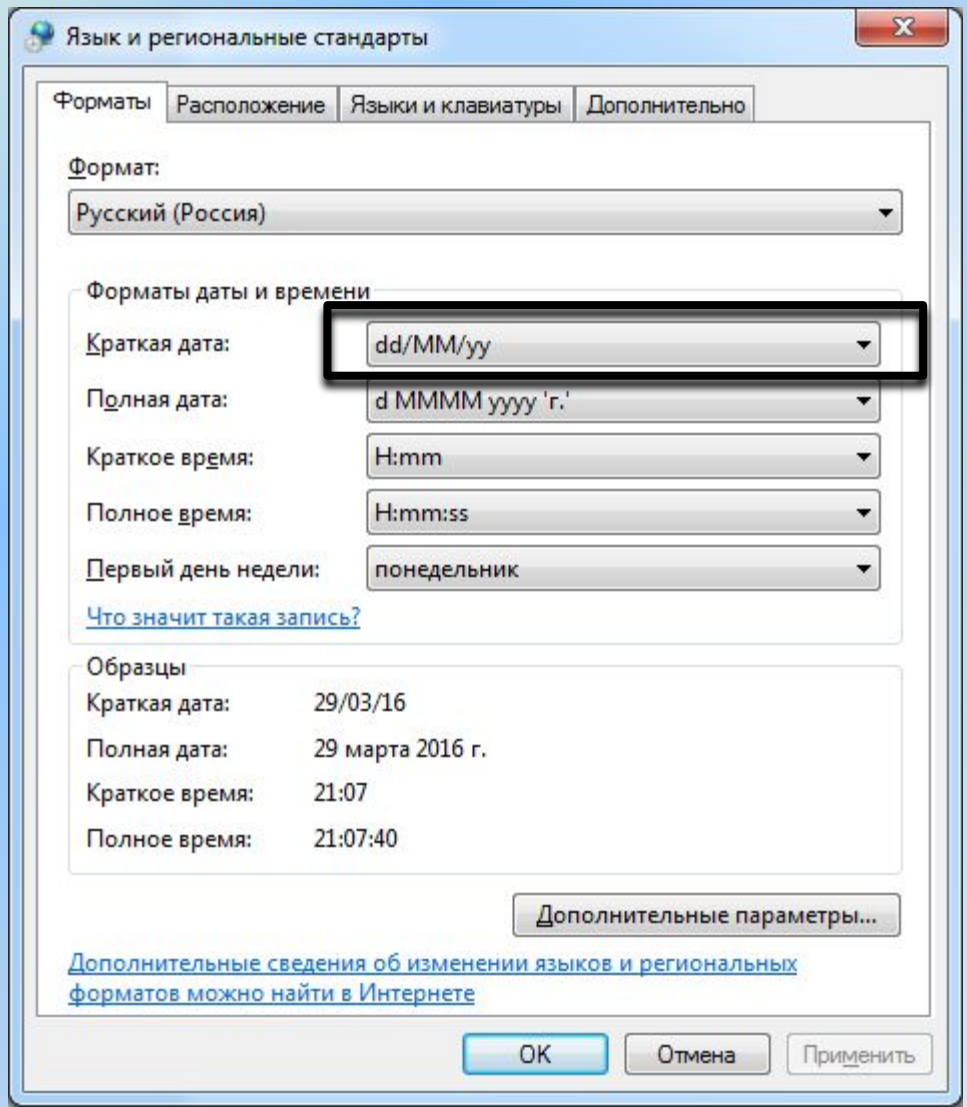
Иногда необходимо не просто получить результаты расчета, но и построить график по результатам расчета. Поведение кривых на построенном графике позволяет оперативно контролировать правильность написанной программы.

Будем проводить расчеты в C++, а график строить в Excel, поскольку Excel имеет более развитый инструментарий для построения графиков и диаграмм, нежели C++. Чтобы автоматизировать перенос результатов расчета из C++ в Excel, результаты расчетов будем выводить в файл, а затем копировать в Excel.

Последовательность действий при построении графика

1. Написать программу, в которой значения аргумента и функции выводятся в файл, а не на консоль. Предусмотреть вывод запятой между аргументом и функцией.
2. Открыть полученный файл в программе Microsoft Office Word.
3. Преобразовать данные в таблицу с помощью опций **Вставка > Таблица > Преобразовать в таблицу**, используя в качестве разделителя данных запятую после аргумента.
4. Скопировать таблицу в Excel и заменить точку в полученных данных на запятую с помощью опций **Главная > Найти и выделить > Заменить.**
5. Построить график в Excel с помощью опций **Вставка > Точечная > Точечная с гладкими кривыми.**

Примечание. Чтобы Excel не переводил автоматически дробное число в дату при копировании из Word в Excel, измените системные настройки даты с «.» на «/» в *Панель управления* > *Язык и региональные стандарты:*



Пример 42. Построить график функции y=sin(x)cos(x)

1)
```
#include <iostream>
#include <fstream> //библиотека потокового ввода/вывода
#include <cmath>
using namespace std;
double trig(double var); // прототип функции trig
int main()
{
setlocale (0, "");
double a, b, h, x;
char s[20]; // строка из 20 символов для имени файла
cout << "Введите начальное и конечное значение
аргумента и шаг: ";
cin >> a >> b >> h;
cout << "Введите имя файла: ";
cin >> s;
ofstream f; //объявляем поток записи в файл
f.open(s); // открываем файл
```

Пример 42 (продолжение)

```
 for (x=a; x<=b; x+=h)
  {f.width(10);
  f << x <<","; /* записываем аргумент x в файл и
одновременно ставим запятую после аргумента, которая
позволит нам преобразовать результаты в таблицу  */
  f.width(15);
  f << trig(x) << endl; /*вызываем функцию с текущим
аргументом x и пишем результаты вычислений в файл */
  }
 f.close(); // закрываем файл
    return 0;
 }
 double trig(double var) {  // функция trig
 return sin(var)+cos(var);
 }
```

2)

Файл с результатами расчетов

```
   0,          1
 0.5,    1.35701
   1,    1.38177
 1.5,    1.06823
   2,   0.493151
 2.5,  -0.202671
   3,  -0.848872
 3.5,   -1.28724
   4,   -1.41045
```

3)

Преобразовываем результаты в таблицу

0	1
0.5	1.35701
1	1.38177
1.5	1.06823
2	0.493151
2.5	-0.202671
3	-0.848872
3.5	-1.28724
4	-1.41045

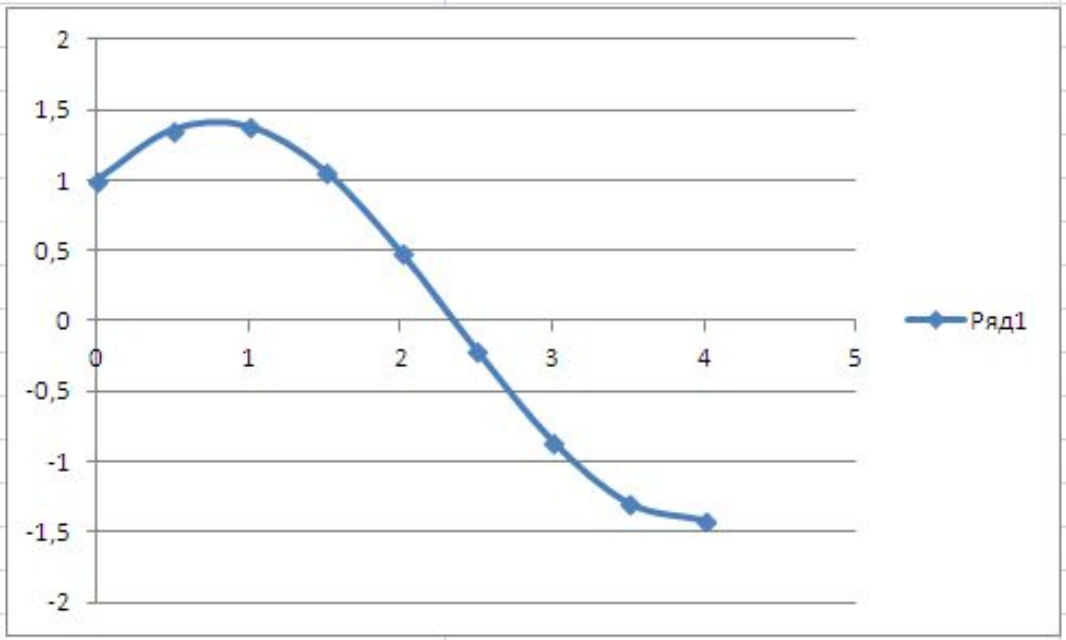
4)

Копируем таблицу в Excel и заменяем разделительные точки в числах на запятые

5)

Строим график

	1	2
1	0	1
2	0,5	1,35701
3	1	1,38177
4	1,5	1,06823
5	2	0,493151
6	2,5	-0,202671
7	3	-0,848872
8	3,5	-1,28724
9	4	-1,41045



Пример 43. При подключении разряженного конденсатора C к источнику напряжения U_0 через резистор R, напряжение на конденсаторе увеличивается от 0 до U_0 по закону

$$U(t) = U_0(1 - e^{-t/\tau})$$

Где $\tau = RC$ постоянная заряда, имеющая размерность [сек].

Написать программу расчета кривой U(t) и построить график этой кривой.
Расчет проведем при следующих исходных данных: R=1 кОм, C=1000 мкФ, время окончания заряда ts=4 с, шаг по времени dt=0.1с.

Пример 43 (начало)

```
#include <iostream>
#include <math.h>
#include <fstream>
using namespace std;
double u0=12;
double tau=1;
double volt(double x); // прототип функции
int main()  {
double t, ts, dt;
setlocale (0, "");
char s[20];
 cout << "Введите время окончания расчета ts: ";
 cin >> ts;
 cout << "Введите шаг dt: ";
 cin >> dt;
 cout << "Введите имя файла ";
 cin >> s;
```

Пример 43 (продолжение)

```
 ofstream f; // вывод в файл
 f.open(s); // открыть файл
 for (t=0; t<=ts; t=t+dt)
 {
   f.width(10);
   f << t << ","; /* Записываем в файл аргумент t и
одновременно пишем после него запятую */
   f.width(15);
   f << volt(t) << endl; /* вызываем функцию volt с текущим
аргументом t и пишем результаты  вычислений в файл*/
}
 f.close(); //Закрываем файл
 return 0;
}
double volt(double x) {
return u0*(1-exp(-x/tau));
}
```

Пример 43 (построение графика)

0	0
0,1	1,14195
0,2	2,17523
0,3	3,11018
0,4	3,95616
0,5	4,72163
0,6	5,41426
0,7	6,04098
0,8	6,60805
0,9	7,12116
1	7,58545
1,1	8,00555
1,2	8,38567
1,3	8,72962
1,4	9,04084
1,5	9,32244
1,6	9,57724
1,7	9,8078
1,8	10,0164
1,9	10,2052
2	10,376

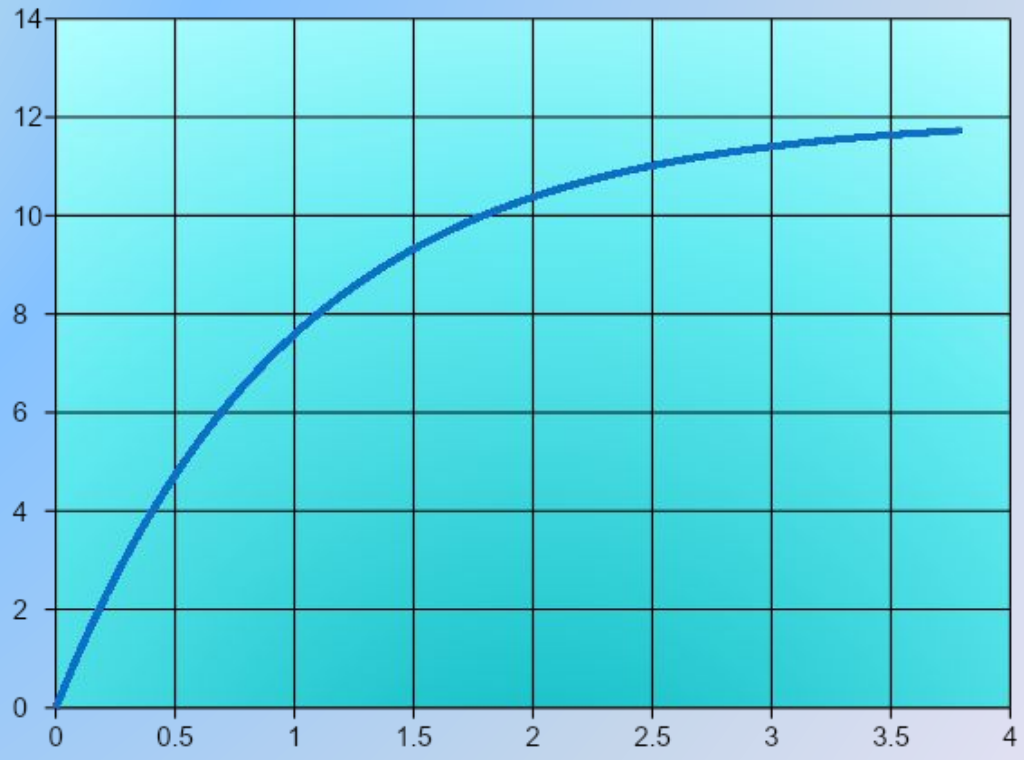


Таблица приведена в сокращенном виде.

Пример 44 (построение траектории)

Мяч падает на пол в точке А со скоростью V1=6 м/с. Угол падения мяча составляет 30 град. от вертикали. При каждом отскоке мяч теряет 30% скорости, т.е. для скорости отскока можно записать V2=0.7*V1. Угол отскока мяча равен углу падения. При втором и каждом последующем падении мяча скорость падения равна скорости предыдущего отскока. Угол падения и угол отскока сохраняют значение 30 град. Написать программу и вывести на монитор траекторию движения мяча при подскоках. Число отскоков задать равным 6.

Высота подъема мяча при отскоке $h = V_{2в}^2/2g$

Время движения мяча между двумя соседними подскоками $t = 2V_{2в}/g$, где $V_{2в}$ - вертикальная составляющая скорости отскока.

Пример 44 (начало)

```
#include <iostream>
#include <cmath>
#include <fstream>     //библиотека для выводы в файл
using namespace std;
double V1=6, g=9.80665, pi=3.14159;
int main()
{
setlocale(LC_ALL, "Russian");
char s[20];
double x, t, tm, dt, V2, Vx, Vy, Vys, h;
int i, j, N;
double xp[7]; // массив координат точек подскока
double tp[7]; //время, соответствующее точкам подскока
cout << "Введите имя файла: ";
cin >> s;
ofstream f;                // указываем поток вывода в файл
f.open(s);                 // открыть файл
tp[0]=0;
xp[0]=0;
```

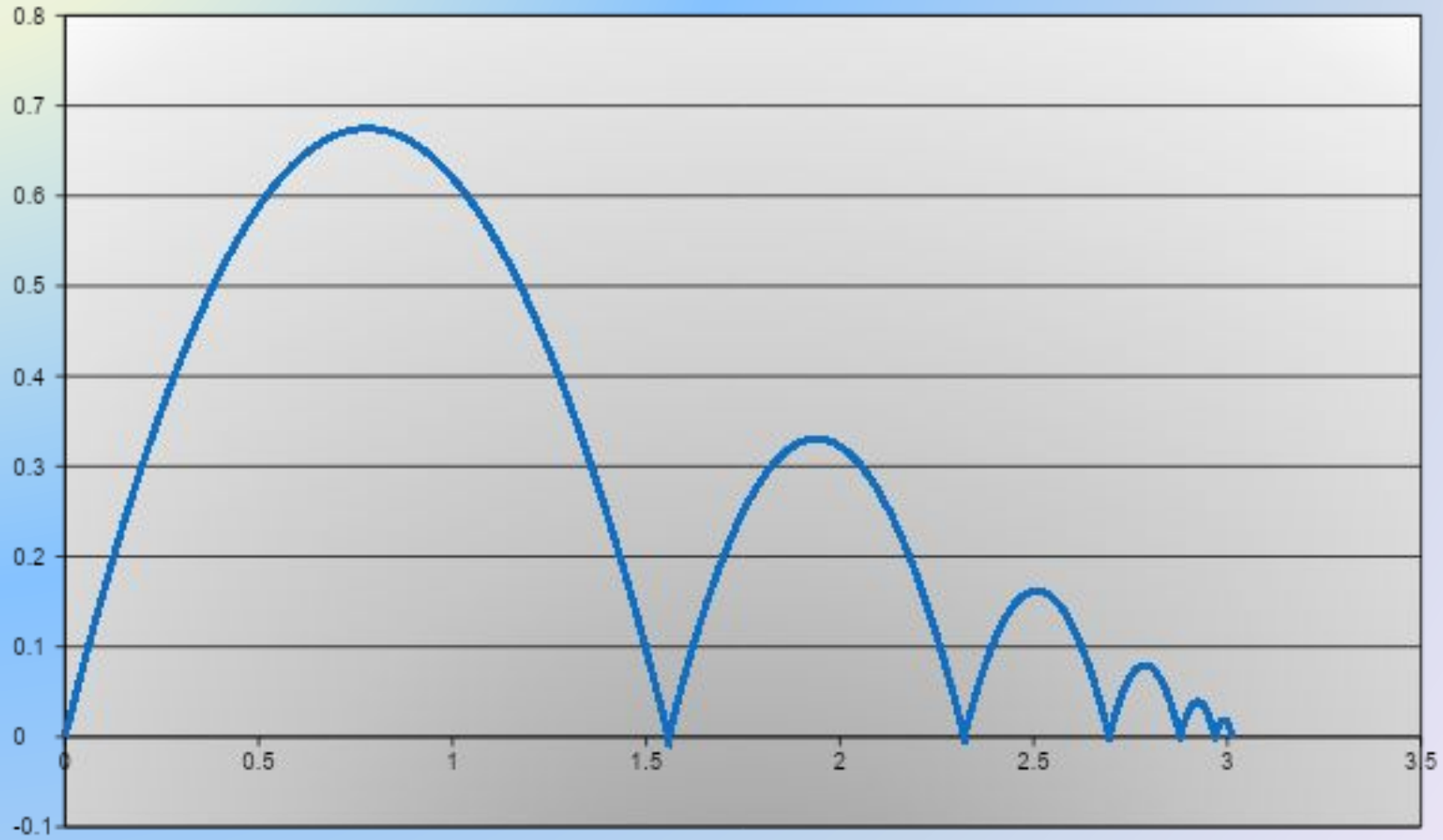
Пример 44 (продолжение)

```
cout << "Введите число N: ";
cin >> N;  // N - число разбиений отрезка между отскоками
for (i=0; i<=6; i=i+1) // цикл по точкам подскока
{
V2=0.7*V1;
Vy=V2*cos(pi/6);
Vx=V2*sin(pi/6);
tm=Vy/g; // время подъема на максим. высоту при отскоке
tp[i+1]=tp[i]+2*tm; //время до следующей точки отскока
xp[i+1]=xp[i]+Vx*2*tm;  //расстояние до след. точки отскока
dt=2*tm/N;              // шаг по времени
for (j=0; j<=N; j=j+1) /*цикл на построение траектории между
соседними точками отскока */
{
if (j==0)
{
h=0;
x=xp[i];
t=tp[i];
Vys=Vy; //запоминание вертик. скорости в предыд. момент
}
```

Пример 44 (окончание)

```
else
{
Vy=Vy-g*dt;
h=h+dt*(Vy+Vys)/2; /* расчет высоты с учетом средней
скорости за промежуток времени dt */
x=x+Vx*dt;
t=t+dt;
Vys=Vy;
}
f << x << ",  ";  //запись координаты x в файл
f << h << endl; //запись высоты, соответствующей коорд. x
}
V1=V2; // переприсвоение скоростей падения и отскока
cout << tp[i] << "  "<< xp[i] << endl;
}
f.close();          //закрываем файл
}
```

Пример 44. Результаты расчета - траектория движения мяча




Задания для
самостоятельной
работы

Написать программы самостоятельно:

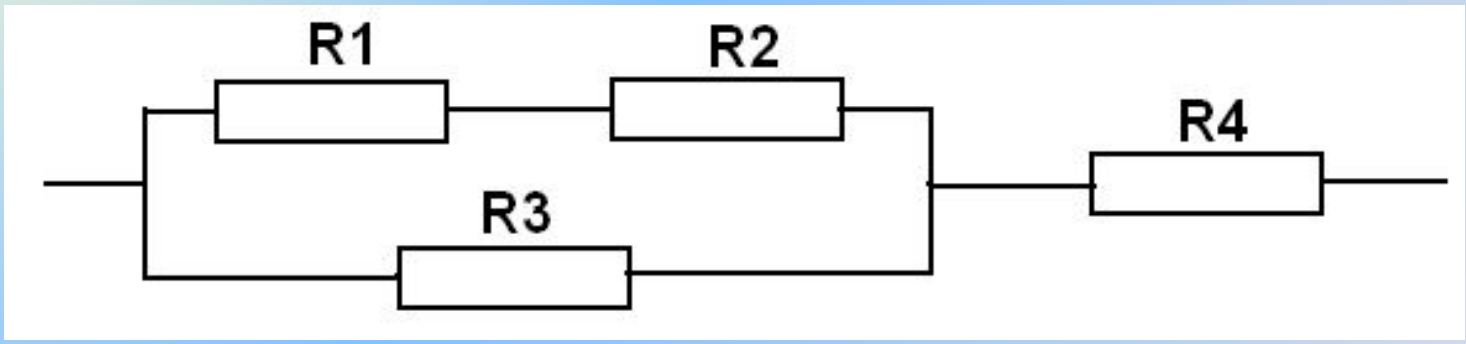
45. Написать программу, выводящую на монитор элементы в диапазоне от 11 до 20 для последовательности чисел 1 3 5 7 9 11 13 15 17 19 21 23 25
46. Написать программу, выводящую на монитор первые 5 элементов последовательности 2 4 8 16 32 64 128
47. Написать программу, вычисляющую факториал натурального числа n, которое пользователь вводит с клавиатуры (n!=1·2·3·4·…·n).
48. Написать программу, переводящую двоичные числа в десятичные:
    1100011111 (ответ: 799),
    11001110001 (ответ: 1649),
    101010111111 (ответ: 2751).

49. Уравнение идеального трансформатора имеет вид

$$\frac{U_2}{U_1} = \frac{N_2}{N_1}$$

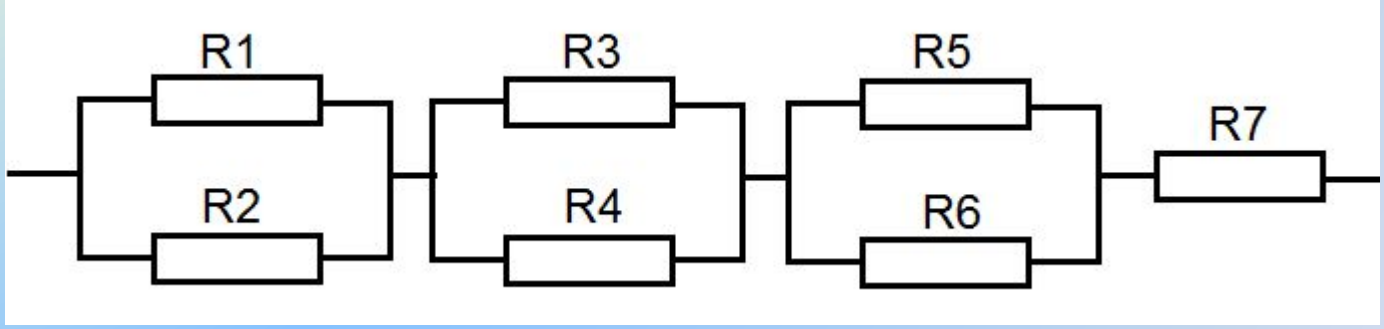
где U_1, U_2 - напряжение в первичной и вторичной обмотках соответственно,
N_1, N_2 число витков в первичной и вторичной обмотках соответственно.
Написать программу для определения напряжения во вторичной обмотке U_2 при известных значениях U_1, N_1, N_2 .
Значения U_1, N_1, N_2 вводить с клавиатуры.

50. Написать программу для расчета сопротивления электрической цепи при параллельно-последовательном соединении резисторов:



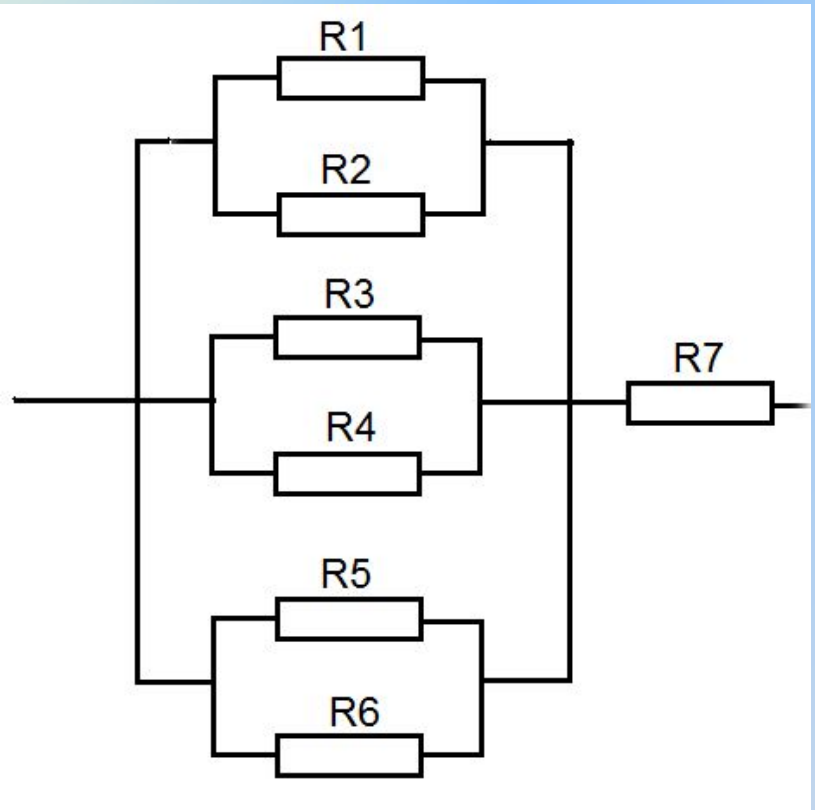
Сопротивления резисторов R1 … R4 вводить с клавиатуры.

51. Написать программу для расчета сопротивления электрической цепи при параллельно-последовательном соединении резисторов:



Сопротивления резисторов R1 … R7 вводить с клавиатуры. Для расчета параллельного участка цепи использовать подпрограмму – функцию.

52. Написать программу для расчета сопротивления электрической цепи при параллельно-последовательном соединении резисторов:



Сопротивления резисторов R1 … R7 вводить с клавиатуры. Для расчета параллельного участка цепи использовать подпрограмму – функцию.

53. Написать программу для вычисления корней квадратного уравнения

$$ax^2 + bx + c = 0$$

Корни квадратного уравнения вычисляются по формуле

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Выражение $b^2 - 4ac$ называется дискриминантом и обозначается буквой D. Если дискриминант больше нуля, то существуют два корня, если D=0, то существует один корень, и если D<0, то действительных корней не существует.

Коэффициенты квадратного уравнения a, b, c вводить с клавиатуры.

54. **Пример на эффект Доплера.** Пусть имеется источник звукового сигнала, который излучает звук частотой f_0 , и приемник, который принимает звуковой сигнал. Частота принимаемого звука зависит от скоростей источника и приемника. Возможны случаи:

Случай 1. Источник сигнала приближается к приемнику со скоростью $V_{ист}$. Тогда частота звука, принимаемого приемником, будет равна

$$f1 = \frac{V \cdot f_0}{V - V_{ист}}$$

Случай 2. Источник сигнала удаляется от приемника со скоростью $V_{ист}$. Тогда частота звука, принимаемого приемником, будет равна

$$f2 = \frac{V \cdot f_0}{V + V_{ист}}$$

Случай 3. Источник сигнала и приемник движутся навстречу друг другу со скоростями соответственно $V_{ист}$ и $V_{пр}$. Тогда частота звука, принимаемого приемником, будет равна

$$f3 = \frac{(V + V_{пр}) \cdot f_0}{V - V_{ист}}$$

Случай 4. Источник сигнала и приемник удаляются друг от друга со скоростями $V_{ист}$ и $V_{пр}$. Тогда частота звука, принимаемого приемником, будет равна

$$f4 = \frac{(V - V_{пр}) \cdot f_0}{V + V_{ист}}$$

Написать программу с оператором **switch** для вычисления частоты звука для рассмотренных случаев. Принять скорость звука V = 340 м/сек. Переменные $\mathbf{f_0}$, $\mathbf{V_{ист}}$, $\mathbf{V_{пр}}$ вводить с клавиатуры.

Написать программы для вычисления значения **y** при следующих исходных данных:

a = 6.5, 7.0, 7.5, 8.0, 8.5, 9.0;

b = 14.3, 15.0, 15.7, 16.4, 17.1

55) $y = a + b$

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

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

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

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

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

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

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

Написать программы для вычисления значения **y** при следующих исходных данных:

α = 10, 20, 30, 40, 50, 60 градусов,

β = 15, 30, 45, 60, 75 градусов

63) $y = sin\ \alpha + tg\ \beta$

64) $y = \sin\alpha + (tg\ \beta)^3$

65) $y = (\sin\alpha)^2 + (\sin\beta)$

66) $y = \sqrt{\sin\alpha} + \sqrt{\cos\beta}$

67) $y = (tg\ \alpha)^2 + \frac{\sin\alpha}{\sin\beta}$

68) $y = \frac{\sin\alpha + \frac{1}{tg\ \beta}}{\sin\alpha + \cos\beta}$

К примеру 63 (вар.1)

```
#include <iostream>
#include <cmath>
using namespace std;
int i,j;
double a, b;                    // углы α и β
double pi=3.1416;
double ar, br, y;
int main()
{
a=0;
for (i=1; i<=6; i++)          // цикл по углу α
{
a=a+10;
ar=(a*pi)/180;                // угол α в радианах
b=0;
      for (j=1; j<=5; j++)    // цикл по углу β
      {
      b=b+15;
      br=(b*pi)/180;          // угол β в радианах
      y=sin(ar)+tan(br);
      cout<<"a="<<a<<"  b="<<b<<"  y="<<y<< endl;
      }
}
return 0;
}
```

К примеру 63 (вар.2)

```
#include <iostream>
#include <cmath>
using namespace std;
double a, b;                              // углы α и β
double pi=3.1416;
double ar, br, y;
int main()
{
for (a=10; a<=60; a=a+10)        // цикл по углу α
{
ar=(a*pi)/180;                        // угол α в радианах
      for (b=15; b<=75; b=b+15)  // цикл по углу β
      {
      br=(b*pi)/180;                 // угол β в радианах
      y=sin(ar)+tan(br);
      cout<<"a="<<a<<"  b="<<b<<"  y="<<y<< endl;
      }
}
return 0;
}
```

Программа варианта 2 короче и более наглядная.

Пример 69

Мяч падает на пол в точке А со скоростью V1=6 м/с. Угол падения мяча составляет 30 град. от вертикали. При каждом отскоке мяч теряет в скорости 30%, т.е. скорость отскока равна V2=0.7*V1, угол отскока мяча равен углу падения. При втором и каждом последующем падении мяча скорость падения равна скорости предыдущего отскока. Угол падения и угол отскока сохраняют значение 30 град.

Написать программу и вывести на монитор максимальную высоту подъема после каждого отскока мяча и расстояние от первоначальной точки падения до текущей точки отскока. Число отскоков задать равным 6.

Высота подъема мяча при отскоке $h = V_{2в}^2/2g$

Время движения мяча между двумя соседними подскоками $t = 2V_{2в}/g$, где $V_{2в}$ - вертикальная составляющая скорости отскока.

Построить графики в диапазоне от -4 до 4

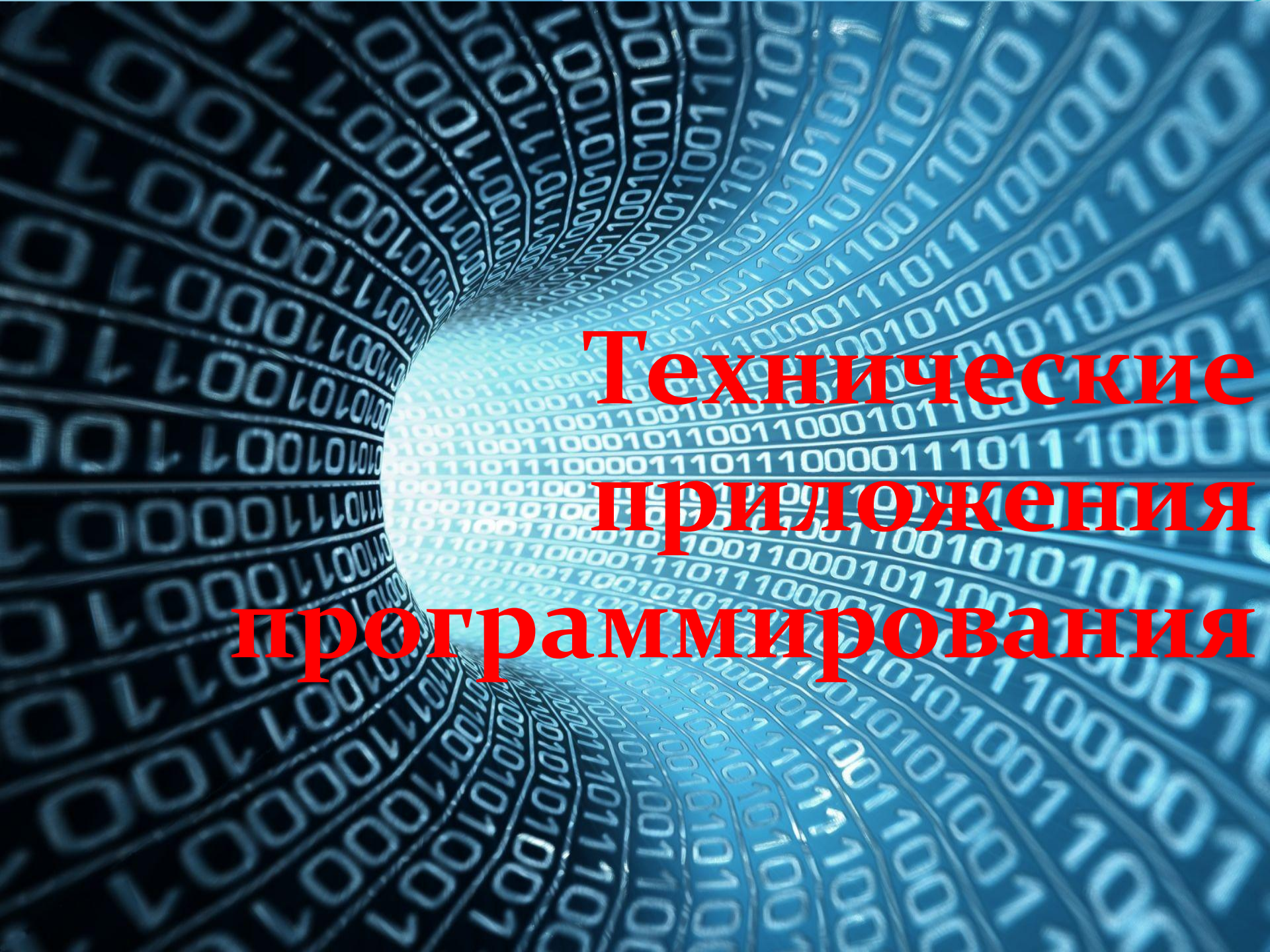
70. $y(x) = 2^{x-1} - 3$

71. $y(x) = \left(\frac{1}{2}\right)^{x} - 3$

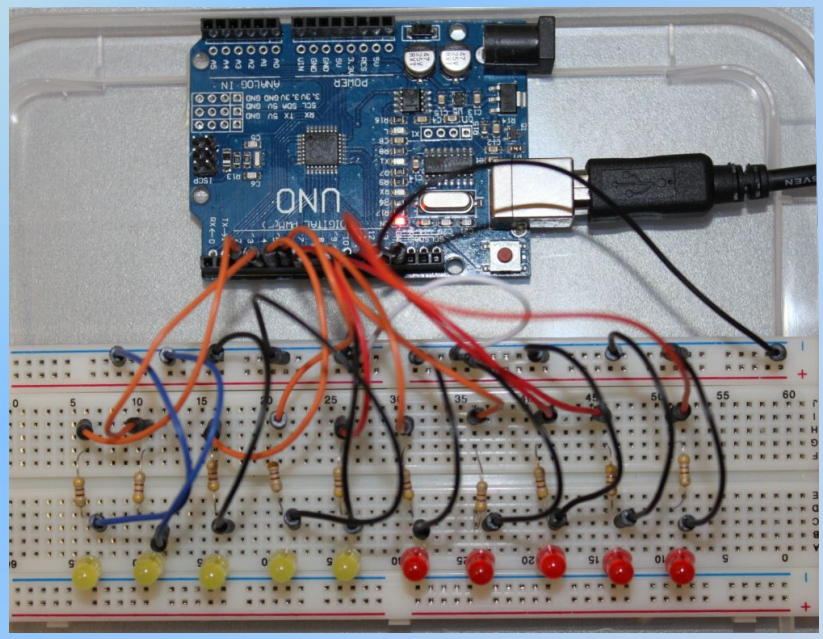
Построить графики в диапазоне от 0 до π

72. $y(x) = 5sinx - cosx + 2cos5x$

73. $y(x) = 2cosx - cos2x$


Технические
приложения
программирования

Пример 74. Рассмотрим, как программирование может пригодиться при создании технического устройства. Техническим устройством, в частности, может служить прибор, управляющий подсветкой различных объектов. На фотографии представлен макет технического устройства, состоящий из микроконтроллерной платы Ардуино и линейки светодиодов.



Программирование микроконтроллерной платы осуществляется с помощью упрощенного аналога C++. Процесс программирования платы подробно описан в литературе [7]. Там же есть описания других устройств на Ардуино.

Программа, управляющая миганием светодиодов, представлена на следующих слайдах.

Пример 74 (начало)

```
#define S1 3
#define S2 4
#define S3 5
#define S4 6
#define S5 7
#define S6 8
#define S7 9
#define S8 10
#define S9 11
#define S10 12
int Led[10]={S1, S2, S3, S4, S5, S6, S7, S8, S9, S10};
int i, j;
void setup()
{
for (i=0; i<=9; i++)
  pinMode (Led[i], OUTPUT);
}
void loop()
{
for (i=0; i<=9; i++) // начальное обнуление
 digitalWrite(Led[i],LOW);
  for (j=1; j<=5; j++) //последовательное мигание светодиодов
 {
```

Пример 74 (продолжение)

```
 for (i=0; i<=9; i++)
    {
      digitalWrite(Led[i],HIGH);
      delay (70);
      digitalWrite(Led[i],LOW);
    }
 }
for (j=1; j<=5; j++) // одновременное мигание линейки
  {
    for (i=0; i<=9; i++)
      digitalWrite(Led[i],LOW);
      delay (500);
    for (i=0; i<=9; i++)
      digitalWrite(Led[i],HIGH);
      delay (500);
  }
for (j=1; j<=5; j++) //переключение между половинами линейки
  {
    for (i=0; i<=4; i++)
      digitalWrite(Led[i],LOW);
    for (i=5; i<=9; i++)
      digitalWrite(Led[i],HIGH);
      delay (500);
        for (i=0; i<=4; i++)
          digitalWrite(Led[i],HIGH);
```

Пример 74 (окончание)

```
 for (i=5; i<=9; i++)
        digitalWrite(Led[i],LOW);
        delay (500);
  }
for (j=1; j<=5; j++) //мигание первой половины линейки
  {
  for (i=0; i<=4; i++)
    digitalWrite(Led[i],HIGH);
    delay (100);
      for (i=0; i<=4; i++)
        digitalWrite(Led[i],LOW);
        delay(100);
  }
for (j=1; j<=5; j++) //мигание второй половины линейки
   {
   for (i=5; i<=9; i++)
     digitalWrite(Led[i],HIGH);
     delay (100);
      for (i=5; i<=9; i++)
        digitalWrite(Led[i],LOW);
        delay(100);
  }
}
```

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)