Начальное программирование на C++. Графики. Технические приложения

(презентация – практикум)

Часть 2

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

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:

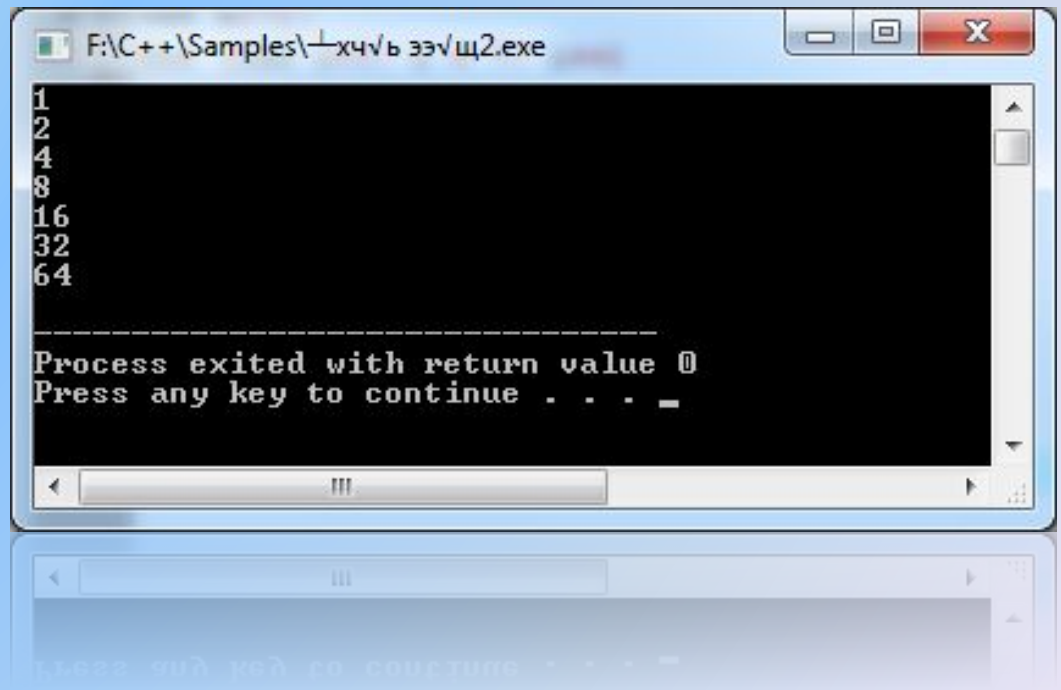
a[0] = 5
a[1] = 10
a[2] = 15
a[3] = 20
a[4] = 25

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;

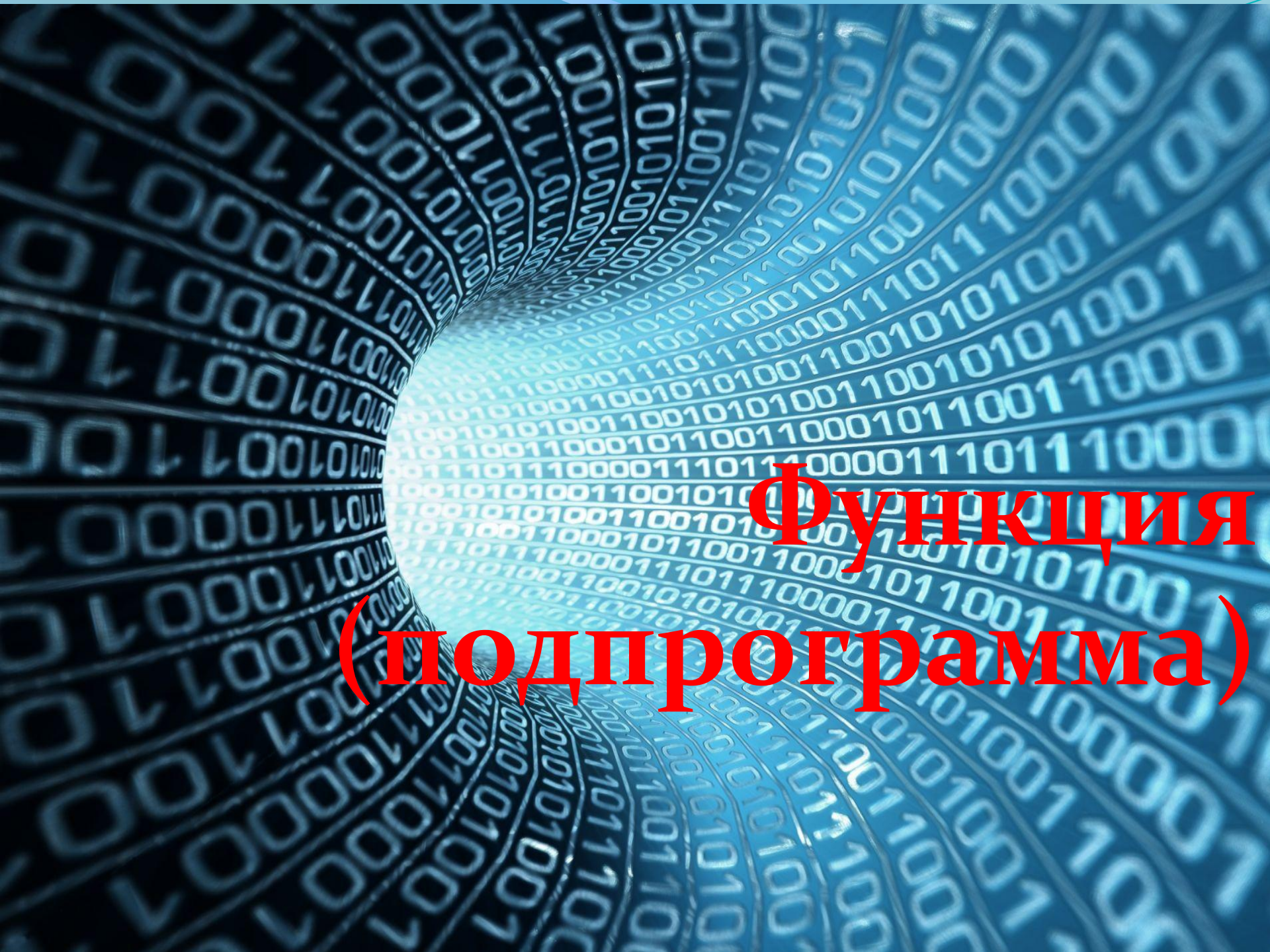
Пример 33. Дописать программу.

```
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;
}
```

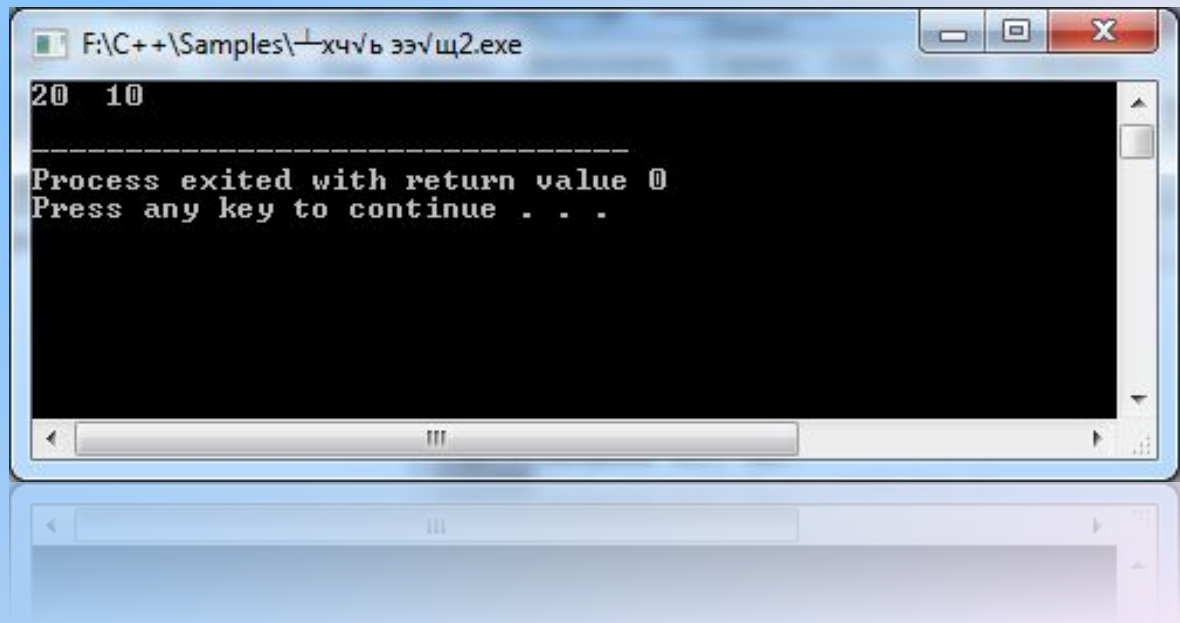
В рассмотренных примерах функция без возвращаемого значения **avrg** становится зависимой от глобальной переменной **z**, что нежелательно. В случае функции с возвращаемым значением **aver** такой зависимости нет, так как все переменные определены внутри функции. Переменная z в этом случае является локальной. 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.

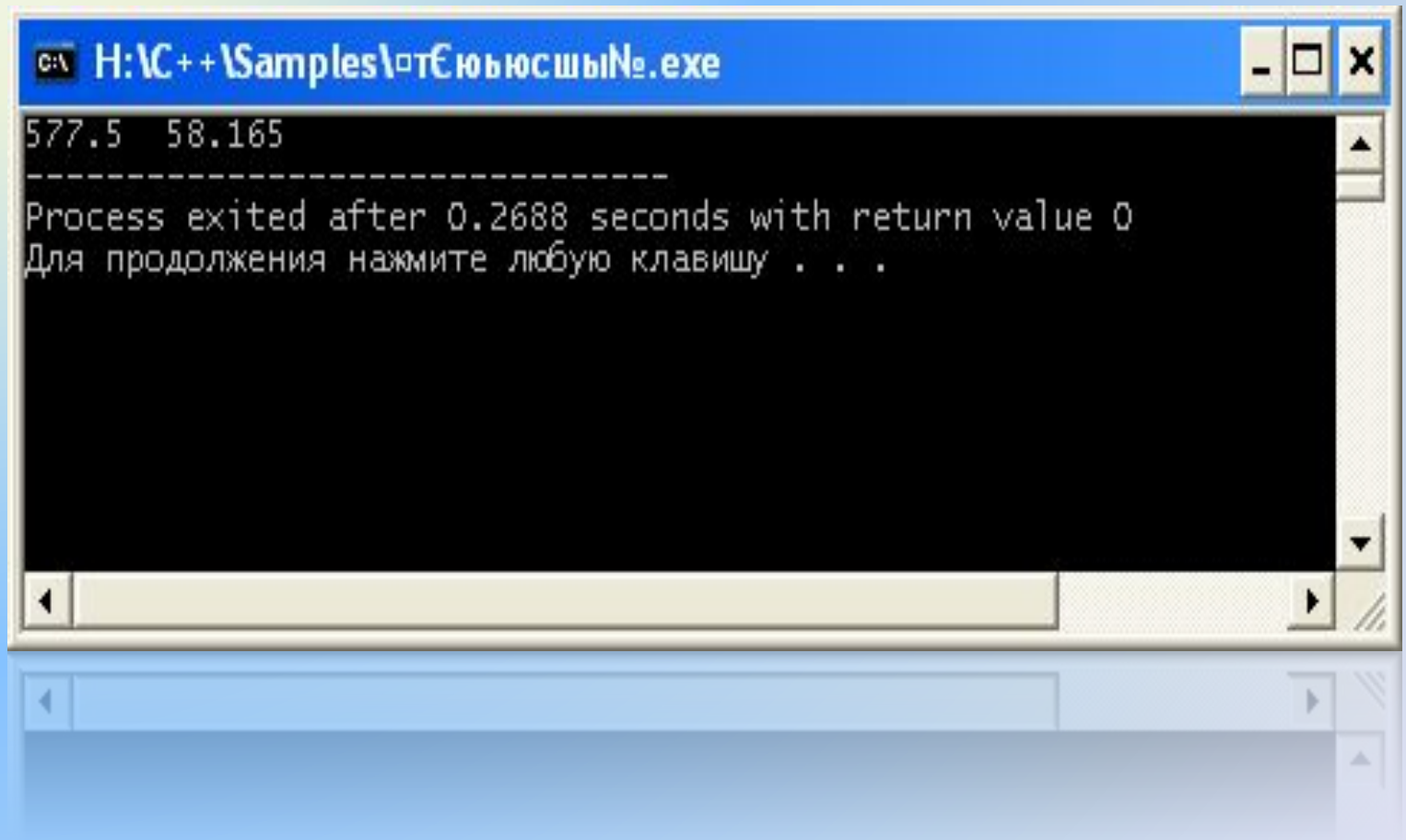
Пример 38 (начало)

```
#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.

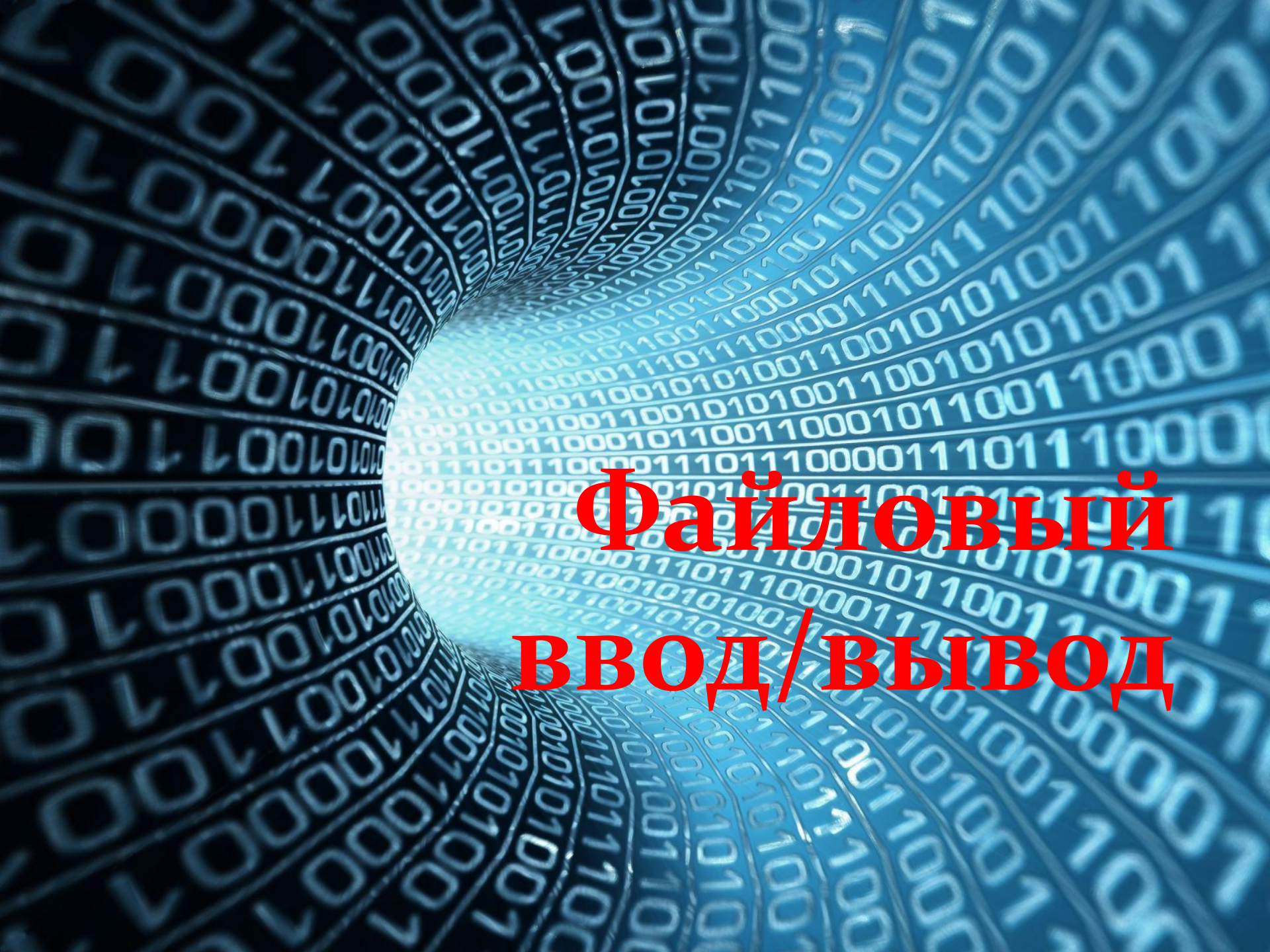
Период колебаний математического маятника определяется по формуле:

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

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

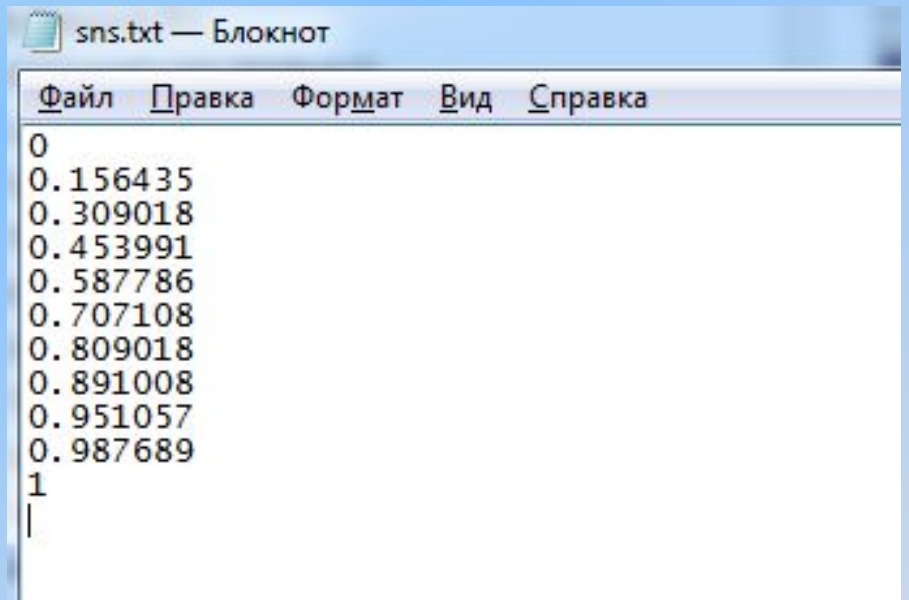
ofstream.f; //output file stream – организовать поток вывода в файл,

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

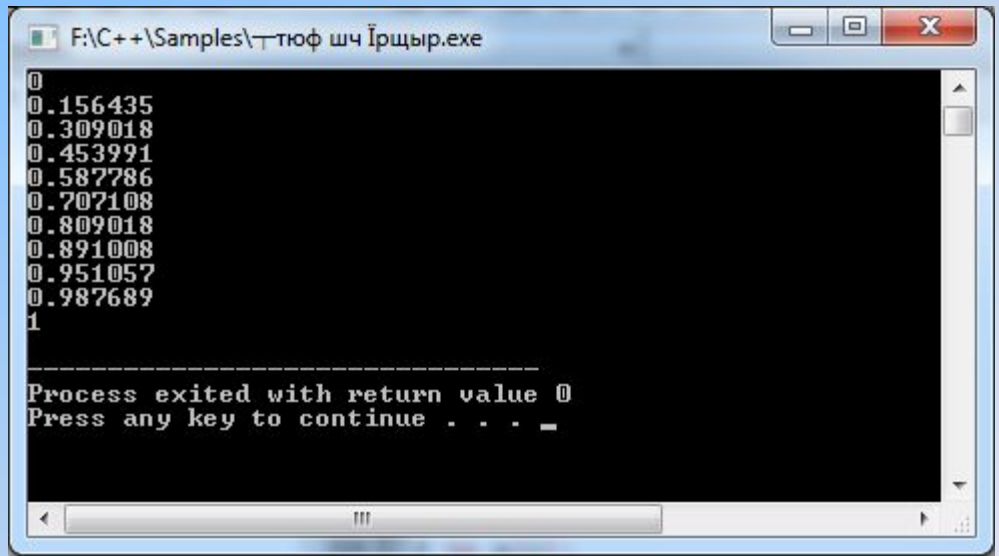
ifstream.f; // input file stream – организовать поток ввода из файла,

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)

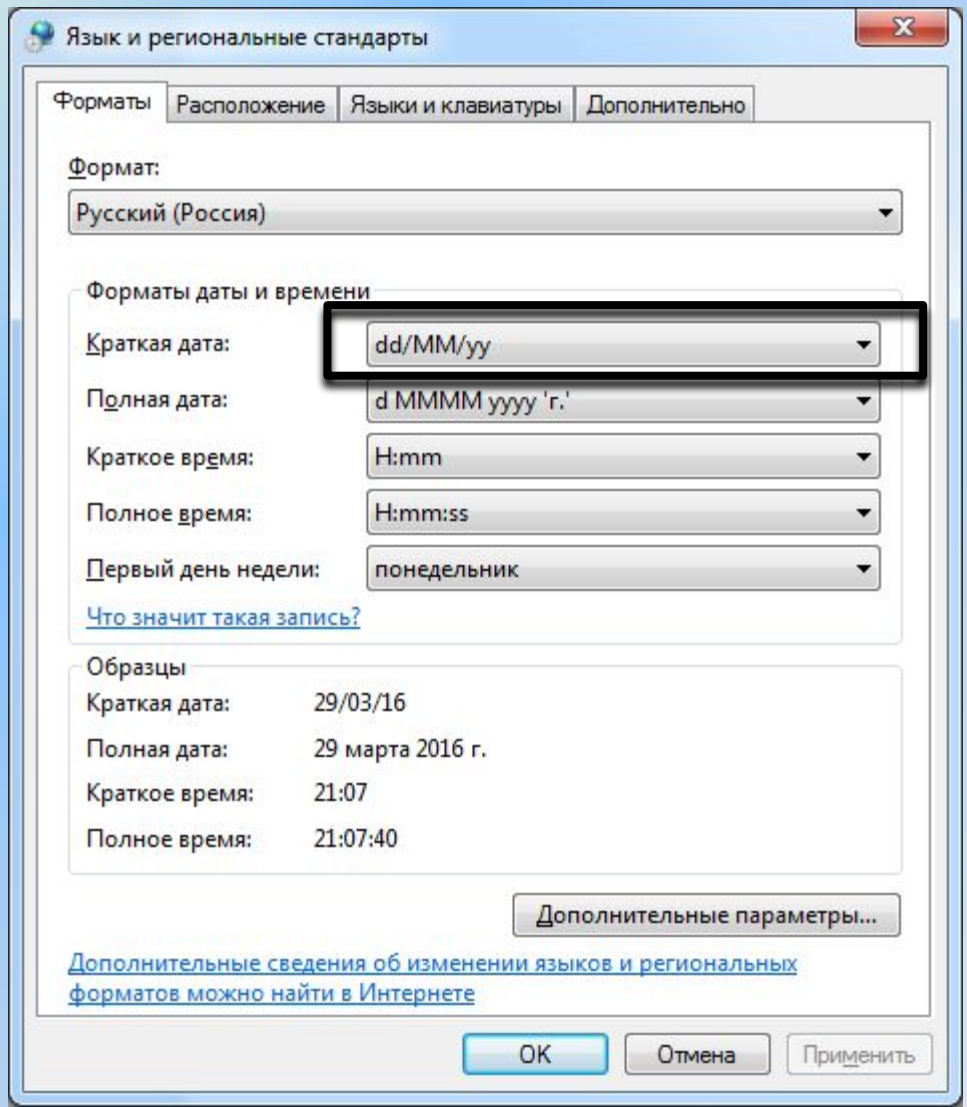
Иногда необходимо не просто получить результаты расчета, но и построить график по результатам расчета. Поведение кривых на построенном графике позволяет оперативно контролировать правильность написанной программы.

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

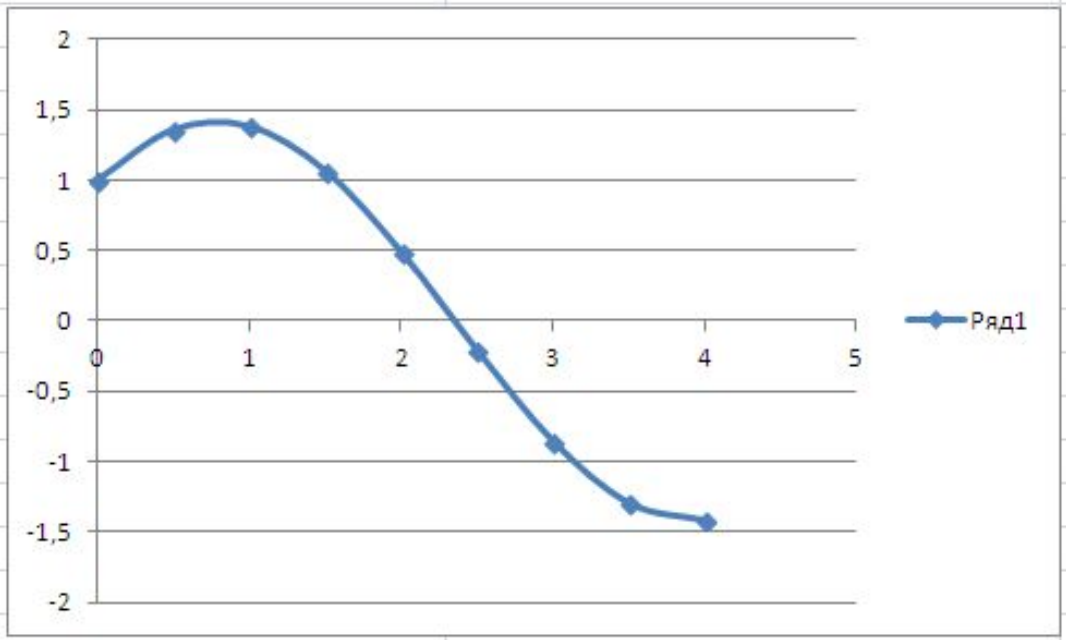
4)

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 С 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

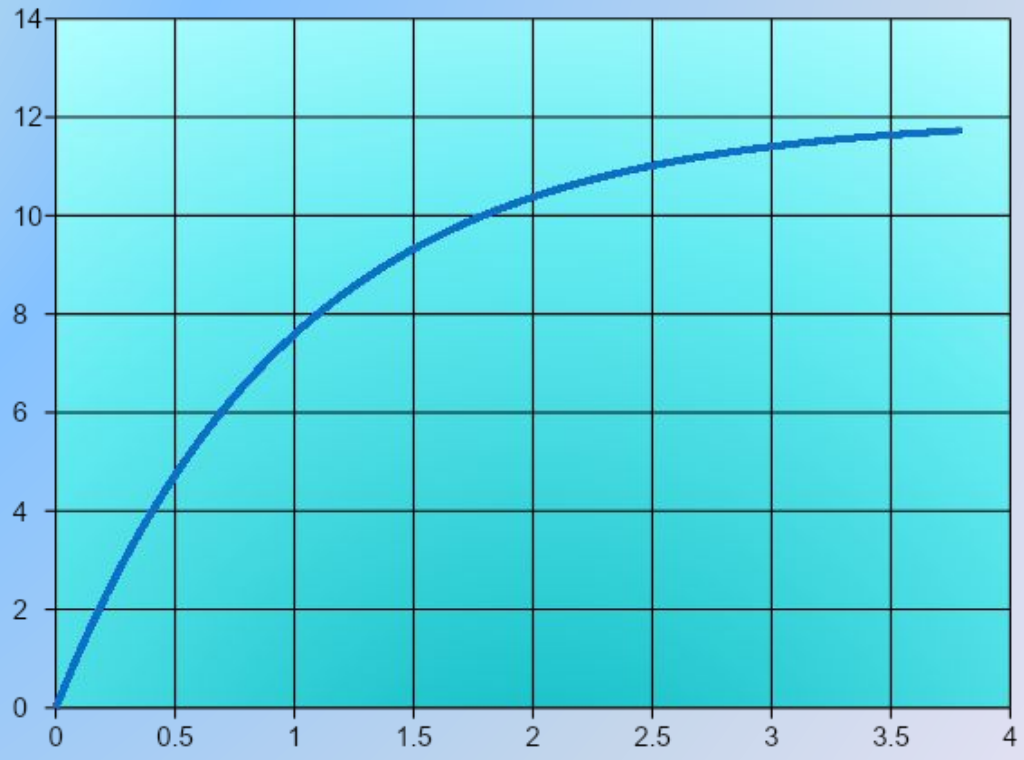


Таблица приведена в сокращенном виде.

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;
```

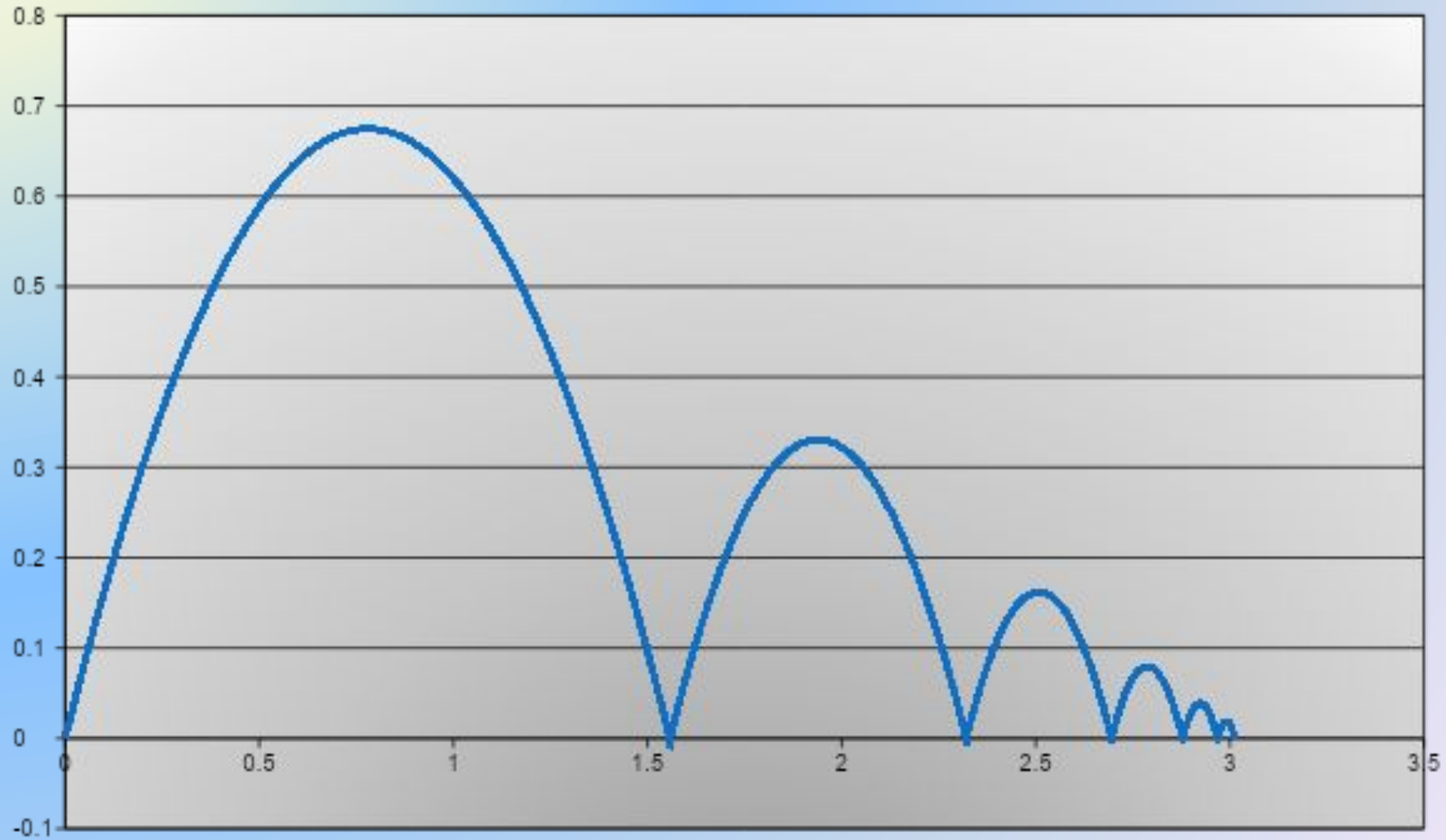
Пример 44 (продолжение)

```
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:

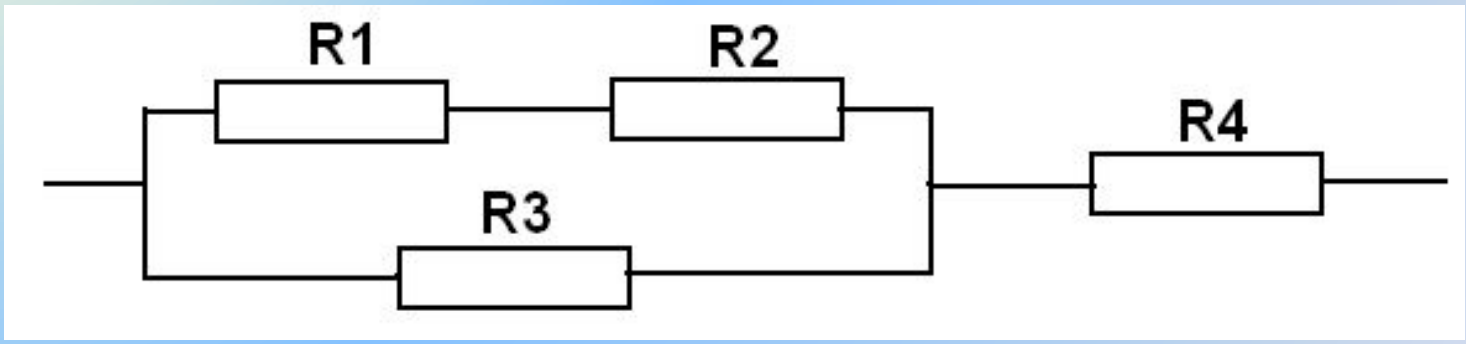
45. Написать программу, выводящую на монитор элементы в диапазоне от 11 до 20 для последовательности чисел 1 3 5 7 9 11 13 15 17 19 21 23 25
46. Написать программу, выводящую на монитор первые 5 элементов последовательности 2 4 8 16 32 64 128
47. Написать программу, вычисляющую факториал натурального числа n, которое пользователь вводит с клавиатуры ($n!=1\cdot2\cdot3\cdot4\cdot\ldots\cdot n$).
48. Написать программу, переводящую двоичные числа в десятичные:
    1100011111 (ответ: 799),
    11001110001 (ответ: 1649),
    101010111111 (ответ: 2751).

49. Уравнение идеального трансформатора имеет вид

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

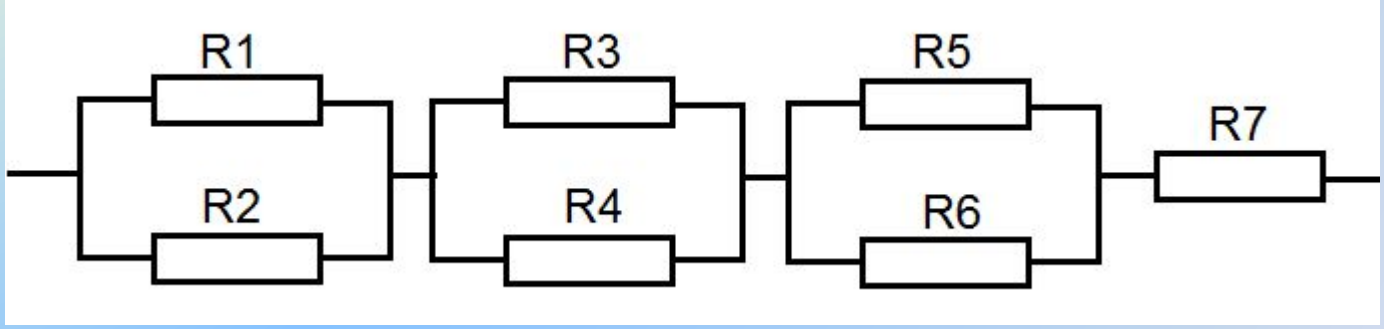
где U_1, U_2 - напряжение в первичной и вторичной обмотках соответственно,
N_1, N_2 число витков в первичной и вторичной обмотках соответственно.
Написать программу для определения напряжения во вторичной обмотке U_2 при известных значениях U_1, N_1, N_2 .
Значения U_1, N_1, N_2 вводить с клавиатуры.

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



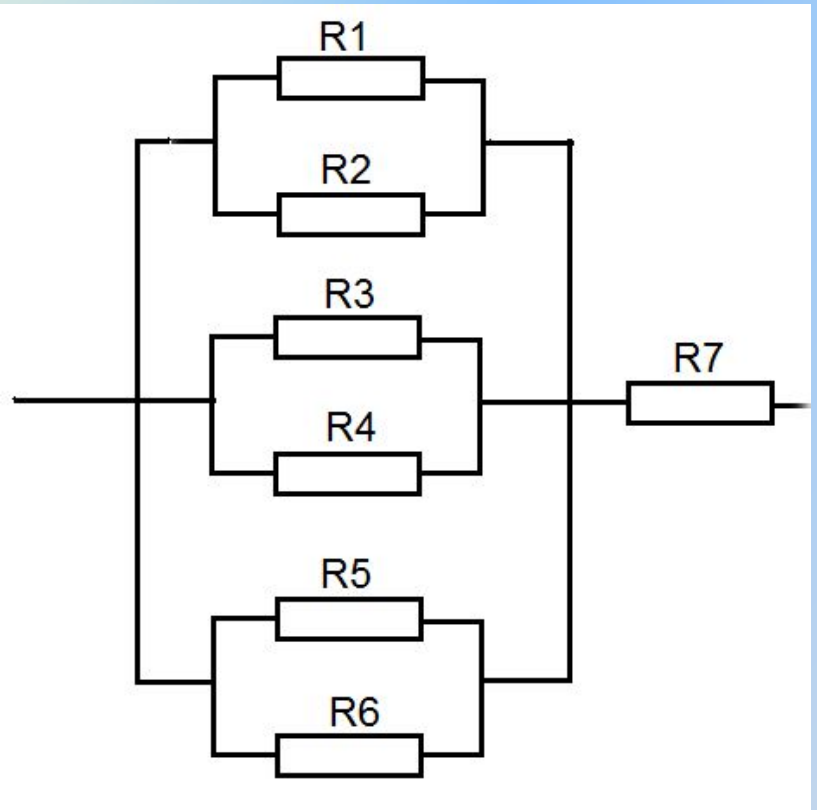
Сопротивления резисторов R1 … R4 вводить с клавиатуры.

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

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

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

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

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

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

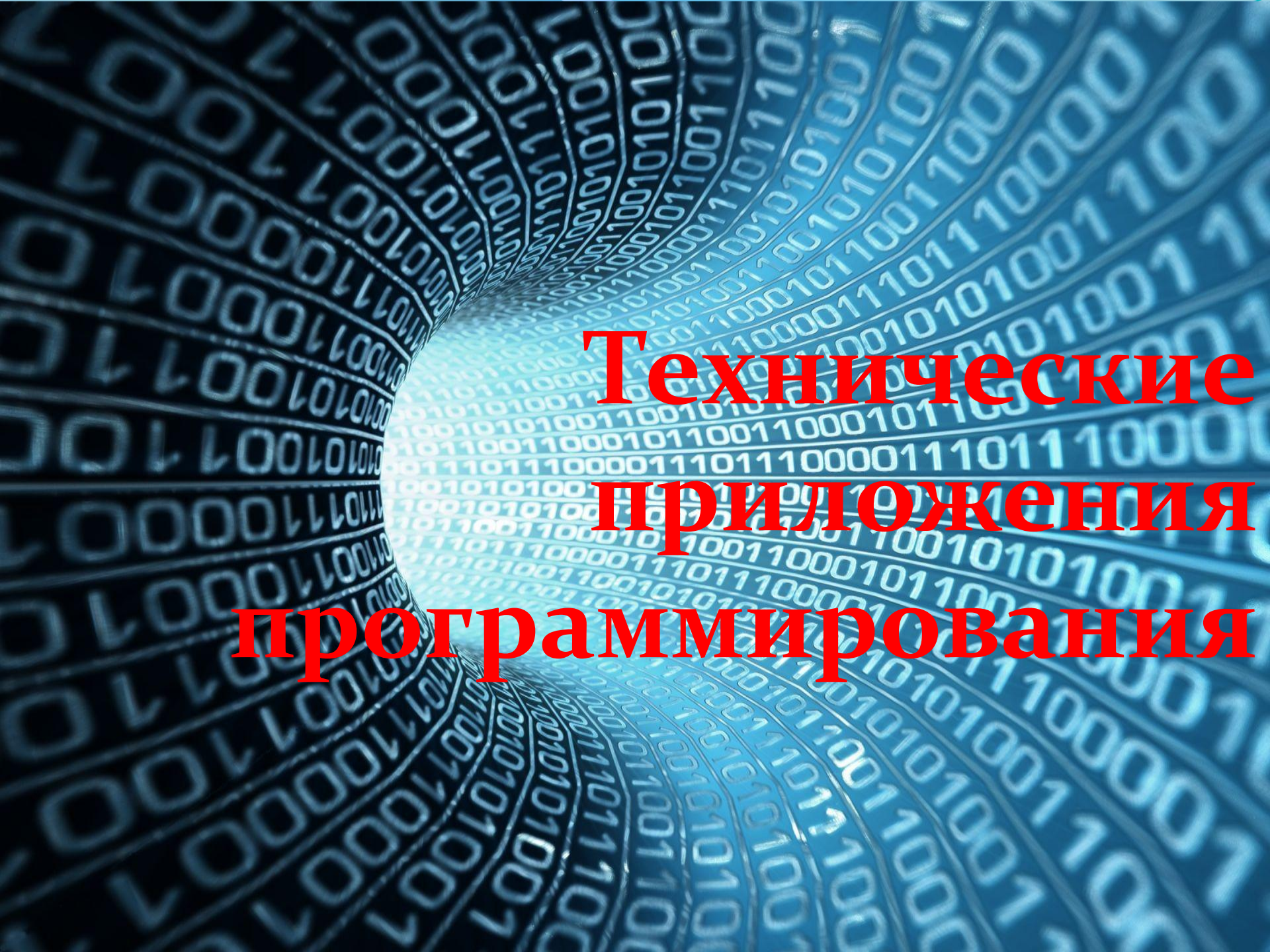
70. $y(x) = 2^{x-1} - 3$

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

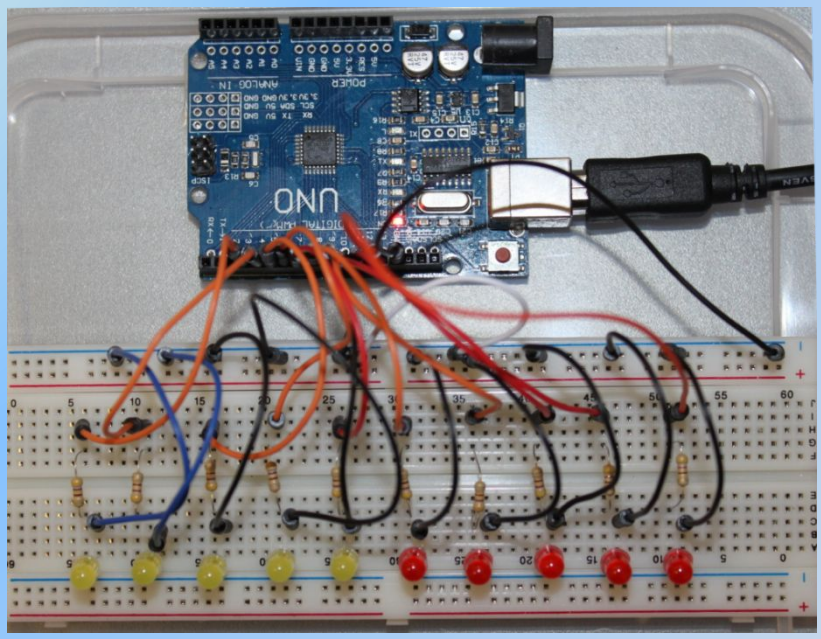
Построить графики в диапазоне от 0 до π

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)

Допускается использование и копирование слайдов при условии ссылки на презентацию.