# Лекция 4

# **Массивы и вектора**

# Определение одномерного массива

**Массив** – это фиксированный набор объектов одного типа, который обозначается одним групповым именем.

**Синтаксис:**

**тип_элементов имя_массива [размерность];**

int m[10]; // m – массив из 10 целочисленных элементов

**Замечания:**

**1.** Тип элементов массива и их количество фиксируется в определении массива. Количество элементов называют **размерностью** массива. Размерность массива должна быть константой целого типа.

```
int n=10;
int m[n];          // ошибка: размерность массива не может быть представлена
                   //переменной величиной
```

**2.** Размерность массива можно задавать с помощью переменной целого типа, определённого ключевым словом const.

```
const int n=10;
int m[n];
```

**3.** В памяти элементы массива располагаются вплотную друг к другу в

# Определение одномерного массива

int m[10];  // m – массив из 10 целочисленных элементов

**Замечания:**

**4.** Элементы массива пронумерованы. Доступ к отдельным элементам осуществляется посредством группового имени и порядкового номера, который называется индексом.

```
m[3]=1;
m[4]=6;
```

**5.** К элементам массива применимы все операции, совместимые с типом, фиксированным в его определении.

```
int i=1;
m[i]=m[2*2]+m[2*i+1];      // m[1]=m[4]+m[3]=7
```

**6.** Нумерация элементов массива начинается с ***нуля***.

```
M[0]=6;    //  Первый элемент
M[9]=7;    //  Последний элемент
M[10]=5;   // Ошибка: элемента с таким
```

номером не существует

**7.** В C++ для встроенных массивов контроля невыхода за границы массива нет, сам программист должен следить за этим.

# Инициализация одномерного массива

Массивы можно ***инициализировать*** при определении, указывая в

фигурных скобках список значений элементов.

**Пример** (массив с количеством дней в каждом месяце):

```
int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

**Замечания:**

**1.** Если размер массива не указан, его длину вычисляет компилятор по списку инициализации. Для приведенного примера это 12.

**2.** Если количество инициализаторов меньше заданного размера массива, оставшиеся элементы будут нулевыми.

**Пример:**

```
int y[6] = {1, 2, 3};
```

Элементы массива y будут иметь следующие значения: y[0] = 1, y[1] = 2, y[2] = 3, y[3] = 0, y[4] = 0, y[5] = 0.

**3.** Задание слишком большого числа инициализаторов

# Определение размера массива

Оператор **sizeof**, примененный к массиву, дает размер массива в

байтах. Например, для массива days

```
int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

выражение sizeof(days) / sizeof(int) равно 12, так как массив состоит из 12 элементов типа int.

Здесь sizeof(days) – размер памяти в байтах, выделенной под массив, sizeof(int) – размер одного элемента массива типа int в байтах.

## Замечание:

Над массивами в целом нельзя выполнять какие-либо действия (присваивать, складывать и т.п.), можно работать только с отдельными элементами массива.

# Циклы работы с массивами

```
const int n=20;      // размерность массива
int m[n];            // объявление массива
```

**Цикл ввода элементов в массив**

```
for(int i = 0; i < n; i++)
{
    cout<<"m["<<i<<"]=";
    cin>>m[i];
}
```

**Цикл вывода элементов из массива**

```
for(int i = 0; i < n; i++)
    cout<<"m["<<i<<"]="<<m[i];
```

**Цикл прохода по элементам массива**

```
for(int i = 0; i < n; i++)
{
Выражения обработки элементов массива;
}
```

# Программа Поиск максимального и минимального элементов массива и вычисления суммы его элементов

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
int x[100] ;          // массив
int n;                // Размер массива
int min, max, sum;
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Задан массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
sum=0;
max=min=x[0];
// Цикл прохода по массиву  элементов
for(int i = 0; i < n ; i++)
{
if(x[i] > max)
max=x[i];
else if (x[i] < min)
min=x[i];
sum+=x[i];
}
cout << "\nМаксимальный элемент:
"<<max;
cout << "\nМинимальный элемент: "<<min;
cout << "\nСумма элементов: "<<sum;
system("pause");
return 0;
}
```

# Программа Проверка упорядоченности массива

Массив называется упорядоченным по возрастанию, если каждый следующий элемент больше предыдущего или равен ему.

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
int x[100] ;          // массив
int n;                // Размер массива
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Задан массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
bool flag = true; // Считаем, что массив
                  //упорядочен по
возрастанию
// Цикл сравнений соседних
элементов
for(int i = 0; i < n - 1; i++)
if(x[i] > x[i + 1]){
flag = false; // Массив не упорядочен
break; // Выход из цикла
}
if(flag) // Вывод результата
cout << "\nМассив упорядочен \n";
else
cout << "\nМассив не упорядочен \n";
system("pause");
return 0;
}
```

# Программа Удаление отрицательных элементов

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
int x[100] ;         // массив
int n;               // Размер массива
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Задан массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
cout<<endl;
//  Проход по элементам массива и поиск
//   отрицательных
for( int i=0;i<n; i++)
if (x[i]<0)  // если отрицательный элемент
{  //  сдвигаем все элементы за ним на
// Цикл сдвига элементов
for (int j = i+1; j < n ; j++)
x[j-1]=x[j];
n--;  // уменьшение размерности массива
i--;  // перепроверка элемента на этом
месте
}
//  вывод преобразованного массива
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
system("pause");
return 0;
}
```

# Программа Вставка элемента в заданную позицию

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
int x[100] ;          // массив
int n;                // Размер массива
int el, poz;          // Элемент и номер позиции
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Задан массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
cout<<endl;
cout << "Введите элемент: \n";
cin>>el;
cout << "Введите номер позиции: \n";
cin>>poz;
// Цикл сдвига элементов справа
налево
for (int j = n-1; j >=poz ; j--)
x[j+1]=x[j];
x[poz]=el;
n++;  // увеличение размерности
массива
//  вывод преобразованного массива
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
system("pause");
return 0;
}
```

# Программа «Сортировка массива по возрастанию»

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
int x[100] ;          // массив
int n;                // Размер массива
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Исходный массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
cout<<endl;
```

```
// Сортировка
for (int i=0; i<n-1; i++)
for (int j=i+1; j<n; j++)
if (x[i]>x[j])
{
int tmp=x[i];
x[i]=x[j];
x[j]= tmp;
}
//  вывод отсортированного массива
for(int i = 0; i < n; i++) // Вывод массива
cout << x[i] << " ";
system("pause");
return 0;
}
```

# Программа Формирование массива без повторений

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{
setlocale(LC_ALL, "Russian");
int x[100] ;        // исходный массив
int y[100] ;        // массив без повторений
int n1;             // Размер исходного
массива
int n2;            // Размер полученного
массива
cout << "Введите количество элементов:
\n";
cin>>n1;
for(int i = 0; i < n1; i++)
{     cout<<"x["<<i<<"]=";
       cin>>x[i];   }
cout << "Исходный массив: \n";
for(int i = 0; i < n1; i++) // Вывод массива
cout << x[i] << " ";
cout<<endl;
// Формирование массива
n2=0;         // Вначале элементов нет
for (int i=0; i<n1; i++) // каждый элемент
 {  // исходного массива ищем в новом
int j=0;
while (j<n2 && x[i]!=y[j])
j++;
if (j==n2)    // если элемент не найден
{
y[n2]=x[i]; // Добавляем новый элемент
n2++;        // Увеличиваем количество
}
}
//  массив без повторений
cout << "Массив без повторений: \n";
for(int i = 0; i < n2; i++) // Вывод массива
cout << y[i] << " ";
system("pause");
return 0;
}
```

# Двумерные массивы

В C++ можно создавать многомерные массивы с несколькими индексами. Многомерный массив рассматриваются как одномерный, элементами которого являются массивы с размерностью на единицу меньше. Двумерные массивы можно представить в виде матриц.

**Синтаксис:**

тип имя[размерность1-число строк][размерность2-число столбцов];

**Пример:**

```
int m[2][3];          //двумерный массив из 2 строк и 3 столбцов,
                      // элементы  массива целые числа.
```

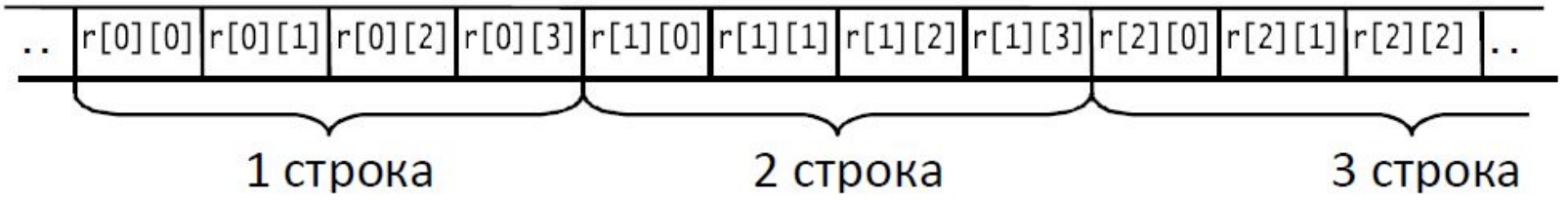
***Доступ к элементам массива осуществляется по двум индексам.***

**Пример:**

```
m[i][j]=1;            //   i – номер строки, j – номер столбца.
```

**Замечания:**

)В оперативной памяти элементы массива располагаются по строкам.




2) Нумерация индексов также начинается с нуля.

# Инициализация двумерных массивов

При определении двумерный массив можно инициализировать с помощью списка констант, заключенных в фигурные скобки.

**Пример:**

```
int m[2][3]={{1,2,3},{4,5,6}};
```

**Замечания:**

1) Заполнение элементов идет по строкам

2) Фигурные скобки, отделяющие строки, могут быть опущены.

3) Список инициализаторов может быть неполным. В этом случае последние элементы инициализируются нулями.

4) При работе с многомерными массивами используются вложенные циклы.

**Пример:**

```
int m[2][3]={1,2,3,4};
```

m[0][0]=1, m[0][1]=2, m[0][2]=3, m[1][0]=4, m[1][1]=0, m[1][2]=0.

# Программа «Сложение матриц»

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{   setlocale(LC_ALL, "Russian");
int a[2][3],b[2][3],c[2][3];
int i,j;
// ввод матрицы a
for (i=0;i<2;i++) // проход по строкам
for (j=0;j<3;j++) // проход по столбцам
{   cout<<"a["<<i+1<<"]["<<j+1<<"]=";
cin>>a[i][j];     }
// ввод матрицы b
for (i=0;i<2;i++) // проход по строкам
for (j=0;j<3;j++) // проход по столбцам
{     cout<<"b["<<i+1<<"]["<<j+1<<"]=";
cin>>b[i][j];    }
// суммирование матриц
for (i=0;i<2;i++) // проход по строкам
for (j=0;j<3;j++) // проход по столбцам
       c[i][j]=a[i][j]+b[i][j];
// вывод матрицы a
cout<<"a"<<endl;
for (i=0;i<2;i++) // проход по строкам
{   for (j=0;j<3;j++) // проход по столбцам
       cout<<a[i][j]<<"    ";
cout<<endl;  } // выделение строки
// вывод матрицы b
cout<<"b"<<endl;
for (i=0;i<2;i++) // проход по строкам
{   for (j=0;j<3;j++) // проход по столбцам
       cout<<b[i][j]<<"    ";
cout<<endl;   }   // выделение строки
// вывод матрицы c
cout<<"a+c"<<endl;
for (i=0;i<2;i++) // проход по строкам
{  for (j=0;j<3;j++) // проход по столбцам
       cout<<c[i][j]<<"    ";
cout<<endl;  }   // выделение строки
system("pause");    return 0;   }
```

# Задача «О предпринимателе»

Пусть некий предприниматель имеет *три* магазина, по которым ведет ежедневный учет выручки от продажи *четырех* видов продуктов (например, чая, сахара, крупы, колбасы). Для учета продаж можно использовать следующий двумерный массив:

**double r[3][4];**

Например:

```
double r[3][4] = {{500, 700, 450, 1000},
                  {600, 710, 480, 1100},
                  {800, 750, 550, 1200}};
```

Пусть предприниматель, о котором речь шла выше, каждый день подводит итоги: вычисляет выручку каждого магазина, выручку от продажи каждого товара и общую выручку за день. Эти расчеты можно выполнить с помощью следующей программы.

# Программа «О предпринимателе»

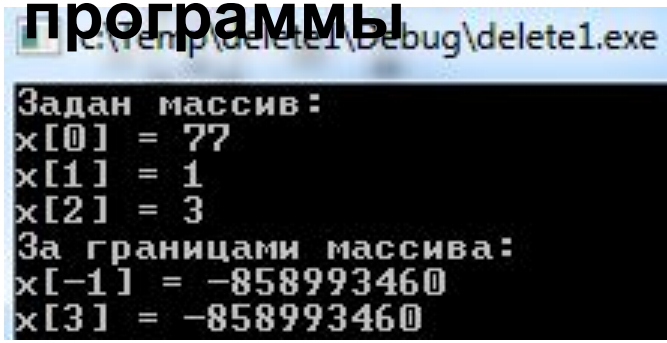
```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{   setlocale(LC_ALL, "Russian");
const int NSHOP = 3; // Число магазинов
const int NGOODS = 4; // Число товаров
double r[NSHOP][NGOODS]; // Двумерный
                         //массив для выручки
double srshop, srg, sum; // Выручка по
                  //магазинам, товарам, общая
cout << "Введите выручку от чая, сахара,
крупы, колбасы \n";
// Ввод данных
for(int i = 0; i < NSHOP; i++){ // Перебор
                                //магазинов
cout << "Магазин " << i + 1 << ": ";
for(int j = 0; j < NGOODS; j++) // Перебор
                                //товаров
cin >> r[i][j]; }   // Ввод выручки i-го магазина
                    // от продажи j-го товара
cout << "Выручка по магазинам: \n";
for(int i = 0; i < NSHOP; i++){ // Перебор
                                //магазинов
srshop = 0;
for(int j = 0; j < NGOODS; j++) // Перебор
                                //товаров
srshop += r[i][j];
cout << srshop << endl;  }
sum = 0;
cout << "Выручка по товарам: \n";
for(int j = 0; j < NGOODS; j++){    // Перебор
                                    //товаров
srg = 0;
for(int i = 0; i < NSHOP; i++) // Перебор
                               //магазинов
srg += r[i][j];
cout << srg << " ";
sum += srg; }     // Подсчет общей выручки
cout << "\nВсего продано за день на: " << sum
<< endl;
system("pause");
return 0;}
```

# Проблемы встроенных массивов

# Программа Выход за границы массива

```
#include <iostream>
#include <locale>
#include <cstdlib>
using namespace std;
int main()
{
setlocale(LC_ALL, "Russian");
int x[] = {77, 1, 3}; // Массив
int n = sizeof(x) / sizeof(int); // Размер массива
cout << "Задан массив: \n";
for(int i = 0; i < n; i++) // Вывод массива
cout << "x[" << i << "] = " << x[i] << endl;
cout << "За границами массива:\n";
cout << "x[-1] = " << x[-1] << endl; // -1 - ошибочный
индекс
cout << "x[3] = " << x[3] << endl; // 3 - ошибочный индекс
system("pause");
return 0;
}
```

**Результат программы**



# Тип данных vector

Более безопасным типом данных, чем массивы являются ***вектора***.

Для того, чтобы использовать вектора надо включить заголовочный файл vector:

```
#include <vector>
```

**Пример:**

```
vector<int> v(3);        // создает вектор из 3-х элементов типа int.
```

Работа с векторами во многом похоже на работу с массивами:

1) обращаться к элементам вектора можно так же как к элементам массива – с помощью индекса, заключенного в квадратные скобки.

2) Нумерация элементов вектора начинается с нуля.

**Особенность:**

При выходе индекса за границы возникает ошибка.

**Замечание:**

Функция size() возвращает текущий размер вектора.

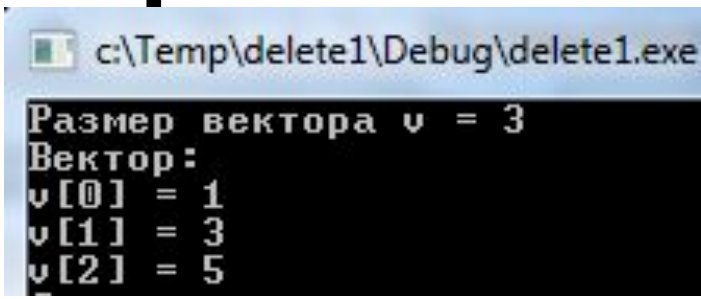
Пример:

```
cout << v.size();                // выведет 3
```

# Программа «Вектора»

```
#include <iostream>
#include <locale>
#include <cstdlib>
#include <vector>
using namespace std;
int main()
{   setlocale(LC_ALL, "Russian");
vector<int> v(3); // Вектор из 3-х элементов
cout << "Размер вектора v = " << v.size() << endl;
for(int i = 0; i < v.size(); i++) // Заполнение вектора
v[i] = 2 * i + 1;
cout << "Вектор:\n";
for(int i = 0; i < v.size(); i++) // Вывод вектора
cout << "v[" << i << "] = " << v[i] << endl;
//cout << "v[-1] = " << v[-1] << endl; // -1 - недопустимый
индекс
//cout << "v[3] = " << v[3] << endl; // 3 - недопустимый индекс
system("pause");     return 0;}
```

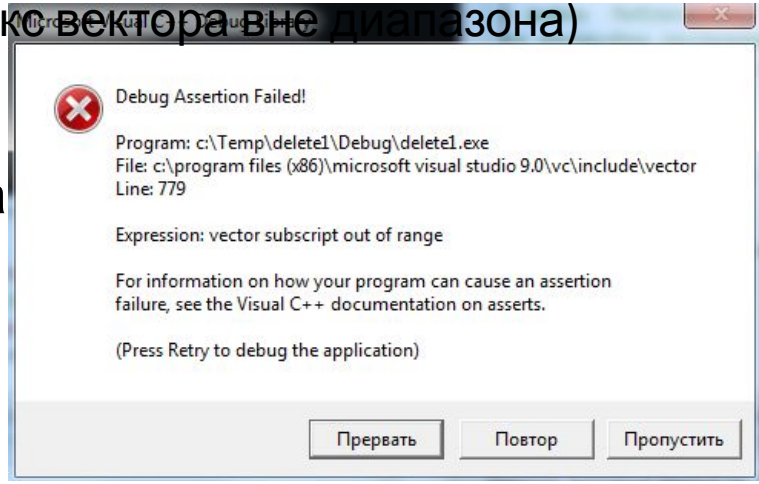
**Результат программы**




Если убрать комментарии – сообщение об ошибке:

vector subscript out of range

(индекс вектора вне диапазона)



# Возможности векторов

Вектора можно создавать из элементов любых типов. Если при создании вектора не указывать число элементов, создается пустой вектор, не имеющий элементов.

```
vector<double> v;  // Пустой вектор
v[0] = 3.5;        // Ошибка, вектор v не имеет элементов
```

1) Функция push_back() добавляет в конец вектора новый элемент.

```
v.push_back(3.5);
v.push_back(7.9);
```

Теперь вектор v содержит элемент v[0] со значение 3.5 и элемент v[1] со значение 7.9.

2) Функция pop_back() удаляет из конца вектора элемент.

```
v.pop_back();
```

Теперь вектор v содержит только элемент v[0] со значение 3.5.

3) Текущий размер вектора можно изменить функцией resize(). Новые элементы заполняются нулями.

```
v.resize(5);       // Теперь размер v равен 5
```

4) Вектора можно присваивать, подобно одиночным переменным, чего нельзя делать массивами.

# Программа Размеры и копирование векторов

```
#include <iostream>
#include <locale>
#include <cstdlib>
#include <vector>
using namespace std;
int main()
{  srand(time(0));              // Инициализация
                    //генератора случайных чисел
setlocale(LC_ALL, "Russian");
const double Max = 1000.0;
vector<double> v;           // Пустой вектор
v.push_back(3.5);       // Добавление в вектор
v.push_back(7.9);         // элементов
cout << "Вектор v размера " << v.size() <<
endl;
for(int i = 0; i < v.size(); i++) // Вывод
cout << v[i] << " "; // вектора
v.resize(5); // Теперь размер v равен 5
cout << "\nТеперь вектор v имеет размер " ;
cout<< v.size() << endl;
for(int i = 0; i < v.size(); i++)         // Вывод
cout << v[i] << " ";                      // вектора
vector<double>w; // Еще один пустой
вектор
cout << "\nРазмер вектора w " << w.size() <<
endl;
w = v; // Присваивание векторов
cout << "Теперь вектор w - копия вектора
v\n";
for(int i = 0; i < w.size(); i++)
cout << w[i] << " ";
for(int i = 0; i < v.size(); i++) // Заполнение
                                  //вектора v
v[i] = rand() / Max; // случайными числами
cout << "\nВектор v заполнен случайными
числами\n";
for(int i = 0; i < v.size(); i++)
cout << v[i] << " ";
cout << endl;
system("pause");
return 0;
}
```

# Программа Удаление отрицательных элементов

```
#include <iostream>
#include <locale>
#include <vector>
#include <cstdlib>
using namespace std;
int main()
{     setlocale(LC_ALL, "Russian");
vector<int> v ;          //  пустой вектор
int n;                   // Размер вектора
int x;                   // вводимый элемент
cout << "Введите количество элементов:
\n";
cin>>n;
for(int i = 0; i < n; i++)
{     cout<<"v["<<i<<"]=";
       cin>>x;
        v.push_back(x);   }
cout << "Задан вектор: \n";
for(int i = 0; i < v.size(); i++) // Вывод вектора
cout << v[i] << " ";
cout<<endl;
```

```
//  Проход по элементам вектора и поиск
//   отрицательных
for( int i=0;i<v.size(); i++)
if (v[i]<0)  // если отрицательный элемент
{  //  сдвигаем все элементы за ним на
    //   одну позицию влево
// Цикл сдвига элементов
for (int j = i+1; j < v.size() ; j++)
v[j-1]=v[j];
v.pop_back();  // уменьшение размерности
                    //вектора
i--;  // перепроверка элемента на этом
месте
}
//  вывод преобразованного вектора
cout << "Преобразованный вектор: \n";
for(int i = 0; i < v.size(); i++) // Вывод вектора
cout << v[i] << " ";
cout<<endl;
system("pause");
return 0;
}
```