Лекция 7

ОПЕРАЦИИ С МНОГОМЕРНЫМИ МАССИВАМИ В VISUAL C++

Цель лекции. Изучить способы и особенности работы с двухмерными массивами в Visual C++ .

Основные вопросы лекции.

1.Двухмерные массивы данных и их инициализация в среде Visual C++.
2. Консольный ввод и вывод двухмерных массивов в среде Visual C++.
3. Присваивание и копирование двухмерных массивов в Visual C++.
4. Поиск элемента в двухмерных массивах в Visual C++.
5. Перестановка и сортировка элементов в двухмерных массивах.

1. Двухмерные массивы данных и их инициализация в Visual C++

Под размерностью массива понимают число индексов, которое необходимо указать для получения доступа к отдельному элементу массива. Массивы, рассмотренные в лабораторной работе № 5, например, были одномерными и требовали только одного индекса. Массивы с более чем одной размерностью, называются многомерными.

Самым простым многомерным массивом является двухмерный массив (матрица).

При объявлении массива указывается тип элементов массива, имя массива и его размер:

Тип ИмяМассива [Размер 1] [Размер 2];

Например, оператор

int A[3][8];

описывает целый двухмерный массив по имени **A** из 24-х целых чисел. В памяти будет зарезервировано место для 24-х целочисленных элементов массива (рис. 1).

В памяти компьютера двухмерный массив располагается непрерывно по строкам, то есть

A[0][0], A[0][1], A[0][2], A[0][3] … A[0][8], A[1][0], A[1][1], A[1][2], A[1][3] … A[1][8].

На рис. 1 приведена схема размещения элементов массива из 24-х целых чисел по имени **A** размерностью 3×8. В памяти будет зарезервировано место для 24-х целочисленных элементов массива, которые располагаются в **непрерывном (!!!)** блоке памяти.

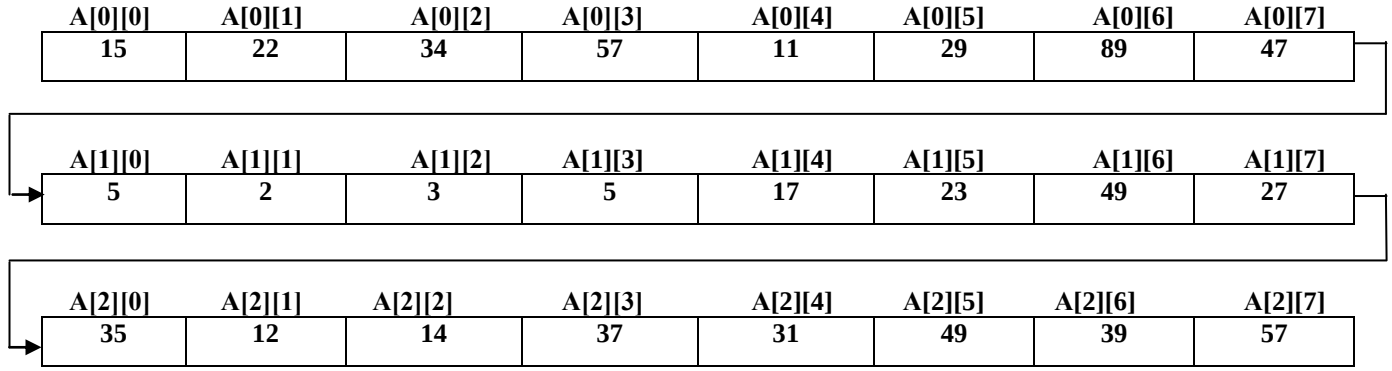


Рис. 1. Значения и расположение в памяти элементов массива **A[3][8]**

Массивы хранятся в памяти компьютера так, что самый правый индекс измеряется быстрее всего.

Возможна инициализации массива непосредственно в программном коде. В этом случае в фигурных скобках приводятся значения элементов массива (рис. 1) в следующем виде:

int A[2][3]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

или

int A[2][3]= {{1, 2, 3, 4} {5, 6, 7, 8} {9, 10, 11, 12}};

Количество инициализаторов не обязано совпадать с количеством элементов массива. Если инициализаторов меньше, то оставшиеся элементы не определены.

Двухмерный массив, например, **int A[3][4]** можно представить в виде следующей матрицы:

Первый индекс – это номер строки в массиве, второй индекс – номер столбца.

В памяти компьютера элементы такой матрицы разместятся в таком порядке:

$$\mathbf{A_{00},\ A_{01},\ A_{02},\ A_{03},\ A_{10},\ A_{11},\ A_{12},\ A_{13},\ A_{20},\ A_{21},\ A_{22},\ A_{23}.}$$

Кроме этого, инициализация массивов возможна в процессе выполнения программы – путем записи данных в отведенные для массивов ячейки памяти.

При работе с массивами, в т. ч. двухмерными, целесообразно использовать оператор цикла **for**, т.к. известен размер обрабатываемого массива (число элементов массива), т. е. число повторений цикла.

В языке С++ не проверяется выход индекса за пределы массива. Если массив **m[100]** целочисленный массив:

int m[100]; ,

а в программе указано

x=m[200]; ,

то сообщение об ошибке не будет, а переменной **x** будет присвоено произвольное значение.

При обработке массивов в Visual C++ 2010 все действия в программе выполняются над элементами массива (!), а не над массивом в целом. При этом индекс элемента может быть задан либо его значением, либо выражением:

A[4], F[i+k+1]; .

Над массивами можно выполнять следующие действия:

1. Вводить массивы в память компьютера.
2. Выводить массивы на экран дисплея, на другое устройство или в файл.
3. Присваивать определенные значения элементам массивов.
4. Копировать массивы.
5. Переставлять элементы массивов.
6. Сортировать элементы массивов.

2. Консольный ввод и вывод двухмерных массивов в среде Visual C++ 2010

Т. к. все действия необходимо выполнять над элементами двухмерных массивов, то для ввода двухмерного массива в память компьютера необходимо организовать его поэлементный ввод посредством оператора цикла **for** (т. к. известен размер массива).

Для консольного вывода двухмерного массива также надо с помощью оператора цикла **for** организовать поэлементный вывод исследуемого массива.

Пример 1. Исследовать способы консольного ввода и вывода двухмерных массивов. Необходимо ввести с клавиатуры матрицу **A** размерностью **M**×**N** в память компьютера и вывести эту информацию на дисплей.

Решение. Для придания программе большей универсальности зададим размерность исходной матрицы с запасом, а реальная размерность будет вводиться в каждом конкретном случае. Число строк обозначим **m**, а столбцов - **n.**

Поэлементный ввод матрицы **A** организован при помощи двух операторов цикла **for** – внешнего и внутреннего (вложенного). Внешний цикл организует перебор элементов матрицы **A** по строкам, а вложенный – по столбцам.

Вывод исходной матрицы на экран будет производиться аналогичным образом.

Блок-схема алгоритма решения данной задачи приведена на рис. 2.

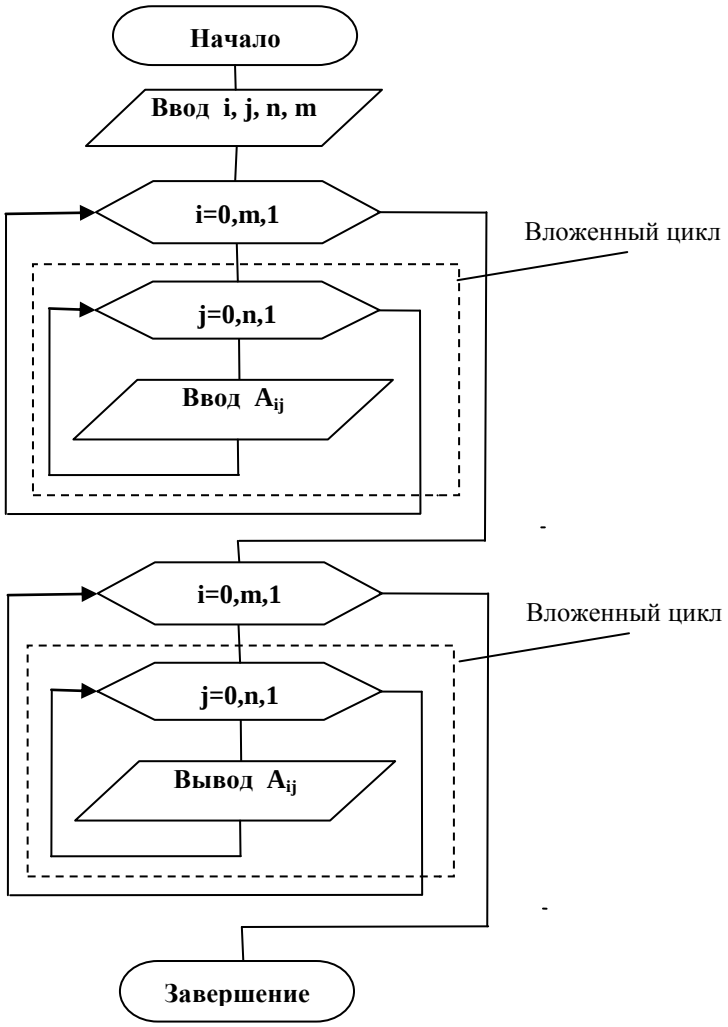


Рис. 2. Блок-схема алгоритма поэлементного ввода и вывода матрицы с использованием цикла **for**

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    const int M=10,N=10;                          //Число строк и столбцов матрицы  A с запасом
    int i, j, m, n, A[M][N];                      //Объявление переменных и матрицы A
    cout<<"Vvedite chislo strok i stolbcov matricu:"<<endl;
    cin>>m>>n;                                    //Ввод числа строк и столбцов исходной матрицы A
    cout<<" Vvedite postrochno elementu matricu:"<<endl;
        for(i=0; i<m; i++)                        //Внешний цикл ввода элементов матрицы A
        for(j=0; j<n; j++)                        //Вложенный цикл ввода элементов матрицы A
                cin>>A[i][j];                     //Ввод элемента матрицы A
                cout<<endl;
    cout<<"Matrica A"<<endl;
        for(i=0; i<m; i++)                        //Внешний цикл вывода элементов матрицы A
            {                                     //Начало составного оператора для внешнего цикла for
            for(j=0; j<n; j++)                    //Вложенный цикл для вывода элементов матрицы A
                cout<<A[i][j]<<"   ";             //Вывод элемента матрицы A
                cout<<endl;
            }                                     //Конец составного оператора для внешнего цикла for

    getch();
    return 0;
    }
```

После выполнения программы получим:

Vvedite chislo strok i stolbcov matricu:
3 4
Vvedite postrochno elementu matricu:
1 2 3 4 5 6 7 8 9 10 11 12
Matrica A
1 2 3 4
5 6 7 8
9 10 11 12

3. Исследование различных операций в двухмерных массивах в среде Visual C++ 2010

Элементам массива могут быть **присвоены** значения выражений. При этом элементы массива и значения выражений должны иметь один и тот же тип.

Например, объявлен массив

double A[3][4];

тогда возможна запись

```
A[0][0]= 3.5;
A[1][3]= 0;
A[2][1]= a*x + b;
```

Копирование – это присваивание значений элементов одного массива элементам другого массива. При копировании оба массива должны иметь одинаковый размер и тип элементов. Копирование массива **A** в **B** будет иметь вид:

```
for (i=0; i<N; i++) B[i][j]= A[i][j];
```

Рассмотрим основные приемы проведения различных операций с матрицами.

Пример 2. Исследуем операцию присваивания в двухмерных массивах, а также операцию транспонирования матрицы. Задана матрица целых чисел **A** размерностью 3×4. Необходимо транспонировать матрицу **A**, т. е. поменять местами ее строки и столбцы, а затем рассчитать матрицу **B**, элементы которой определяются по формуле

$$B_{ij} = \sqrt[3]{A_{ij}^2} + \sin^2(A^3) \ .$$

Элементы исходной матрицы **A** введем непосредственно в программном коде. Исходную матрицу **A**, транспонированную **AT** и полученную матрицу **B** выведем на экран. Для наглядности примем

$$A(3,4) = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{vmatrix} .$$

Блок-схема алгоритма решения данной задачи приведена на рис. 3.

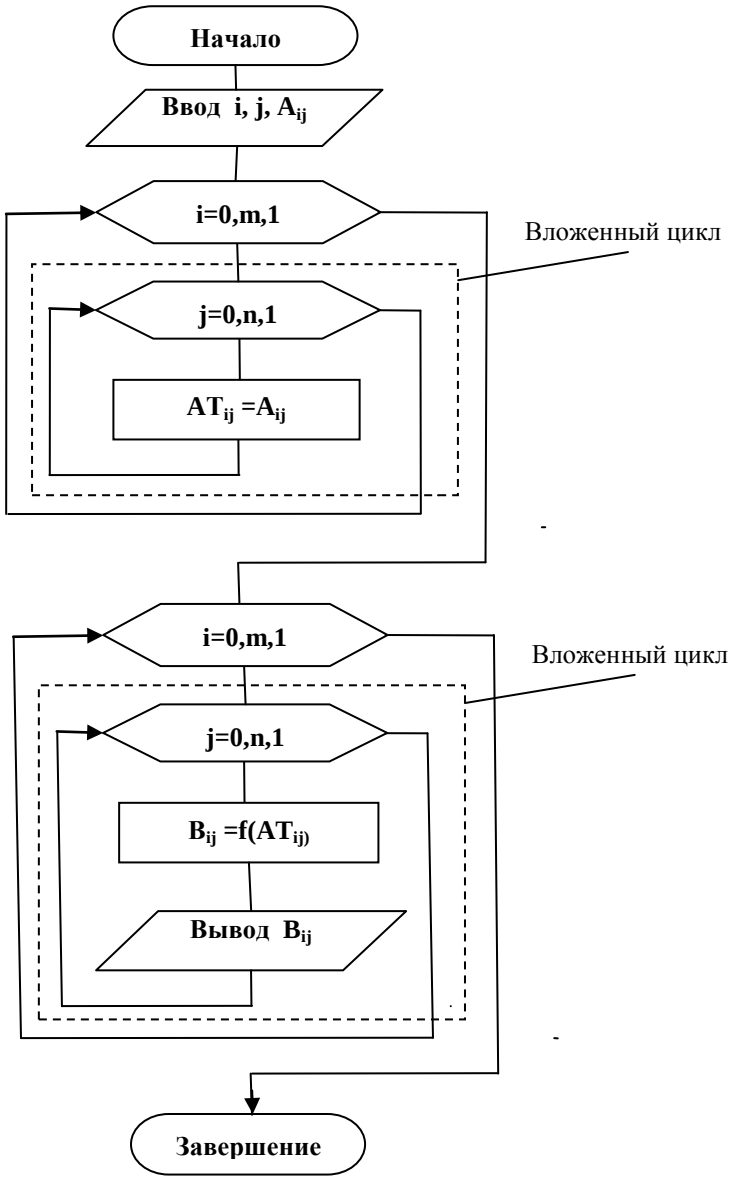


Рис. 3. Блок-схема алгоритма транспонирования и присваивания матрицы

Реализующая данный алгоритм программа имеет следующий вид:

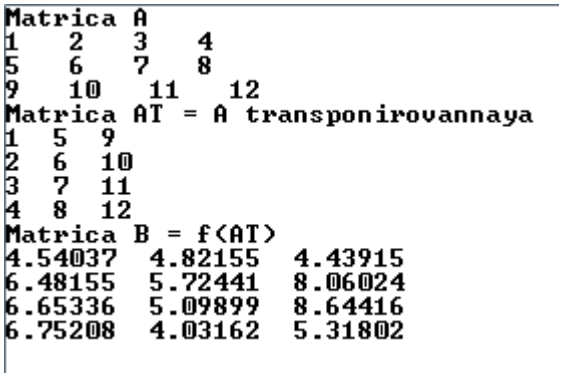
```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j;
double AT[4][3], B[4][3];                    //Объявление переменных и матриц A и B
double A[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};  //Инициализация исходной матрицы A
//Вывод исходной матрицы
cout<<"Matrica A"<<endl;                     //Вывод названия

    for(i=0; i<3; i++)                       //Внешний цикл вывода элементов матрицы A
    {
        for(j=0; j<4; j++)                   //Вложенный цикл для вывода элементов матрицы A
            cout<<A[i][j]<<"   ";            //Вывод элементов исходной матрицы A
            cout<<endl;
    }
                    //Транспонирование исходной матрицы
    for(i=0; i<3; i++)                       //Внешний цикл для транспонирования
        for(j=0; j<4; j++)                   // Вложенный цикл для транспонирования
            AT[j][i]=A[i][j];                //Операция транспонирования матрицы A
  cout<<"Matrica AT = A transponirovannaya"<<endl;  //Вывод названия
    for(i=0; i<4; i++)                       //Начало цикла
```

```
    {
        for(j=0; j<3; j++)                    //Вложенный цикл для вывода матрицы AT
            cout<<AT[i][j]<<"  ";             //Вывод элементов матрицы AT
            cout<<endl;
    }
cout<<"Matrica B = f(AT)"<<endl;              //Вывод названия
//Вычисление матрицы B
    for(i=0; i<4; i++)                        //Внешний цикл для поэлементного вычисление матрицы B
    {
        for(j=0; j<3; j++)                    // Вложенный цикл для поэлементного вычисление матрицы B
        {
            B[i][j]=5*pow(sin(pow(AT[i][j],3)),2)+pow((AT[i][j]*AT[i][j]),1./3.);   // Bij = f(ATji)
            cout<<B[i][j]<<"  ";                                                    //Вывод элементов матрицы B

        }
    cout<<endl;
    }
getch();
return 0;
}
```

После запуска программы на выполнение экран дисплея должен иметь вид:



.

Пример 3. Исследуем операцию поиска максимального элемента матрицы. Задана матрица **A** из 20 произвольных чисел размера 4×5. Необходимо найти максимальный элемент матрицы **A** и номера его строки и столбца (индексы элемента матрицы). Элементы матрицы **A** введем в программе прямым образом. На экран выведем исходную матрицу **A**, максимальный элемент и его индексы.

Блок-схема алгоритма решения данной задачи приведена на рис. 4.

Для решения задачи используем следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;ij
int _tmain(int argc, _TCHAR* argv[])
{
int i,  j, m, n;                                          // Объявление переменных
int A[4][5]={1,2,18,3,20,4,5,6,7,8,9,10,19,11,12,13,14,15,16,17};  //Инициализация матрицы A
int B;
cout<<"Matrica A"<<endl;
    for(i=0; i<4; i++)                                    //Внешний цикл вывода  матрицы A
    {
    for(j=0; j<5; j++)                                    //Внутренний цикл вывода  матрицы A
    cout<<A[i][j]<<"\t";                                  //Вывод  элемента  матрицы A
    cout<<endl;
    }
B=A[0][0];                                    //Подготовка к поиску максимального элемента
m=0; n=0;
    for(i=0; i<4; i++)                        //Внешний  цикл по перебору элементов матрицы A
        for(j=0; j<5; j++)                    //Внутренний  цикл по перебору элементов матрицы A
            if(A[i][j]>B)                     //Выбор максимального элемента
            {
            B=A[i][j];                        //Запоминание  в B большего элемента Aij
```

```
            n=i;                                  //Запоминание в  n номера строки i элемента Aij
            m=j;                                  //Запоминание в m номера столбца j элемента Aij
            }
cout<<endl<<"Max= "<<B;                           //Вывод  максимального элемента
cout<<"   Stroka  "<<n+1;                         //Вывод  номера строки
cout<<"   Stolbec  "<<m+1<<endl;                  //Вывод   номера столбца
getch();
return 0;
}
```

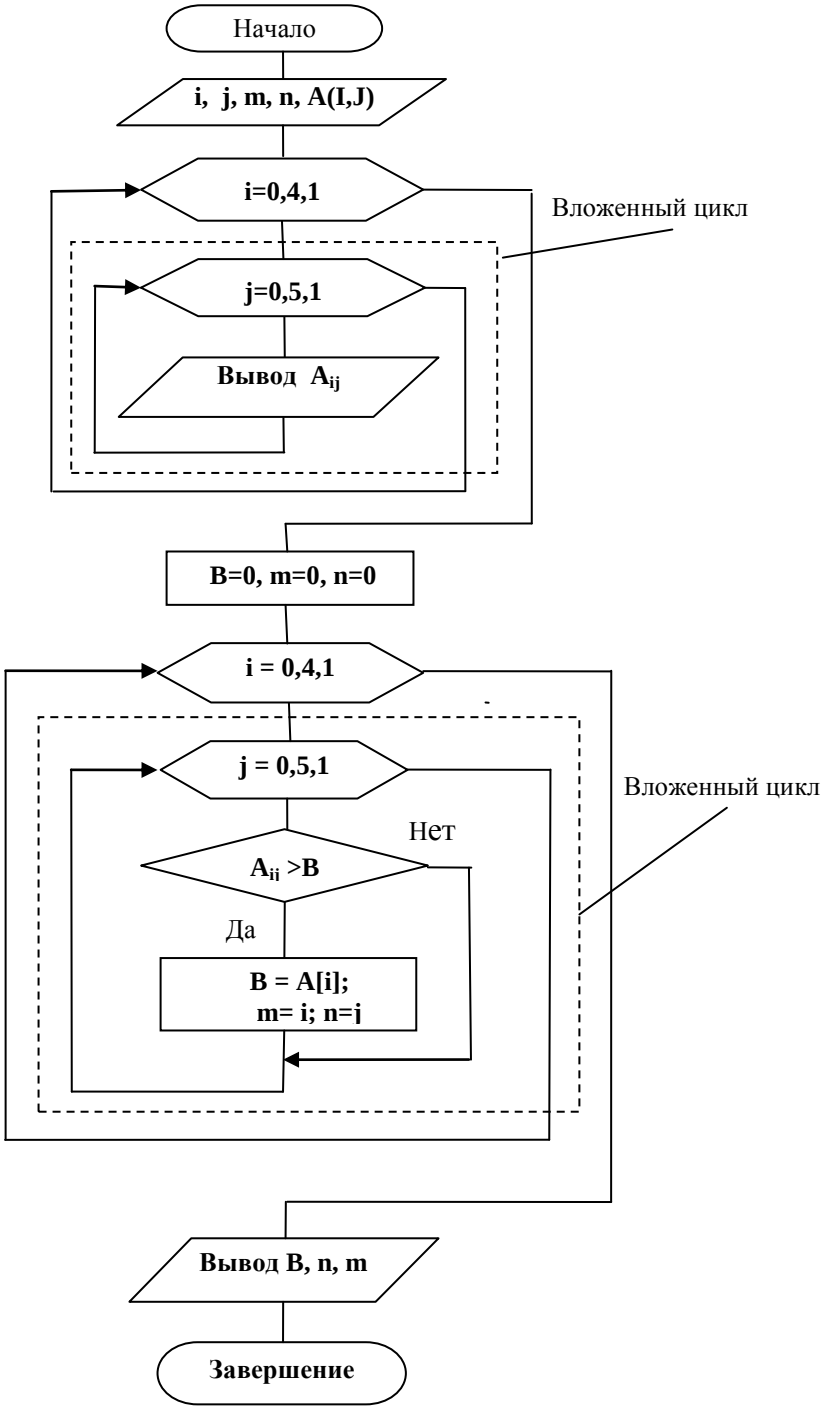


Рис. 4. Блок-схема алгоритма поиска максимального элемента матрицы

Экран после выполнения программы должен иметь следующий вид:

```
Matrica A
1      2      18    3      20
4      5      6      7      8
9      10    19    11    12
13    14    15    16    17
Max= 20      Stroka  1      Stolbec  5
```

Пример 4. Исследуем способы ввода и вывода двухмерных массивов, а также следующие действия с матрицей: найти минимальный элемент каждой строки матрицы **A**, составить из них вектор **R** (одномерный массив) и определить суммы его четных и нечетных элементов. Задана матрица из 20 произвольных чисел размера 4×5. Элементы исходной матрицы **A** введем с клавиатуры. На экран выведем исходную матрицу **A,** вектор **R** и вычисленные суммы.

Блок-схема алгоритма решения данной задачи не приводится, т. к. она аналогична блок-схеме на рис. 3.

Для решения задачи предлагается следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int M=10, N=10;                          //Максимальное число строк и столбцов матрицы
int A[M][N], R[M];                             //Декларирование матрицы A и вектора R
int i, j, k, m, S1, S2;                        // Декларирование индексов и вспомогательных переменных
cout<<"Vvedite chislo strok  i stolbcov matricu  A :"<<endl;
cin>>m>>k;                                     //Ввод реальных размеров исходной матрицы A
cout<<"Vvedite postrochno elementu matricu  A :"<<endl;
    for(i=0; i<m; i++)        //Внешний  цикл вывода элементов матрицы A
        for(j=0; j<k; j++)
            cin>>A[i][j];                      //Ввод элементов исходной матрицы A
    //Поиск минимального элемента каждой строки исходной матрицы A
    for(i=0; i<m; i++)
    {                                          //Начало составного оператора в цикле
        R[i]=A[i][0];                          //Задание начального значения минимального элемента строки
            for(j=0; j<k; j++)                 //Определение минимального элемента
                if(A[i][j]<R[i])               //каждой строки исходной матрицы A
                    R[i]=A[i][j];              //Запись его в  массив R
    }                                          //Конец составного оператора в цикле
S1=0;                                          //Задание начальных значений сумм четных
S2=0;                                          //и нечетных элементов  вектора R
    //Операции по определению суммы четных и нечетных элементов массива R
    for(i=0; i<m; i++)
    {                                          //Начало составного оператора в цикле
        if(R[i]%2==0)                          //Условие – есть ли остаток от деления на 2 (четное или нет)
            S2=S2+R[i];                        //Определение  суммы четных элементов вектора R
        else S1=S1+R[i];                       //Определение суммы нечетных элементов вектора R
    }                                          //Конец составного оператора в цикле
cout<<"Matrica A"<<endl;
    for(i=0; i<m; i++)                         //Внешний  цикл вывода элементов матрицы A
    {                                          //Начало составного оператора в цикле
        for(j=0; j<k; j++)
            cout<<A[i][j]<<"\t";               //Вывод элементов матрицы A
            cout<<endl;
    }                                          //Конец составного оператора в цикле
 cout<<endl<<"Vektor R"<<endl;
        for(i=0; i<m; i++)
            cout<<R[i]<<"    ";                //Вывод вектора R
cout<<endl;
cout<<"S1= "<<S1<<endl;                        //Вывод суммы S1 четных элементов
cout<<"S2= "<<S2<<endl;                        //Вывод суммы S2 нечетных элементов
getch();
return 0;
}
```

В результате выполнения вышеприведенной программы получим следующие результаты:

```
Matrica A
1    2    3    4
5    6    7    8
9    10    11    12
Matrica AT = A transponirovannaya
1   5   9
2   6   10
3   7   11
4   8   12
Matrica B = f(AT)
4.54037   4.82155   4.43915
6.48155   5.72441   8.06024
6.65336   5.09899   8.64416
6.75208   4.03162   5.31802
```

Вопросы для самоконтроля.

1. Каким образом в программе на С++ объявляется двухмерный массив?
   1) **double A[0..2][0..3];** 2) **double A[2, 3];** 3) **double A[2][3];**
2. В памяти компьютера двухмерный массив располагается…
   1) непрерывно строками 2) непрерывно столбцами 3) в виде обычной матрицы
3. Возможно ли следующее объявление массивов в С++: **int i, j, k, m ; int A[i][j], B[k][m];**?
   1) Возможно 2) Не возможно 3) если **i>j**, а **k>m**
4. Как в программе на С++ будет выведен на экран двухмерный массив **A[k][k]**оператором
   for(i=0; i<k; i++) for(j=0; j<k; j++) cout<<A[i][j]<<endl; ?
   1) В виде строки
   2) В виде столбца
   3) В виде матрицы
5. Что нужно изменить в операторе **for(i=0; i<k; i++) for(j=0; j<k; j++) cout<<A[i][j]<<endl;** чтобы массив **A[k][k]** был выведен на экран в виде строки?
   1) Вместо **i** подставить **k** 2) Убрать **<<endl** 3) Вместо **j** подставить **k**
6. Как будет выведен на экран двухмерный массив в фрагменте программы на С++?

```
int i, j, k, m;
for(i=0; i<k; i++)
    for ( j=0; j<m; j++)
    cout<<A[i][j]<<" ";
```

   1) В виде строки 2) В виде столбца 3) В виде матрицы
7. Как будет выведен на экран двухмерный массив в фрагменте программы на С++?

```
int i, j, k, m
for(i=0; i<k; i++)
{
    for ( j=0; j<m; j++)
    cout<<A[i][j];
    cout<<endl;
}
```

   1) В виде строки 2) В виде столбца 3) В виде матрицы
8. Что будет выведено на экран в фрагменте программы на С++?

```
    for(i=0; i<n; i++)
    for (j=0; j<n; j++)
    cout<<A[0][j];
```

   1) Первая строка матрицы, повторенная **n** раз
   2) Первый столбец матрицы, повторенный **n** раз
   3) Полная матрица
9. Что нужно изменить в фрагменте программы

```
int i, j, k, m;
for(i=0; i<k; i++)
    for ( j=0; j<m; j++)
    cout<<A[i][j]<<" ";
```

для вывода массива **A[k][m]** на экран в виде матрицы?
   1) Вместо **i** подставить **k** 2) Добавить **cout<<endl;** 3) Вместо **j** подставить **m**
10. Почему при работе с массивами в С++ целесообразно использовать оператор цикла **for**?
   1) Т.к. оператор цикла **for** безошибочный
   2) Т.к. известен размер массива, т. е. число повторений цикла.
   3) Т.к. оператор цикла **for** быстродействующий
11. Какое действие выполняет оператор **for (i=0; i<N; i++) B[i][j]= A[i][j];**?
   1) Выводит на экран матрицу **B[i][j]**
   2) Выводит на экран матрицу **A[i][j]**
   3) Копирует матрицу **A[i][j]** в матрицу **B[i][j]**