**Л Е К Ц И Я № 05 УКАЗАТЕЛИ**

Оглавление

[1. МНОГОМЕРНЫЕ МАССИВЫ 1](#_Toc37677160)

[Пример 4.6 Объявление и инициализация многомерных массивов 2](#_Toc37677161)

[Пример \*\* Работа с матрицами БЕЗ ФУНКЦИЙ 7](#_Toc37677162)

[Комментарии к Лабораторной работе на матрицы 19](#_Toc37677163)

[Пример \*\* Работа с матрицами ЧЕРЕЗ ФУНКЦИИ 21](#_Toc37677164)

[2. ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ 35](#_Toc37677165)

[ОПЕРАТОРЫ new и delete 35](#_Toc37677166)

[Пример 4.7 Динамическое выделение памяти для переменных 37](#_Toc37677167)

[3. СОЗДАНИЕ ДИНАМИЧЕСКИХ МНОГОМЕРНЫХ МАССИВОВ 39](#_Toc37677168)

1. МНОГОМЕРНЫЕ МАССИВЫ

Специального типа многомерных массивов в С++ не существует. Многомерный массив размерности N можно представить как одномерный массив из массивов размерности N-1. Таким образом, трехмерный массив – это одномерный массив, элементами которого являются матрицы. Многомерные массивы инициализируются в порядке наискорейшего изменения самого правого индекса (как разряды в числах)

Пример 4.6 Объявление и инициализация многомерных массивов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex03\_03 \*

\* File Name : Ex04\_06.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : ПРИМЕР МНОГОМЕРНЫХ МАССИВОВ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

cout << "\tEx04\_06.cpp\n";

//примеры инициализации

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

, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24 };

// или

int y[3][2][4] = { { { 1, 2, 3, 4 } //три двумерных массива

,{ 5, 6, 7, 8 } } //1

,{ { 9, 10, 11, 12 }

,{ 13, 14, 15, 16 } } //2

,{ { 17, 18, 19, 20 }

,{ 21, 22, 23, 24 } } //3

};

int i, j, k;

//массивы x и y идентичны

//1

cout << "/1\n";

//массив x

// " Массив x "

**cout << "\n\t Array x \n";**

for (k = 0; k < 3; ++k) //по номеру плоскости

{

for (i = 0; i < 2; ++i) //по номеру строки

{

for (j = 0; j < 4; ++j) //по номеру столбца

cout << x[k][i][j] << " "; //по горизонтали

cout << endl;

} //for i

cout << "\n";

} //for k

cout << "\n";

//массив y

// " Массив y "

**cout << "\n\t Array y \n";**

for (k = 0; k < 3; ++k) //по номеру плоскости

{

for (i = 0; i < 2; ++i)) //по номеру строки

{

for (j = 0; j < 4; ++j) //по номеру столбца

cout << y[k][i][j] << " "; //по горизонтали

cout << endl;

} //for i

cout << "\n";

} //for k

cout << "\n";

//2 для многомерных массивов можно опускать величину первой размерности

cout << "/2\n";

**int z[][3]** = { { 71, 72, 73 } //коды ASCII

,{ 74, 75, 76 } //560 % 256 = 48 код "0"

,{ 77, 78, 79 }

,{ 80, 81, 82 } };

for (i = 0; i< **sizeof(z) / (3 \* sizeof(int))**; i++) //цикл по первой координате

{

for (j = 0; j<3; j++) //цикл по второй координате

cout << z[i][j] << " " << (char)z[i][j] << " ";

cout << endl;

} //for i

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex04\_06.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Чтобы создать функцию, принимающую в качестве аргумента двумерный массив, надо помнить, что имя массива обрабатывается как адрес, то есть является указателем. Надо только правильно объявить указатель.

Пусть функции sum() передается массив matr :

int matr [3][4] = { {1,2,3,4},{9,8,7,6},{2,4,6,8}};

int total = sum (matr, 3);

Как должен выглядеть прототип функции sum()? Почему ей передается только число строк (3), а не число строк (3) и столбцов (4)?

Здесь matr представляет собой имя массива, содержащего 3 элемента, каждый из которых является массивом на 4 элемента. matr указывает на первый из четырех значений первой строки типа int. Прототип выглядит следующим образом:

int sum (int (\*arr2)[4], int size);

здесь int (\*arr2)[4] – указатель на массив из 4 элементов, а

int \*arr2[4] - массив из четырех указателей на данные типа int, а параметр функции не может быть массивом.

int \*arr2[4]; //массив из 4 указателей на элемент типа int

int (\*arr2)[4]; //указатель на массив из 4 int элементов

Существует еще один формат записи прототипа

**int sum (int arr2[ ][4], int size);**

int sum (int (\*arr2)[4] , int size);

(аналогично int years[] и int \*years)

Оба варианта означают, что arr2 представляет собой **указатель**, а не массив. Кроме того, обозначение типа указателя сообщает, что он ссылается на массив из четырех значений типа int. Таким образом, тип указателя уже определяет количество столбцов и отдельным параметром в функцию не передается. Функция sum() может обрабатывать только массивы с четырьмя столбцами, а количество строк может быть любым (!) и задается переменной size (в функции не надо жестко прописывать размер массива).

int sum (int arr2[ ][4], int size)

{

int total = 0;

for (int i = 0; i < size; i++)

for (int j = 0; j < 4; j++)

total +=arr2[i][j];

return total;

}

Пример \*\* Работа с матрицами БЕЗ ФУНКЦИЙ

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* КАФЕДРА № 302 1 КУРС ПЯВУ \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_УМНОЖЕНИЕ МАТРИЦ БЕЗ ФУНКЦИЙ \*

\* File Name : Ex04\_УМНОЖЕНИЕ МАТРИЦ БЕЗ ФУНКЦИЙ.CPP \*

\* Language : C/C++ \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modified By : \*

\* Created : 10/03/15 \*

\* Last Revision : 07/04/20 \*

\* Comment(s) : **ПРИМЕРЫ РАБОТЫ С ДВУМЕРНЫМИ МАССИВАМИ** \*

\* ВВОД МАТРИЦЫ С КЛАВИАТУРЫ \*

\* ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ И ПО СТОЛБЦАМ \*

\* СУММА ЭЛЕМЕНТОВ МАТРИЦЫ \*

\* ПОСЧИТАТЬ СЛЕД МАТРИЦЫ \*

\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*

\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

int main()

{

//стандартная молитва

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

system("color F0"); //экран белый, буквы черные

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* ВВОД МАТРИЦЫ С КЛАВИАТУРЫ \*/**

**/\* ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ И ПО СТОЛБЦАМ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block ВВОД МАТРИЦЫ С КЛАВИАТУРЫ

// ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ И ПО СТОЛБЦАМ

//локальные константы

const int NMAX = 2; //число строк

const int MMAX = 3; //число столбцов

//объявление переменных

int AMatr[NMAX][MMAX];

int i, j;

cout << "ВВЕДИТЕ МАТРИЦУ\n";

for (i = 0; i < NMAX; i++) //по строкам

{

cout << "Введите " << i << "-ю строку матрицы\n";

for (j = 0; j < MMAX; j++) //по столбцам

{

cout << "AMart [" << i << "][" << j << "] = ";

cin >> AMatr[i][j];

}//for j

}//for i

//печать матрицы по строкам

cout << "\n\tПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ\n\t";

for (i = 0; i < NMAX; i++) //по строкам

{

for (j = 0; j < MMAX; j++) //по столбцам

{

cout << AMatr[i][j] << " ";

}//for j

cout << "\n\t";

}//for i

cout << "\n\tПЕЧАТЬ МАТРИЦЫ ПО СТОЛБЦАМ\n\t";

for (j = 0; j < MMAX; j++) //по столбцам

{

for (i = 0; i < NMAX; i++) //по строкам

{

cout << AMatr[i][j] << " ";

}//for i

cout << "\n\t";

}//for j

cout << endl;

//СУММА ЭЛЕМЕНТОВ МАТРИЦЫ

int SumMatr = 0;

for (i = 0; i < NMAX; i++) //по строкам

{

for (j = 0; j < MMAX; j++) //по столбцам

{

SumMatr = SumMatr + AMatr[i][j];

}//for j

}//for i

cout << "СУММА ЭЛЕМЕНТОВ МАТРИЦЫ = " << SumMatr << endl;

}//block ВВОД МАТРИЦЫ С КЛАВИАТУРЫ

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* ПОСЧИТАТЬ СЛЕД МАТРИЦЫ \*/**

**/\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block ПОСЧИТАТЬ СЛЕД МАТРИЦЫ

//block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

//Транспонировать матрицу саму в себя

//без использования дополнительного массива

//локальные константы

const int NMAX = 5; //число строк и столбцов (квадратная матрица)

//объявление переменных

int AMatr[NMAX][NMAX];

int i, j;

int Temp;

cout << "\n\t\tПОСЧИТАТЬ СЛЕД МАТРИЦЫ \n";

cout << "\t\tТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ\n";

//ввод исходных данных

cout << "\tВВОД МАТРИЦЫ AMatr\n";

for (i = 0; i < NMAX; i++)

{

cout <<"\tВведите " << i <<"- ю строку матрицы из "

<< NMAX << " элементов\n";

for (j = 0; j < NMAX; j++)

{

cin >> AMatr[i][j];

}; // for j

cout << endl;

}; //for i

//Печать исходных данных

cout << "\n\t МАТРИЦА AMatr\n";

for (i = 0; i < NMAX; i++)

{

for (j = 0; j < NMAX; j++)

{

cout << "\t" << AMatr[i][j];

}; // for j

cout << endl;

}; //for i

cout << endl;

//ПОСЧИТАТЬ СЛЕД МАТРИЦЫ

int TraceMatr = 0;

for (i = 0; i < NMAX; i++)

TraceMatr = TraceMatr + AMatr[i][i];

cout << "\tСЛЕД МАТРИЦЫ = " << TraceMatr << endl;

//ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

for (i = 0; i < NMAX; i++)

{

//for (j = 0; j < NMAX; j++) **//с ошибкой!!!!**

/\*

//верхний треугольник

i j

0 1..NMAX - 1

1 2..NMAX - 1

2 3..NMAX - 1

3 4..NMAX - 1

\*/

//элементы на главной диагонали не трогаем (**j > i**)

for (j = **i + 1**; j < NMAX; j++) **//без ошибки**

{

//меняем элементы местами

Temp = AMatr[i][j];

AMatr[i][j] = AMatr[j][i];

AMatr[j][i] = Temp;

}; // for j

}; //for i

cout << endl;

cout << "\n\tТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr\n";

for (i = 0; i < NMAX; i++)

{

for (j = 0; j < NMAX; j++)

{

cout << "\t" << AMatr[i][j];

}; // for j

cout << endl;

}; //for i

cout << endl;

system("pause");

/\*

Работа программы.

МАТРИЦА AMatr **i j**

1 2 3 4 5 **0 1..NMAX-1**

6 7 8 9 10 **1 2..NMAX-1**

11 12 13 14 15 **2 3..NMAX-1**

16 17 18 19 20 **3 4..NMAX-1**

21 22 23 24 25 **не используется**

СЛЕД МАТРИЦЫ = 65

ТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr

1 6 11 16 21

2 7 12 17 22

3 8 13 18 23

4 9 14 19 24

5 10 15 20 25

Для продолжения нажмите любую клавишу . . .

\*/

} // end block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

//локальные константы

const int MMAX = 5; //число столбцов (длина вектора)

//объявление переменных

int AVec[MMAX] = { 1, 2, 3, 4, 5 }; //вектор A

int BVec[MMAX] = { 6, 7, 8, 9, 10 }; //вектор B

int ResMult\_VV = 0; //результат умножения вектора на вектор

int j; //j -для движения по столбцам (по строке)

cout << "\n\tУМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР\n";

//печать исходных данных

cout << "\tВЕКТОР А\n";

for (j = 0; j < MMAX; j++)

{

**cout <<** /\*"AVec[" << j << "]= " <<\*/ **AVec[j] << " ";**

} //end for j

cout << endl;

cout << "\tВЕКТОР B\n";

for (j = 0; j < MMAX; j++)

{

**cout <<** /\*"BVec[" << j << "]= " <<\*/ **BVec[j] << " ";**

} //end for j

cout << endl;

**ResMult\_VV = 0; //результат умножения вектора на вектор**

**//умножение вектора на вектор**

**for (j = 0; j < MMAX; j++)**

**{**

**ResMult\_VV = ResMult\_VV + AVec[j] \* BVec[j];**

**} //end for j**

//печать результата

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ ВЕКТОРА НА ВЕКТОР = "

<< ResMult\_VV << "\n\n";

system("pause");

}//end block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{

//Умножение матрицы размером N\*M на вектор размером M.

const int NMAX = 2; //число строк

const int MMAX = 3; //число столбцов (длина вектора)

//объявление переменных

int AMatr[NMAX][MMAX] = { { 1, 2, 3 } //1 строка

,{ 3, 2, 1 } //2 строка

};

int AVec[MMAX] = { 4, 5, 6 }; //вектор A

int ResVec[NMAX];

int i; //i - для смены строки

int j; //j - для движения по столбцам (по строке)

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР\n";

////ввод исходных данных

//cout << "\tВВОД МАТРИЦЫ AMatr\n";

//for (i = 0; i < NMAX; i++)

//{

// cout << "Введите " << i << "- ю строку матрицы из "

// << MMAX << " элементов\n";

// for (j = 0; j < MMAX; j++)

// {

// //cout <<"AMatr[" << i << "][" << j << "]= ";

// cin >> AMatr[i][j];

// //cout << "\t" << AMatr[i][j]<< "\t";

// }; // for j

// cout << endl;

//}; //for i

//cout << "\tВВОД ВЕКТОРА-столбца AVec\n";

//for (j = 0; j < MMAX; j++)

//{

// cout << "AVec[" << j << "]= ";

// cin >> AVec[j];

// //cout << "\t" << AVec[j]<< "\t";

//}; //for j

//cout << endl;

//Печать исходных данных

**cout << "\n\t МАТРИЦА AMatr\n";**

for (i = 0; i < NMAX; i++)

{

for (j = 0; j < MMAX; j++)

{

cout << " " << AMatr[i][j];

}; // for j

cout << endl;

}; //for i

cout << endl;

**cout << "\n\tВЕКТОР AVec\n";**

for (j = 0; j < MMAX; j++)

{

cout << "\t " << AVec[j] << endl;

}; //for j

cout << endl;

// УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР\n";

for (i = 0; i < NMAX; i++) //по строкам - выбор очередного вектора из матрицы

{

//умножение очередного вектора AMatr[i] из матрицы на вектор AVec

**ResVec[i] = 0;**

**//движение по столбцам (по строке)**

**for (j = 0; j < MMAX; j++) //по столбцам**

**{**

**ResVec[i] = ResVec[i] + AMatr[i][j] \* AVec[j];**

**}//for j**

**//результат умножения двух векторов**

**cout << "\t\t" << ResVec[i] << endl;**

}//for i

cout << endl;

system("pause");

/\*

Работа программы.

Введите 1-ю строку матрицы

1 2 3

1.00 2.00 3.00

Введите 2-ю строку матрицы

3 2 1

3.00 2.00 1.00

Введите вектор-столбец строкой

4 5 6

4.00

5.00

6.00

Результат

32.00

28.00

\*/

} // end block УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

//локальные константы

const int NMAX = 4; //число строк

const int MMAX = 3; //число столбцов (длина вектора)

const int KMAX = 2;

//объявление переменных

int AMatr[NMAX][MMAX] =

**{ 1, 2, 3**

**, 4, 5, 6**

**, 7, 8, 9**

**, 0, 1, 2**

**};**

int BMatr[MMAX][KMAX] =

**{**

**1, 3**

**, 2, 2**

**, 3, 1**

**};**

int ResMatr\_MM[NMAX][KMAX];//результат умножения MM

int i, j, k;

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ\n";

////ввод исходных данных

//cout << "\tВВОД МАТРИЦЫ AMatr\n";

//for (i = 0; i < NMAX; i++) //по числу строк

//{

// cout << "Введите " << i << "-ю строку матрицы из " << MMAX << " элементов\n";

// for (j = 0; j < MMAX; j++) //по длине строки

// {

// cin >> AMatr[i][j];

// }; // for j

// // cout << endl;

//}; //for i

//cout << endl;

//cout << "\tВВОД МАТРИЦЫ BMatr\n";

//for (j = 0; j < MMAX; j++) //по числу строк

//{

// cout << "Введите " << j << "-ю строку матрицы из " << KMAX << " элементов\n";

// for (k = 0; k < KMAX; k++) //по длине строки

// {

// cin >> BMatr[j][k];

// }; // for j

// // cout << endl;

//}; //for i

//cout << endl;

//Печать исходных данных

**cout << "\n\t МАТРИЦА AMatr\n";**

for (i = 0; i < NMAX; i++) //по числу строк

{

for (j = 0; j < MMAX; j++) //по числу столбцов

{

cout << " " << AMatr[i][j];

}; // for j

cout << endl;

}; //for i

cout << endl;

**cout << "\n\t МАТРИЦА BMatr\n\t";**

for (j = 0; j < MMAX; j++) //по числу строк

{

for (k = 0; k < KMAX; k++) //по числу столбцов

{

cout << " " << BMatr[j][k];

}; // for j

cout << "\n\t";

}; //for i

cout << endl;

//УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

**for (i = 0; i < NMAX; i++) //вектор-строка из матрицы A**

**{**

**for (k = 0; k < KMAX; k++) //вектор-столбец из матрицы B**

**{**

**ResMatr\_MM[i][k] = 0;**

**for (j = 0; j < MMAX; j++)//по длине векторов из матриц А и В**

**{**

**ResMatr\_MM[i][k] = ResMatr\_MM[i][k] + AMatr[i][j] \* BMatr[j][k];**

**};//for j**

**};// for k**

**};//for i**

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ \n\t\t";

for (i = 0; i < NMAX; i++) //по числу строк

{

for (k = 0; k < KMAX; k++) //по числу столбцов

{

cout << " " << ResMatr\_MM[i][k];

} //for k

cout << "\n\t\t";

} // for i

cout << endl;

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

МАТРИЦА AMatr

1 2 3

4 5 6

7 8 9

0 1 2

МАТРИЦА BMatr

1 1

2 2

3 3

РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ

14 14

32 32

50 50

8 8

Для продолжения нажмите любую клавишу . . .

1 2 3 1 0 0 1 2 3

3 2 1 умножить 0 1 0 равно 3 2 1

0 0 1

Результат

1 2 3

3 2 1

\*/

}//end block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

cout << endl;

system("pause");

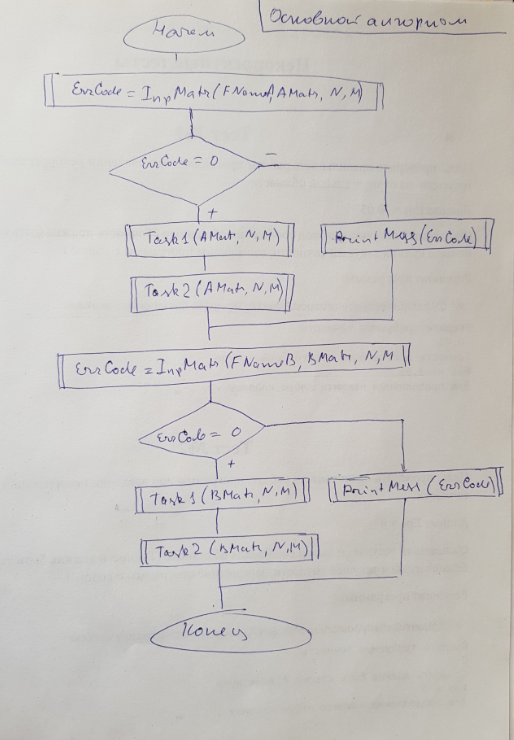
return 0;

}//end main()

/\*\*\*\*\* END Of Ex04\_УМНОЖЕНИЕ МАТРИЦ БЕЗ ФУНКЦИЙ.CPP FILE \*\*\*\*\*/

Комментарии к Лабораторной работе на матрицы

**Общая блок-схема программы**



**Описание функции InpMatr**

1. Назначение

Ввод данных из файла и входной контроль

2. Прототип функции

int InpMatr( char \* xFname //имя файла исх данных

, int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

);

3. Обращение

ErrCode = InpMatr(FNAME\_A, AMatr, N, M);

4. Описание параметров

|  |  |  |  |
| --- | --- | --- | --- |
| **Идентификатор** | **Тип** | **Назначение** | **Входной/Выходной** |
| InpMatr | int | Код ошибки | Выходной |
| xFname | char \* | имя файла исх данных | Входной |
| xMatr | (xMatr \*)[MMAX]  *(xMatr[][MMAX])* | указатель на матрицу | Выходной |
| xN | int | число строк | Выходной |
| xM | int | число столбцов | Выходной |

5. Блок-схема функции InpMatr

. . .

Пример \*\* Работа с матрицами ЧЕРЕЗ ФУНКЦИИ

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* КАФЕДРА № 302 1 КУРС ПЯВУ \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Pr\_05\_Matr\_Func \*

\* File Name : Pr\_05\_Matr\_Func.CPP \*

\* Language : C/C++ \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 10/03/15 \*

\* Last Revision : 18/03/18 \*

\* Comment(s) : ПРИМЕРЫ РАБОТЫ С ДВУМЕРНЫМИ МАССИВАМИ \*

\* ЧЕРЕЗ ФУНКЦИИ \*

\* ВВОД МАТРИЦЫ С КЛАВИАТУРЫ \*

\* ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ И ПО СТОЛБЦАМ \*

\* СУММА ЭЛЕМЕНТОВ МАТРИЦЫ \*

\* ПОСЧИТАТЬ СЛЕД МАТРИЦЫ \*

\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*

\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const int NMAX = 5; //число строк

const int MMAX = 5; //число столбцов

const int KMAX = 5; //число столбцов

const int VERT = 1; //печать вектора по вертикали

const int HORIZ = 0; //печать вектора по горзонтали

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ВВОД МАТРИЦЫ С КЛАВИАТУРЫ -----------------------

void Input\_Matr\_Kb(int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

);

//ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ -----------------------

void Print\_Matr\_Row(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

);

//ПЕЧАТЬ МАТРИЦЫ ПО СТОЛБЦАМ -----------------------

void Print\_Matr\_Col(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

);

//СУММА ЭЛЕМЕНТОВ МАТРИЦЫ -----------------------

int Sum\_Matr\_1(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

);

//-----------------------

void Sum\_Matr\_2(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

, int &xSumM //результат - СУММА

);

//ПОСЧИТАТЬ СЛЕД МАТРИЦЫ -----------------------

int Trace\_Matr( const int xMatr[][MMAX] //указатель на матрицу

, int xM //число строк = столбцов

);

//ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ -----------------------

void Transpon\_Matr( int xMatr[][MMAX] //указатель на матрицу

, int xM //число строк = столбцов

);

//ПЕЧАТЬ ВЕКТОРА -----------------------

void Print\_Vec( const int xVec[] //указатель на вектор

, int xM //длина вектора

, int xVert //1 - по вертикали; 0 - по горизонтали

);

//УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР -----------------------

int Mult\_V\_V( const int xVec[] //указатель на 1 вектор , const int yVec[] //указатель на 2 вектор

, int xM //длина векторов

);

//УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР -----------------------

void Mult\_Matr\_Vect(

const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

, const int xVec[] //указатель на вектор длина вектора = числу столбцов

, int xResVec[] //указатель на вектор-результат длина вектора = числу строк

);

//УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ -----------------------

//A[N][M] \* B[M]K] = R[N][K]

void Mult\_Matr\_Matr(

const int xMatr[][MMAX] //указатель на матрицу A[N][M]

, int xN //число строк

, int xM //число строк/столбцов

, const int yMatr[][KMAX] //указатель на матрицу B[M]K]

, int xK //число столбцов

, int xResMatr[][KMAX] //указатель на матрицу-результат R[N][K]

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

//стандартная молитва

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

system("color F0");//экран белый, буквы черные

{ //block **ВВОД МАТРИЦЫ С КЛАВИАТУРЫ**

**// ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ И ПО СТОЛБЦАМ**

//локальные константы

//const int NMAX = 2; //число строк

//const int MMAX = 3; //число столбцов

//объявление переменных

int AMatr[NMAX][MMAX];

//ВВОД МАТРИЦЫ С КЛАВИАТУРЫ

Input\_Matr\_Kb(AMatr, NMAX, MMAX);

//ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

Print\_Matr\_Row(AMatr, NMAX, MMAX);

//ПЕЧАТЬ МАТРИЦЫ ПО СТОЛБЦАМ

Print\_Matr\_Col(AMatr, NMAX, MMAX);

//СУММА ЭЛЕМЕНТОВ МАТРИЦЫ - через имя функции

int SumM1 = Sum\_Matr\_1(AMatr, NMAX, MMAX);

cout << "Сумма элементов матрицы SumM1 = " << SumM1 << endl;

//СУММА ЭЛЕМЕНТОВ МАТРИЦЫ - через выходной параметр

int SumM2 = 0;

Sum\_Matr\_2(AMatr, NMAX, MMAX, SumM2);

cout << "Сумма элементов матрицы SumM2 = " << SumM2 << endl;

system("pause");

}//block ВВОД МАТРИЦЫ С КЛАВИАТУРЫ

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* ПОСЧИТАТЬ СЛЕД МАТРИЦЫ \*/**

**/\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block ПОСЧИТАТЬ СЛЕД МАТРИЦЫ

//block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

//Транспонировать матрицу саму в себя

//без использования дополнительного массива

//локальные константы

//const int NMAX = 5; //число строк и столбцов (квадратная матрицаа)

//квадратная матрица

int AMatr[MMAX][MMAX] = { 1, 2, 3, 4, 5

, 6, 7, 8, 9, 10

, 11, 12, 13, 14, 15

, 16, 17, 18, 19, 20

, 21, 22, 23, 24, 25 };

//ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

cout << "\n\t\tИСХОДНАЯ МАТРИЦА AMatr\n";

Print\_Matr\_Row(AMatr, MMAX, MMAX);

cout << "\n\t\tПОСЧИТАТЬ СЛЕД МАТРИЦЫ \n";

int TraceM = Trace\_Matr(AMatr, MMAX);

cout << "\tСЛЕД МАТРИЦЫ = " << TraceM << endl;

cout << "\n\t\tТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ\n";

Transpon\_Matr(AMatr, MMAX);

//ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

cout << "\tТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr\n";

Print\_Matr\_Row(AMatr, MMAX, MMAX);

system("pause");

/\*

Работа программы.

МАТРИЦА AMatr

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25

СЛЕД МАТРИЦЫ = 65

ТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr

1 6 11 16 21

2 7 12 17 22

3 8 13 18 23

4 9 14 19 24

5 10 15 20 25

Для продолжения нажмите любую клавишу . . .

\*/

}// end block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

//локальные константы

const int MMAX = 5; //число столбцов (длина вектора)

//объявление переменных

int AVec[MMAX] = { 1, 2, 1, 2, 1 }; //вектор A

int BVec[MMAX] = { 10, 5, 10, 5, 10 }; //вектор B

//int AVec[MMAX] = { 1, 2, 3, 4, 5 }; //вектор A

//int BVec[MMAX] = { 6, 7, 8, 9, 10 }; //вектор B

int ResMult\_VV = 0; //результат умножения вектора на вектор

cout << "\n\t\tУМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР\n";

//печать исходных данных

cout << "\tВЕКТОР А = ";

Print\_Vec(AVec, MMAX, HORIZ); //по горизонтали

cout << "\tВЕКТОР B = ";

Print\_Vec(BVec, MMAX, HORIZ); //по горизонтали

//печать результата

int ResMultVV = Mult\_V\_V(AVec, BVec, MMAX);

cout << "\tРезультат умножения вектора на вектор = "

<< ResMultVV << endl;

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

ВЕКТОР А = 1 2 1 2 1

ВЕКТОР B = 10 5 10 5 10

Результат умножения вектора на вектор = 50

\*/

}//end block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{

//Умножение матрицы размером N\*M на вектор размером M.

//реальные размеры исходной матрицы и вектора

const int NMAX\_0 = 2; //число строк

const int MMAX\_0 = 3; //число столбцов (длина вектора)

//объявление переменных

//матрицу и вектор объявляем по максимуму

int AMatr[NMAX][MMAX] = { { 1, 2, 3 } //1 строка

,{ 3, 2, 1 } //2 строка

};

int AVec[MMAX] = { 4, 5, 6 }; //вектор A

int ResVec[NMAX];

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР\n";

//Печать исходных данных

cout << "\n\t МАТРИЦА AMatr";

//по реальной длине

Print\_Matr\_Row(AMatr, NMAX\_0, MMAX\_0);

cout << "\n\tВЕКТОР AVec\n\t";

//по реальной длине

Print\_Vec(AVec, MMAX\_0, HORIZ); //по горизонтали

// УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР";

//по реальной длине

Mult\_Matr\_Vect(AMatr, NMAX\_0, MMAX\_0, AVec, ResVec);

cout << "\tВЕКТОР ResVec\n\t";

//по реальной длине

Print\_Vec(ResVec, NMAX\_0, VERT); //по вертикали

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

МАТРИЦА AMatr

/Print\_Matr\_Row(): ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

1 2 3

3 2 1

ВЕКТОР AVec

4 5 6

РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР

ВЕКТОР ResVec

32

28

Для продолжения нажмите любую клавишу . . .

\*/

} // end block УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

//A[N][M] \* B[M]K] = R[N][K]

//локальные константы

//реальные размеры матриц

const int NMAX\_0 = 4; //число строк

const int MMAX\_0 = 3; //число строк/столбцов

const int KMAX\_0 = 2; //число столбцов

//объявление переменных

int AMatr[NMAX][MMAX] = { {1, 2, 3}

, {4, 5, 6}

, {7, 8, 9}

, {0, 1, 2}

};

int BMatr[MMAX][KMAX] = { {1, 1}

, {2, 2}

, {3, 3}

};

int ResMatr\_MM[NMAX][KMAX];//результат умножения MM

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ\n";

cout << "\n\t МАТРИЦА AMatr";

//по реальной длине

Print\_Matr\_Row(AMatr, NMAX\_0, MMAX\_0); //A[N][M]

cout << "\n\tМАТРИЦА BMatr";

//по реальной длине

Print\_Matr\_Row(BMatr, MMAX\_0, KMAX\_0);//B[M]K]

// УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

Mult\_Matr\_Matr(AMatr, NMAX\_0, MMAX\_0, BMatr, KMAX\_0, ResMatr\_MM);

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ\n";

cout << "\tМАТРИЦА ResMatr\_MM";

//по реальной длине

Print\_Matr\_Row(ResMatr\_MM, NMAX\_0, KMAX\_0); //R[N][K]

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

МАТРИЦА AMatr

/Print\_Matr\_Row(): ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

1 2 3

4 5 6

7 8 9

0 1 2

МАТРИЦА BMatr

/Print\_Matr\_Row(): ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

1 1

2 2

3 3

МАТРИЦА ResMatr\_MM

РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ

/Print\_Matr\_Row(): ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ

14 14

32 32

50 50

8 8

Для продолжения нажмите любую клавишу . . .

//проверить самостоятельно

1 2 3 1 0 1 2

3 2 1 умножить 0 1 равно 3 2

0 0 1 0 0 0 0

\*/

}//end block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

cout << endl;

system("pause");

return 0;

}//end main()

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*------------------------------------------------------------\*/**

**/\* ввод матрицы \*/**

**/\* с клавиатуры \*/**

**/\*---------------\*/**

void Input\_Matr\_Kb(int xMatr[][MMAX], int xN, int xM)

{

int i, j;

cout << "/Input\_Matr\_Kb(): ВВЕДИТЕ МАТРИЦУ " << xN

<< " на " << xM << "\n";

for (i = 0; i < xN; i++) //по строкам

{

cout << "Введите " << i << "-ю строку матрицы\n";

for (j = 0; j < xM; j++) //по столбцам

{

cout << "xMatr [" << i << "][" << j << "] = ";

cin >> xMatr[i][j];

}//for j

}//for i

}//Input\_Matr\_Kb()

**/\*------------------------------------------------------------\*/**

**/\* печать матрицы \*/**

**/\* по строкам \*/**

**/\*----------------\*/**

void Print\_Matr\_Row(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM) //число столбцов

{

int i, j;

cout << "\n\t/Print\_Matr\_Row(): ПЕЧАТЬ МАТРИЦЫ ПО СТРОКАМ\n\t";

for (i = 0; i < xN; i++) //по строкам

{

for (j = 0; j < xM; j++) //по столбцам

{

cout << xMatr[i][j] << " ";

}//for j

cout << "\n\t";

}//for i

}//Print\_Matr\_Row()

**/\*------------------------------------------------------------\*/**

**/\* печать матрицы \*/**

**/\* по столбцам \*/**

**/\*----------------\*/**

void Print\_Matr\_Col(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM) //число столбцов

{

int i, j;

cout << "\n\t/Print\_Matr\_Col(): ПЕЧАТЬ МАТРИЦЫ ПО СТОЛБЦАМ\n\t";

for (j = 0; j < xM; j++) //по столбцам

{

for (i = 0; i < xN; i++) //по строкам

{

cout << xMatr[i][j] << " ";

}//for i

cout << "\n\t";

}//for j

cout << endl;

}//Print\_Matr\_Col()

**/\*-----------------------------------------------------\*/**

**/\* изменение матрицы \*/**

**/\*-------------------\*/**

void ChangeMatr(int XMatr[][MMAX] //указатель на матрицу

, int Xn //число строк

, int Xm //число столбцов

)

{

int i, j;

cout << "\n\n\t/FROM ChangeMatr: \n";

//меняем и печатаем матрицу

for (i = 0; i<Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

XMatr[i][j] \*= 2; //удвоили значение

cout << XMatr[i][j] << "\t"; //через индекс

}//for j

cout << "\n";

}//for i

} //end ChangeMatr()

**/\*-----------------------------------------------------\*/**

**/\* сумма элементов \*/**

**/\* матрицы \*/**

**/\*-----------------\*/**

//возвращаемый параметр - имя функции Sum\_Matr\_1

int Sum\_Matr\_1(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM) //число столбцов

{

int SumMatr = 0;

int i, j;

for (i = 0; i < xN; i++) //по строкам

{

for (j = 0; j < xM; j++) //по столбцам

{

SumMatr = SumMatr + xMatr[i][j];

}//for j

}//for i

return SumMatr; //результат в имени функции

}//Sum\_Matr\_1()

/\*-----------------------------------------------------\*/

**//возвращаемый параметр - xSumMatr**

void Sum\_Matr\_2(const int xMatr[][MMAX], int xN, int xM, int &xSumMatr)

{

int i, j;

xSumMatr = 0; //СУММА ЭЛЕМЕНТОВ МАТРИЦЫ

for (i = 0; i < xN; i++) //по строкам

{

for (j = 0; j < xM; j++) //по столбцам

{

xSumMatr = xSumMatr + xMatr[i][j];

}//for j

}//for i

return; //без значения

}//Sum\_Matr\_2()

**/\*-----------------------------------------------------\*/**

**/\* след матрицы \*/**

**/\*--------------\*/**

int Trace\_Matr(const int xMatr[][MMAX], int xN)

{

int i;

int TraceMatr = 0;

for (i = 0; i < xN; i++)

TraceMatr = TraceMatr + xMatr[i][i];

return TraceMatr;

}//Trace\_Matr()

**/\*-----------------------------------------------------\*/**

**/\* транспонировать матрицу \*/**

**/\* саму в себя \*/**

**/\*-------------------------\*/**

//ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

void Transpon\_Matr(int xMatr[][MMAX], int xM)

{

int i, j;

int Temp;

for (i = 0; i < xM; i++)

{

// for (j = 0; j < xM; j++) //с ошибкой!!!!

//элементы на главной диагонали не трогаем

for (j = i + 1; j < xM; j++) //без ошибки

{

Temp = xMatr[i][j];

xMatr[i][j] = xMatr[j][i];

xMatr[j][i] = Temp;

} // for j

} //for i

}//Transpon\_Matr()

**/\*-----------------------------------------------------\*/**

**/\* печать вектора по \*/**

**/\* вертикали/горизонтали \*/**

**/\*-----------------------\*/**

void Print\_Vec(const int xVec[] //указатель на вектор

, int xM //длина вектора

, int xVert //1 - по вертикали; 0 - по горизонтали

)

{

int i;

for (i = 0; i < xM; i++) //ПЕЧАТЬ ВЕКТОРА

{

cout << xVec[i] << " ";

if (xVert) //по вертикали

cout << "\n\t";

}//for i

cout << endl;

}//Print\_Vec()

**/\*-----------------------------------------------------\*/**

**/\* умножение вектора \*/**

**/\* на вектор \*/**

**/\*-------------------\*/**

//УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

int Mult\_V\_V(const int xVec[] //указатель на 1 вектор

, const int yVec[] //указатель на 2 вектор

, int xM //длина векторов

)

{

int resMult\_VV = 0;

int i;

//умножение вектора на вектор

for (i = 0; i < xM; i++)

{

resMult\_VV = resMult\_VV + xVec[i] \* yVec[i];

}//for i

return resMult\_VV;

}//Mult\_V\_V()

**/\*-----------------------------------------------------\*/**

**/\* умножение матрицы \*/**

**/\* на вектор \*/**

**/\*-------------------\*/**

//УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

void Mult\_Matr\_Vect(const int xMatr[][MMAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

, const int xVec[] //указатель на вектор длина вектора = числу столбцов

, int xResVec[] //указатель на вектор-результат длина вектора = числу строк

)

{

int i, j;

for (i = 0; i < xN; i++)

{

xResVec[i] = 0;

for (j = 0; j < xM; j++)

{

xResVec[i] = xResVec[i] + xMatr[i][j] \* xVec[j];

}//for j

//cout << "\t\t" << xResVec[i] << endl;

}//for i

cout << endl;

}//Mult\_Matr\_Vect()

**/\*-----------------------------------------------------\*/**

**/\* умножение матрицы \*/**

**/\* на матрицу \*/**

**/\*-------------------\*/**

//УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

//A[N][M] \* B[M]K] = R[N][K]

void Mult\_Matr\_Matr(const int xMatr[][MMAX] //указатель на матрицу A[N][M]

, int xN //число строк

, int xM //число строк/столбцов

, const int yMatr[][KMAX] //указатель на матрицу B[M]K]

, int xK //число столбцов

, int xResMatr[][KMAX] //указатель на матрицу-результат R[N][K]

)

{

int i, j, k;

for (i = 0; i < xN; i++) //строка из матрицы A

{

for (k = 0; k < xK; k++) //столбец из матрицы B

{

xResMatr[i][k] = 0;

for (j = 0; j < xM; j++) //по длине векторов

{

xResMatr[i][k] = xResMatr[i][k] + xMatr[i][j] \* yMatr[j][k];

}//for j

}; // for k

}; //for i

}//Mult\_Matr\_Matr()

**/\*\*\*\*\*\*\*\*\*\*\*\* END Of Pr\_05\_Matr\_Func.CPP FILE \*\*\*\*\*\*\*\*\*\*\*\*/**

1. ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

ОПЕРАТОРЫ new и delete

Переменная является *именованной* областью памяти, которая резервируется во время компиляции (статическая область памяти; имя переменной – это значение, которое хранится где-то в памяти; & - адрес), а указатели всего лишь служат псевдонимами (второе имя) для областей памяти, к которым в любом случае можно обратиться напрямую по имени (указатель – переменная, хранящая адрес значения вместо самого значения). Но истинная ценность указателей проявляется тогда, когда для хранения данных выделяется **неименованная** область памяти *во время* выполнения программы (динамическая область памяти). В этом случае указатели становятся единственным способом доступа к этим областям памяти. Для динамического выделения памяти используется оператор new .

Оператору new мы указываем для какого типа данных (сколько байт, фактически) выделяется память. Оператор new ищет блок памяти нужного размера и возвращает адрес этого блока. Выделение памяти происходит **при выполнении** программы. Присваивая этот адрес указателю, мы можем работать с выделенным блоком. Общая форма записи для выделения памяти под основной тип или тип *структура* имеет вид:

ИмяТипа \*ИмяУказателя = new ИмяТипа;

Например:

int \*pn = new int; или int \*pn; pn = new int;

Для *освобождения* выделенной памяти (возврат ее в общую область) используется оператор delete . Освобожденная память может быть повторно использована в других местах другими частями программы. В тексте программы за оператором delete должен следовать указатель на блок памяти, ранее зарезервированный при помощи оператора new . Например:

delete pn;

В данном случае очищается область памяти, на которую указывает pn , но сам pn **не удаляется**. Его можно использовать повторно для указания на другую выделенную область памяти **того же** типа. Операторы new и delete всегда нужно использовать в паре, иначе может произойти утечка памяти, что может привести к останову программы при поиске дополнительной памяти.

Нельзя повторно очищать блок памяти, который уже был освобожден. Результат такого действия неопределен. Нельзя использовать оператор delete для освобождения памяти, созданной при объявлении переменной (статическая память):

delete pn; //допустимо

delete pn; //уже недопустимо

int Temp = 5; //допустимо

int \*pT = &Temp; //допустимо

delete pT //недопустимо, память не выделена

//оператором new

Пример 4.7 Динамическое выделение памяти для переменных

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* КАФЕДРА № 302 1 КУРС ПЯВУ \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_Matrix\_01\_Function \*

\* File Name : Ex04\_Matrix\_011\_Function.CPP \*

\* Language : C/C++ \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 10/03/15 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ И ОСВОБОЖДЕНИЕ \*

\* ПАМЯТИ ДЛЯ ПЕРЕМЕННЫХ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

int main()

{

//стандартная молитва

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

system("color F0"); //экран белый, буквы черные

cout << "\t Оператор NEW для ПЕРЕМЕННЫХ !! \n\n";

//выделение области памяти для значения типа int

//- место хранения значения

int \* pn = new int;

\*pn = 1235; //собственно значение

cout << " int значение \*pn = " << \*pn << "\t\t Ячейка pn = "

<< pn << endl;

double \* pd = new double; //выделение памяти для double

\*pd = 2.718281828; //собственно значение

cout << "double значение \*pd = " << \*pd << "\t\t Ячейка pd = "

<< pd << "\n\n";

cout << "размер pn = " << sizeof pn

<< "\tразмер \*pn = " << sizeof \*pn << "\n";

cout << "размер pd = " << sizeof pd

<< "\tразмер \*pd = " << sizeof \*pd << "\n\n";

//освобождаем память

//очищается область памяти, **НО САМ УКАЗАТЕЛЬ НЕ УДАЛЯЕТСЯ**

delete pn;

cout << " ПОСЛЕ удаления pn - E R R O R ! ! !\n";

**cout << " int значение \*pn = " << \*pn << "\t Ячейка pn = "**

**<< pn << "\n\n"; // E R R O R ! ! !**

cout << endl;

double \* pd2 = pd;

cout <<"double значение \*pd2 = " << \*pd2 << "\t Ячейка pd2 = "

<< pd2 << "\n\n";

//освобождаем адрес, на который укзывал указатель, а НЕ УКАЗАТЕЛЬ !

delete pd2;

cout << " ПОСЛЕ удаления pd2 - E R R O R ! ! !\n";

**//cout <<"double значение \*pd2 = " << \*pd2 <<"\tЯчейка pd2 = "**

**// << pd2 << "\n"; // E R R O R ! ! !**

cout << endl;

cout << "double значение \*pd = " << \*pd << "\t Ячейка pd = "

<< pd << "\n\n";

//**float \*pn = new float; //менять тип pn НЕЛЬЗЯ !!!**

//повторное использование указателя на динамическую память

cout << "\tПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ pn\n";

int Len = 55555;

pn = new int;

\*pn = Len; //через переменную

cout << " int значение \*pn = " << \*pn

<< "\t Ячейка pn = " << pn << endl;

cout << "размер pn = " << sizeof pn

<< "\tразмер \*pn = " << sizeof \*pn << "\n\n";

//освобождаем память

delete pn; //очищается область памяти, НО САМ УКАЗАТЕЛЬ НЕ УДАЛЯЕТСЯ

//**delete pd; //ERROR повторное освобождение pd = pd2**

cout << endl;

system("pause");

return 0;

}//end main()

/\*\*\*\*\*\*\*\*\*\*\*\* END Of Ex04\_07.CPP FILE \*\*\*\*\*\*\*\*\*\*\*\*/

1. СОЗДАНИЕ ДИНАМИЧЕСКИХ МНОГОМЕРНЫХ МАССИВОВ

Предположим, что создается программа, для которой массив может либо потребоваться, либо нет в зависимости от информации, полученной в ходе работы программы. Длина массива будет определяться после запуска программы.

Если массив создается путем объявления, то память под него отводится на этапе компиляции, независимо от того будет использоваться массив в программе или нет. Массив уже встроен в программу и размер его заранее определен. Выделение памяти для массива в процессе компиляции называется *статическим связыванием*. Оператор new позволяет создавать или не создавать массив в зависимости от потребности. Можно даже выбирать размер массива в процессе выполнения программы. Процесс создания массива в ходе выполнения программы называется *динамическим связыванием*, а сам массив – *динамическим*.

При создании динамического массива необходимо указать оператору new тип элементов массива и их количество. Общая форма выделения и присвоения области памяти для массива имеет вид:

ИмяТипа \*ИмяУказателя = new ИмяТипа[ЧислоЭлементов];

В результате выполнения оператора new резервируется блок памяти, достаточный для хранения элементов типа ИмяТипа в количестве ЧислоЭлементов , а указатель ИмяУказателя устанавливается на первый элемент массива.

Например:

int \*pnArr = new int [10]; //блок на 10 значений типа int

Для высвобождения памяти, выделенной под динамический массив с помощью оператора new , используется другая форма оператора delete .

delete [ ] pnArr;

Квадратные скобки указывают, что необходимо высвободить память, отведенную для *целого* массива, а не только *элемента*, на который указывает указатель pnArr (delete pnArr;).

pnArr является указателем на *один* элемент типа int – первый элемент блока из 10 элементов. Так как для динамического массива оператор sizeof не работает, то программист должен сам реализовывать проверку количества элементов массива в программе.

A1

A1

Адрес

Зн 1

Значение

int \*pVec = new int [10]; //вектор

Зн 1

Зн 2

Зн m

**Пример 4.8** Динамическое выделение памяти для массивов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*--------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Lesson\_04\_Массивы\_Структуры\Ex04\_08 \*

\* File Name : Ex04\_08.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 20/04/04 \*

\* Last Revision : 07/09/06 \*

\* Comment(s) : ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ ДЛЯ \*

\* ОДНОМЕРНЫХ МАССИВОВ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> //по стандарту C++

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

// "\n\t new[] ДЛЯ МАССИВА \n"

cout <<"\t new[] FOR ARRAY \n\n";

int n, i;

// " Введите число элементов N = "

cout << " Enter Number of elements N = ";

cin >> n;

//место хранения значения

double \* pdmas = new double [n];// pdmas трактуется как имя массива

// и как указатель на начало массива

// sizeof не работает

//заполнение массива

for (i = 0; i < n; i++)

pdmas[i] = 1.0 / (i+1);

// "\n\tДИНАМИЧЕСКИЙ МАССИВ\n"

cout << "\n\tDYNAMIC ARRAY \n";

// печать массива

for (i = 0; i < n; i++)

{

// " pdmas[ " "] значение = "

cout << " pdmas[" << i << "] value = " << pdmas[i]

// "\t\t8 Ячейка = "

<< "\t\t Location = "<<&pdmas[i]<<"\n";

}

//**функция sizeof() не дает информации о длине массива - pdmas - указатель**

// " Размер (pdmas) = "

cout << " Size of (pdmas) = " << sizeof(pdmas) << "\n\n";

// печать массива через указатель

for (i = 0; i < n; i++)

cout << \*(pdmas+i) << " "<< pdmas[i] << " "; //в строчку

cout << "\n\n";

*/\**

*----------------------------------------*

*delete pdmas; // !!! освобождается весь массив !!!*

*cout <<"\t AFTER delete \n";*

*// печать массива*

*for (i = 0; i < n; i++)*

*{*

*cout << " pdmas[" << i << "] value = " << pdmas[i]*

*<<"\t Location = "<<&pdmas[i]<<"\n";*

*}*

*------------------------------------------*

*\*/*

**delete [] pdmas;**

// "\t ПОСЛЕ delete [] \n";

cout <<"\t AFTER delete [] \n";

// печать массива

for (i = 0; i < n; i++)

{

cout << " pdmas[" << i << "] value = " << pdmas[i]

<<"\t Location = "<<&pdmas[i]<<"\n";

}

return 0;

}

int \*\*ppMatr; //двойной указатель на матрицу n\*m

int \*\*ppMatr; //двойной указатель на матрицу n\*m

ppMatr = new int \* [n];//массив указателей

ppMatr

Аn

int \*\*ppMatr; //двойной указатель на матрицу

ppMatr = new int \* [n];//массив указателей

for (i=0; i < n; ++i)

ppMatr[i] = new int [m];//массив чисел

ppMatr

ppMatr

А1

А2

Аn

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

ppMatr[i][j] \*( \*(ppMatr + i) +j)

pppCube

А1

А2

Аn

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

А1

А1

А2

Аn

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Аk

А2

int \*\*\*pppCube; //тройной указатель на куб k\*n\*m

pppCube = new int \*\* [k];//массив указателей на

//2-мерный массив n\*m

for (l=0; l<k; ++l) //по числу матриц

{ pppCube [l] = new int \* [n];//массив указателей на

// строки

for (i=0;i<n;++i) //по числу строк

pppCube[l][i] = new int [m]; ];//массив чисел

}//for l

А1

А2

Аn

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн 1

Зн m

pppCube[k][i][j] \*(\*(\*(pppCube + k) + i) +j)

**Пример 4.9** Чтение **динамических** массивов разных размерностей из файла, передача массивов в функцию, запись массива в файл.

На занятии рассмотреть часть примера.

Остальное самостоятельно.

Подготовить в текстовом редакторе файл исходных данных **SRC1.TXT**

4

10 11 12 13

3 5

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

2 3 4

31 32 33 34

41 42 43 44

51 52 53 54

64 63 62 61

74 73 72 71

84 83 82 81

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Lesson\_03\_Указатели\_Ссылки\Ex04\_09 \*

\* File Name : Ex04\_09.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 06/06/05 \*

\* Last Revision : 10/10/05 \*

\* Comment(s) : **ДИНАМИЧЕСКИЕ МНОГОМЕРНЫЕ МАССИВЫ** \*

\* УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ \*

\* ЧТЕНИЕ РАЗМЕРА ВЕКТОРА И ВЕКТОРА ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА МАТРИЦЫ И МАТРИЦЫ ИЗ ФАЙЛА \*

\* ЗАПИСЬ МАТРИЦЫ В ФАЙЛ \*

\* ЧТЕНИЕ РАЗМЕРА КУБА И КУБА ИЗ ФАЙЛА \*

\* ПЕРЕДАЧА ВЕКТОРА, МАТИРИЦЫ И КУБА В ПРОЦЕДУРУ \*

\* ВСЕ ФАЙЛЫ ТЕКСТОВЫЕ \*

\* scanf() работает с **адресами** \*

\* printf() работает со **значениями** \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// читаем данные из файла

#include <stdio.h>

#include <conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char fNAME[]="SRC1.dat"; //файл исходных данных

const char fNAME\_Out[]="RES1.dat"; //файл результатов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintVec (int XVec[], int Xm); //(int \*pVec, int Xm);

void PrintMatr (int \*\*ppMatr, int Xn, int Xm);

void PrintCube (int \*\*\*pppCube, int Xk, int Xn, int Xm);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* О С Н О В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

FILE \*fp=NULL; // объявим указатель на входной файл

FILE \*fp\_out=NULL; // объявим указатель на выходной файл

int n; //число строк

int m; //число столбцов

int k; //число плоскостей

int i,j,l;

int TotSum;

fp = fopen (fNAME,"rt"); //открытие входного файла

if (fp == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME);

return 1;

}

**//================================================================**

printf ("\n\t VECTOR \n");

TotSum = 0;

int \*pVec = 0; //указатель на вектор

fscanf (fp,"%d ", &m); //читаем размер вектора

printf ("/1: m = %d \n",m);

pVec = new int[m]; //указатель на массив **чисел**

for (j = 0; j < m; j++) //читаем вектор

{

fscanf (fp, " %d ", pVec+j);

printf ("/2: \*(pVec +%d) = %4d \n", j, \*(pVec+j));

// printf ("/2: \*(pVec +%d) = %4d %4d %4d %4d \n"

// , j, \*(pVec+j), \*(j+pVec)

// , pVec[j] , [j]pVec );

TotSum = TotSum + \*(pVec+j);

}//for j

printf ("\n\tTotSum VEC = %d \n",TotSum);

PrintVec (pVec, m); //печать вектора из процедуры

//освобождение памяти

delete [] pVec;

//==================================================================

printf ("\n\n\t MATRIX \n");

TotSum = 0;

//указатель на указатель, который указывает на целое число

int \*\*ppMatr; //двойной **указатель** на массив целых

fscanf (fp," %d %d ",&n, &m); //читаем размер матрицы

printf ("/3: n = %d, m = %d \n", n,m);

ppMatr=new int **\*** [n]; //**указатель на массив диких указателей**

//на n пустых строк неопределенной длины

printf ("\n ppMatr = %p \n", ppMatr);

for (i=0; i<n; i++)

{

//выделение памяти под каждый элемент строки конкретной длины

ppMatr[i]=new int [m]; //указатель на массив **чисел**

printf (" ppMatr[%d] = %p \n", i, ppMatr[i]);

}

printf ("\n\t READ MATRIX \n");

for (i=0; i<n; i++) //читаем матрицу

{

for (j = 0; j < m; j++)

{

/\*

//работает

fscanf (fp, " %d" , ppMatr[i]+j); //считать элемент

printf ("%4d " , \*(ppMatr[i]+j)); //эхо-печать

\*/

/\*

//работает

fscanf (fp, " %d", \*(ppMatr+i)+j );

printf ("%4d " , \*(\*(ppMatr+i)+j));

TotSum += \*(\*(ppMatr+i)+j);

\*/

/\*

//работает

fscanf (fp, " %d", &ppMatr[i][j]); //считать элемент

printf ("%4d " , ppMatr[i][j]); //эхо-печать

TotSum = TotSum + ppMatr[i][j];

\*/

//работает

**fscanf (fp, " %d", &ppMatr[i][j]); //считать элемент**

**printf ("%4d" , ppMatr[i][j]); //эхо-печать**

}//for j

printf ("\n");

}//for i

// printf ("\n\tTotSum MATR = %d \n\n",TotSum);

/\*

for (i=0; i<n; i++) //выделенная память

{

printf (" ppMatr[%d] = %p \n", i, ppMatr[i]);

}

\*/

PrintMatr (ppMatr, n, m);

**//------------------------------------------------------------------**

**//запись результатов в файл**

// fp\_out = fopen (fNAME\_Out,"wt");

// if (fp\_out == NULL)

if ((fp\_out = fopen (fNAME\_Out,"wt")) == NULL)

{

printf("\a\n\tCannot open file %s ! \n\n", fNAME\_Out);

return 1;

}

// пишем float матрицу в файл

for (i=0; i<n; i++)

{

for (j = 0; j < m; j++)//форматная запись в файл

fprintf (fp\_out, " %5f", 2\*ppMatr[i][j]+i/10.+j/100.);

fprintf (fp\_out, "\n");

}//for i

fclose(fp\_out); // закрываем выходной файл

**//-----------------------------------------------------------------**

**//освобождение памяти в обратном порядке**

printf ("\n ERASE MEMORY ppMatr\n");

for (i=0; i<n; i++)

{

// printf (" delete [] ppMatr[%d] = %p \n", i, ppMatr[i]);

delete [] ppMatr[i]; //удаляем строки (массивы)

}

delete [] ppMatr; //удаляем массив указателей на строки

**//==================================================================**

**printf ("\n\n\t CUBE \n");**

int **\*\*\***pppCube; //тройной **указатель** на трехмерный массив целых

TotSum = 0;

fscanf (fp," %d %d %d",&k, &n, &m); //читаем размер матрицы

printf ("/5: k = %d, n = %d, m = %d \n", k,n,m);

pppCube = new int **\*\*** [k]; //массив **указателей** на плоскости (матрицы)

// printf ("\n new \*\* pppCube = %p \n", pppCube);

for (l = 0; l<k; l++) //по числу матриц

{

pppCube[l] = new int **\*** [n]; //массив **указателей** на строки

// printf ("\tnew \* pppCube[%d] = %p \n\n", l, pppCube[l]);

for (i=0; i<n; i++) //по числу строк

{

pppCube[l][i] = new int [m]; //указатель на столбец **чисел**

// printf (" \t\tnew \* pppCube[%d][%d] = %p \n"

// , l, i, pppCube[l][i]);

for (j=0; j<m; j++) //по числу столбцов

{

fscanf (fp, "%d ", \*(\*(pppCube + l) + i) +j );

printf ("%4d " , \*(\*(\*(pppCube + l) + i) +j) );

} //for j

printf ("\n");

} //for i

printf ("\n");

} //for l

PrintCube (pppCube, k, n, m);

**//-----------------------------------------------------------------**

**//освобождение памяти в обратном порядке**

printf ("\n ERASE MEMORY pppCube\n");

for (l=0; l<k; l++) //по числу матриц

{

for (i=0; i<n; i++) //по числу строк

{

//printf (" delete [] pppCube[%d][%d] = %p \n", l, i, pppCube[l][i]);

**delete [] pppCube[l][i]; //удаляем строки (массивы)**

}

// printf (" delete [] pppCube[%d] = %p \n\n", l, pppCube[l]);

**delete [] pppCube[l];** //указатель на плоскости

}

// printf ("\n delete [] \*\* pppCube = %p \n", pppCube);

**delete [] pppCube;** //удаляем массив указателей на плоскости

// printf ("\n");

//==================================================================

fclose(fp); // закрываем файл исходных данных

getch(); // ждем

return 0; // выходим из программы

}//end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-----------------\*/

/\* Печать вектора \*/

/\*---------------------------------------------------------------------\*/

void PrintVec (int \*pXVec, int Xm)

{

int j;

printf ("\n/PrintVec:\tRECEIVE VECTOR \n");

printf ("/PrintVec: Xm = %d \n",Xm);

for (j = 0; j < Xm; j++) //читаем вектор

printf (" %d ", \*(pXVec+j));

printf ("\n");

} //end PrintVec()

/\*-----------------\*/

/\* Печать матрицы \*/

/\*---------------------------------------------------------------------\*/

void PrintMatr (int \*\*ppXMatr, int Xn, int Xm)

{

int i, j;

printf ("\n\n/PrintMatr: RECEIVE MATRIX \n\n");

for (i=0; i<Xn; i++) //читаем матрицу

{

for (j = 0; j < Xm; j++)

{

printf ("%4d", \*(\*(ppXMatr+i)+j)); //эхо-печать

}//for j

printf ("\n");

}//for i

} //end PrintMatr()

/\*-------------\*/

/\* Печать куба \*/

/\*---------------------------------------------------------------------\*/

void PrintCube (int \*\*\*pppXCube, int Xk, int Xn, int Xm)

{

int i,j,l;

printf ("\n\n/PrintCube: RECEIVE CUBE \n\n");

for (l=0; l<Xk; l++) //число плоскостей

//читаем матрицу

{

for (i = 0; i < Xn; i++)

{

for (j = 0; j < Xm; j++)

{

printf ("%4d", pppXCube[l][i][j]); //эхо-печать

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

} //end PrintCube()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of file Ex04\_09.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**НА ДОМ**: разобрать пример с чтением из файла и заполнением **статических** массивов разных размерностей.

**Пример 4.10Дом** Чтение массивов разных размерностей из файла, передача массивов в функцию, запись массива в файл.

File Name :Ex04\_10\_стат\_масс\_файл.cpp

Подготовить в текстовом редакторе файл исходных данных **SRC1.TXT**

4

10 11 12 13

3 5

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

2 3 4

31 32 33 34

41 42 43 44

51 52 53 54

64 63 62 61

74 73 72 71

84 83 82 81

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\1 КУРС\MATRIX\ Ex04\_10 \*

\* File Name : Ex04\_10\_стат\_масс\_файл.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 11/03/05 \*

\* Last Revision : 10/10/05 \*

\* Comment(s) : **МАССИВЫ СТАТИЧЕСКИЕ \***

**\***  ЧТЕНИЕ РАЗМЕРА ВЕКТОРА И ВЕКТОРА ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА МАТРИЦЫ И МАТРИЦЫ ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА КУБА И КУБА ИЗ ФАЙЛА \*

\* ЗАПИСЬ МАТРИЦЫ В ФАЙЛ \*

\* УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ \*

\* ПЕРЕДАЧА ВЕКТОРА, МАТИРИЦЫ И КУБА В ПРОЦЕДУРУ \*

\* ВСЕ ФАЙЛЫ ТЕКСТОВЫЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// читаем данные из айла

#include <stdio.h>

#include <conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const N\_MAX = 10; //максимальное число строк в матрице

const M\_MAX = 10; //максимальное число столбцов в строке (векторе)

const K\_MAX = 10; //максимальное число матриц в кубе

const char fNAME[]="SRC1.dat"; //файл исходных данных

const char fNAME\_Out[]="RES1.dat"; //файл результатов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintVec (int XVec[], int Xm); //через индексы

void PrintMatr (int XMatr[][M\_MAX], int Xn, int Xm); //через индексы

void PrintMatrP (int \*pXM, int Xn, int Xm); //через указатель

void PrintCube (int XCube[][N\_MAX][M\_MAX], int Xk, int Xn, int Xm);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е П Е Р Е М Е Н Н Ы Е \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* О С Н О В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

FILE \*fp=NULL; // объявим указатель на входной файл

FILE \*fp\_out=NULL; // объявим указатель на выходной файл

int n; //число строк

int m; //число столбцов

int k; //число плоскостей

int AVec[M\_MAX]; //вектор

int AMatr[N\_MAX][M\_MAX]; //матрица

int ACube[N\_MAX][M\_MAX][K\_MAX]; //куб

int i,j,l;

int TotSum; //сумма элементов массива

fp = fopen (fNAME,"rt"); //открытие входного файла

if (fp == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME);

return 1;

}

**//================================================================**

**printf ("\n\t VECTOR \n");**

TotSum = 0; //сумма элементов массива

fscanf (fp,"%d ", &m); //читаем размер вектора

printf ("/1: m = %d \n",m); //эхо-печать

//читаем вектор

for (j = 0; j < m; j++) //по числу столбцов вектор-строки

{

fscanf (fp, " %d ", &AVec[j]);

printf ("/2: AVec[%d] = %d \n", j, AVec[j]);

TotSum = TotSum + AVec[j];

}//for j

printf ("\n\tTotSum VEC = %d \n",TotSum); //заголовок

PrintVec (AVec, m); //печать вектора из процедуры

**//==================================================================**

**printf ("\n\n\t MATRIX \n");**

TotSum = 0; //сумма элементов массива

fscanf (fp," %d %d ",&n, &m); //читаем размер матрицы

printf ("/3: n = %d, m = %d \n", n,m); //эхо-печать

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

fscanf (fp, " %d", &AMatr[i][j]); //считать элемент

printf ("%4d", AMatr[i][j]); //эхо-печать

TotSum = TotSum + AMatr[i][j];

}//for j

printf ("\n");

}//for i

printf ("\n\tTotSum MATR = %d \n",TotSum);

PrintMatr (AMatr, n, m);

// PrintMatrP (AMatr, n, m); //можно через указатель

**//-----------------------------------------------------------------**

**// УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ**

int \*pint; //указатель на целое число

int Sum3 = 0; //сумма элементов матрицы, кратных 3

//инициализация указателя

pint = &AMatr[0][0]; //адресом первого элемента

// pint = AMatr; //адресом начала массива

printf ("/Pointer AMatr \n");

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

if ((\*(pint + i\*M\_MAX + j)) % 3 == 0) //числа, кратные 3

{

// Sum3 += AMatr[i][j]; //через индекс

Sum3 += \*(pint + i\*M\_MAX + j); //через указатель

//эхо-печать чисел, кратных 3

printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j], \*(pint + i\*M\_MAX + j));

}//if

//эхо-печать всех чисел

// printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j], \*(pint + i\*M\_MAX + j));

}//for j

// printf ("\n");

}//for i

printf ("\n\tSum3 MATR = %d \n",Sum3);

getch(); //бряк по клавише без эха на экране

**//-----------------------------------------------------------**

**// int (\*pVec)[M\_MAX]; //указатель на массив из M\_MAX элементов**

/\*

int \*\*pAM; //указатель на указатель, указывающий на целое

int Sum3p = 0; //сумма элементов матрицы, кратных 3

\*pAM = &AMatr[0]; //указатель на массив строк

printf ("/Pointer AMatr \n");

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

if ((\*(pint + i\*M\_MAX + j)) % 3 == 0) //числа, кратные 3

{

// Sum3 += AMatr[i][j]; //через индекс

Sum3 += \*(pint + i\*M\_MAX + j); //через указатель

//эхо-печать чисел, кратных 3

printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j]

, \*(pint + i\*M\_MAX + j));

}//if

//эхо-печать всех чисел

// printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j]

, \*(pint + i\*M\_MAX + j));

}//for j

// printf ("\n");

}//for i

printf ("\n\tSum3 MATR = %d \n",Sum3);

getch(); //бряк по клавише без эха на экране

\*/

**//==================================================================**

**//работа с трехмерным массивом**

printf ("\n\n\t CUBE \n");

TotSum = 0; //сумма элементов массива

fscanf (fp," %d %d %d",&k, &n, &m); //читаем размер матрицы

printf ("/5: k = %d, n = %d, m = %d \n", k,n,m);

for (l=0; l<k; l++) //по числу плоскостей

//читаем матрицу

{

for (i = 0; i < n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

fscanf (fp,"%d",&ACube[l][i][j]); //через индекс

printf ("%4d", ACube[l][i][j]); //эхо-печать

TotSum = TotSum + ACube[l][i][j];

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

printf ("\n\tTotSum CUBE = %d \n",TotSum);

PrintCube (ACube, k, n, m);

//==================================================================

fclose(fp); // закрываем файл

**//==================================================================**

**//запись результатов в файл**

// fp\_out = fopen (fNAME\_Out,"wt");

// if (fp\_out == NULL)

if ((fp\_out = fopen (fNAME\_Out,"wt")) == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME\_Out);

return 1;

}

// пишем матрицу в файл

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

//форматная запись в файл

fprintf (fp\_out, " %5d", **2\*AMatr[i][j]**); //поменяли значения

}//for j

fprintf (fp\_out, "\n");

}//for i

fclose(fp\_out); // закрываем выходной файл

getch(); // ждем бряк по клавише без эха на экране

return 0; // выходим из программы

} //main

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**/\*-----------------\*/**

**/\* Печать вектора \*/**

**/\*---------------------------------------------------------------------\*/**

void PrintVec (int XVec[], int Xm)

{

int j;

printf ("\n/PrintVec:\tRECEIVE VECTOR \n");

printf ("/PrintVec: Xm = %d \n",Xm);

//читаем вектор

for (j = 0; j < Xm; j++) //по числу столбцов вектор-строки

{

printf (" %d ", XVec[j]); //через индекс

}//for j

printf ("\n");

}

**/\*-----------------\*/**

**/\* Печать матрицы \*/**

**/\* через индексы \*/**

**/\*---------------------------------------------------------------------\*/**

void PrintMatr (int XMatr[][M\_MAX], int Xn, int Xm)

{

int i, j;

printf ("\n\n/PrintMatr: RECEIVE MATRIX \n\n");

//читаем матрицу

for (i=0; i<Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", XMatr[i][j]); //через индекс

}//for j

printf ("\n");

}//for i

}

**/\*-----------------\*/**

**/\* Печать матрицы \*/**

**/\* через указатель \*/**

**/\*---------------------------------------------------------------------\*/**

void PrintMatrP (int \*pXM, int Xn, int Xm)

{

int i, j;

printf ("\n\n/PrintMatrP: RECEIVE MATRIX \n\n");

//читаем матрицу

for (i=0; i<Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", pXM+i\*M\_MAX+j); //через указатель

}//for j

printf ("\n");

}//for i

}

**/\*-------------\*/**

**/\* Печать куба \*/**

**/\*---------------------------------------------------------------------\*/**

void PrintCube (int XCube[][N\_MAX][M\_MAX], int Xk, int Xn, int Xm)

{

int i,j,l;

printf ("\n\n/PrintMatr: RECEIVE CUBE \n\n");

for (l=0; l<Xk; l++) //по числу плоскостей

{

for (i = 0; i < Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", XCube[l][i][j]); //через индекс

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of file Ex04\_10.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

<К О Н Е Ц З А Н Я Т И Я № 7>

**КОНЕЦ**