Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 17

По дисциплине «Основы алгоритмизации и программирования»

На тему «Указатели и ссылки при работе с функциями»

Выполнил:

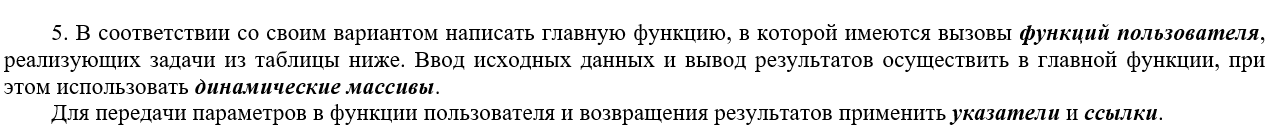
Студент 1 курса 6 группы

Романов Игорь Вячеславович

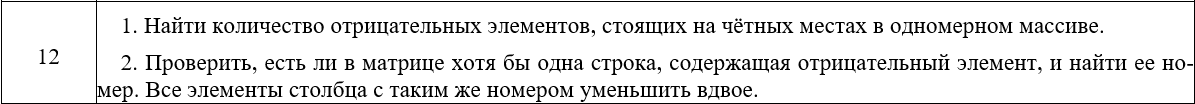
Преподаватель: асс. Андронова М.В.

2023, Минск

Задание 5



Вариант 12 (основной)



1.

#include <iostream>

using namespace std;

//прототип функции

int countNeg(int\* A, int size);

//главная функция

void main()

{

setlocale(LC\_ALL, "Russian");

int size; //ввод размера массива

cout << "Введите размер массива: "; cin >> size;

int\* A = new int[size]; //выделение динамической памяти для массива

cout << "Введите элементы одномерного массива" << endl;

for (int i = 0; i < size; i++) { //ввод элементов с клавиатуры

cout << "Элемент [" << i << "]: "; cin >> \*(A + i);

}

cout << "\nКоличество отрицательных элементов, стоящих на четных местах: " << countNeg(A, size);

delete[] A; //освобождение памяти

}

//функция для подсчета отрицательных элементов

int countNeg(int\* A, int size)

{

int counter\_neg = 0; //ввод счетчика отрицательных элементов

for (int i = 0; i < size; i++) { //если элемент совпадает с условием, то увеличить счетчик

if (i % 2 == 0 && \*(A + i) < 0) {

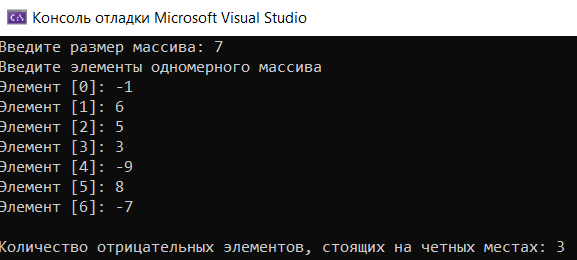
counter\_neg++;

}

}

return counter\_neg; //вернуть значение в главную функцию

}



2.

#include <iostream>

using namespace std;

//прототипы функции

int &returnIndex(float\*\* A, int rows, int cols);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int rows, cols, i, j; //ввод размера строк и столбцов матрицы; переменных итераторов

cout << "Введите размер строк в матрице: "; cin >> rows;

cout << "Введите размер строк в матрице: "; cin >> cols;

//выеделение памяти для создания матрицы

float\*\* A = new float\* [rows];

for (i = 0; i < rows; i++) {

A[i] = new float[cols];

}

//добавление в матрицу элементов с клавиатуры

cout << "Введите элементы матрицы: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

cout << "Элемент (" << i << "," << j << "): "; cin >> A[i][j];

}

}

//вывод матрицы на экран

cout << "Исходная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

//вывод соответствующего сообщения

cout << "Номер строки, содержащий отрицательный элемент: " << returnIndex(A, rows, cols);

//освобождение памяти

for (i = 0; i < rows; i++) {

delete[] A[i];

}

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения отрицательного элемента и его индекса в строке

int &returnIndex(float\*\* A, int rows, int cols)

{

int check = 0, index, i, j;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

if (A[i][j] < 0) {

check++;

index = i;

}

if (check == 1) { //если в строке есть хотя бы 1 отрицательный элемент, то выйти из цикла

break;

}

}

//преобразование матрицы, если нашлась хотя бы одна строка из отрицательных элементов

if (check == 1) {

j = index;

for (i = 0; i < rows; i++) {

A[i][j] /= 2; //элементы соответствующего столбца уменьшить вдвое

}

break;

}

}

//вывод матрицы на экран

cout << "Преобразованная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

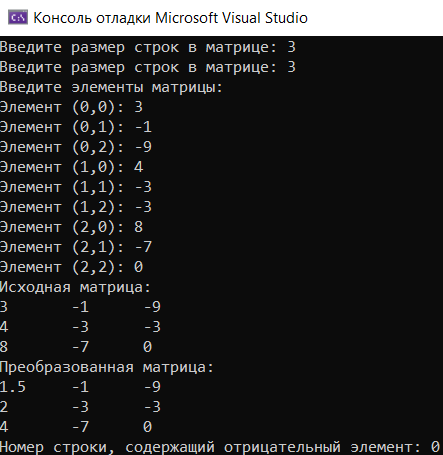
}

cout << "\n";

}

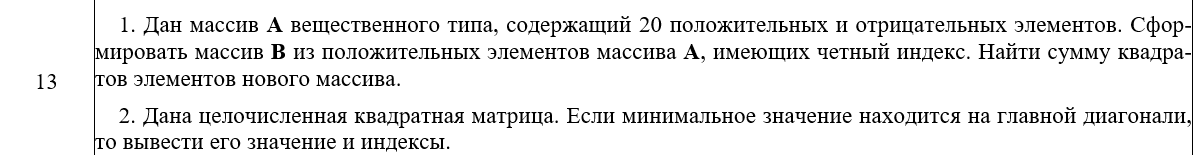
return index; //вернуть значение в главную функцию

}



Дополнительные задания:

Вариант 13



1.

#include <iostream>

using namespace std;

//прототипы функции

int getPos(float\* A, int size);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int size = 20;

float\* A = new float[size]; //выделение динамической памяти для массива

cout << "Введите элементы одномерного массива" << endl;

for (int i = 0; i < size; i++) { //ввод элементов с клавиатуры

cout << "Элемент [" << i << "]: "; cin >> \*(A + i);

}

cout << "Сумма квадратов элементов массива B: " << getPos(A, size); //вывод соответствующего сообщения

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения положительных элементов под четным номером

int getPos(float\* A, int size)

{

int j = 0; //ввод переменной итератора

float square\_pos = 0; //ввод квадрата положительных чисел

float\* B = new float[size]; //выделение динамической памяти для массива

for (int i = 0; i < size; i++) { //нахождение минимального элемента и его позиции в массиве

if (i % 2 == 0 && \*(A + i) > 0) {

\*(B + j) = \*(A + i); //заполнение массива B положительными элементами на четной позиции массива A

square\_pos += \*(B + j) \* \*(B + j); //нахождение квадрата положительных элементов

j++; //увеличение индекса массива B

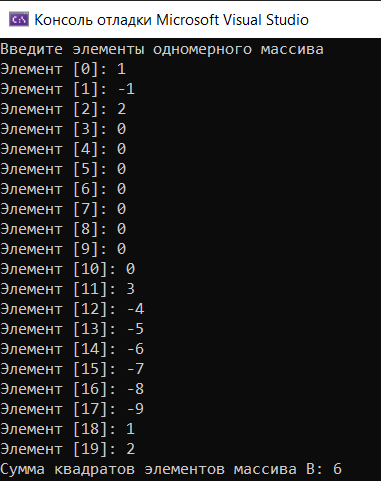
}

}

delete[] B; //освобождение памяти

return square\_pos;; //вернуть значение в главную функцию

}



2.

#include <iostream>

using namespace std;

//прототипы функции

int &returnIndex(int\*\* A, int rows, int cols);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int rows, cols, i, j; //ввод размера строк и столбцов матрицы; переменных итераторов

cout << "Введите размер строк в матрице: "; cin >> rows;

cout << "Введите размер строк в матрице: "; cin >> cols;

//выеделение памяти для создания матрицы

int\*\* A = new int\* [rows];

for (i = 0; i < rows; i++) {

A[i] = new int[cols];

}

//добавление в матрицу элементов с клавиатуры

cout << "Введите элементы матрицы: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

cout << "Элемент (" << i << "," << j << "): "; cin >> A[i][j];

}

}

//вывод матрицы на экран

cout << "Исходная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

//вывод соответствующего сообщения

cout << "Минимальный элемент: " << returnIndex(A, rows, cols);

//освобождение памяти

for (i = 0; i < rows; i++) {

delete[] A[i];

}

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения минимального элемента и его индекса в строке

int &returnIndex(int\*\* A, int rows, int cols)

{

int index\_i, index\_j, i, j, i\_min = A[0][0]; //ввод индексов минимальных элементов, переменных итератора, минимального элемента

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

if (A[i][j] < i\_min) { //если находится минимальный элемент, то запомнить его и его индексы

i\_min = A[i][j];

index\_i = i;

index\_j = j;

}

}

}

//вывод сообщения, если минимальных элемент находится на главной диагонали

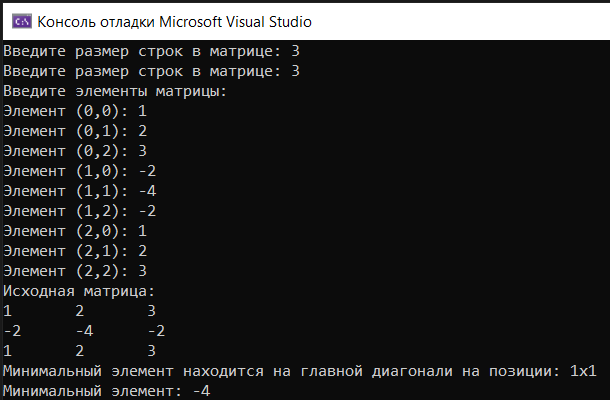
if (index\_i == index\_j) {

cout << "Минимальный элемент находится на главной диагонали на позиции: " << index\_i << "x" << index\_j << endl;

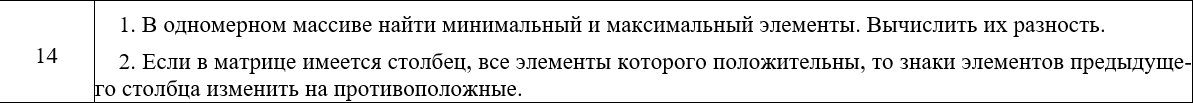
}

return i\_min; //вернуть значение в главную функцию

}



Вариант 14



1.

#include <iostream>

using namespace std;

//прототипы функции

int returnMin(int\* A, int size);

int returnMax(int\* A, int size);

int getDiff(int\* A, int size);

//главная функция

void main()

{

setlocale(LC\_ALL, "Russian");

int size; //ввод размера массива

cout << "Введите размер массива: "; cin >> size;

int\* A = new int[size]; //выделение динамической памяти для массива

cout << "Введите элементы одномерного массива" << endl;

for (int i = 0; i < size; i++) { //ввод элементов с клавиатуры

cout << "Элемент [" << i << "]: "; cin >> \*(A + i);

}

cout << "\nМинимальный элемент массива: " << returnMin(A, size);

cout << "\nМаксимальный элемент массива: " << returnMax(A, size);

cout << "\nРазница между минимальным и максимальным элементом: " << getDiff(A, size);

delete[] A; //освобождение памяти

}

//функция для подсчета минимальных элементов

int returnMin(int\* A, int size)

{

int\* i\_min = A; //ввод минимального элемента

for (int i = 1; i < size; i++) { //если элемент совпадает с условием, то увеличить счетчик

if (\*(A + i) < \*i\_min) {

i\_min = (A + i);

}

}

return \*i\_min; //вернуть значение в главную функцию

}

//функция для подсчета максимальных элементов

int returnMax(int\* A, int size)

{

int\* i\_max = A; //ввод минимального элемента

for (int i = 1; i < size; i++) { //если элемент совпадает с условием, то увеличить счетчик

if (\*(A + i) > \*i\_max) {

i\_max = (A + i);

}

}

return \*i\_max; //вернуть значение в главную функцию

}

//функция для разности элементов

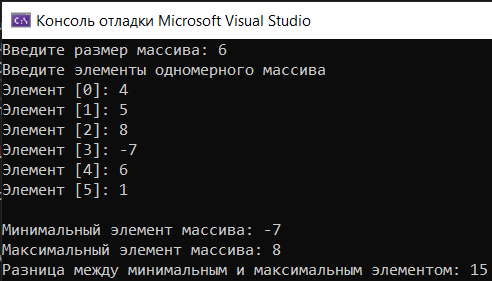
int getDiff(int\* A, int size)

{

int diff = returnMax(A, size) - returnMin(A, size); //ввод разницы элементов

return diff; //вернуть значение в главную функцию

}



2.

#include <iostream>

using namespace std;

//прототипы функции

int &getPos(int\*\* A, int rows, int cols);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int rows, cols, i, j; //ввод размера строк и столбцов матрицы; переменных итераторов

cout << "Введите размер строк в матрице: "; cin >> rows;

cout << "Введите размер строк в матрице: "; cin >> cols;

//выеделение памяти для создания матрицы

int\*\* A = new int\* [rows];

for (i = 0; i < rows; i++) {

A[i] = new int[cols];

}

//добавление в матрицу элементов с клавиатуры

cout << "Введите элементы матрицы: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

cout << "Элемент (" << i << "," << j << "): "; cin >> A[i][j];

}

}

//вывод матрицы на экран

cout << "Исходная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

//вывод соответствующего сообщения

cout << "Индекс столбца, содержащего только положительные элементы: " << getPos(A, rows, cols) << endl;

//вывод преобразованной матрицы

cout << "Преобразованная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

//освобождение памяти

for (i = 0; i < rows; i++) {

delete[] A[i];

}

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения положительных элементов в столбце

int &getPos(int\*\* A, int rows, int cols)

{

int check, i, j, index = 0; //ввод счетчика положительных чисел, переменных итераторов, номер столбца полож. элементов

for (j = 0; j < cols; j++) {

check = 0; //по умолчанию считаются, что все элемент отрицательные

for (i = 0; i < rows; i++) {

if (A[i][j] > 0) { //если в столбце находится положительный элемент, то увеличить счетчик и запомнить номер строки

check++;

index = j;

}

}

if (check == rows) { //если в столбце все элементы положительные,

if (index == 0) { //если положительные элементы находятся в первом столбце, то поменять значения в последнем

index = cols;

}

for (i = 0; i < rows; i++) { //то в предыдущем столбце заменить элементы на противоположные

A[i][index - 1] \*= -1;

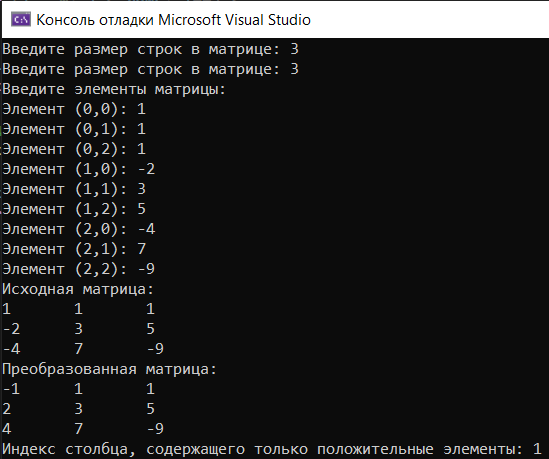
}

}

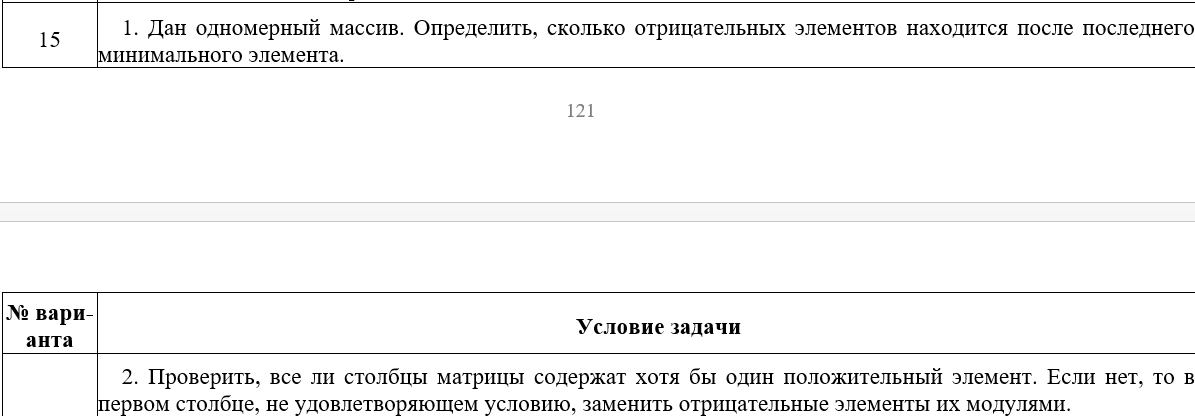
}

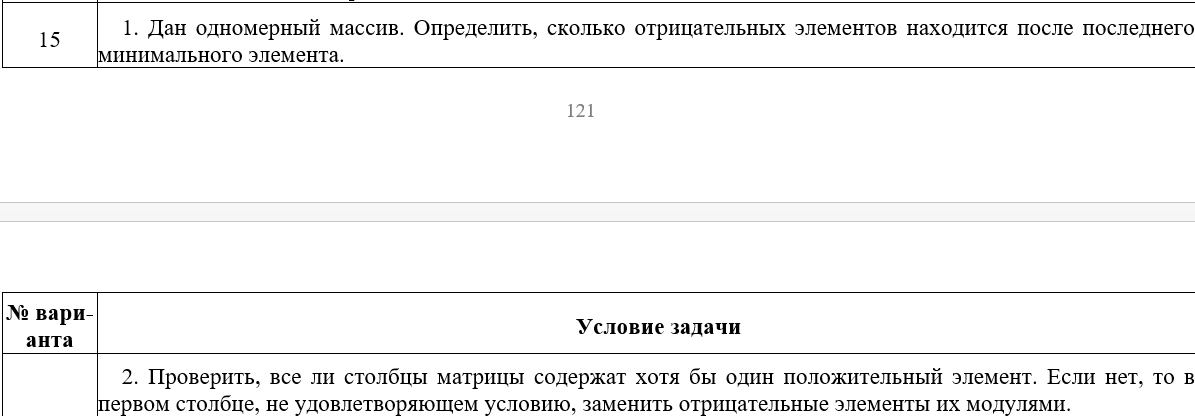
return index; //вернуть значение в главную функцию

}



Вариант 15.





1.

#include <iostream>

using namespace std;

//прототипы функции

int returnMin(int\* A, int size);

int countNeg(int\* A, int size);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int size; //ввод размера массива

cout << "Введите размер массива: "; cin >> size;

int\* A = new int[size]; //выделение динамической памяти для массива

cout << "Введите элементы одномерного массива" << endl;

for (int i = 0; i < size; i++) { //ввод элементов с клавиатуры

cout << "Элемент [" << i << "]: "; cin >> \*(A + i);

}

cout << "\nКоличество отрицательных чисел после последнего минимального элемента: " << countNeg(A, size);

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения вхождения минимального элемента

int returnMin(int\* A, int size)

{

int pos\_min;

int\* i\_min = A; //ввод минимального элемента

for (int i = 1; i < size; i++) { //нахождение минимального элемента и его позиции в массиве

if (\*(A + i) < \*i\_min) {

i\_min = (A + i);

pos\_min = i;

}

}

return pos\_min; //вернуть значение в главную функцию

}

//функция для подсчета отрицательных элементов

int countNeg(int\* A, int size)

{

int count\_neg = 0; //ввод счетчика отрицательных чисел

for (int i = returnMin(A, size) + 1; i < size; i++) {

if (\*(A + i) < 0) {

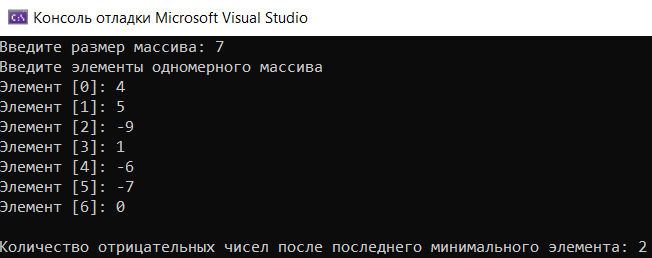
count\_neg++;

}

}

return count\_neg; //вернуть значение в главную функцию

}



2.

#include <iostream>

using namespace std;

//прототип функции

int &getPos(int\*\* A, int rows, int cols);

//главная функция

int main()

{

setlocale(LC\_ALL, "Russian");

int rows, cols, i, j; //ввод размера строк и столбцов матрицы; переменных итераторов

cout << "Введите размер строк в матрице: "; cin >> rows;

cout << "Введите размер строк в матрице: "; cin >> cols;

//выделение памяти для создания матрицы

int\*\* A = new int\* [rows];

for (i = 0; i < rows; i++) {

A[i] = new int[cols];

}

//добавление в матрицу элементов с клавиатуры

cout << "Введите элементы матрицы: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

cout << "Элемент (" << i << "," << j << "): "; cin >> A[i][j];

}

}

//вывод матрицы на экран

cout << "Исходная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

//вывод соответствующего сообщения

cout << "Индекс столбца, содержащего только отрицательные элементы: " << getPos(A, rows, cols);

//освобождение памяти

for (i = 0; i < rows; i++) {

delete[] A[i];

}

delete[] A; //освобождение памяти

return 0;

}

//функция для нахождения положительных элементов в столбце

int &getPos(int\*\* A, int rows, int cols)

{

int check, i, j, index = 0; //ввод счетчика положительных чисел, переменных итераторов, номер столбца отриц. элементов

for (j = 0; j < cols; j++) {

check = 0; //по умолчанию считаются, что все элементы отрицательные

for (i = 0; i < rows; i++) {

if (A[i][j] > 0) { //если в столбце находится хотя бы один положительный элемент,

check = 1; //то присвоить счетчику значение истина

break; //выйти из цикла

}

}

if (check == 0) { //если в столбце все элементы отрицательные

for (i = 0; i < rows; i++) { //то в этом столбце заменить элементы на противоположные

A[i][j] \*= -1;

}

break;

}

index++; //увеличение счетчика столбцов

}

//вывод преобразованной матрицы

cout << "Преобразованная матрица: " << endl;

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

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

}

cout << "\n";

}

return index; //вернуть значение в главную функцию

}

