НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»»

ИНСТИТУТ ДИСТАНЦИОННОГО И ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированный анализ и программирование»

Тема: *«Проектирование приложений обработки матриц в среде MS Visual Studio 2008»*

Студент группы: Арещенко С. Ф.
(Ф.И.О.)

Руководитель: доцент Батасова В. С.
(уч. степень, звание, Ф.И.О.)

Сдана на проверку		
Возвращена на доработку		
Допущена к защите		
Оценка		

Оглавление

Введение 3
Условие задачи 3
Выбор функций, назначение функций 3
Шаблоны функции: 3
void input(int n, int m, float mat[][7]) 3
float* search(int n, int m, float mat[][7]) 3
void output(float* Sup) 3
Описание данных функции main 4
Форма ввода для функции input ..... **Ошибка! Закладка не определена.**
Форма вывода для функции output ..... **Ошибка! Закладка не определена.**
Блок-схема алгоритма функции main 5
Описание данных функции обработки 6
Блок схема алгоритма функции search 6
Код консольного приложения с подпрограммами 8
Изменения в коде под динамические массивы **Ошибка! Закладка не определена.**
Многофайловая программа ..... **Ошибка! Закладка не определена.**
Описание программы с использованием классов 12
Оконное приложение 15
Код формы с комментариями 16
Тесты 18
Заключение 20
Было создано: ..... **Ошибка! Закладка не определена.**

Введение

Целью курсовой работы является приобретение навыков проектирования алгоритмов обработки матриц, кодирования их на алгоритмическом языке C++ и реализация в среде MS Visual Studio:

В ходе выполнения курсовой работы будут созданы:

- консольные приложения с подпрограммами, с локальными и динамическими массивами (версии 1 и 2 консольного приложения);
- консольные приложения с классами (версии 3 и 4 консольного приложения);
- оконное приложение.

Условие задачи

Определить индексы максимального элемента (в предположении, что он единственный) в каждой из трех матриц A (7×5), B (3×7), C (4×5).

Раздел 1. Решение задачи с использованием функций

Выбор функций, назначение функций

Алгоритм вычисления индексов максимального элемента матрицы одинаков (с точностью до обозначений и констант) для A, B, C. Поэтому вычисление индексов максимального элемента произвольной матрицы следует оформить как функцию (назовем ее search), а затем применить ее три раза к конкретным (фактическим) матрицам A, B, C. Алгоритм ввода одинаков для всех матриц, поэтому ввод также оформим как функцию (назовем ее input). Аналогично вывод результатов будет осуществлять функция output.

Шаблоны функций:

- функция, с помощью которой происходит ввод матриц, имеет шаблон:

```
void input(int n, int m, float mat[][7]);
```

- Функция, с помощью которой происходит поиск значения и индексов максимального элемента заданной матрицы, имеет шаблон:

```
float* search(int n, int m, float mat[][7]);
```

- Функция, которая выводит значение и индексы максимального элемента, записанные в массив Sup из трех элементов, имеет шаблон:

```
void output(float* Sup);
```

Состав данных функции main

Имя	Смысл	Тип	Структура
Входные данные			
A	заданная матрица	float (вещественный)	двумерный массив с 7 столбцами
B	заданная матрица	float (вещественный)	двумерный массив с 7 столбцами
C	заданная матрица	float (вещественный)	двумерный массив с 7 столбцами
Выходные данные			
Sup	Значение и индексы максимального элемента матрицы	float (вещественный)	динамический массив

Блок-схема алгоритма функции main

Блок-схема алгоритма функции main, представлена на рисунке 1.

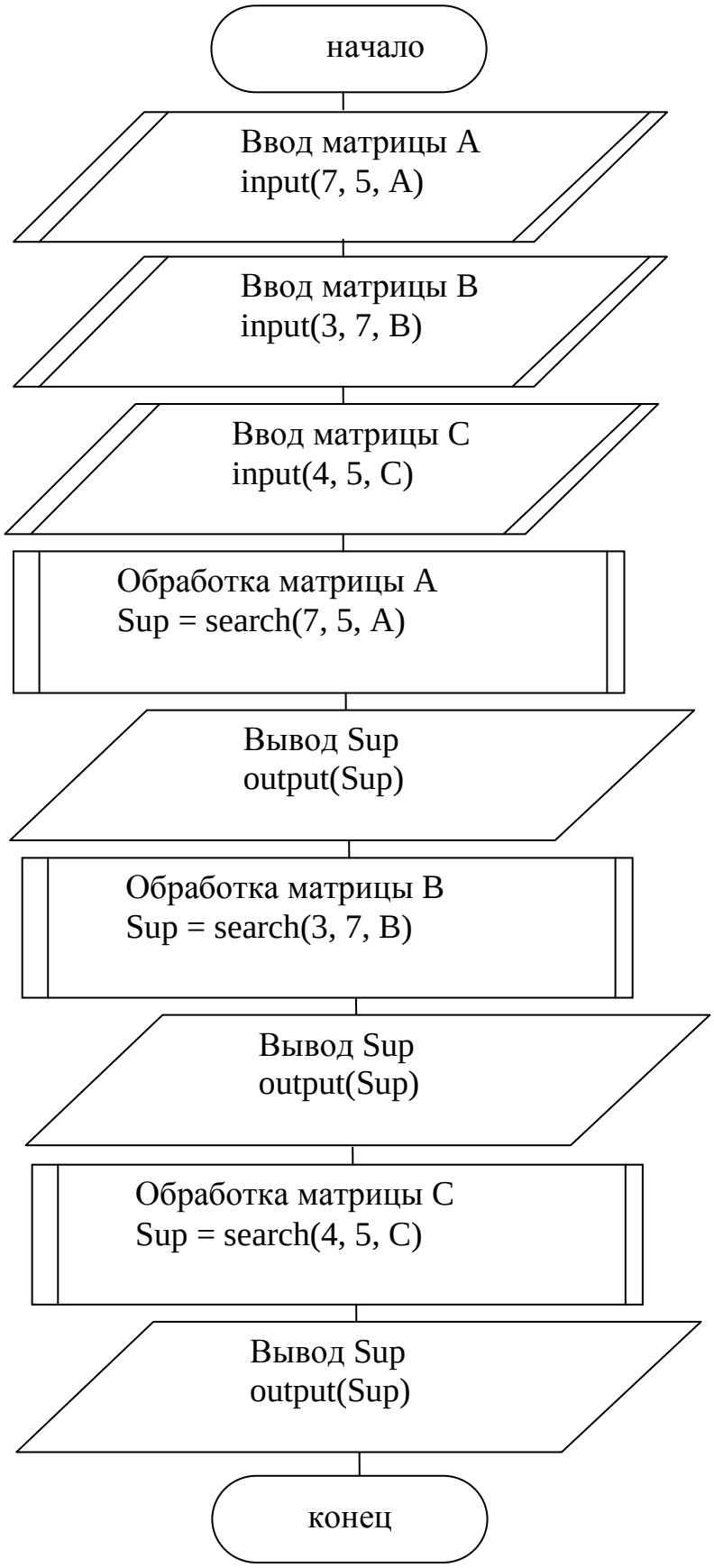


Рис. 1. Блок-схема функции main

Описание данных функции обработки

Описание данных функции search, представлены в таблице 2.

Имя	Смысл	Тип	Структура
Входные данные			
mat	заданная матрица	float (вещественный)	двумерный массив с 7 столбцами
n	количество строк	int (целый)	простая переменная
m	количество столбцов	int (целый)	простая переменная
Выходные данные			
Sup	Значение и индексы максимального элемента д матрицы	float (вещественный)	динамический массив
Промежуточные данные			
max	максимальный элемент	float (вещественный)	простая переменная
mini	индекс строки максимального элемента	int (целый)	простая переменная
minj	индекс столбца максимального элемента	int (целый)	простая переменная
i	индекс строки (текущий)	int (целый)	простая переменная
j	индекс столбца (текущий)	int (целый)	простая переменная

Таблица 1

Блок схема алгоритма функции search

Блок-схема алгоритма функции search, представлена на рисунке 2.

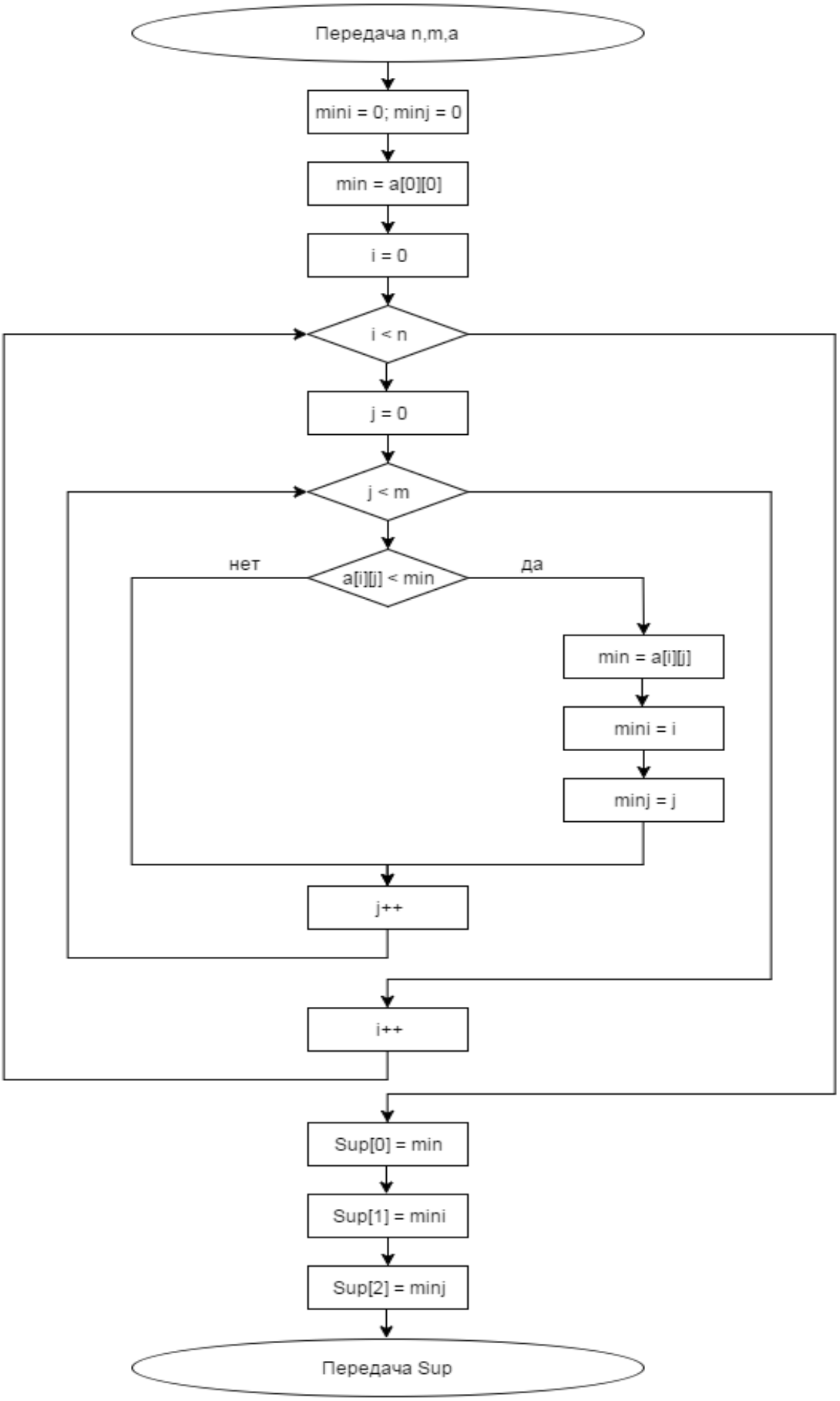


 Рис. 2. Блок-схема функции search

Код консольного приложения с подпрограммами (версия 1)

```
#include <iostream>
#include <conio.h>
#include <ctime>

using namespace std;

/*Функция ввода*/
void input(int n, int m, float mat[][7])
{
	int i, j;

	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
		{
			cin >> mat[i][j];
		}
		cout << endl;
	}
}

/*Функция обработки*/
float* search(int n, int m, float mat[][7])
{
	int i, j;
	float *Sup;
	Sup = new float[3];
	int mini=0, minj=0;
	float max = mat[0][0];
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			if (mat[i][j] < max)
			{
				max = mat[i][j];
				mini = i;
				minj = j;
			}
	Sup[0] = max;
	Sup[1] = mini;
	Sup[2] = minj;
	return Sup;
}

/*Функция вывода индексов*/
void output(float* Sup)
{
	cout << "Min = " << Sup[0] << endl;
	cout << "i = " << Sup[1] << endl;
	cout << "j = " << Sup[2] << endl;
}

int main()
{
	float A[7][7], B[7][7], C[7][7];		//A[7][5], B[3][7], C[4][5]
	float *Sup;

	cout << "Matrix A" << endl;
		input(7, 5, A);
	cout << "Matrix B" << endl;
		input(3, 7, B);
```

```
	cout << "Matrix C" << endl;
		input(4, 5, C);

	cout << "Min element in matrix A:\n";
		Sup = search(7, 5, A);
		output(Sup);
	cout << "\nMin element in matrix B:\n";
		Sup = search(3, 7, B);
		output(Sup);
	cout << "\nMin element in matrix C:\n";
		Sup = search(4, 5, C);
		output(Sup);

	_getch();
	return 0;
}
```

Код консольного приложения с динамической матрицей (версия 2)

```
/*Функция ввода и выделения памяти под массив*/
float** input(int n, int m)
{
	int i, j;
	float **mat;
	mat = new float* [n];
	for (i = 0; i < n; i++)
	{
		mat[i] = new float[m];
		for (j = 0; j < m; j++)
		{
			cin >> mat[i][j];
		}
		cout << endl;
	}
	return mat;
}

/*Функция освобождения памяти*/
void delMat(float** mat, int n)
{
	for (int i = 0; i < n; i++)
	{
		delete[] mat[i];
	}
	delete[] mat;
}

/*Функция обработки*/
float* search(int n, int m, float **mat)
{
	int i, j;
	float *Sup;
	Sup = new float[3];
	int mini=0, minj=0;
	float max = mat[0][0];
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			if (mat[i][j] > max)
			{
				max = mat[i][j];
				mini = i;
				minj = j;
			}
```

```
	Sup[0] = min;
	Sup[1] = mini;
	Sup[2] = minj;
	return Sup;
}


/*Функция main*/
int main()
{
	float **A, **B, **C;	//A[7][5], B[3][7], C[4][5]
	float *Sup;

	cout << "Matrix A" << endl;
	A = input(7, 5);
	cout << "Matrix B" << endl;
	B = input(3, 7);
	cout << "Matrix C" << endl;
	C = input(4, 5);

	cout << "Max element in matrix A:\n";
	Sup = search(7, 5, A);
	output(Sup);
	cout << "\nMax element in matrix B:\n";
	Sup = search(3, 7, B);
	output(Sup);
	cout << "\nMax element in matrix C:\n";
	Sup = search(4, 5, C);
	output(Sup);

	delMat(A,7);
	delMat(B,3);
	delMat(C,4);

	_getch();
	return 0;
}
```

Раздел 2. Консольная программа с использованием класса (версия 3)

Объявление класса начинается с зарезервированного ключевого слова class, после которого написано имя этого класса – maker. Далее объявлен спецификатор доступа private, дающий доступ к полям класса только методам текущего класса. Для методов класса объявлен спецификатор доступа public, то есть методы можно использовать в любой точке программы, где доступно описание класса. Конструктор maker(имя конструктора такое же, как и имя класса) - специальный метод класса, предназначенный для инициализации элементов класса и распределения памяти под поля класса. Деструктор ~maker - специальный метод класса, который служит для уничтожения класса. Методы (функции) класса, нужные для данной задачи: input – функция ввода матрицы, search – функция для определения индексов максимального элемента матрицы, функция output выводящая в консольный экран вычисленные индексы.

```
#include <iostream>
#include <conio.h>
#include <ctime>

using namespace std;

class maker
{
private:
	float **mat;
	int n, m;
	char c;
	float max;
	int mini, minj;
public:
	maker(int n, int m, char c);
	~maker();
	void input();
	void output();
	void search();
};

maker::maker(int n, int m, char c)
{
	this->n = n;
	this->m = m;
	this->c = c;
	mat = new float*[n];
	for (int i = 0; i < n; i++)
	{
		mat[i] = new float[m];
	}
}

void maker::input()
{
	int i, j;
	cout << "Matrix " << c << endl;
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
		{
			cin >> mat[i][j];
		}

	}
}

void maker::output()
{
	cout << "\nMax element in matrix " << c << ":\n";
	cout << "Max = " << max << endl;
	cout << "i = " << mini + 1 << " j = " << minj + 1 << endl;
}

maker::~maker()
{
	for (int i = 0; i < n; i++)
	{
		delete[] mat[i];
	}
	delete[] mat;
}
```

```
void maker::search()
{
	int i, j;
	mini = 0, minj = 0;
	max = mat[0][0];
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			if (mat[i][j] > max)
			{
				max = mat[i][j];
				mini = i;
				minj = j;
			}
}

void main()
{
	maker *A, *B, *C;;		//A[7][5], B[3][7], C[4][5]

	A = new maker(7, 5, 'A');
	B = new maker(3, 7, 'B');
	C = new maker(4, 5, 'C');

	A->input();
	B->input();
	C->input();

	cout << endl;

	A->search();
	A->output();

	B->search();
	B->output();

	C->search();
	C->output();

	_getch();
}
```

Раздел 3. Определение класса-наследника

В классе-предке maker уровень доступа к полям класса изменен с private на protected, чтобы доступ к полям класса имели не только методы текущего класса, но и его классов-наследников. Методы класса maker по-прежнему имеют уровень доступа public.

Создан производный класс (наследник) класса maker, его имя newmaker. В класс-наследник добавлен новый метод – summa, который ищет сумму элементов матрицы. В наследнике также был переопределен метод search, который теперь ищет индексы минимального элемента матрицы.

```
#include <iostream>
#include <conio.h>
#include <ctime>
```

```
using namespace std;

class maker
{
protected:
	float **mat;
	int n, m;
	char c;
	float max;
	int mini, minj;
public:
	maker(int n, int m, char c);
	~maker();
	void input();
	void output();
	void search();
};

maker::maker(int n, int m, char c)
{
	this->n = n;
	this->m = m;
	this->c = c;
	mat = new float*[n];
	for (int i = 0; i < n; i++)
	{
		mat[i] = new float[m];
	}
}

void maker::input()
{
	int i, j;
	cout << "Matrix " << c << endl;
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
		{
			cin >> mat[i][j];
		}
	}
}

void maker::output()
{
	cout << "\nMax element in matrix " << c << ":\n";
	cout << "Max = " << max << endl;
	cout << "i = " << mini + 1 << " j = " << minj + 1 << endl;
}

maker::~maker()
{
	for (int i = 0; i < n; i++)
	{
		delete[] mat[i];
	}
	delete[] mat;
}

void maker::search()
{
	int i, j;
	mini = 0, minj = 0;
```

```
	max = mat[0][0];
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			if (mat[i][j] > max)
			{
				min = mat[i][j];
				mini = i;
				minj = j;
			}
}

class newmaker : public maker
{
private:
	float min;
	int maxi;
	int maxj;
	float sum;
public:
	float summa();
	void search();
	void output();
	newmaker(int n, int m, char c);
	~newmaker();
};

newmaker::newmaker(int n, int m, char c) :maker(n, m, c) {}
newmaker::~newmaker() {}

/*Добавленный метод*/
float newmaker::summa()
{
	int i, j;
	sum = 0;
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			sum += mat[i][j];
	return sum;
}
 /*Переопределенный метод*/
void newmaker::search()
{
	int i, j;
	maxi = 0, maxj = 0;
	min = mat[0][0];
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			if (mat[i][j] < min)
			{
				min = mat[i][j];
				maxi = i;
				maxj = j;
			}
}

void newmaker::output()
{
	cout << "\nMax element in matrix " << c << ":\n";
	cout << "Max = " << min << endl;
	cout << "i = " << maxi + 1 << " j = " << maxj + 1 << endl;
}

void main()
{
```

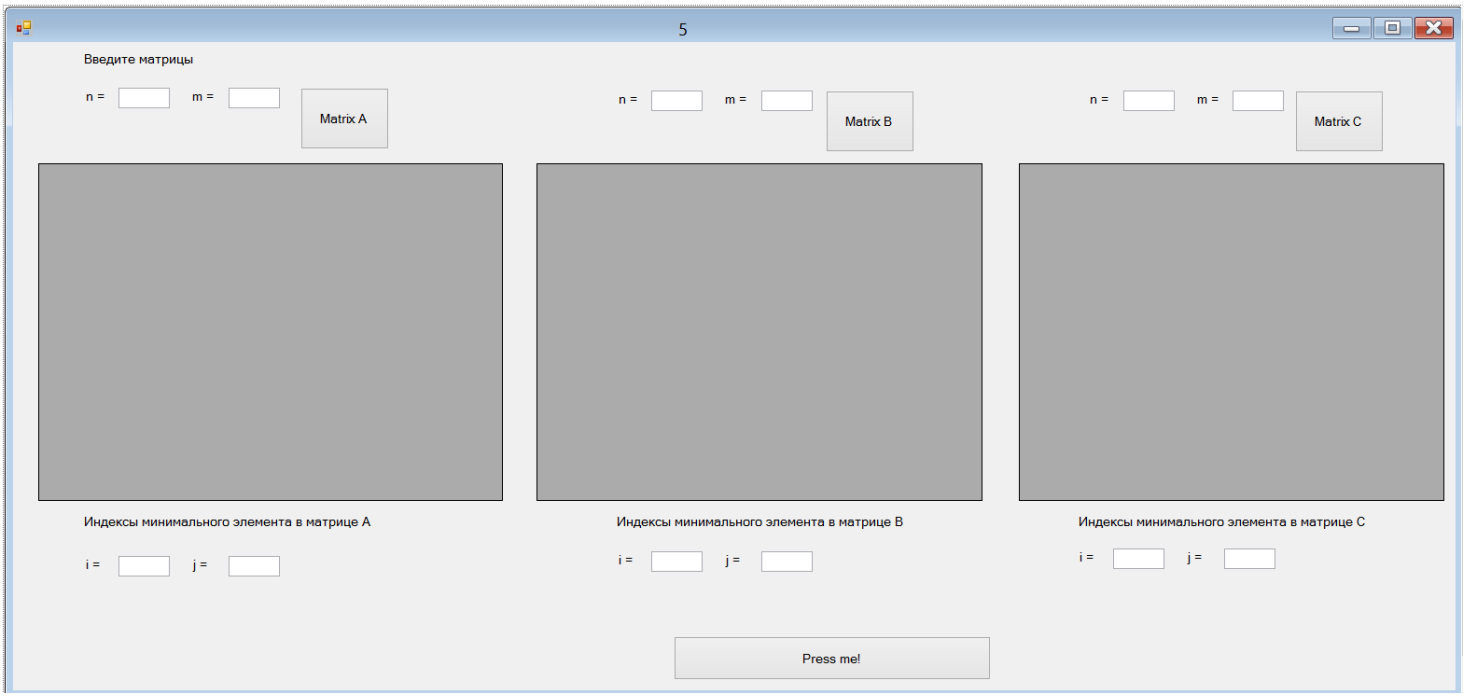
```
        maker *A;
        newmaker *B, *C;

        A = new maker(7, 5, 'A');
        B = new newmaker(3, 7, 'B');
        C = new newmaker(4, 5, 'C');

        A->input();
        B->input();
        C->input();

        cout << endl;

        A->search();
        A->output();

        B->search();
        B->output();

        C->search();
        C->output();

        //cout << "The sum of elements in array: " << A->summa() << endl;
        cout << "\n\nThe sum of elements in array B: " << B->summa() << endl;
        cout << "The sum of elements in array C: " << C->summa() << endl;



        A->~maker();
        B->~newmaker();
        C->~newmaker();

        _getch();
}
```

Раздел 4. Оконное приложение

Разработка основной формы приложения

Первым шагом в разработке оконного приложения является создание основной формы приложения, на которой надо расположить необходимые элементы интерфейса. Таким образом, в нашей задаче необходимо создать для каждой из трех матриц:

1. Два Textbox’a в верхней части интерфейса для ввода значений n и m (количества строк и столбцов).
2. dataGridView для ввода значений матрицы
3. Два Textbox’a в нижней части интерфейса для вывода индексов минимального элемента матрицы
4. Кнопку для создания матрицы на основании значений n и m
5. Общую кнопку для нахождения индексов максимального элемента в каждой из трех матриц



Коды обработчиков основных событий с комментариями

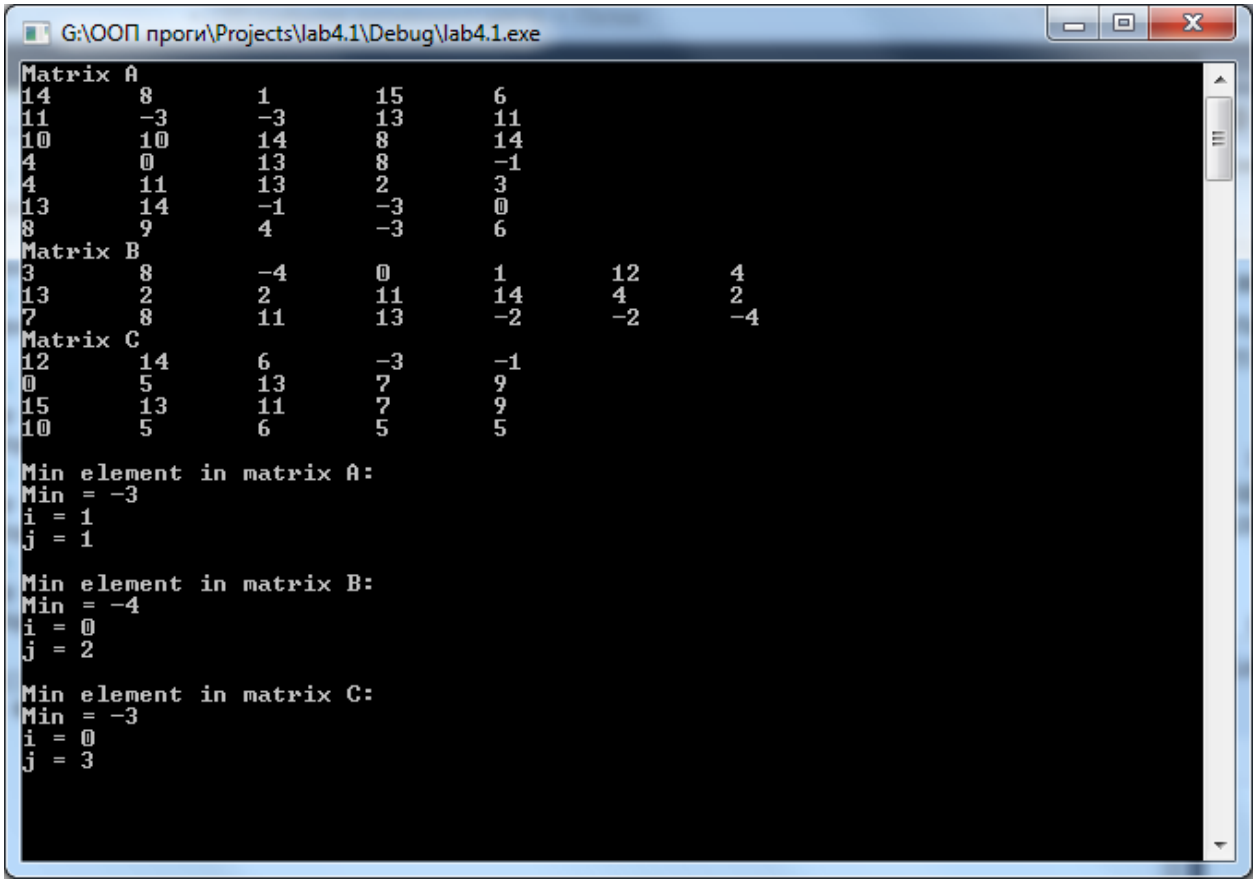
```
/*описание обработчика нажатия кнопки button1 – ввод размеров первой матрицы*/
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                        /* Считывание значений из текстовых окон*/
                        n1=System::Convert::ToInt16(nA->Text);
                        m1=System::Convert::ToInt16(mA->Text);
                        /*Предупреждение при вводе 0*/
                        if(n1==0||m1==0)
                                MessageBox::Show("Пожалуйста, введите значения N и M не
равные нулю.", "Error!");
                        else
                        {
                                dataGridView1->RowCount=n1;
                                dataGridView1->ColumnCount=m1;
                        }

                        if(n1!=0 && m1!=0)
                        /*Вывод приглашения для ввода матрицы*/

                    MessageBox::Show("Введите матрицу "+System::Convert::ToString(n1)+"
x "+System::Convert::ToString(m1), "Matrix");
                }
/*описание кнопки button2 – ввод размеров второй матрицы*/
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
                        n2=System::Convert::ToInt16(nB->Text);
                        m2=System::Convert::ToInt16(mB->Text);

                        if(n2==0||m2==0)
                                MessageBox::Show("Пожалуйста, введите значения N и M не
равные нулю.", "Error!");
                        else
                        {
                                dataGridView2->RowCount=n2;
                                dataGridView2->ColumnCount=m2;
                        }
```

```
                        if(n2!=0 && m2!=0)
                    MessageBox::Show("Введите матрицу "+System::Convert::ToString(n2)+"
x "+System::Convert::ToString(m2), "Matrix");
                }

/*описание кнопки button3 – ввод размеров третьей матрицы*/
private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
        n3=System::Convert::ToInt16(nC->Text);
        m3=System::Convert::ToInt16(mC->Text);

        if(n3==0||m3==0)
        MessageBox::Show("Пожалуйста, введите значения N и M не равные нулю.", "Error!");
        else
        {
                dataGridView3->RowCount=n3;
                dataGridView3->ColumnCount=m3;
        }

        if(n3!=0 && m3!=0)
        MessageBox::Show("Введите матрицу "+System::Convert::ToString(n3)+" x
        "+System::Convert::ToString(m3), "Matrix");
                }

/*описание кнопки button4 – вычисление индексов максимальных элементов матриц*/
private: System::Void button4_Click(System::Object^  sender, System::EventArgs^  e) {
                    int i,j;
                    double **a, *c;
                    n = n1; m = m1;
                    c = new double [2];
                    a = new double *[n];
                    for (i=0; i<n; i++)
                        a[i] = new double [m];

                    for (i=0; i<n; i++)
                        for (j=0; j<m; j++)
                    a[i][j] = System::Convert::ToDouble(dataGridView1[j,i]->Value);

                    search(n,m,a,c);

                    a1->Text = System::Convert::ToString(c[0]+1);
                    a2->Text = System::Convert::ToString(c[1]+1);
//-------------------------------------------------------------------------------------
                    n = n2; m = m2;
                    for (i=0; i<n; i++)
                        for (j=0; j<m; j++)
                    a[i][j] = System::Convert::ToDouble(dataGridView2[j,i]->Value);

                    search(n,m,a,c);

                    b1->Text = System::Convert::ToString(c[0]+1);
                    b2->Text = System::Convert::ToString(c[1]+1);

//-------------------------------------------------------------------------------------
                    n = n3; m = m3;
                    for (i=0; i<n; i++)
                        for (j=0; j<m; j++)
                    a[i][j] = System::Convert::ToDouble(dataGridView3[j,i]->Value);

                    search(n,m,a,c);

                    c1->Text = System::Convert::ToString(c[0]+1);
                    c2->Text = System::Convert::ToString(c[1]+1);

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

```
            delete [] a[i];
        delete [] a;
        delete [] c;
    }
```

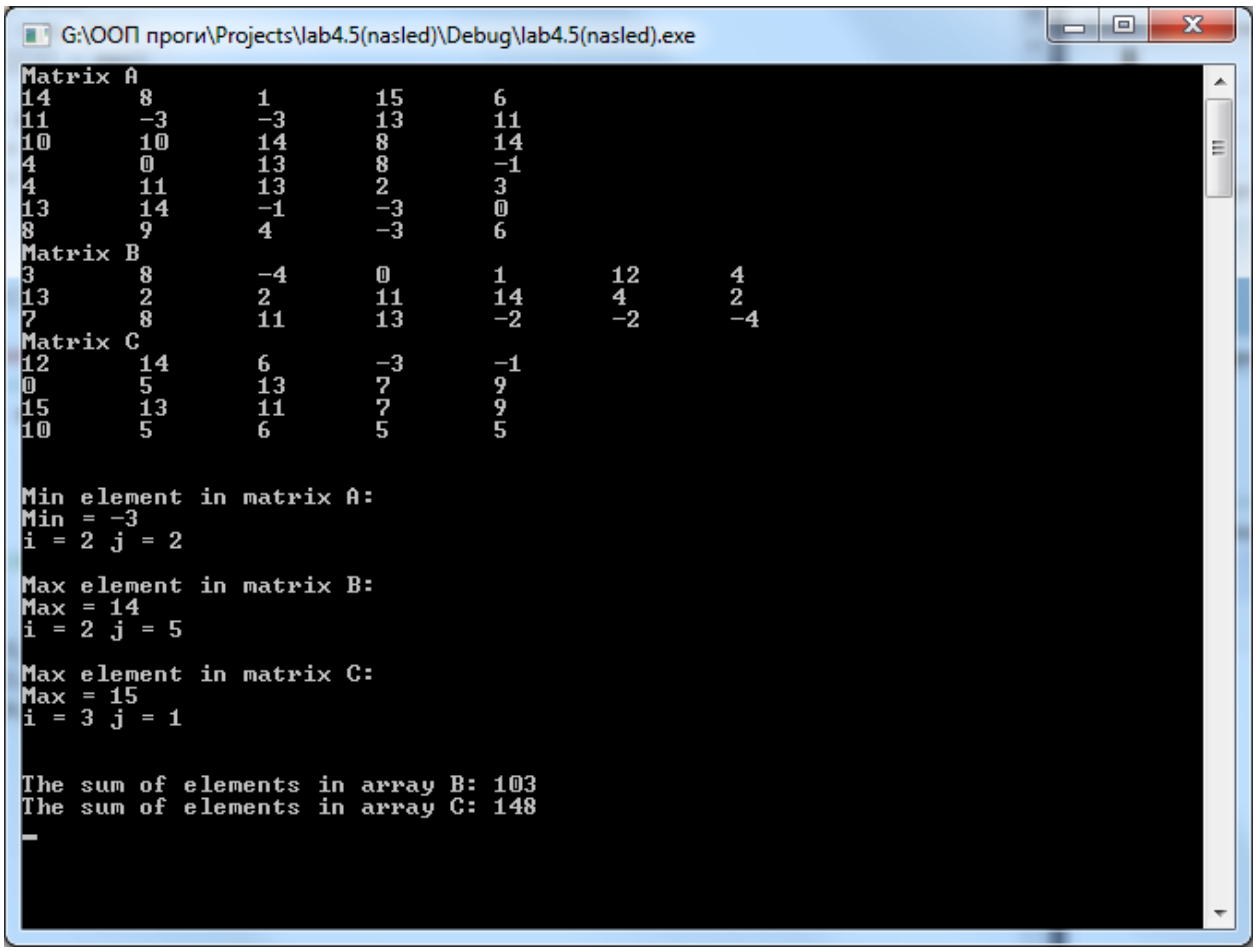
Тесты

Тест консольного *однофайлового* приложения

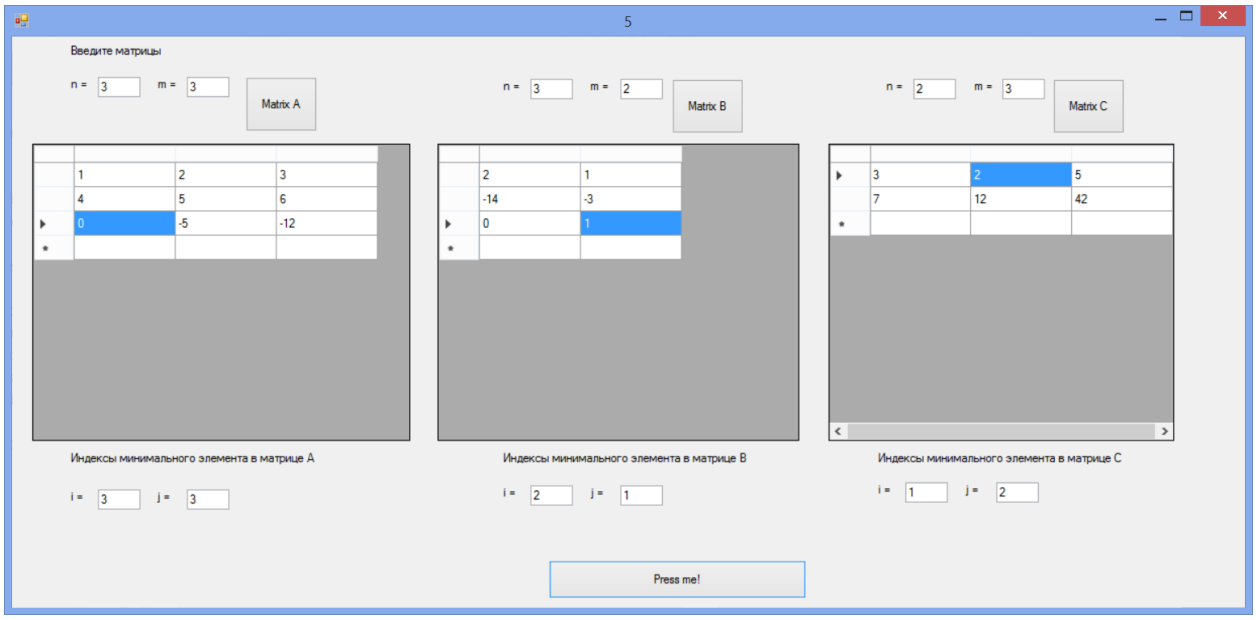


Тест консольного приложения с *подпрограммами* (многофайловая) и тест консольного приложения с *динамическими массивами* аналогичны результату консольного *однофайлового* приложения.

Тест консольного приложения с *классами*



Тест работы *оконного приложения*



Заключение

В ходе курсового проектирования была проделана следующая работа:

1. Определены функции (методы), необходимые для грамотного проектирования алгоритма.
2. Разработан алгоритм консольного приложения в виде блок-схемы функции main и основной вычислительной функции search.
3. Разработано консольное приложение с использованием функций в двух версиях – с локальными и динамическими матрицами.
4. Разработано консольные приложения с классами, в том числе с использованием класса-наследника.
5. Разработано оконное приложение.
6. Все приложения опробованы на тестах.