Министерство образования и науки Российской Федерации

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**Разработка структур хранения для векторов и матриц**

**Выполнил**:студент группы 381606-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимакин Н.Е.

Подпись

**Проверил**: к.ф.-м.н., доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Баркалов К.А.

Подпись

Нижний Новгород 2017

**Введение**

В языке программирования C++ изначально не предусмотрен механизм хранения и взаимодействия для таких объектов, как вектор и матрица. Так что для решения конкретных задач, в которых они используются, например, в линейной алгебре, необходимо описать такой механизм.

**Постановка задачи**

Разработать структуру хранения для векторов и матриц и с помощью неё реализовать калькулятор, который бы выполнял простейшие операции с матрицами.

**Описание структуры программы**

Решение задачи строится на основе классов – векторы и матрицы реализованы как пользовательские типы данных. Так же в программе присутствует класс исключений, который позволяет отловить ошибки, связанный с неправильной индексацией вектора или матрицы. Сам же калькулятор выполнен с помощью Microsoft Windows Forms.

Файлы, входящие в программу:

Exception.h – содержит класс исключений, его поля и методы

Поля:

int line; - номер строки, в которой возбудилось исключение

char \*func; - имя функции, в которой возбудилось исключение

char \*msg; - сообщение для пользователя

Методы:

Exception(int \_line, char \*\_func, char \*\_msg); - конструктор- инициализатор

Exception(const Exception& a); - конструктор копирования

~Exception(); - деструктор

void print(); - печать исключения

Exception.cpp – содержит реализацию методов класса исключений

vector.h – содержит класс вектор, его поля и методы

Поля:

double \*arr; - массив, содержащий значения вектора

int size; - размер вектора

Методы:

vector(); - конструктор по умолчанию

vector(int \_size); - конструктор инициализатор 1

vector(const vector& v); - конструктор копирования

vector(int \_size, double a); - конструктор инициализатор 2

~vector(); - деструктор

int getsize(); - возвращает размер вектора

double getelem(int n); - возвращает значение элемента под номером n

void setelem(double a, int n); - вставляет элемент a на позицию n

void setsize(int \_size); - изменяет размер вектора

vector& operator=(const vector& v); - перегрузка операции присваивания

double& operator[](int n) - перегрузка операции индексации

vector operator+(const vector& v); - перегрузка операции + (сложение векторов)

void operator+=(const vector& v); - перегрузка операции += (сложение векторов)

vector operator-(const vector& v); - перегрузка операции – (вычитание векторов)

void operator-=(const vector& v); - перегрузка операции -= (вычитание векторов)

double operator\*(const vector& v); - перегрузка операции \* (скалярное произведение векторов)

friend ostream& operator<<(ostream& os, const vector& v); - вывод вектора в поток

friend istream& operator>>(istream& is, vector& v); - ввод вектора из потока

vector.cpp – содержит реализацию методов класса вектор

matrix.h – содержит класс матрица, его поля и методы

Поля:

int size; - размер матрицы

vector \*M; - массив, содержащий строки матрицы

Методы:

matrix(int \_size=5); - конструктор по умолчанию и инициализатор

matrix(int \_size, double a); - конструктор-инициализатор

matrix(const matrix& A); - конструктор копирования

~matrix(); - деструктор

int getsize(); - возвращает размер матрицы

double& getelem(int i, int j); - возвращает элемент матрицы

vector& operator[](int i); - возвращает i-ую строку матрицы

void setsize(int \_size); - изменяет размер матрицы

matrix& operator=(const matrix& A); - перегрузка операции присваивания

matrix operator+(const matrix& A); - перегрузка операции + (сложение матриц)

matrix operator-(const matrix& A); - перегрузка операции - (вычитание матриц)

void operator+=(const matrix& A); - перегрузка операции += (сложение матриц)

void operator-=(const matrix& A); - перегрузка операции -= (вычитание матриц)

matrix transpl() const; - транспонирование матрицы

void print(); - печать матрицы

vector operator\*(const vector& v); - перегрузка операции \* (произведение матрицы на вектор)

matrix operator\*(const matrix& A); - перегрузка операции \* (произведение матриц)

matrix operator\*(double a); - перегрузка операции \* (произведение матрицы на число)

void operator\*=(double a); - перегрузка операции \*= (умножение матрицы на число)

friend istream& operator>>(istream& is, matrix& A); - ввод матрицы из потока

friend ostream& operator<<(ostream& os, const matrix& A); - вывод матрицы в поток

matrix.cpp – содержит реализацию методов класса матрица

MyForm.h , MyForm.cpp – содержат реализацию визуальной составляющей калькулятора

**Описание алгоритмов**

Класс вектор:

vector(); - размеру присваивается 1, выделяется память под один элемент, и нулевому элементу присваивается 0

vector(int \_size); - размеру присваивается \_size, выделяется память под \_size элементов, и всем элементам присваивается 0

vector(const vector& v); -данному вектору присваивается размер вектора v, выделяется память под этот размер, и поочерёдно присваиваются все элементы из вектора v

vector(int \_size, double a); - размеру присваивается \_size, выделяется память под \_size элементов, и всем элементам присваивается a

~vector(); - удаляется указатель arr

int getsize(); - возвращается размер вектора

double getelem(int n); - возвращается n–ый элемент вектора

void setelem(double a, int n); - n-ому элементу вектора присваивается значение a

void setsize(int \_size); - создаётся копия данного вектора, данному вектору присваивается новый размер, удаляется старый указатель на arr, выделяется новая память. Если новый размер больше старого, то присвоить данному вектору все элементы копии, а оставшиеся заполнить нулями. Иначе присвоить данному вектору соответствующие элементы копии

vector& operator=(const vector& v); - если размеры не равны, то присвоить данному вектору размер вектора v, удалить старый указатель на arr и выделить новую память. Каждому элементу данного вектора присвоить соответствующий элемент вектора v. Вернуть данный вектор

double& operator[](int n) – если n больше размера вектора или меньше 0, то вызвать исключение. Вернуть n-ый элемент

vector operator+(const vector& v); - создать вектор-результат размера вектора v, каждому его элементу присвоить сумму соответствующих элементов данного вектора и v. Вернуть результат

void operator+=(const vector& v); - к каждому элементу данного вектора прибавить соответствующий элемент вектора v

double operator\*(const vector& v); - создать результат типа double, присвоить ему 0. К результату поочерёдно прибавить все произведения соответствующих координат. Вернуть результат

friend ostream& operator<<(ostream& os, const vector& v); - поочерёдно вывести в поток все элементы вектора v с пробелом. Вернуть поток

friend istream& operator>>(istream& is, vector& v); - поочерёдно ввести в поток все элементы вектора v. Вернуть поток

Класс матрица:

matrix(int \_size=5); - размеру присваивается \_size, выделяется память под \_size элементов, всем строкам присваивается вектор того же размера

matrix(int \_size, double a); - размеру присваивается \_size, выделяется память под \_size элементов, всем строкам присваивается вектор того же размера с элементами a

matrix(const matrix& A); - размеру данной матрицы присваивается размер матрицы A, выделяется память под этот размер, каждой строке данной матрицы присваивается строка матрицы A

~matrix(); - удаляется указатель на массив строк

int getsize(); - возвращается размер данной матрицы

double& getelem(int i, int j); - возвращается j-ый элемент i-ой строки данной матрицы

vector& operator[](int i); - возвращается i-ая строка данной матрицы

void setsize(int \_size); - создаётся копия данной матрицы, размеру данной матрицы присваивается \_size, удаляется указатель на массив строк данной матрицы и выделяется новая память. Если размер больше размера копии, то присвоить соответствующим элементам данной матрицы все элементы копии, а остальные заполнить нулями. Иначе присвоить всем элементам данной матрицы соответствующие элементы копии

matrix& operator=(const matrix& A); - если размеры не равны, то присвоить размеру данной матрицы размер матрицы A, удалить указатель на массив строк данной матрицы и выделить новую память. Всем элементам данной матрицы присвоить элементы матрицы A. Вернуть данную матрицу

matrix operator+(const matrix& A); - создать результат-матрицу размера матрицы A. Каждому элементу матрицы A присвоить сумму соответствующих элементов данной матрицы и A. Вернуть результат

void operator+=(const matrix& A); - к каждому элементу данной матрицы прибавить соответствующий элемент матрицы A

matrix transpl() const; - создать результат-матрицу размера данной матрицы. Для каждого элемента результата: j-ому элементу i-ой строки присвоить i-ый элемент j-ой строки. Вернуть результат

void print(); - напечатать поочерёдно каждую строку

vector operator\*(const vector& v); - создать вектор-результат, каждому его элементу присвоить произведение соответствующей строки на вектор v. Вернуть результат

matrix operator\*(const matrix& A); - создать матрицу-результат размера данной матрицы, создать матрицу, транспонированную к A. Каждому элементу результата присвоить сумму произведений соответствующих строк данной и транспонированной матрицы. Вернуть результат

matrix operator\*(double a); - создать матрицу-результат равную данной матрице, каждый её элемент умножить на a. Вернуть результат

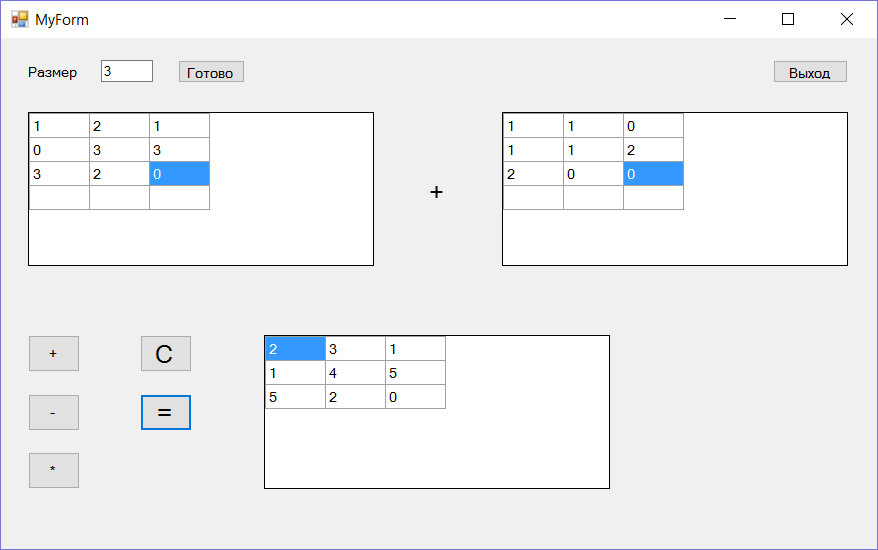
void operator\*=(double a); -

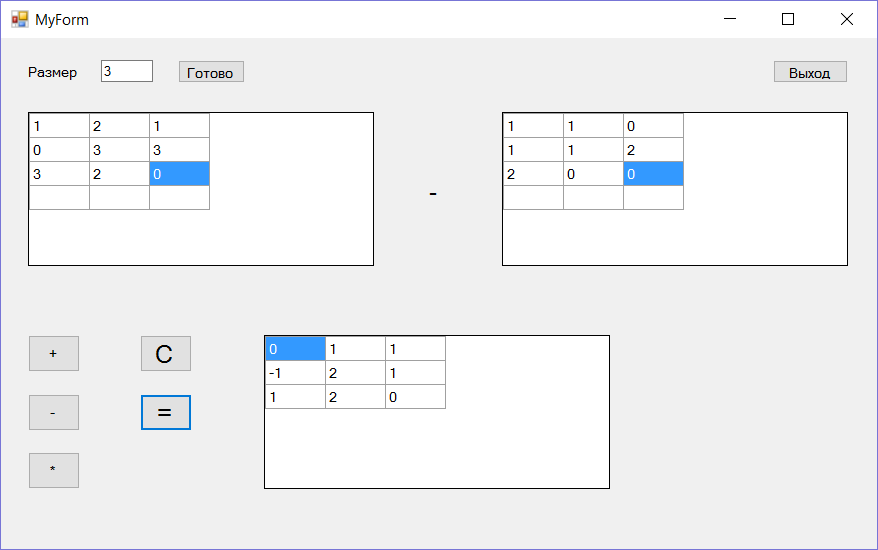
friend istream& operator>>(istream& is, matrix& A); - поочерёдно ввести из потока каждую строку матрицы A. Вернуть поток

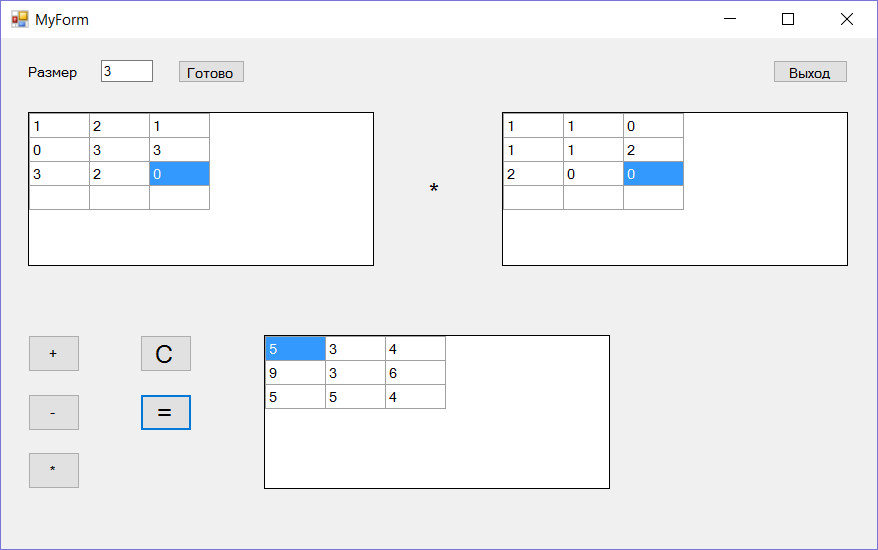
friend ostream& operator<<(ostream& os, const matrix& A); - поочерёдно вывести в поток каждую строку матрицы A. Вернуть поток

Графический интерфейс реализован с помощью Microsoft Windows Forms, а именно с помощью таких объектов как Button, Label, TextBox, DataGridView. В интерфейсе предусмотрено окно, в которое вводится размер матрицы. После подтверждения размера таблицы становятся активными, и в них записываются матрицы. После выбора действия и нажатия кнопки “=” выполняется операция

**Результаты**







**Вывод**

Калькулятор корректно выполняет все операции, а значит, у меня получилось разработать структуру хранения для векторов и матриц, реализованную в виде классов

**Литература**

Брайан Керниган, Деннис Ритчи «Язык программирования Си»

Брюс Эккель «Философия C++. Введение в стандартный C++»

Стивен Прата «Язык программирования C++. Лекции и упражнения»

**Приложение**

Exception.h

#pragma once

#include <iostream>

#include <string.h>

using namespace std;

class Exception

{

int line;

char \*func;

char \*msg;

public:

Exception(int \_line, char \*\_func, char \*\_msg);

Exception(const Exception& a);

~Exception();

void print();

};

Exception.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Exception.h"

Exception::Exception(int \_line, char \*\_func, char \*\_msg){

line = \_line;

int size = strlen(\_func);

func = new char[size + 1];

strcpy(func, \_func);

size = strlen(\_msg);

msg = new char[size + 1];

strcpy(msg, \_msg);

}

Exception::Exception(const Exception& a) {

line = a.line;

int size = strlen(a.func);

func = new char[size + 1];

strcpy(func, a.func);

size = strlen(a.msg);

msg = new char[size + 1];

strcpy(msg, a.msg);

}

Exception::~Exception(){

delete[] func;

delete[] msg;

}

void Exception::print() {

cout << "Line: " << line << endl << "Function: " << func << endl << msg << endl;

}

vector.h

#pragma once

#include "Exception.h"

class vector

{

private:

double \*arr;

int size;

public:

vector();

vector(int \_size);

vector(const vector& v);

vector(int \_size, double a);

~vector();

int getsize() { return size; }

double getelem(int n) { return arr[n]; }

void setelem(double a, int n) { arr[n] = a; }

void setsize(int \_size);

vector& operator=(const vector& v);

double& operator[](int n) {

if (n >= size) throw Exception(\_\_LINE\_\_,\_\_FUNCTION\_\_, "Error 001");

if (n < 0) throw Exception(\_\_LINE\_\_, \_\_FUNCTION\_\_, "Error 002");

return arr[n];

}

vector operator+(const vector& v);

void operator+=(const vector& v);

vector operator-(const vector& v);

void operator-=(const vector& v);

double operator\*(const vector& v);

friend ostream& operator<<(ostream& os, const vector& v) {

for (int i = 0; i < v.size; i++) {

os << v.arr[i] << " ";

}

return os;

}

friend istream& operator >> (istream& is, vector& v) {

for (int i = 0; i < v.size; i++) {

is >> v.arr[i];

}

return is;

}

};

vector.cpp

#include "vector.h"

vector::vector(){

size = 1;

arr = new double[1];

arr[0] = 0;

}

vector::vector(int \_size) {

size = \_size;

arr = new double[size];

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

arr[i] = 0;

}

}

vector::vector(int \_size, double a) {

size = \_size;

arr = new double[size];

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

arr[i] = a;

}

}

vector::vector(const vector& v) {

size = v.size;

arr = new double[size];

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

arr[i] = v.arr[i];

}

}

vector::~vector(){

delete[] arr;

}

void vector::setsize(int \_size) {

vector temp = \*this;

size = \_size;

delete[] arr;

arr = new double[size];

if (size > temp.size) {

for (int i = 0; i < temp.size; i++) {

arr[i] = temp.arr[i];

}

for (int i = temp.size; i < size; i++) {

arr[i] = 0;

}

}

else {

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

arr[i] = temp.arr[i];

}

}

}

vector& vector::operator=(const vector& v) {

if (size != v.size) {

delete[] arr;

arr = new double[v.size];

size = v.size;

}

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

arr[i] = v.arr[i];

}

return \*this;

}

vector vector::operator+(const vector& v) {

vector res(v.size);

for (int i = 0; i < v.size; i++) {

res.arr[i] =arr[i] + v.arr[i];

}

return res;

}

void vector::operator+=(const vector& v) {

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

arr[i] += v.arr[i];

}

}

vector vector::operator-(const vector& v) {

vector res(v.size);

for (int i = 0; i < v.size; i++) {

res.arr[i] = arr[i] - v.arr[i];

}

return res;

}

void vector::operator-=(const vector& v) {

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

arr[i] -= v.arr[i];

}

}

double vector::operator\*(const vector& v) {

double res=0;

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

res += arr[i] \* v.arr[i];

}

return res;

}

matrix.h

#pragma once

#include "vector.h"

class matrix

{

private:

int size;

vector \*M;

public:

matrix(int \_size=5);

matrix(int \_size, double a);

matrix(const matrix& A);

~matrix();

int getsize() { return size; }

double& getelem(int i, int j) { return M[i][j]; }

vector& operator[](int i) { return M[i]; }

matrix& operator=(const matrix& A);

matrix operator+(const matrix& A);

matrix operator-(const matrix& A);

void operator+=(const matrix& A);

void operator-=(const matrix& A);

matrix transpl() const;

vector operator\*(const vector& v);

matrix operator\*(const matrix& A);

matrix operator\*(double a);

void operator\*=(double a);

void print();

friend istream& operator>> (istream& is, matrix& A) {

for (int i = 0; i < A.size; i++) {

is >> A[i];

}

return is;

}

friend ostream& operator<<(ostream& os, const matrix& A) {

for (int i = 0; i < A.size; i++) {

os << A.M[i] << endl;

}

return os;

}

void setsize(int \_size);

};

matrix.cpp

#include "matrix.h"

matrix::matrix(int \_size){

size = \_size;

M = new vector[size];

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

//M[i].setsize(size);

M[i] = vector(size);

}

}

matrix::matrix(int \_size, double a) {

size = \_size;

M = new vector[size];

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

M[i] = vector(size, a);

}

}

matrix::matrix(const matrix& A) {

size = A.size;

M = new vector[size];

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

M[i] = A.M[i];

}

}

matrix::~matrix(){

delete[] M;

}

matrix& matrix::operator=(const matrix& A) {

if (size != A.size) {

size = A.size;

delete[] M;

M = new vector[size];

}

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

M[i] = A.M[i];

}

return \*this;

}

matrix matrix::operator+(const matrix& A) {

matrix res(A.size);

for (int i = 0; i < res.size; i++) {

for (int j = 0; j < res.size; j++) {

res[i][j] = M[i][j] + A.M[i][j];

}

}

return res;

}

matrix matrix::operator-(const matrix& A) {

matrix res(A.size);

for (int i = 0; i < res.size; i++) {

for (int j = 0; j < res.size; j++) {

res[i][j] = M[i][j] - A.M[i][j];

}

}

return res;

}

void matrix::operator+=(const matrix& A) {

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

for (int j = 0; j < size; j++) {

M[i][j] += A.M[i][j];

}

}

}

void matrix::operator-=(const matrix& A) {

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

for (int j = 0; j < size; j++) {

M[i][j] -= A.M[i][j];

}

}

}

matrix matrix::transpl() const {

matrix res(size);

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

for (int j = 0; j < size; j++) {

res[i][j] = M[j][i];

}

}

return res;

}

vector matrix::operator\*(const vector& v) {

vector res(size);

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

res[i] = M[i] \* v;

}

return res;

}

matrix matrix::operator\*(const matrix& A) {

matrix res(size), At = A.transpl();

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

for (int j = 0; j < size; j++) {

res[i][j] = M[i] \* At[j];

}

}

return res;

}

matrix matrix::operator\*(double a) {

matrix res = \*this;

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

for (int j = 0; j < size; j++) {

res[i][j] \*= a;

}

}

return res;

}

void matrix::operator\*=(double a) {

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

for (int j = 0; j < size; j++) {

M[i][j] \*= a;

}

}

}

void matrix::print() {

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

cout << M[i] << endl;

}

}

void matrix::setsize(int \_size) {

size = \_size;

matrix tmp = \*this;

delete[] M;

M = new vector[size];

if (\_size > tmp.size) {

for (int i = 0; i < tmp.size; i++) {

for (int j = 0; j < tmp.size; j++) {

M[i][j] = tmp[i][j];

}

for (int j = tmp.size; j < size; j++) {

M[i][j] = 0;

}

}

for (int i = tmp.size; i < size; i++) {

for (int j = 0; j < size; j++) {

M[i][j] = 0;

}

}

}

else {

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

for (int j = 0; j < size; j++) {

M[i][j] = tmp[i][j];

}

}

}

}

MyForm.h

#pragma once

#include "matrix.h"

namespace Матричный\_калькулятор\_1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::DataGridView^ dataGridView2;

private: System::Windows::Forms::DataGridView^ dataGridView3;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Button^ button5;

private: System::Windows::Forms::Button^ button6;

private: System::Windows::Forms::Button^ button7;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->label1 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->button1 = (gcnew System::Windows::Forms::Button());

this->dataGridView2 = (gcnew System::Windows::Forms::DataGridView());

this->dataGridView3 = (gcnew System::Windows::Forms::DataGridView());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

this->label2 = (gcnew System::Windows::Forms::Label());

this->button5 = (gcnew System::Windows::Forms::Button());

this->button6 = (gcnew System::Windows::Forms::Button());

this->button7 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(24, 25);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(57, 17);

this->label1->TabIndex = 0;

this->label1->Text = L"Размер";

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(100, 22);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(52, 22);

this->textBox1->TabIndex = 1;

//

// dataGridView1

//

this->dataGridView1->BackgroundColor = System::Drawing::SystemColors::ControlLightLight;

this->dataGridView1->ColumnHeadersBorderStyle = System::Windows::Forms::DataGridViewHeaderBorderStyle::None;

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->ColumnHeadersVisible = false;

this->dataGridView1->Location = System::Drawing::Point(27, 74);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->RowHeadersVisible = false;

this->dataGridView1->RowTemplate->Height = 24;

this->dataGridView1->Size = System::Drawing::Size(346, 154);

this->dataGridView1->TabIndex = 2;

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(139, 356);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(52, 37);

this->button1->TabIndex = 3;

this->button1->Text = L"=";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// dataGridView2

//

this->dataGridView2->BackgroundColor = System::Drawing::SystemColors::ControlLightLight;

this->dataGridView2->ColumnHeadersBorderStyle = System::Windows::Forms::DataGridViewHeaderBorderStyle::None;

this->dataGridView2->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView2->ColumnHeadersVisible = false;

this->dataGridView2->Location = System::Drawing::Point(501, 74);

this->dataGridView2->Name = L"dataGridView2";

this->dataGridView2->RowHeadersVisible = false;

this->dataGridView2->RowTemplate->Height = 24;

this->dataGridView2->Size = System::Drawing::Size(346, 154);

this->dataGridView2->TabIndex = 4;

//

// dataGridView3

//

this->dataGridView3->BackgroundColor = System::Drawing::SystemColors::ControlLightLight;

this->dataGridView3->ColumnHeadersBorderStyle = System::Windows::Forms::DataGridViewHeaderBorderStyle::None;

this->dataGridView3->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView3->ColumnHeadersVisible = false;

this->dataGridView3->Location = System::Drawing::Point(263, 297);

this->dataGridView3->Name = L"dataGridView3";

this->dataGridView3->RowHeadersVisible = false;

this->dataGridView3->RowTemplate->Height = 24;

this->dataGridView3->Size = System::Drawing::Size(346, 154);

this->dataGridView3->TabIndex = 5;

//

// button2

//

this->button2->Location = System::Drawing::Point(27, 297);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(52, 37);

this->button2->TabIndex = 6;

this->button2->Text = L"+";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(27, 356);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(52, 37);

this->button3->TabIndex = 7;

this->button3->Text = L"-";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// button4

//

this->button4->Location = System::Drawing::Point(27, 414);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(52, 37);

this->button4->TabIndex = 8;

this->button4->Text = L"\*";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4\_Click);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(423, 139);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(27, 29);

this->label2->TabIndex = 9;

this->label2->Text = L"L";

//

// button5

//

this->button5->Location = System::Drawing::Point(177, 22);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(67, 23);

this->button5->TabIndex = 10;

this->button5->Text = L"Готово";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &MyForm::button5\_Click);

//

// button6

//

this->button6->Location = System::Drawing::Point(772, 22);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(75, 23);

this->button6->TabIndex = 11;

this->button6->Text = L"Выход";

this->button6->UseVisualStyleBackColor = true;

this->button6->Click += gcnew System::EventHandler(this, &MyForm::button6\_Click);

//

// button7

//

this->button7->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button7->Location = System::Drawing::Point(139, 297);

this->button7->Name = L"button7";

this->button7->Size = System::Drawing::Size(52, 37);

this->button7->TabIndex = 12;

this->button7->Text = L"C";

this->button7->UseVisualStyleBackColor = true;

this->button7->Click += gcnew System::EventHandler(this, &MyForm::button7\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(876, 511);

this->Controls->Add(this->button7);

this->Controls->Add(this->button6);

this->Controls->Add(this->button5);

this->Controls->Add(this->label2);

this->Controls->Add(this->button4);

this->Controls->Add(this->button3);

this->Controls->Add(this->button2);

this->Controls->Add(this->dataGridView3);

this->Controls->Add(this->dataGridView2);

this->Controls->Add(this->button1);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label1);

this->Name = L"MyForm";

this->Text = L"MyForm";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int size = Convert::ToInt32(textBox1->Text);

matrix A(size), B(size), res(size);

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

for (int j = 0; j < size; j++) {

A[i][j] = Convert::ToDouble(dataGridView1->Rows[i]->Cells[j]->Value);

B[i][j] = Convert::ToDouble(dataGridView2->Rows[i]->Cells[j]->Value);

}

}

char sgn = Convert::ToChar(label2->Text);

switch (sgn) {

case '+': res = A + B; break;

case '-': res = A - B; break;

case '\*': res = A \* B; break;

}

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

for (int j = 0; j < size; j++) {

dataGridView3->Rows[i]->Cells[j]->Value = Convert::ToString(res[i][j]);

}

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

label2->Text = "+";

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

label2->Text = "-";

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

label2->Text = "\*";

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

int size = Convert::ToInt32(textBox1->Text);

dataGridView1->ColumnCount = size;

dataGridView1->RowCount = size;

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

dataGridView1->Columns[i]->Width = 60;

dataGridView1->Rows[i]->Cells[0]->Value = " ";

}

dataGridView2->ColumnCount = size;

dataGridView2->RowCount = size;

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

dataGridView2->Columns[i]->Width = 60;

dataGridView2->Rows[i]->Cells[0]->Value = " ";

}

dataGridView3->ColumnCount = size;

dataGridView3->RowCount = size;

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

dataGridView3->Columns[i]->Width = 60;

dataGridView3->Rows[i]->Cells[0]->Value = " ";

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

Close();

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) {

int size = Convert::ToInt32(textBox1->Text);

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

for (int j = 0; j < size; j++) {

dataGridView1->Rows[i]->Cells[j]->Value = "";

dataGridView2->Rows[i]->Cells[j]->Value = "";

dataGridView3->Rows[i]->Cells[j]->Value = "";

}

}

}

};

}

MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void Main(/\*array<String^>^ args\*/)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Матричный\_калькулятор\_1::MyForm form;

Application::Run(%form);

}