Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**по творческому заданию**

Тема: «Калькулятор логарифмов и решение задачи коммивояжера методом ветвей и границ»

«Основы алгоритмизации и программирования»

Семестр 2, Группа ИВТ-21-2б

Выполнил: Фаттахов Никита Маратович

Проверила: Полякова Ольга Андреевна

**Пермь, 2022**

**Творческая работа задача коммивояжера**

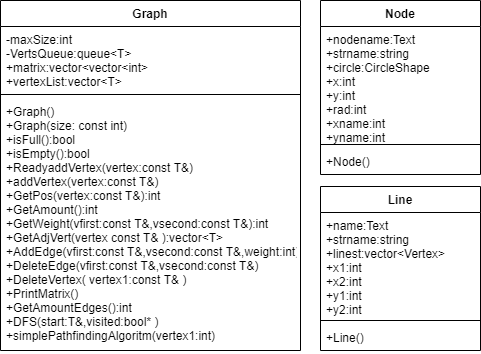
**Постановка задачи**

Задача коммивояжера заключается в поиске самого выгодного маршрута, проходящего через все вершины по одному разу с последующим возвратом в исходную вершину.

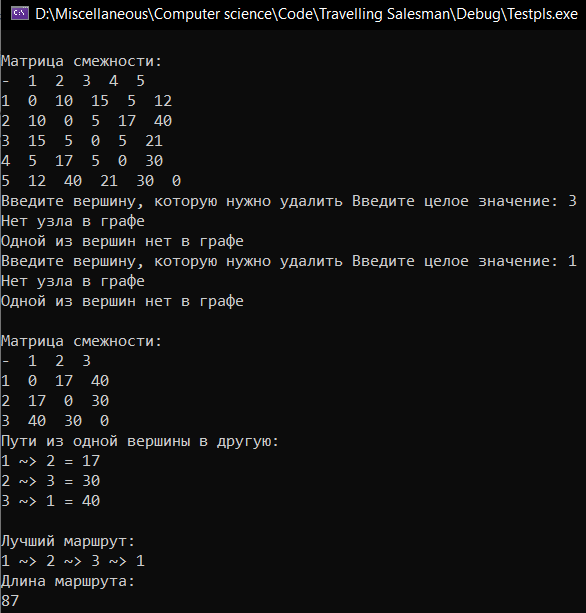
**Анализ задачи**

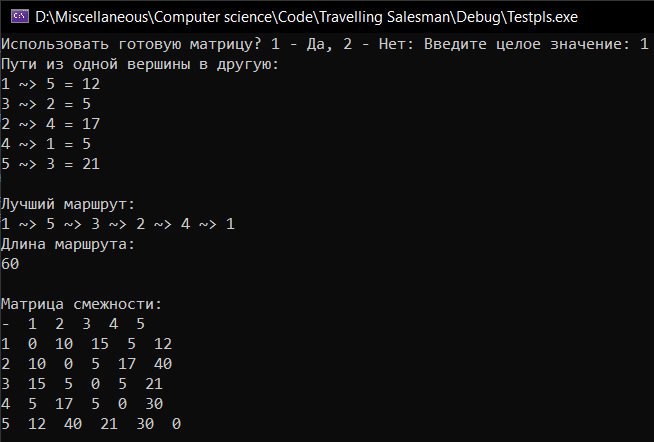
1. Создание и заполнение графа.
2. Решение задачи коммивояжера методом ветвей и границ.
   1. Построение матрицы с исходными данными — в таблицу заносятся расстояния (Cij) между городами (в ячейки типа A-A, B-B и т. д. ставится символ M — условно бесконечно большое число); при этом строкам соответствуют города отбытия, а столбцам города прибытия;
   2. Нахождение минимумов по строкам — в каждой строке определяется минимальное число (di) и выписывается в отдельный столбец; Редукция строк — из значений ячеек каждой строки вычитаем соответствующий минимум (Cij = Cij — di), не затрагивая при этом клетки с M;
   3. Нахождение минимумов по столбцам — в каждом столбце определяется минимальное число (dj) и выписывается в отдельную строку;
   4. Редукция столбцов — из значений ячеек каждого столбца вычитаем соответствующий минимум (Cij = Cij — dj), не затрагивая при этом клетки с M;
   5. Нахождение корневой нижней границы (делаем это только один раз, в следующие разы пункт 6 пропускаем) — вычисляем нижнюю границу (минимально возможную на текущем этапе длину маршрута) в стартовой (корневой) точке решения, как сумму найденных ранее минимумов (H0 = ∑di + ∑dj) и начинаем построение графа (схемы) решения с внесения в него корневой вершины;
   6. Вычисление оценок нулевых клеток — считаем оценки (pij) для каждой ячейки с нулями, как сумму минимумов по строке и столбцу, в которых располагается нулевая клетка, не учитывая при этом саму нулевую клетку;
   7. Выбор нулевой клетки с максимальной оценкой — ищем среди нулевых клеток обладающую наибольшей оценкой (если таких ячеек несколько, выбираем любую), и получаем пару ветвей (вариантов) решения задачи: с включением в маршрут отрезка пути относящегося к выбранной ячейке и без включения;
   8. Редукция матрицы — вычеркиваем относящиеся к выбранной клетке строку и столбец, а также заменяем значение ячейки соответствующей обратному пути на M;
   9. Вычисление нижней границы первой ветви (включающей отрезок пути) — вновь находим минимумы по строкам, проводим редукцию строк, находим минимумы по столбцам, проводим редукцию столбцов, после чего вычисляем локальную нижнюю границу, как сумму предыдущей локальной нижней границы и минимумов (Hk = Hk-1 + ∑di + ∑dj), и добавляем вершину в граф;
   10. Вычисление нижней границы второй ветви (не включающей отрезок пути) — считаем локальную нижнюю границу, как сумму предыдущей локальной нижней границы и оценки выбранной ранее нулевой клетки (Hk\* = Hk-1 + pij), и добавляем вершину в граф;
   11. Выбор ветви с минимальным значением нижней границы — среди еще не ветвившихся вершин выбираем обладающую минимальным значением локальной нижней границы (вне зависимости от того, какую ветвь рассматриваем в данный момент);
   12. Если полный маршрут еще не найден, продолжаем решение, если найден — переходим к пункту 10 — если маршрут еще не найден, то ход дальнейшего решения зависит от выбранной ветви: (а) первая ветвь — переходим к пункту 7, (б) вторая ветвь — в клетку с максимальной оценкой ставим M и переходим к пункту 2, (в) другая ветвь — возвращаемся к соответствующим ей этапу решения и таблице данных;
   13. Построение полного маршрута и определение его длины — соединяем все найденные ранее отрезки пути в полный маршрут и считаем его общую длину (данные берем из исходной таблицы).
3. Визуализация графа и интерфейса с помощью SFML.

**UML Диаграмма**

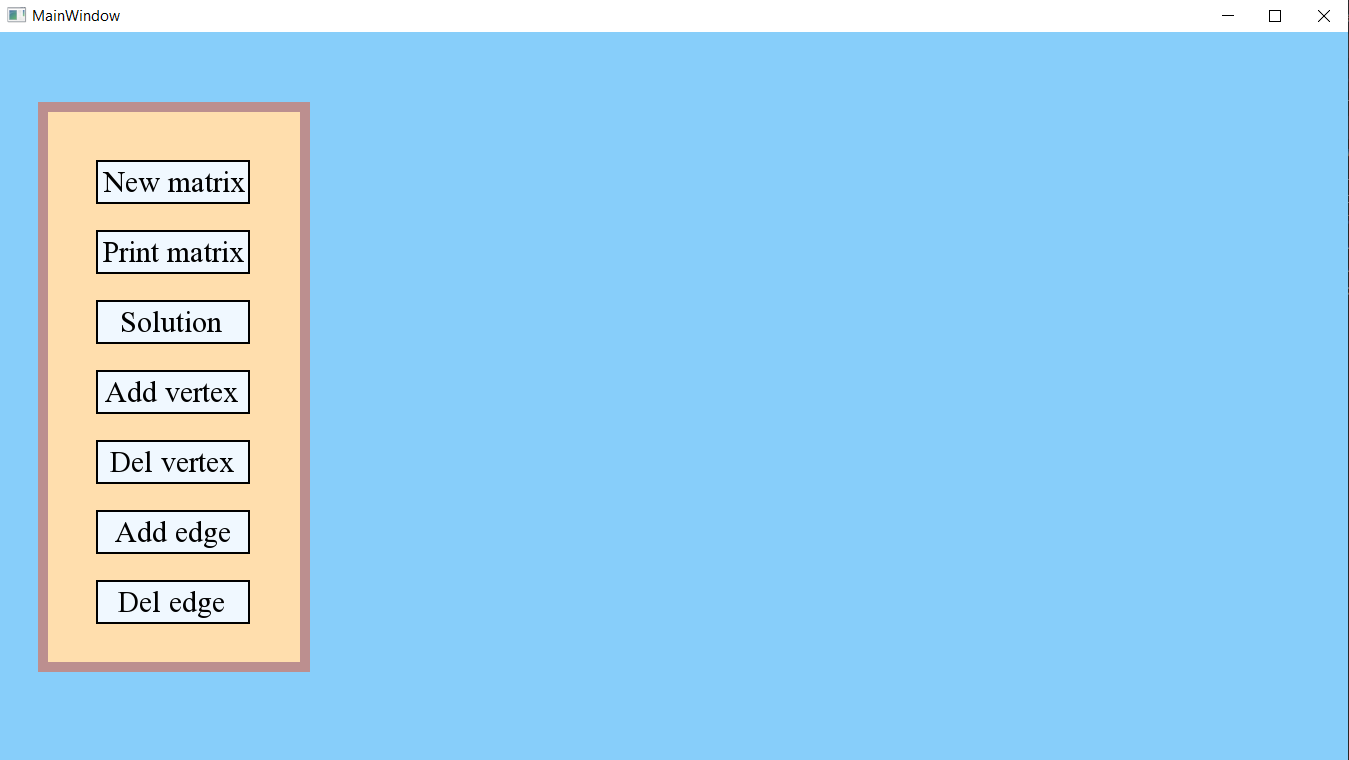


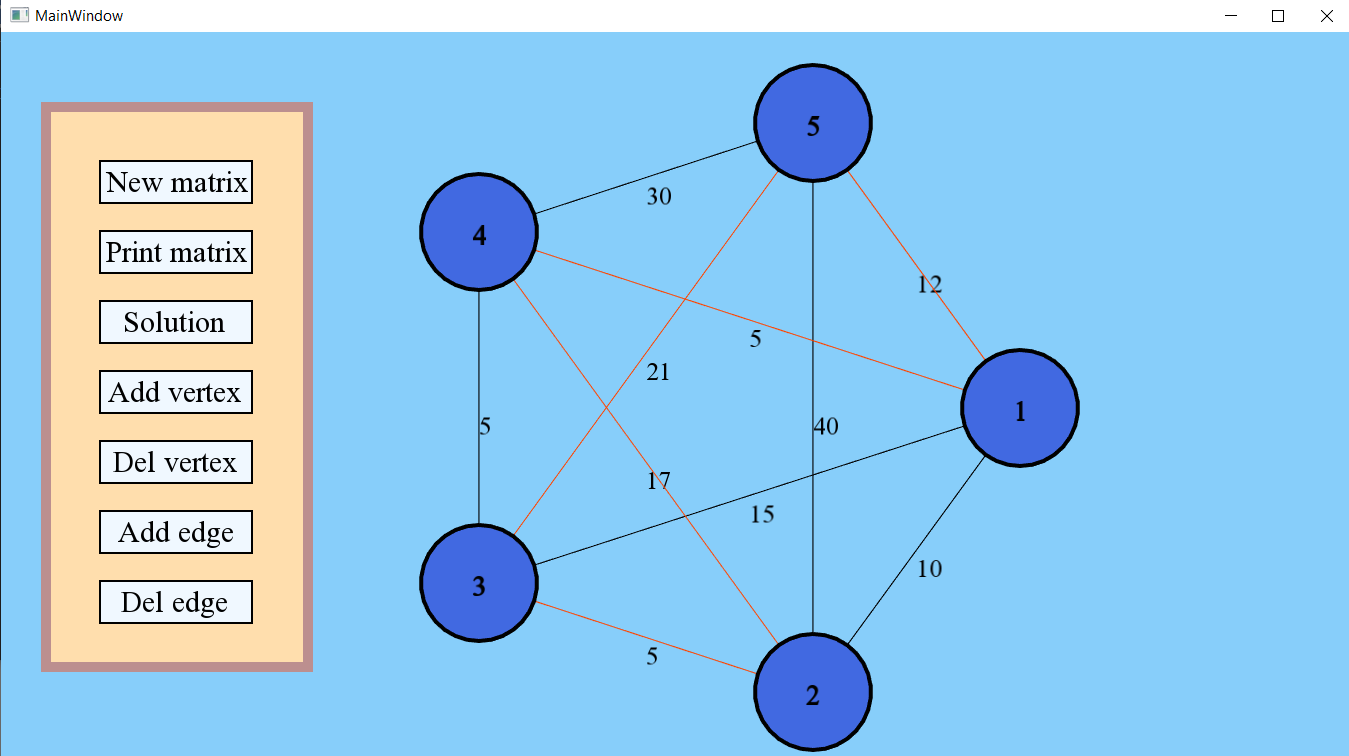
**Скриншоты консоли**

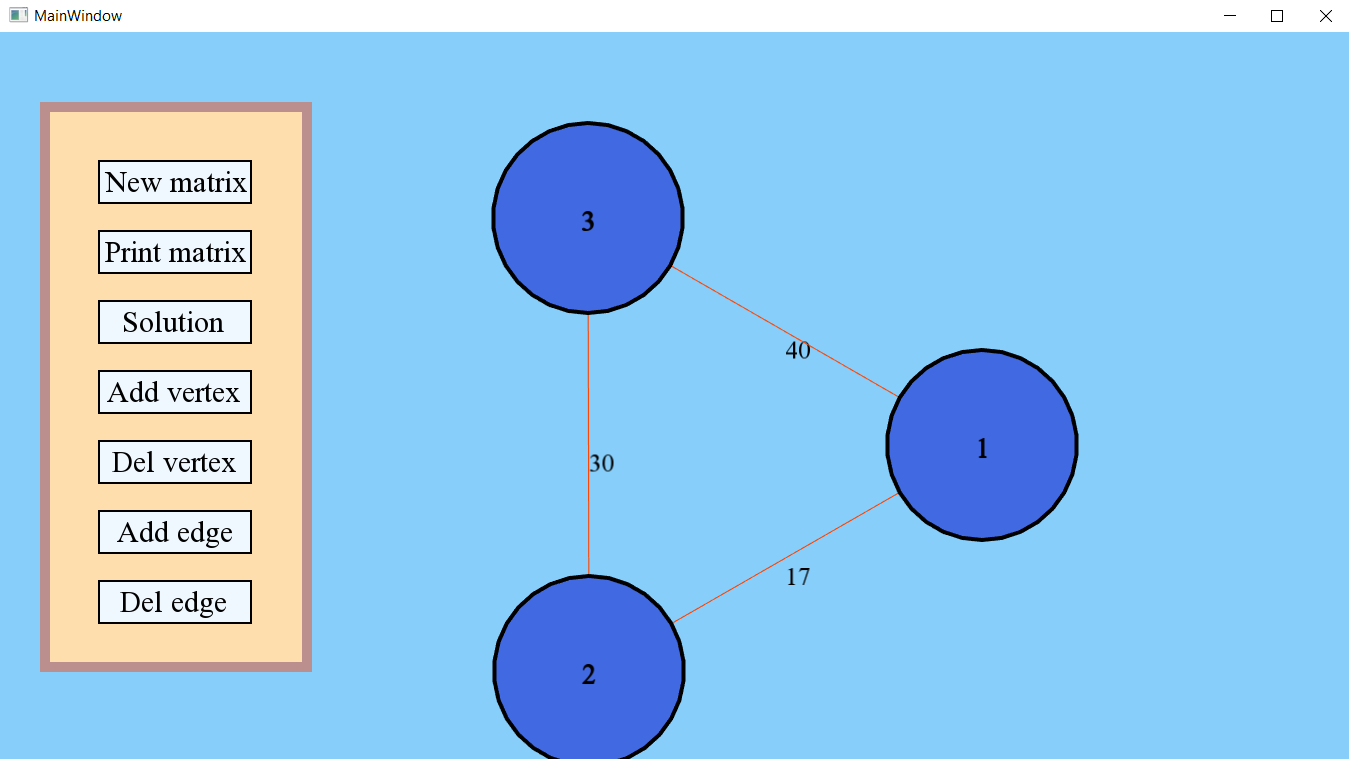


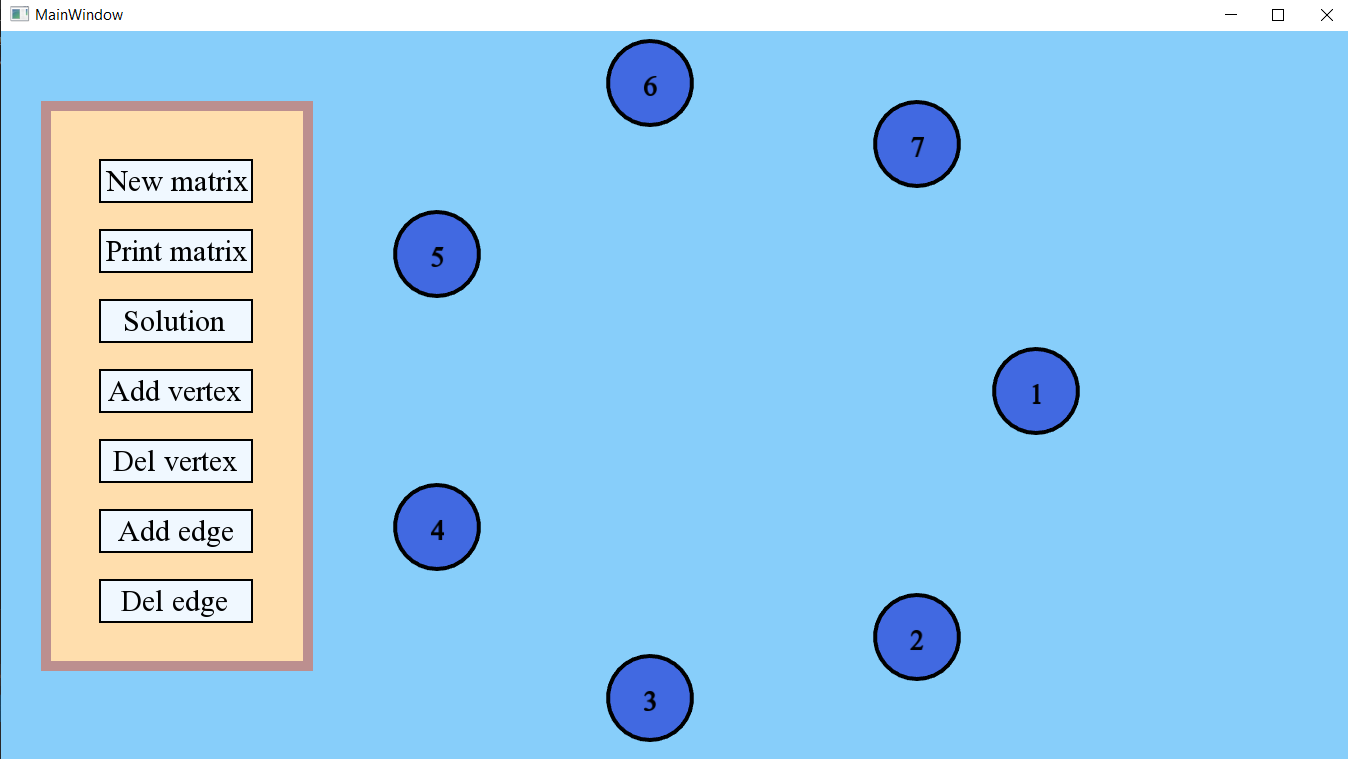


**Скриншоты визуализации**









**Код программы**

#include <random>

#include <vector>

#include <iostream>

#include <SFML/Graphics.hpp>

#include <iomanip>

#include <string>

#include <climits>

#include <vector>

#include <queue>

using namespace std;

using namespace sf;

vector<vector<int>> copyMatrix;

vector<vector<int>> grafMatrix;

vector<pair<int, int>> list\_for\_paths;

vector<int>copyVertexList;

Font font;

int WinW = 1350;

int WinH = 730;

template<class T>

class Graph

{

private:

int maxSize;

queue<T> VertsQueue;

public:

vector<vector<int>> matrix;

vector<T> vertexList;

Graph() : Graph(20) {}

Graph(const int& size) {

maxSize = size;

matrix = vector<vector<T>>(size, vector<T>(size));

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

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

matrix[i][j] = 0;

}

}

vector<T> GetList() {

return vertexList;

}

bool isFull() {

return vertexList.size() == maxSize;

}

bool isEmpty() {

return vertexList.size() == 0;

}

void ReadyaddVertex(const T& vertex) {

if (isFull()) {

cout << "Кол-во вершин превышает максимум." << endl;

return;

}

vertexList.push\_back(vertex);

copyVertexList.push\_back(vertex);

}

void addVertex(const T& vertex) {

if (isFull()) {

cout << "Кол-во вершин превышает максимум." << endl;

return;

}

vertexList.push\_back(vertex);

copyVertexList.push\_back(vertex);

int vertPos1 = GetPos(vertex);

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

matrix[i][vertPos1] = 0;

matrix[vertPos1][i] = 0;

}

}

int GetPos(const T& v) {

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

if (vertexList[i] == v) {

return i;

}

}

return -1;

}

int GetAmount() {

return vertexList.size();

}

int GetWeight(const T& vfirst, const T& vsecond) {

if (isEmpty()) {

return 0;

}

int Pos\_vfirst = GetPos(vfirst);

int Pos\_vsecond = GetPos(vsecond);

if (Pos\_vfirst == -1 || Pos\_vsecond == -1) {

cout << "Нет узла в графе " << endl;

return 0;

}

return matrix[Pos\_vfirst][Pos\_vsecond];

}

vector<T> GetAdjVert(const T& vertex) {

vector<T> AdjVertList;

int pos = GetPos(vertex);

if (pos != -1) {

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

if (matrix[pos][i] != 0) {

AdjVertList.push\_back(vertexList[i]);

}

}

}

return AdjVertList;

}

*//oriented or not*

void AddEdge(const T& vertex1, const T& vertex2, int weight) {

if (GetPos(vertex1) != (-1) && GetPos(vertex2) != (-1)) {

int vertPos1 = GetPos(vertex1);

int vertPos2 = GetPos(vertex2);

if (matrix[vertPos1][vertPos2] != 0 && matrix[vertPos2][vertPos1] != 0) {

cout << "Ребро уже есть" << endl;

return;

}

else {

matrix[vertPos1][vertPos2] = weight; *//1 для ориентированного графа*

matrix[vertPos2][vertPos1] = weight; *//1,2 для неориентированного графа*

}

}

else {

cout << "Одной из вершин нет в графе" << endl;

return;

}

}

*//oriented or not*

void DeleteEdge(const T& vertex1, const T& vertex2) {

if (GetPos(vertex1) != (-1) && GetPos(vertex2) != (-1)) {

int vertPos1 = GetPos(vertex1);

int vertPos2 = GetPos(vertex2);

if (matrix[vertPos1][vertPos2] != 0 && matrix[vertPos2][vertPos1] != 0) {

matrix[vertPos1][vertPos2] = 0; *//1 для ориентированного графа*

matrix[vertPos2][vertPos1] = 0; *//1,2 для неориентированного графа*

return;

}

}

else {

cout << "Одной из вершин нет в графе" << endl;

return;

}

}

*//not oriented*

void DeleteVertex(const T& vertex1) {

vector<T> AdjVert = GetAdjVert(vertex1);

int vertPos1 = GetPos(vertex1);

matrix.erase(matrix.begin() + vertPos1);

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

matrix[i].erase(matrix[i].begin() + vertPos1);

}

auto it = remove(vertexList.begin(), vertexList.end(), vertex1);

vertexList.erase(it, vertexList.end());

auto tit = remove(copyVertexList.begin(), copyVertexList.end(), vertex1);

copyVertexList.erase(tit, copyVertexList.end());

}

void PrintMatrix() {

if (!isEmpty()) {

cout << endl;

cout << "Матрица смежности: " << endl;

cout << "- ";

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

cout << vertexList[i] << " ";

}

cout << endl;

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

cout << vertexList[i] << " ";

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

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

}

cout << endl;

}

}

else {

cout << "Граф пуст" << endl;

}

}

*//oriented or not*

int GetAmountEdges() {

int amount = 0;

if (!isEmpty()) {

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

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

if (matrix[i][j] != 0) {

amount++;

}

}

}

}

*//return amount; //для ориентированного графа*

return amount / 2; *//для неориентированного графа*

}

void DFS(T& start, bool\* visited) {

cout << "Вершина " << start << " пройдена" << endl;

visited[GetPos(start)] = true;

vector<T> AdjVert = GetAdjVert(start);

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

if (!visited[GetPos(AdjVert[i])]) {

DFS(AdjVert[i], visited);

}

}

}

void BFS(T& start, bool\* visited) {

if (visited[GetPos(start)] == false) {

VertsQueue.push(start);

cout << "Вершина " << start << " пройдена" << endl;

visited[GetPos(start)] = true;

}

vector<T> AdjVerts = GetAdjVert(start);

VertsQueue.pop();

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

if (visited[GetPos(AdjVerts[i])] != true) {

VertsQueue.push(AdjVerts[i]);

visited[GetPos(AdjVerts[i])] = true;

cout << "Вершина " << start << " пройдена" << endl;

}

}

if (VertsQueue.empty()) return;

BFS(VertsQueue.front(), visited);

}

void simplePathfindingAlgoritm(int vertex1)

{

int start = GetPos(vertex1);

bool\* Visited = new bool[vertexList.size()];

unsigned\* distances = new unsigned[vertexList.size()];

unsigned minimalWeight, minimalIndex;

for (unsigned i = 0; i < vertexList.size(); ++i)

{

Visited[i] = false;

distances[i] = INT\_MAX;

}

distances[start] = 0;

do

{

minimalIndex = INT\_MAX;

minimalWeight = INT\_MAX;

for (unsigned i = 0; i < vertexList.size(); ++i)

{

if (!Visited[i] && distances[i] < minimalWeight)

{

minimalIndex = i;

minimalWeight = distances[i];

}

}

if (minimalIndex != INT\_MAX)

{

for (unsigned i = 0; i < vertexList.size(); ++i)

{

if (matrix[minimalIndex][i])

{

unsigned temp = minimalWeight + matrix[minimalIndex][i];

if (temp < distances[i])

distances[i] = temp;

}

}

Visited[minimalIndex] = true;

}

} while (minimalIndex < INT\_MAX);

for (unsigned i = 0; i < vertexList.size(); ++i)

{

if (distances[i] != INT\_MAX)

{

cout << "Вес: " << start << " ~> " << i << " = " << setw(6) << left << distances[i] << "\t";

unsigned end = i;

unsigned weight = distances[end];

string way = to\_string(end) + " >~ ";

while (end != start)

{

for (unsigned j = 0; j < vertexList.size(); ++j)

{

if (matrix[j][end])

{

int temp = weight - matrix[j][end];

if (temp == distances[j])

{

end = j;

weight = temp;

way += to\_string(j) + " >~ ";

}

}

}

}

cout << "Путь: ";

for (int j = way.length() - 5; j >= 0; --j)

cout << way[j];

cout << endl;

}

else

cout << "Вес: " << start << " ~ " << i << " = " << "маршрут недоступен" << endl;

}

delete[] Visited;

delete[] distances;

}

};

void ExtraMatrix(vector<vector<int> > matrix) {

int amount = matrix.size();

copyMatrix.clear();

grafMatrix.clear();

vector<int> temp\_vec;

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

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

if (i == j)

temp\_vec.push\_back(-1);

else

temp\_vec.push\_back(matrix[i][j]);

copyMatrix.push\_back(temp\_vec);

temp\_vec.clear();

}

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

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

temp\_vec.push\_back(matrix[i][j]);

grafMatrix.push\_back(temp\_vec);

temp\_vec.clear();

}

}

void RowReduction()

{

int amount = copyMatrix.size();

int size = amount;

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

{

int min = INT\_MAX;

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

{

if (copyMatrix[i][j] < min && copyMatrix[i][j] != -1)

min = copyMatrix[i][j];

}

if (min != INT\_MAX && min != 0)

{

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

{

if (copyMatrix[i][j] > 0 && copyMatrix[i][j] != INT\_MAX)

copyMatrix[i][j] -= min;

}

}

}

}

void ColumnReduction()

{

int amount = copyMatrix.size();

int size = amount;

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

{

int min = INT\_MAX;

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

{

if (copyMatrix[j][i] < min && copyMatrix[j][i] != -1)

min = copyMatrix[j][i];

}

if (min != INT\_MAX && min != 0)

{

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

{

if (copyMatrix[j][i] > 0 && copyMatrix[j][i] != INT\_MAX)

copyMatrix[j][i] -= min;

}

}

}

}

int FindMinInLine(int h, int g)

{

int amount = copyMatrix.size();

int min = INT\_MAX;

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

{

if (copyMatrix[h][i] < min && i != g && copyMatrix[h][i] != -1)

min = copyMatrix[h][i];

}

return min;

}

int FindMinInCol(int h, int g)

{

int amount = copyMatrix.size();

int min = INT\_MAX;

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

{

if (copyMatrix[i][g] < min && i != h && copyMatrix[i][g] != -1)

min = copyMatrix[i][g];

}

return min;

}

pair<int, int> FindPaths()

{

int amount = copyMatrix.size();

int max = INT\_MIN;

int size = amount;

vector<vector < int>> cop = copyMatrix;

std::pair<int, int> way;

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

{

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

{

if (copyMatrix[i][j] == 0)

{

int mark = FindMinInLine(i, j) + FindMinInCol(i, j);

if (max < mark)

{

max = mark;

way.first = i;

way.second = j;

}

}

}

}

if (copyMatrix[way.second][way.first] == 0)

copyMatrix[way.second][way.first] = -1;

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

{

copyMatrix[way.first][i] = -1;

copyMatrix[i][way.second] = -1;

}

return way;

}

void BranchBoundMethod()

{

int amount = copyMatrix.size();

copyMatrix = grafMatrix;

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

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

{

if (i == j)

copyMatrix[i][j] = -1;

if (copyMatrix[i][j] <= 0)

copyMatrix[i][j] = -1;

}

int ii = 0;

while (ii != amount)

{

RowReduction();

ColumnReduction();

std::pair<int, int> tmp;

tmp = FindPaths();

if (tmp != std::pair<int, int>(0, 0))

list\_for\_paths.push\_back(tmp);

ii++;

}

}

void Print\_road() {

cout << "Лучший маршрут:\n";

int begin = 0;

for (int i = 0; i < list\_for\_paths.size(); i++) {

for (int j = 0; j < list\_for\_paths.size(); j++) {

if (list\_for\_paths[j].first == begin) {

cout << list\_for\_paths[j].first + 1;

cout << " ~> ";

begin = list\_for\_paths[j].second;

break;

}

}

}

bool lastElementPrinted = false;

for (int j = 0; j < list\_for\_paths.size(); j++)

if (list\_for\_paths[j].first == begin) {

cout << list\_for\_paths[j].first + 1;

lastElementPrinted = true;

break;

}

if (!lastElementPrinted) {

for (int j = 0; j < list\_for\_paths.size(); j++)

if (list\_for\_paths[j].second == begin)

cout << list\_for\_paths[j].second + 1;

}

bool hasRoad = false;

for (int i = 0; i < list\_for\_paths.size(); i++)

if (list\_for\_paths[i].second == 0) {

hasRoad = true;

break;

}

if (!hasRoad)

cout << "\nЗадача не имеет решения\n";

cout << endl;

}

void Print\_paths(vector<vector<int> > matrix) {

cout << "Пути из одной вершины в другую:" << endl;

for (int i = 0; i < list\_for\_paths.size(); i++)

cout << list\_for\_paths[i].first + 1 << " ~> " << list\_for\_paths[i].second + 1

<< " = " << matrix[list\_for\_paths[i].first][list\_for\_paths[i].second] << endl;

}

void Sum\_of\_paths(vector<vector<int> > matrix) {

int sum = 0;

cout << "Длина маршрута:\n";

for (int i = 0; i < list\_for\_paths.size(); i++)

sum += matrix[list\_for\_paths[i].first][list\_for\_paths[i].second];

cout << sum << endl;

}

class Node {

public:

Text nodename;

string strname;

CircleShape circle;

int x;

int y;

int rad;

int xname;

int yname;

Node() {

xname = x + rad;

yname = y + rad;

}

};

class Line {

public:

Text name;

string strname;

vector<Vertex> linest;

int x1;

int y1;

int x2;

int y2;

Line() {

}

};

void SetCoord(int i, int amountVert, vector<Node>& cities, Graph<int> graph)

{

Node city;

int R\_;

int x0 = WinW / 2;

int y0 = WinH / 2;

if (WinW > WinH)

{

city.rad = 5 \* (WinH / 13) / amountVert;

R\_ = WinH / 2 - city.rad - 10;

}

else {

city.rad = 5 \* (WinW / 13) / amountVert;

R\_ = WinW / 2 - city.rad - 10;

}

float theta = 2.0f \* 3.1415926f \* float(i) / float(amountVert);

city.x = R\_ \* cos(theta) + y0 + R\_;

city.y = R\_ \* sin(theta) + x0 - R\_ - city.rad;

city.xname = city.x + city.rad;

city.yname = city.y + city.rad;

city.circle.setPosition(city.x, city.y);

city.circle.setFillColor(Color(65, 105, 225));

city.circle.setRadius(city.rad);

city.circle.setOutlineThickness(4);

city.circle.setOutlineColor(Color::Black);

city.nodename.setFont(font);

city.nodename.setCharacterSize(30);

city.nodename.setStyle(Text::Bold);

city.nodename.setFillColor(sf::Color::Black);

city.nodename.setPosition(city.xname-7, city.yname-17);

city.nodename.setString(to\_string(graph.vertexList[i]));

city.strname = (to\_string(graph.vertexList[i]));

cities.push\_back(city);

}

int main()

{

setlocale(LC\_ALL, "Russian");

font.loadFromFile("TimesNewRoman.ttf");

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> range(1, 500);

vector<Line> connections;

vector<Line> bestconnections;

vector<Node> cities;

Graph<int> graph;

*//win path*

*/\*for (int i = 0; i < list\_for\_paths.size(); i++) {*

*Line connect;*

*connect.x1 = cities[graph.GetPos(list\_for\_paths[i].first + 1)].xname;*

*connect.y1 = cities[graph.GetPos(list\_for\_paths[i].first + 1)].yname;*

*connect.x2 = cities[graph.GetPos(list\_for\_paths[i].second + 1)].xname;*

*connect.y2 = cities[graph.GetPos(list\_for\_paths[i].second + 1)].yname;*

*connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color(255, 69, 0)));*

*connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color(255, 69, 0)));*

*bestconnections.push\_back(connect);*

*}\*/*

ContextSettings settings;

settings.antialiasingLevel = 8;

const float W = 150;

const float H = 40;

const float fy = 130;

const float ax = 100;

const float ter = 70;

RectangleShape back(Vector2f(W+102, fy+ter\*6));

back.move(ax-50, fy-50);

back.setFillColor(Color(255, 222, 173));

back.setOutlineThickness(10.f);

back.setOutlineColor(Color(188, 143, 143));

float AnswerW = W;

float AnswerH = H;

float Answerx = ax;

float Answery = fy + ter;

RectangleShape Answer(Vector2f(AnswerW, AnswerH));

Answer.move(Answerx, Answery);

Answer.setFillColor(Color(240, 248, 255));

Answer.setOutlineThickness(2.f);

Answer.setOutlineColor(Color(0, 0, 0));

Text TextAnswer;

TextAnswer.setFont(font);

TextAnswer.setCharacterSize(30);

TextAnswer.setFillColor(sf::Color::Black);

TextAnswer.setPosition(Answerx+22, Answery);

TextAnswer.setString("Solution");

float DrawW = W;

float DrawH = H;

float Drawx = ax;

float Drawy = fy;

RectangleShape Draw(Vector2f(DrawW, DrawH));

Draw.move(Drawx, Drawy);

Draw.setFillColor(Color(240, 248, 255));

Draw.setOutlineThickness(2.f);

Draw.setOutlineColor(Color(0, 0, 0));

Text TextDraw;

TextDraw.setFont(font);

TextDraw.setCharacterSize(30);

TextDraw.setFillColor(sf::Color::Black);

TextDraw.setPosition(Drawx+5, Drawy);

TextDraw.setString("Print matrix");

float NewW = W;

float NewH = H;

float Newx = ax;

float Newy = fy+ter\*2;

RectangleShape New(Vector2f(NewW, NewH));

New.move(Newx, Newy);

New.setFillColor(Color(240, 248, 255));

New.setOutlineThickness(2.f);

New.setOutlineColor(Color(0, 0, 0));

Text TextNew;

TextNew.setFont(font);

TextNew.setCharacterSize(30);

TextNew.setFillColor(sf::Color::Black);

TextNew.setPosition(Newx + 5, Newy);

TextNew.setString("New matrix");

float AddW = W;

float AddH = H;

float Addx = ax;

float Addy = fy+ter\*3;

RectangleShape Add(Vector2f(AddW, AddH));

Add.move(Addx, Addy);

Add.setFillColor(Color(240, 248, 255));

Add.setOutlineThickness(2.f);

Add.setOutlineColor(Color(0, 0, 0));

Text TextAdd;

TextAdd.setFont(font);

TextAdd.setCharacterSize(30);

TextAdd.setFillColor(sf::Color::Black);

TextAdd.setPosition(Addx + 7, Addy);

TextAdd.setString("Add vertex");

float DelW = W;

float DelH = H;

float Delx = ax;

float Dely = fy+ter\*4;

RectangleShape Del(Vector2f(DelW, DelH));

Del.move(Delx, Dely);

Del.setFillColor(Color(240, 248, 255));

Del.setOutlineThickness(2.f);

Del.setOutlineColor(Color(0, 0, 0));

Text TextDel;

TextDel.setFont(font);

TextDel.setCharacterSize(30);

TextDel.setFillColor(sf::Color::Black);

TextDel.setPosition(Delx + 12, Dely);

TextDel.setString("Del vertex");

float DelEdgeW = W;

float DelEdgeH = H;

float DelEdgex = ax;

float DelEdgey = fy+ter\*6;

RectangleShape DelEdge(Vector2f(DelEdgeW, DelEdgeH));

DelEdge.move(DelEdgex, DelEdgey);

DelEdge.setFillColor(Color(240, 248, 255));

DelEdge.setOutlineThickness(2.f);

DelEdge.setOutlineColor(Color(0, 0, 0));

Text TextDelEdge;

TextDelEdge.setFont(font);

TextDelEdge.setCharacterSize(30);

TextDelEdge.setFillColor(sf::Color::Black);

TextDelEdge.setPosition(DelEdgex + 20, DelEdgey);

TextDelEdge.setString("Del edge");

float AddEdgeW = W;

float AddEdgeH = H;

float AddEdgex = ax;

float AddEdgey = fy+ter\*5;

RectangleShape AddEdge(Vector2f(AddEdgeW, AddEdgeH));

AddEdge.move(AddEdgex, AddEdgey);

AddEdge.setFillColor(Color(240, 248, 255));

AddEdge.setOutlineThickness(2.f);

AddEdge.setOutlineColor(Color(0, 0, 0));

Text TextAddEdge;

TextAddEdge.setFont(font);

TextAddEdge.setCharacterSize(30);

TextAddEdge.setFillColor(sf::Color::Black);

TextAddEdge.setPosition(AddEdgex + 17, AddEdgey);

TextAddEdge.setString("Add edge");

RenderWindow MainWindow(VideoMode(WinW, WinH), "MainWindow", Style::Default, settings);

while (MainWindow.isOpen()) {

Event event;

while (MainWindow.pollEvent(event)) {

Vector2i position = Mouse::getPosition(MainWindow);

int x = position.x;

int y = position.y;

if (x > 0 && y > 0 && x >= Answerx && x <= Answerx + AnswerW && y >= Answery && y <= Answery + AnswerH) {

Answer.setFillColor(Color(0, 250, 154));

}

else if(x > 0 && y > 0 && x >= Newx && x <= Newx + NewW && y >= Newy && y <= Newy + NewH) {

New.setFillColor(Color(0, 250, 154));

}

else if (x > 0 && y > 0 && x >= Drawx && x <= Drawx + DrawW && y >= Drawy && y <= Drawy + DrawH) {

Draw.setFillColor(Color(0, 250, 154));;

}

else if (x > 0 && y > 0 && x >= Addx && x <= Addx + AddW && y >= Addy && y <= Addy + AddH) {

Add.setFillColor(Color(0, 250, 154));;

}

else if (x > 0 && y > 0 && x >= Delx && x <= Delx + DelW && y >= Dely && y <= Dely + DelH) {

Del.setFillColor(Color(0, 250, 154));;

}

else if (x > 0 && y > 0 && x >= DelEdgex && x <= DelEdgex + DelEdgeW && y >= DelEdgey && y <= DelEdgey + DelEdgeH) {

DelEdge.setFillColor(Color(0, 250, 154));

}

else if (x > 0 && y > 0 && x >= AddEdgex && x <= AddEdgex + AddEdgeW && y >= AddEdgey && y <= AddEdgey + AddEdgeH) {

AddEdge.setFillColor(Color(0, 250, 154));;

}

else {

Answer.setFillColor(Color(240, 248, 255));

New.setFillColor(Color(240, 248, 255));

Draw.setFillColor(Color(240, 248, 255));

Add.setFillColor(Color(240, 248, 255));

Del.setFillColor(Color(240, 248, 255));

DelEdge.setFillColor(Color(240, 248, 255));

AddEdge.setFillColor(Color(240, 248, 255));

}

switch (event.type) {

case Event::MouseButtonPressed:

if (event.key.code == Mouse::Left)

{

Vector2i position = Mouse::getPosition(MainWindow);

int x = position.x;

int y = position.y;

if (x > 0 && y > 0 && x >= Answerx && x <= Answerx + AnswerW && y >= Answery && y <= Answery + AnswerH) {

vector<vector<int>> tcopyMatrix;

vector<vector<int>> tgrafMatrix;

vector<pair<int, int>> tlist\_for\_paths;

ExtraMatrix(graph.matrix);

BranchBoundMethod();

Print\_paths(graph.matrix);

cout << endl;

Print\_road();

Sum\_of\_paths(graph.matrix);

for (int i = 0; i < list\_for\_paths.size(); i++) {

Line connect;

connect.x1 = cities[graph.GetPos(list\_for\_paths[i].first + 1)].xname;

connect.y1 = cities[graph.GetPos(list\_for\_paths[i].first + 1)].yname;

connect.x2 = cities[graph.GetPos(list\_for\_paths[i].second + 1)].xname;

connect.y2 = cities[graph.GetPos(list\_for\_paths[i].second + 1)].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color(255, 69, 0)));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color(255, 69, 0)));

bestconnections.push\_back(connect);

}

grafMatrix.clear();

copyMatrix.clear();

list\_for\_paths.clear();

*/\*for (int i = 0; i < copyVertexList.size(); ++i) {*

*cout << copyVertexList[i] <<" ";*

*}\*/*

}

if (x > 0 && y > 0 && x >= Newx && x <= Newx + NewW && y >= Newy && y <= Newy + NewH) {

vector<Line> tempconnections;

vector<Line> tempbestconnections;

vector<Node> tempcities;

Graph<int> tempgraph;

int amountVert, amountEdge, edgeWeight;

int vertex, initialVertex, endVertex;

int answer = 0;

while (answer != 1 && answer != 2) {

cout << "Использовать готовую матрицу? 1 - Да, 2 - Нет: "; cin >> answer;

}

if (answer == 2) {

cout << "Введите кол-во вершин графа: "; cin >> amountVert;

cout << "Введите кол-во ребер графа: "; cin >> amountEdge;

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

tempgraph.addVertex(i + 1);

cout << "Вершина " << i + 1 << " создана" << endl;

SetCoord(i, amountVert, tempcities, tempgraph);

}

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

cout << "Исходная вершина: "; cin >> initialVertex;

cout << "Конечная вершина: "; cin >> endVertex;

cout << "Вес ребра: "; cin >> edgeWeight;

tempgraph.AddEdge(initialVertex, endVertex, edgeWeight);

Line connect;

connect.x1 = tempcities[tempgraph.GetPos(initialVertex)].xname;

connect.y1 = tempcities[tempgraph.GetPos(initialVertex)].yname;

connect.x2 = tempcities[tempgraph.GetPos(endVertex)].xname;

connect.y2 = tempcities[tempgraph.GetPos(endVertex)].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0);

connect.name.setString(to\_string(edgeWeight));

tempconnections.push\_back(connect);

}

graph = tempgraph;

connections = tempconnections;

bestconnections = tempbestconnections;

cities = tempcities;

}

if (answer == 1) {

*//graph.matrix = {*

*// // 1 2 3 4 5 6*

*// /\*1\*/ { 0, 12, 18, 5, 0, 34},*

*// /\*2\*/ { 12, 0, 16, 0, 53, 45 },*

*// /\*3\*/ { 18, 16, 0, 3, 21, 0 },*

*// /\*4\*/ { 5, 0, 3, 0, 0, 0 },*

*// /\*5\*/ { 0, 53, 21, 0, 0, 0 },*

*// /\*6\*/ { 34, 45, 0, 0, 0, 0 }*

*//};*

*//tempgraph.matrix = {*

*// // 1 2 3 4 5*

*// /\*1\*/ { 0, 10, 15, 5, 12},*

*// /\*2\*/ { 10, 0, 5, 17, 40},*

*// /\*3\*/ { 15, 5, 0, 5, 21},*

*// /\*4\*/ { 5, 17, 5, 0, 30},*

*// /\*5\*/ { 12, 40, 21, 30, 0},*

*//};*

amountVert = 5;

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

tempgraph.ReadyaddVertex(i + 1);

SetCoord(i, amountVert, tempcities, tempgraph);

}

tempgraph.AddEdge(1, 2, 10);

tempgraph.AddEdge(1, 3, 15);

tempgraph.AddEdge(1, 4, 5);

tempgraph.AddEdge(1, 5, 12);

tempgraph.AddEdge(2, 3, 5);

tempgraph.AddEdge(2, 4, 17);

tempgraph.AddEdge(2, 5, 40);

tempgraph.AddEdge(3, 4, 5);

tempgraph.AddEdge(3, 5, 21);

tempgraph.AddEdge(4, 5, 30);

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

{

for (int j = i + 1; j < amountVert; j++)

{

int a = tempgraph.matrix[i][j];

if (a != 0)

{

Line connect;

connect.x1 = tempcities[i].xname;

connect.y1 = tempcities[i].yname;

connect.x2 = tempcities[j].xname;

connect.y2 = tempcities[j].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0 + 2);

connect.name.setString(to\_string(a));

tempconnections.push\_back(connect);

}

}

}

graph = tempgraph;

connections = tempconnections;

bestconnections = tempbestconnections;

cities = tempcities;

}

}

if (x > 0 && y > 0 && x >= Drawx && x <= Drawx + DrawW && y >= Drawy && y <= Drawy + DrawH) {

graph.PrintMatrix();

}

if (x > 0 && y > 0 && x >= Addx && x <= Addx + AddW && y >= Addy && y <= Addy + AddH) {

vector<Line> tempconnections;

vector<Line> tempbestconnections;

vector<Node> tempcities;

Graph<int> tempgraph = graph;

tempbestconnections = bestconnections;

tempconnections = connections;

if (graph.vertexList.size() != 0)graph.addVertex(\*max\_element(graph.vertexList.begin(), graph.vertexList.end()) + 1);

if (graph.vertexList.size() == 0)graph.addVertex(graph.vertexList.size() + 1);

int amountVert = graph.vertexList.size();

cout << "Вершина " << graph.vertexList.size() << " создана" << endl;

cities.clear();

connections.clear();

bestconnections.clear();

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

SetCoord(i, amountVert, cities, graph);

}

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

{

for (int j = i + 1; j < amountVert; j++)

{

int a = graph.matrix[i][j];

if (a != 0)

{

Line connect;

connect.x1 = cities[i].xname;

connect.y1 = cities[i].yname;

connect.x2 = cities[j].xname;

connect.y2 = cities[j].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0 + 2);

connect.name.setString(to\_string(a));

connections.push\_back(connect);

}

}

}

}

if (x > 0 && y > 0 && x >= Delx && x <= Delx + DelW && y >= Dely && y <= Dely + DelH) {

int delvert;

if (graph.vertexList.empty()) {

cout << "Вершин для удаления нет" << endl;

break;

}

vector<Line> tempconnections;

tempconnections = connections;

connections.clear();

cout << "Введите вершину, которую нужно удалить "; cin >> delvert;

bool t1 = false;

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

if (graph.vertexList[i] == delvert)

t1 = true;

}

if (!t1) {

cout << "В графе нет данной вершины " << endl;

cities.clear();

connections.clear();

bestconnections.clear();

int amountVert = graph.vertexList.size();

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

SetCoord(i, amountVert, cities, graph);

}

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

{

for (int j = i + 1; j < amountVert; j++)

{

int a = graph.matrix[i][j];

if (a != 0)

{

Line connect;

connect.x1 = cities[i].xname;

connect.y1 = cities[i].yname;

connect.x2 = cities[j].xname;

connect.y2 = cities[j].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0 + 2);

connect.name.setString(to\_string(a));

connections.push\_back(connect);

}

}

}

break;

}

else {

vector<int> xx;

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

xx.push\_back(graph.GetWeight(delvert, i));

}

int vertPos2 = graph.GetPos(delvert);

for (auto i : tempconnections) {

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

if (i.strname == to\_string(xx[j])) {

tempconnections.erase(tempconnections.begin() + vertPos2);

vertPos2 += 1;

}

}

}

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

graph.DeleteEdge(delvert, i);

}

int vertPos1 = graph.GetPos(delvert);

graph.DeleteVertex(delvert);

for (auto i : cities) {

if (i.strname == to\_string(delvert)) {

cities.erase(cities.begin() + vertPos1);

break;

}

}

connections = tempconnections;

cities.clear();

connections.clear();

bestconnections.clear();

int amountVert = graph.vertexList.size();

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

SetCoord(i, amountVert, cities, graph);

}

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

{

for (int j = i + 1; j < amountVert; j++)

{

int a = graph.matrix[i][j];

if (a != 0)

{

Line connect;

connect.x1 = cities[i].xname;

connect.y1 = cities[i].yname;

connect.x2 = cities[j].xname;

connect.y2 = cities[j].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0 + 2);

connect.name.setString(to\_string(a));

connections.push\_back(connect);

}

}

}

}

}

if (x > 0 && y > 0 && x >= DelEdgex && x <= DelEdgex + DelEdgeW && y >= DelEdgey && y <= DelEdgey + DelEdgeH) {

if (graph.vertexList.empty()) {

cout << "Ребер для удаления нет" << endl;

break;

}

if (graph.GetAmountEdges() == 0) {

cout << "Ребер для удаления нет" << endl;

break;

}

int deledge1, deledge2;

cout << "Введите первую вершину ребра для удаления "; cin >> deledge1;

cout << endl;

cout << "Введите вторую вершину ребра для удаления "; cin >> deledge2;

bool t1 = false;

bool t2 = false;

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

if (graph.vertexList[i] == deledge1)

t1 = true;

if (graph.vertexList[i] == deledge2)

t2 = true;

}

if (!t1 || !t2) {

cout << "В графе нет данного ребра" << endl;

break;

}

else {

graph.DeleteEdge(deledge1, deledge2);

cities.clear();

connections.clear();

bestconnections.clear();

int amountVert = graph.vertexList.size();

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

SetCoord(i, amountVert, cities, graph);

}

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

{

for (int j = i + 1; j < amountVert; j++)

{

int a = graph.matrix[i][j];

if (a != 0)

{

Line connect;

connect.x1 = cities[i].xname;

connect.y1 = cities[i].yname;

connect.x2 = cities[j].xname;

connect.y2 = cities[j].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0 + 2);

connect.name.setString(to\_string(a));

connections.push\_back(connect);

}

}

}

}

}

if (x > 0 && y > 0 && x >= AddEdgex && x <= AddEdgex + AddEdgeW && y >= AddEdgey && y <= AddEdgey + AddEdgeH) {

if (graph.vertexList.empty()) {

cout << "Вершин для добавления ребер нет" << endl;

break;

}

int initialVertex, endVertex, edgeWeight;

cout << "Исходная вершина: "; cin >> initialVertex;

cout << "Конечная вершина: "; cin >> endVertex;

cout << "Вес ребра: "; cin >> edgeWeight;

bool t1 = false;

bool t2 = false;

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

if (graph.vertexList[i] == initialVertex)

t1 = true;

if (graph.vertexList[i] == endVertex)

t2 = true;

}

if (!t1 || !t2) {

cout << "В графе нет данной вершины " << endl;

break;

}

else {

graph.AddEdge(initialVertex, endVertex, edgeWeight);

Line connect;

connect.x1 = cities[graph.GetPos(initialVertex)].xname;

connect.y1 = cities[graph.GetPos(initialVertex)].yname;

connect.x2 = cities[graph.GetPos(endVertex)].xname;

connect.y2 = cities[graph.GetPos(endVertex)].yname;

connect.linest.push\_back(Vertex(Vector2f(connect.x1, connect.y1), Color::Black));

connect.linest.push\_back(Vertex(Vector2f(connect.x2, connect.y2), Color::Black));

connect.name.setFont(font);

connect.name.setCharacterSize(25);

connect.name.setFillColor(sf::Color::Black);

connect.name.setPosition((connect.x2 + connect.x1) / 2.0, (connect.y2 + connect.y1) / 2.0);

connect.name.setString(to\_string(edgeWeight));

connections.push\_back(connect);

}

*/\*graph = tempgraph;*

*connections = tempconnections;*

*bestconnections = tempbestconnections;*

*cities = tempcities;\*/*

}

break;

}

else break;

case Event::Closed:

MainWindow.close();

break;

}

}

MainWindow.clear(Color (135, 206, 250));

MainWindow.draw(back);

MainWindow.draw(Answer);

MainWindow.draw(TextAnswer);

MainWindow.draw(Draw);

MainWindow.draw(TextDraw);

MainWindow.draw(New);

MainWindow.draw(TextNew);

MainWindow.draw(Add);

MainWindow.draw(TextAdd);

MainWindow.draw(Del);

MainWindow.draw(TextDel);

MainWindow.draw(DelEdge);

MainWindow.draw(TextDelEdge);

MainWindow.draw(AddEdge);

MainWindow.draw(TextAddEdge);

for (auto i : connections) {

MainWindow.draw(&i.linest[0], i.linest.size(), Lines);

MainWindow.draw(i.name);

}

for (auto i : bestconnections) {

MainWindow.draw(&i.linest[0], i.linest.size(), Lines);

}

for (auto i : cities) {

MainWindow.draw(i.circle);

MainWindow.draw(i.nodename);

}

MainWindow.display();

}

return 0;

}

**Творческая работа калькулятор логарифмов**

**Постановка задачи**

Создать калькулятор вычисляющий математические операции, тригонометрические и обратные тригонометрические функции.

**Анализ задачи**

* + - 1. Собрать интерфейс в конструкторе Windows Forms.
      2. Добавить обработку неверного ввода.

Если введены не числа, вызывает окно с сообщением о неверном вводе.

bool check\_input\_textbox\_1() {

if (textBox1->Text->Length == 0) return false;

for (int i = 0; i < textBox1->Text->Length; ++i) {

if ((textBox1->Text[i] >= '0' && textBox1->Text[i] <= '9') || textBox1->Text[i] == ',' || textBox1->Text[i] == '-') {}

else return false;

return true;

}

3.Добавить события при нажатии кнопок.

Кнопка очистки

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

textBox4->Text = ""; textBox5->Text = "";

}

Кнопка вычисления

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

if (check\_input\_textbox\_1() && check\_input\_textbox\_2()) {

double number\_1, number\_2, result = 0;

number\_1 = System::Convert::ToDouble(textBox1->Text);

number\_2 = System::Convert::ToDouble(textBox2->Text);

if (comboBox1->SelectedItem == "Сложение") {

result = number\_1 + number\_2;

}

else if (comboBox1->SelectedItem == "Вычитание") {

result = number\_1 - number\_2;

}

else if (comboBox1->SelectedItem == "Умножение") {

result = number\_1 \* number\_2;

}

else if (comboBox1->SelectedItem == "Деление") {

if (number\_2 != 0) result = number\_1 / number\_2;

else MessageBox::Show("Нельзя делить на 0");

}

else if (comboBox1->SelectedItem == "Нахождение остатка") {

if (number\_2 != 0) result = (int)number\_1 % (int)number\_2;

else MessageBox::Show("Нельзя делить на 0");

}

else if (comboBox1->SelectedItem == "Возведение в степень") {

result = pow(number\_1,number\_2);

}

else if (comboBox1->SelectedItem == "Нахождение корня") {

if (number\_2 > 0 && number\_1 >= 0) result = pow(number\_1, 1/number\_2);

if (number\_2 <= 0) MessageBox::Show("Степень корня должна быть больше нуля");

if (number\_1 < 0) MessageBox::Show("Число под корнем должно быть не меньше нуля");

}

else if (comboBox1->SelectedItem == "Нахождение логарифма") {

if (number\_1 > 0 && number\_2 > 0 && number\_1!=1) result = log(number\_2)/log(number\_1);

if (number\_1 <= 0) MessageBox::Show("Логарифмическое основание должно быть больше 0");

if (number\_2 <= 0) MessageBox::Show("Логарифмируемое число должно быть больше 0 ");

if (number\_1 == 1) MessageBox::Show("Логарифмическое основание не должно быть равно 1");

}

else if (comboBox1->SelectedItem == "Нахождение факториала") {

textBox2->Text = System::Convert::ToString(0);

long long int F = 1;

for (int i = 1; i <= number\_1; ++i)

{

F \*= i;

}

result = F;

}

else return;

textBox3->Text = System::Convert::ToString(result);

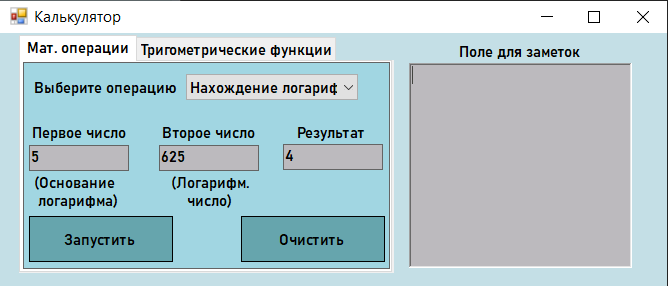
}

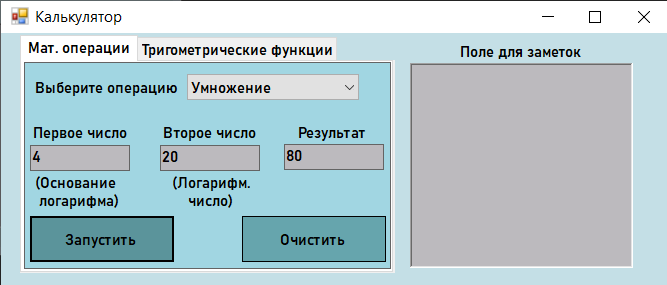
else MessageBox::Show("Данные введены некорректно");

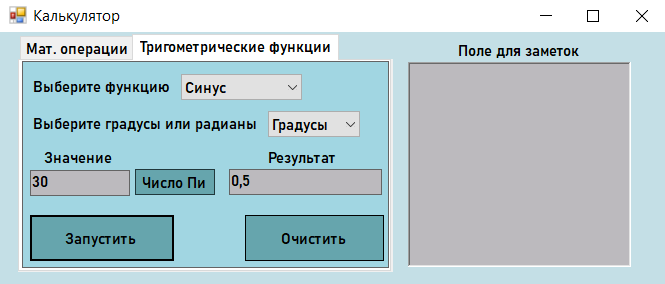
}

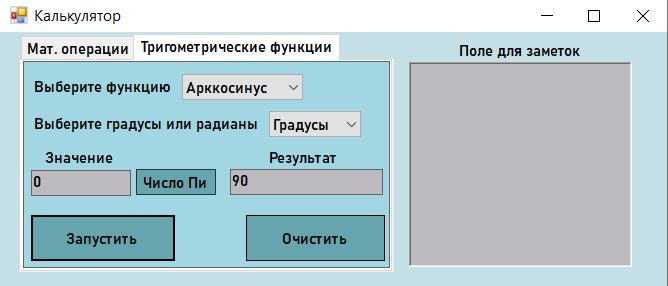
4.Добавить вычисления операций.

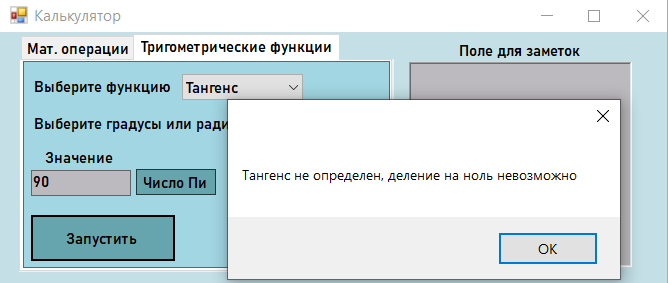
**Скриншоты работы**

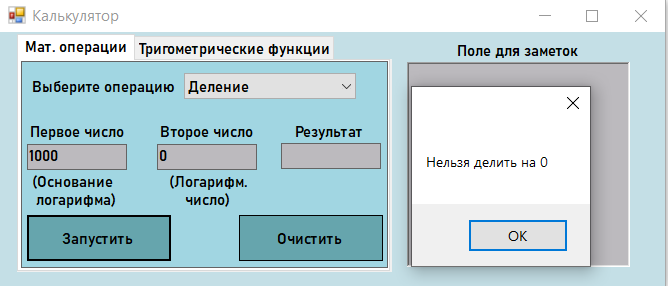












**Код программы (MyForm.cpp)**

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(array<String^>^ arg) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Calculator::MyForm Calculator;

Application::Run(% Calculator);

}

**Код программы (MyForm.h)**

#define \_USE\_MATH\_DEFINES

#include<cmath>

#pragma once

namespace Calculator {

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>*

*/// Сводка для MyForm*

*/// </summary>*

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

{

public:

MyForm(void)

{

InitializeComponent();

*//*

*//TODO: добавьте код конструктора*

*//*

}

protected:

*/// <summary>*

*/// Освободить все используемые ресурсы.*

*/// </summary>*

~MyForm()

{

if (components)

{

delete components;

}

}

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

private: System::Windows::Forms::ComboBox^ comboBox1;

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

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

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

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

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TabControl^ tabControl1;

private: System::Windows::Forms::TabPage^ tabPage1;

private: System::Windows::Forms::TabPage^ tabPage2;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::ComboBox^ comboBox2;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::ComboBox^ comboBox3;

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

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

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

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Label^ label11;

private: System::ComponentModel::IContainer^ components;

protected:

private:

*/// <summary>*

*/// Обязательная переменная конструктора.*

*/// </summary>*

#pragma region Windows Form Designer generated code

*/// <summary>*

*/// Требуемый метод для поддержки конструктора — не изменяйте*

*/// содержимое этого метода с помощью редактора кода.*

*/// </summary>*

void InitializeComponent(void)

{

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

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

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

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

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

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

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->tabControl1 = (gcnew System::Windows::Forms::TabControl());

this->tabPage1 = (gcnew System::Windows::Forms::TabPage());

this->label11 = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->tabPage2 = (gcnew System::Windows::Forms::TabPage());

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

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

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

this->comboBox3 = (gcnew System::Windows::Forms::ComboBox());

this->label8 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->label9 = (gcnew System::Windows::Forms::Label());

this->tabControl1->SuspendLayout();

this->tabPage1->SuspendLayout();

this->tabPage2->SuspendLayout();

this->SuspendLayout();

*//*

*// textBox1*

*//*

this->textBox1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->textBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox1->Location = System::Drawing::Point(5, 82);

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

this->textBox1->Size = System::Drawing::Size(100, 26);

this->textBox1->TabIndex = 0;

*//*

*// comboBox1*

*//*

this->comboBox1->BackColor = System::Drawing::SystemColors::Window;

this->comboBox1->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(9) {

L"Сложение", L"Вычитание", L"Умножение", L"Деление",

L"Возведение в степень", L"Нахождение корня", L"Нахождение логарифма", L"Нахождение остатка", L"Нахождение факториала"

});

this->comboBox1->Location = System::Drawing::Point(162, 11);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(172, 26);

this->comboBox1->TabIndex = 1;

this->comboBox1->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::comboBox1\_SelectedIndexChanged);

*//*

*// button1*

*//*

this->button1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(102)), static\_cast<System::Int32>(static\_cast<System::Byte>(165)),

static\_cast<System::Int32>(static\_cast<System::Byte>(173)));

this->button1->FlatAppearance->MouseDownBackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->button1->Location = System::Drawing::Point(5, 153);

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

this->button1->Size = System::Drawing::Size(144, 46);

this->button1->TabIndex = 2;

this->button1->Text = L"Запустить";

this->button1->UseVisualStyleBackColor = false;

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

*//*

*// button2*

*//*

this->button2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(102)), static\_cast<System::Int32>(static\_cast<System::Byte>(165)),

static\_cast<System::Int32>(static\_cast<System::Byte>(173)));

this->button2->FlatAppearance->MouseDownBackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->button2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->button2->Location = System::Drawing::Point(217, 153);

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

this->button2->Size = System::Drawing::Size(144, 46);

this->button2->TabIndex = 3;

this->button2->Text = L"Очистить";

this->button2->UseVisualStyleBackColor = false;

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

*//*

*// label1*

*//*

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(5, 60);

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

this->label1->Size = System::Drawing::Size(102, 18);

this->label1->TabIndex = 4;

this->label1->Text = L"Первое число";

this->label1->Click += gcnew System::EventHandler(this, &MyForm::label1\_Click);

*//*

*// label2*

*//*

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(135, 60);

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

this->label2->Size = System::Drawing::Size(101, 18);

this->label2->TabIndex = 4;

this->label2->Text = L"Второе число";

this->label2->Click += gcnew System::EventHandler(this, &MyForm::label2\_Click);

*//*

*// textBox2*

*//*

this->textBox2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->textBox2->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox2->Location = System::Drawing::Point(135, 82);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(100, 26);

this->textBox2->TabIndex = 0;

*//*

*// label3*

*//*

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(270, 60);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(76, 18);

this->label3->TabIndex = 6;

this->label3->Text = L"Результат";

this->label3->Click += gcnew System::EventHandler(this, &MyForm::label3\_Click);

*//*

*// textBox3*

*//*

this->textBox3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->textBox3->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox3->Location = System::Drawing::Point(259, 81);

this->textBox3->Name = L"textBox3";

this->textBox3->ReadOnly = true;

this->textBox3->Size = System::Drawing::Size(100, 26);

this->textBox3->TabIndex = 5;

*//*

*// label4*

*//*

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(7, 15);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(151, 18);

this->label4->TabIndex = 4;

this->label4->Text = L"Выберите операцию";

this->label4->Click += gcnew System::EventHandler(this, &MyForm::label1\_Click);

*//*

*// tabControl1*

*//*

this->tabControl1->Controls->Add(this->tabPage1);

this->tabControl1->Controls->Add(this->tabPage2);

this->tabControl1->Font = (gcnew System::Drawing::Font(L"Bahnschrift SemiBold", 9, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->tabControl1->Location = System::Drawing::Point(19, 2);

this->tabControl1->Name = L"tabControl1";

this->tabControl1->SelectedIndex = 0;

this->tabControl1->Size = System::Drawing::Size(375, 238);

this->tabControl1->TabIndex = 7;

*//*

*// tabPage1*

*//*

this->tabPage1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(161)), static\_cast<System::Int32>(static\_cast<System::Byte>(214)),

static\_cast<System::Int32>(static\_cast<System::Byte>(226)));

this->tabPage1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->tabPage1->Controls->Add(this->label11);

this->tabPage1->Controls->Add(this->label10);

this->tabPage1->Controls->Add(this->comboBox1);

this->tabPage1->Controls->Add(this->button1);

this->tabPage1->Controls->Add(this->button2);

this->tabPage1->Controls->Add(this->label3);

this->tabPage1->Controls->Add(this->textBox1);

this->tabPage1->Controls->Add(this->textBox3);

this->tabPage1->Controls->Add(this->textBox2);

this->tabPage1->Controls->Add(this->label2);

this->tabPage1->Controls->Add(this->label1);

this->tabPage1->Controls->Add(this->label4);

this->tabPage1->Font = (gcnew System::Drawing::Font(L"Bahnschrift SemiBold", 9, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->tabPage1->Location = System::Drawing::Point(4, 27);

this->tabPage1->Name = L"tabPage1";

this->tabPage1->Padding = System::Windows::Forms::Padding(3);

this->tabPage1->Size = System::Drawing::Size(367, 207);

this->tabPage1->TabIndex = 0;

this->tabPage1->Text = L"Мат. операции";

this->tabPage1->Click += gcnew System::EventHandler(this, &MyForm::tabPage1\_Click);

*//*

*// label11*

*//*

this->label11->AutoSize = true;

this->label11->Location = System::Drawing::Point(144, 110);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(86, 36);

this->label11->TabIndex = 8;

this->label11->Text = L"(Логарифм.\r\n число)";

*//*

*// label10*

*//*

this->label10->AutoSize = true;

this->label10->Location = System::Drawing::Point(7, 110);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(92, 36);

this->label10->TabIndex = 7;

this->label10->Text = L"(Основание\r\n логарифма)";

this->label10->Click += gcnew System::EventHandler(this, &MyForm::label10\_Click);

*//*

*// tabPage2*

*//*

this->tabPage2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(161)), static\_cast<System::Int32>(static\_cast<System::Byte>(214)),

static\_cast<System::Int32>(static\_cast<System::Byte>(226)));

this->tabPage2->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->tabPage2->Controls->Add(this->button5);

this->tabPage2->Controls->Add(this->button4);

this->tabPage2->Controls->Add(this->button3);

this->tabPage2->Controls->Add(this->comboBox3);

this->tabPage2->Controls->Add(this->label8);

this->tabPage2->Controls->Add(this->label7);

this->tabPage2->Controls->Add(this->textBox5);

this->tabPage2->Controls->Add(this->textBox4);

this->tabPage2->Controls->Add(this->label6);

this->tabPage2->Controls->Add(this->label5);

this->tabPage2->Controls->Add(this->comboBox2);

this->tabPage2->Font = (gcnew System::Drawing::Font(L"Bahnschrift SemiBold", 9, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->tabPage2->Location = System::Drawing::Point(4, 27);

this->tabPage2->Name = L"tabPage2";

this->tabPage2->Padding = System::Windows::Forms::Padding(3);

this->tabPage2->Size = System::Drawing::Size(367, 207);

this->tabPage2->TabIndex = 1;

this->tabPage2->Text = L"Тригометрические функции";

*//*

*// button5*

*//*

this->button5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(102)), static\_cast<System::Int32>(static\_cast<System::Byte>(165)),

static\_cast<System::Int32>(static\_cast<System::Byte>(173)));

this->button5->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->button5->Location = System::Drawing::Point(112, 107);

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

this->button5->Size = System::Drawing::Size(80, 26);

this->button5->TabIndex = 8;

this->button5->Text = L"Число Пи";

this->button5->UseVisualStyleBackColor = false;

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

*//*

*// button4*

*//*

this->button4->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(102)), static\_cast<System::Int32>(static\_cast<System::Byte>(165)),

static\_cast<System::Int32>(static\_cast<System::Byte>(173)));

this->button4->FlatAppearance->MouseDownBackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->button4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->button4->Location = System::Drawing::Point(7, 153);

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

this->button4->Size = System::Drawing::Size(144, 46);

this->button4->TabIndex = 8;

this->button4->Text = L"Запустить";

this->button4->UseVisualStyleBackColor = false;

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

*//*

*// button3*

*//*

this->button3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(102)), static\_cast<System::Int32>(static\_cast<System::Byte>(165)),

static\_cast<System::Int32>(static\_cast<System::Byte>(173)));

this->button3->FlatAppearance->MouseDownBackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(128)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(128)));

this->button3->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->button3->Location = System::Drawing::Point(222, 153);

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

this->button3->Size = System::Drawing::Size(139, 46);

this->button3->TabIndex = 8;

this->button3->Text = L"Очистить";

this->button3->UseVisualStyleBackColor = false;

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

*//*

*// comboBox3*

*//*

this->comboBox3->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox3->FormattingEnabled = true;

this->comboBox3->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"Градусы", L"Радианы" });

this->comboBox3->Location = System::Drawing::Point(245, 49);

this->comboBox3->Name = L"comboBox3";

this->comboBox3->Size = System::Drawing::Size(92, 26);

this->comboBox3->TabIndex = 10;

*//*

*// label8*

*//*

this->label8->AutoSize = true;

this->label8->Location = System::Drawing::Point(7, 52);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(232, 18);

this->label8->TabIndex = 9;

this->label8->Text = L"Выберите градусы или радианы";

this->label8->Click += gcnew System::EventHandler(this, &MyForm::label8\_Click);

*//*

*// label7*

*//*

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(242, 86);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(76, 18);

this->label7->TabIndex = 8;

this->label7->Text = L"Результат";

*//*

*// textBox5*

*//*

this->textBox5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->textBox5->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox5->Location = System::Drawing::Point(7, 108);

this->textBox5->Name = L"textBox5";

this->textBox5->Size = System::Drawing::Size(100, 26);

this->textBox5->TabIndex = 7;

*//*

*// textBox4*

*//*

this->textBox4->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->textBox4->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox4->Location = System::Drawing::Point(206, 107);

this->textBox4->Name = L"textBox4";

this->textBox4->ReadOnly = true;

this->textBox4->Size = System::Drawing::Size(153, 26);

this->textBox4->TabIndex = 7;

*//*

*// label6*

*//*

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(18, 86);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(76, 18);

this->label6->TabIndex = 2;

this->label6->Text = L"Значение";

*//*

*// label5*

*//*

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(7, 15);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(145, 18);

this->label5->TabIndex = 1;

this->label5->Text = L"Выберите функцию";

this->label5->Click += gcnew System::EventHandler(this, &MyForm::label5\_Click);

*//*

*// comboBox2*

*//*

this->comboBox2->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(8) {

L"Синус", L"Косинус", L"Тангенс", L"Котангенс",

L"Арксинус", L"Арккосинус", L"Арктангенс", L"Арккотангенс"

});

this->comboBox2->Location = System::Drawing::Point(158, 12);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(121, 26);

this->comboBox2->TabIndex = 0;

this->comboBox2->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::comboBox2\_SelectedIndexChanged);

*//*

*// richTextBox1*

*//*

this->richTextBox1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(188)), static\_cast<System::Int32>(static\_cast<System::Byte>(186)),

static\_cast<System::Int32>(static\_cast<System::Byte>(190)));

this->richTextBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->richTextBox1->Location = System::Drawing::Point(409, 30);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(223, 205);

this->richTextBox1->TabIndex = 8;

this->richTextBox1->Text = L"";

*//*

*// label9*

*//*

this->label9->AutoSize = true;

this->label9->Font = (gcnew System::Drawing::Font(L"Bahnschrift SemiBold", 9, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->label9->Location = System::Drawing::Point(456, 9);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(129, 18);

this->label9->TabIndex = 9;

this->label9->Text = L"Поле для заметок";

*//*

*// MyForm*

*//*

this->AutoScaleDimensions = System::Drawing::SizeF(8, 18);

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

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(196)), static\_cast<System::Int32>(static\_cast<System::Byte>(223)),

static\_cast<System::Int32>(static\_cast<System::Byte>(230)));

this->ClientSize = System::Drawing::Size(667, 254);

this->Controls->Add(this->label9);

this->Controls->Add(this->richTextBox1);

this->Controls->Add(this->tabControl1);

this->Font = (gcnew System::Drawing::Font(L"Bahnschrift SemiBold", 9, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->Name = L"MyForm";

this->Text = L"Калькулятор";

this->tabControl1->ResumeLayout(false);

this->tabPage1->ResumeLayout(false);

this->tabPage1->PerformLayout();

this->tabPage2->ResumeLayout(false);

this->tabPage2->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void comboBox1\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

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

textBox1->Text = ""; textBox2->Text = ""; textBox3->Text = "";

}

bool check\_input\_textbox\_1() {

if (textBox1->Text->Length == 0) return false;

for (int i = 0; i < textBox1->Text->Length; ++i) {

if ((textBox1->Text[i] >= '0' && textBox1->Text[i] <= '9') || textBox1->Text[i] == ',' || textBox1->Text[i] == '-') {}

else return false;

return true;

}

}

bool check\_input\_textbox\_2() {

if (textBox2->Text->Length == 0) return false;

for (int i = 0; i < textBox2->Text->Length; ++i) {

if ((textBox2->Text[i] >= '0' && textBox2->Text[i] <= '9') || textBox2->Text[i] == ',' || textBox2->Text[i] == '-') {}

else return false;

return true;

}

}

bool check\_input\_textbox\_5() {

if (textBox5->Text->Length == 0) return false;

for (int i = 0; i < textBox5->Text->Length; ++i) {

if ((textBox5->Text[i] >= '0' && textBox5->Text[i] <= '9') || textBox5->Text[i] == ',' || textBox5->Text[i] == '-') {}

else return false;

return true;

}

}

private: System::Void label2\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label3\_Click(System::Object^ sender, System::EventArgs^ e) {

}

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

if (check\_input\_textbox\_1() && check\_input\_textbox\_2()) {

double number\_1, number\_2, result = 0;

number\_1 = System::Convert::ToDouble(textBox1->Text);

number\_2 = System::Convert::ToDouble(textBox2->Text);

if (comboBox1->SelectedItem == "Сложение") {

result = number\_1 + number\_2;

}

else if (comboBox1->SelectedItem == "Вычитание") {

result = number\_1 - number\_2;

}

else if (comboBox1->SelectedItem == "Умножение") {

result = number\_1 \* number\_2;

}

else if (comboBox1->SelectedItem == "Деление") {

if (number\_2 != 0) result = number\_1 / number\_2;

else MessageBox::Show("Нельзя делить на 0");

}

else if (comboBox1->SelectedItem == "Нахождение остатка") {

if (number\_2 != 0) result = (int)number\_1 % (int)number\_2;

else MessageBox::Show("Нельзя делить на 0");

}

else if (comboBox1->SelectedItem == "Возведение в степень") {

result = pow(number\_1,number\_2);

}

else if (comboBox1->SelectedItem == "Нахождение корня") {

if (number\_2 > 0 && number\_1 >= 0) result = pow(number\_1, 1/number\_2);

if (number\_2 <= 0) MessageBox::Show("Степень корня должна быть больше нуля");

if (number\_1 < 0) MessageBox::Show("Число под корнем должно быть не меньше нуля");

}

else if (comboBox1->SelectedItem == "Нахождение логарифма") {

if (number\_1 > 0 && number\_2 > 0 && number\_1!=1) result = log(number\_2)/log(number\_1);

if (number\_1 <= 0) MessageBox::Show("Логарифмическое основание должно быть больше 0");

if (number\_2 <= 0) MessageBox::Show("Логарифмируемое число должно быть больше 0 ");

if (number\_1 == 1) MessageBox::Show("Логарифмическое основание не должно быть равно 1");

}

else if (comboBox1->SelectedItem == "Нахождение факториала") {

textBox2->Text = System::Convert::ToString(0);

long long int F = 1;

for (int i = 1; i <= number\_1; ++i)

{

F \*= i;

}

result = F;

}

else return;

textBox3->Text = System::Convert::ToString(result);

}

else MessageBox::Show("Данные введены некорректно");

}

private: System::Void label5\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void comboBox2\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label8\_Click(System::Object^ sender, System::EventArgs^ e) {

}

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

textBox4->Text = ""; textBox5->Text = "";

}

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

if (check\_input\_textbox\_5()) {

double number\_5,tresult = 0;

number\_5 = System::Convert::ToDouble(textBox5->Text);

if (comboBox2->SelectedItem == "Синус" && comboBox3->SelectedItem == "Градусы") {

if ((int)number\_5 % 180 == 0) tresult = 0;

else tresult = sin(number\_5 \* M\_PI/180.0);

}

else if (comboBox2->SelectedItem == "Синус" && comboBox3->SelectedItem == "Радианы") {

if (number\_5 == M\_PI) tresult = 0;

else tresult = sin(number\_5);

}

else if (comboBox2->SelectedItem == "Косинус" && comboBox3->SelectedItem == "Градусы") {

if ((int)number\_5 % 90 == 0 && (int)number\_5 % 180 != 0) tresult = 0;

else tresult = cos(number\_5 \* M\_PI / 180.0);

}

else if (comboBox2->SelectedItem == "Косинус" && comboBox3->SelectedItem == "Радианы") {

tresult = cos(number\_5);

}

else if (comboBox2->SelectedItem == "Тангенс" && comboBox3->SelectedItem == "Градусы") {

if ((int)number\_5 % 90 == 0 && (int)number\_5 % 180 != 0) MessageBox::Show("Тангенс не определен, деление на ноль невозможно");

else if ((int)number\_5 % 180 == 0) tresult = 0;

else tresult = tan(number\_5 \* M\_PI / 180.0);

}

else if (comboBox2->SelectedItem == "Тангенс" && comboBox3->SelectedItem == "Радианы")

{

tresult = tan(number\_5);

}

else if (comboBox2->SelectedItem == "Котангенс" && comboBox3->SelectedItem == "Градусы") {

if ((int)number\_5 % 180 == 0) MessageBox::Show("Котангенс не определен, деление на ноль невозможно");

else if ((int)number\_5 % 90 == 0) tresult = 0;

else tresult = 1.0/tan(number\_5 \* M\_PI / 180.0);

}

else if (comboBox2->SelectedItem == "Котангенс" && comboBox3->SelectedItem == "Радианы")

{

if (number\_5 == 3, 14159265358979) MessageBox::Show("Котангенс не определен, деление на ноль невозможно");

else tresult = 1.0/tan(number\_5);

}

else if (comboBox2->SelectedItem == "Арксинус" && comboBox3->SelectedItem == "Градусы") {

if (number\_5<-1 || number\_5>1) MessageBox::Show("Значения должны быть в пределах от -1 до 1");

else tresult = asin(number\_5) \* 180.0 / M\_PI;

}

else if (comboBox2->SelectedItem == "Арксинус" && comboBox3->SelectedItem == "Радианы") {

if (number\_5 < -1 || number\_5>1) MessageBox::Show("Значения должны быть в пределах от -1 до 1");

else tresult = asin(number\_5);

}

else if (comboBox2->SelectedItem == "Арккосинус" && comboBox3->SelectedItem == "Градусы") {

if (number\_5 < -1 || number\_5>1) MessageBox::Show("Значения должны быть в пределах от -1 до 1");

else tresult = acos(number\_5) \* 180.0 / M\_PI;

}

else if (comboBox2->SelectedItem == "Арккосинус" && comboBox3->SelectedItem == "Радианы") {

if (number\_5 < -1 || number\_5>1) MessageBox::Show("Значения должны быть в пределах от -1 до 1");

else tresult = acos(number\_5);

}

else if (comboBox2->SelectedItem == "Арктангенс" && comboBox3->SelectedItem == "Градусы") {

tresult = atan(number\_5) \* 180.0 / M\_PI;

}

else if (comboBox2->SelectedItem == "Арктангенс" && comboBox3->SelectedItem == "Радианы") {

tresult = atan(number\_5);

}

else if (comboBox2->SelectedItem == "Арккотангенс" && comboBox3->SelectedItem == "Градусы") {

tresult = (M\_PI/2.0 - atan(number\_5)) \* 180.0 / M\_PI;

}

else if (comboBox2->SelectedItem == "Арккотангенс" && comboBox3->SelectedItem == "Радианы") {

tresult = (M\_PI / 2.0 - atan(number\_5));

}

else return;

textBox4->Text = System::Convert::ToString(tresult);

}

else MessageBox::Show("Данные введены некорректно");

}

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

textBox5->Text = System::Convert::ToString(M\_PI);

}

private: System::Void tabPage1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label10\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}