Министерство образования и науки Российской Федерации

Федеральное государственное образовательное автономное учреждение высшего образования

"Пермский национальный исследовательский политехнический университет"

ОТЧЕТ

ПО ТВОРЧЕСКОЙ РАБОТЕ.

Дисциплина: Основы алгоритмизации и программирования.

Тема: АРМ Флориста. Задача Комиваязжера.

Выполнила работу

студентка группы РИС-23-3б

Мокроусова А.A.

Проверила

Доцент кафедры ИТАС

Полякова О.А.

Пермь – 2024

Введение

На текущем этапе развития информационных технологий люди смогли найти общий язык с компьютером. Создаются различные компьютерные игры, мобильные приожения, сайты и многое другое, что позволяет произовдить взаимодействие человека с машиной. В современном мире, где информация перегружает нас со всех сторон, визуализация становится не просто инструментом, а необходимым навыком для любого разработчика. Не важно, создаете ли вы веб-приложения, анализируете данные, проектируете игры или разрабатываете научные модели. Разработка визуальных приложений и инструментов становится все более востребованной. Осваивая базовые методы визуализации, вы расширяете свой арсенал инструментов, улучшаете коммуникационные навыки и повышаете свою ценность на рынке труда.

Целью работы является освоение базовых методов визуализации

Для достижения поставленной цели необходимо решить следующие задачи:

1. Произвести анализ предментной области.
2. Создать автоматизированное место для работы флориста.
3. Решить задачу комивозжера.

# **Анализ предметной области.**

В мире компьютерной графики существует множество инструментов для создания визуальных эффектов. Два из самых популярных и доступных: SFML (Simple and Fast Multimedia Library) и OpenGL. Каждый из них обладает своими сильными сторонами, предоставляя разработчикам выбор в зависимости от потребностей проекта.

SFML - это кроссплатформенная библиотека C++, которая предоставляет простой и интуитивный API для работы с графикой, звуком, вводом и сетью. SFML отлично подходит для создания 2D-игр, графических приложений и визуализации данных. Она предлагает множество готовых классов для рисования спрайтов, текста, геометрических фигур, а также работы с текстурами, шрифтами и анимацией.

OpenGL - это стандартный API для работы с 3D-графикой. Он предоставляет низкоуровневый доступ к графическому процессору (GPU) и позволяет создавать сложные визуальные эффекты, такие как освещение, текстурирование, теневые эффекты и многие другие. OpenGL дает большую свободу, но требует более глубокого понимания графической архитектуры.

# **АРМ Флориста.**

## Постановка задачи:

Создать приложение для автоматизации работы флориста, визуализировать его работу при помощи графической библиотеки SFML на языке C++.

## Анализ задачи:

* Необходимо создание каталога цветов и каталога стандартный ваз.
* Реализовать возможность введения пользователем вышеперечисленного в приложение
* Необходим сбор информации о различных видов цветов и их особеностях.
* Необходимо реальзовать функцию позволяющую расфосовать цветы по вазам с учетом заданных ограничений.
* Необходимо реальзовать сохранение полученного результата на внешние носители.
* Вдумчиво продумать интерфейс и логику работу будующего приложения.

## Преимущества:

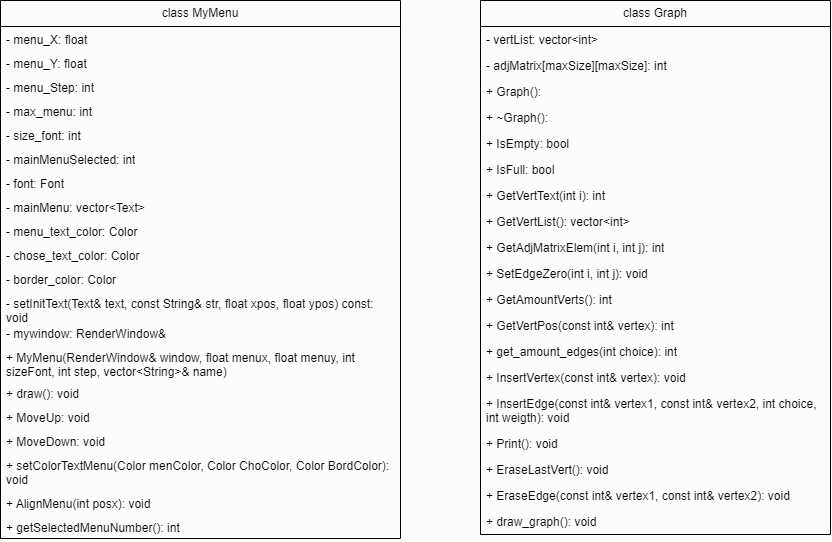
* Упрощение работы флориста: Автоматизация рутинных задач позволяет флористам сосредоточиться на творческой работе.
* Повышение производительности: Приложение позволяет сократить время при поступлении крупных партий товаров.
* Снижение необходимой кваификации: флористу для расфасовки цветов уже не будут требоваться знания об ограничениях в совмещении цветов разных видов.
* Визуализация: Помогает флористам лучше представить конечный результат.
* Учет ограничений: Уверенность в том, что цветы будут правильно размещены в соответствии с совместимостью.
* Появление дополнительного метода отслеживания дат поставок цветов а значит и контроля качества продукции.

## Технические требования:

* Язык программирования: C++.
* Графическая библиотека: SFML.
* Среда разработки: Visual Studio 2022.
* Операционная система: Windows.

## UML диаграмма:

На рисунке 1 представлена UML диаграмма алгоритма программы.



***Рисунок 1 – UML диаграмма АРМ Флориста.***

## Код на С++ с комментариями:

### *//main.cpp*

lude <SFML/Graphics.hpp>

#include<SFML/Audio.hpp>

#include"MyMenu.h"

#include<Windows.h>

#include<iostream>

#include <vector>

#include <fstream>

#include <map>

#include <ctime>

#include <iomanip>

using namespace sf;

using namespace std;

struct TextFormat

{

int size\_font = 60; // Размер шрифта

Color menu\_text\_color = Color::White; // Цвет текста

float bord = 0.0f; // Толщина обводки

Color border\_color = Color::Black; // Цвет обводки

};// Структура для хранения параметров форматирования текста

// Функция настройки текста

void InitText(Text& mtext, float xpos, float ypos, const String& str, TextFormat Ftext)

{

// Установка параметров текста

mtext.setCharacterSize(Ftext.size\_font);

mtext.setPosition(xpos, ypos);

mtext.setString(str);

mtext.setFillColor(Ftext.menu\_text\_color);

mtext.setOutlineThickness(Ftext.bord);

mtext.setOutlineColor(Ftext.border\_color);

}

// Структура для представления элемента

struct Element {

int type; // Тип элемента

int count; // Количество элементов

int not\_type; // Тип элемента, который не должен быть в одной коробке с данным

};

// Функция для проверки, можно ли добавить элемент в коробку

bool canAddToBox(vector<int>& box, int type, int not\_type) {

for (int t : box) {

if (t == not\_type) {

return false; // Найден элемент, который не должен быть вместе

}

}

return true;

}

// Функция для расфасовки элементов по коробкам

void packElements(int z1, int z2, int z3, int z4, int z5, int zv1, int zv2, int zv3) {

int boxCount10 = zv3, boxCount5 = zv2, boxCount3 = zv1;

// Создаем вектор boxSizes

vector<int> boxSizes(boxCount10 + boxCount5 + boxCount3);

for (int i = 0; i < boxCount10; i++) {

boxSizes[i] = 10;

}

for (int i = boxCount10; i < boxCount10 + boxCount5; i++) {

boxSizes[i] = 5;

}

for (int i = boxCount10 + boxCount5; i < boxCount10 + boxCount5 + boxCount3; i++) {

boxSizes[i] = 3;

}

vector<Element> elements = {

{1, z1, 5}, // Тип 1, 4 элемента, не должен быть с типом 2

{2, z2, 0}, // Тип 2, 3 элемента, не должен быть с типом 1

{3, z3, 4}, // Тип 3, 2 элемента, не должен быть с типом 3

{4, z4, 3}, // Тип 4, 5 элементов, не должен быть с типом 2

{5, z5, 1}, // Тип 5, 1 элемент, не должен быть с типом 3

};

// Создаем коробки

vector<vector<int>> boxes(boxSizes.size());

// Используем map для подсчета не влезших элементов

map<int, int> notFitElements;

// Перебираем все элементы

for (auto& elem : elements) {

int type = elem.type; // Тип текущего элемента

int count = elem.count; // Количество текущего элемента

int not\_type = elem.not\_type; // Тип, который не должен быть в одной коробке

// Расфасовываем элемент по коробкам

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

bool placed = false;

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

if (boxes[j].size() < boxSizes[j] && canAddToBox(boxes[j], type, not\_type)) {

boxes[j].push\_back(type);

placed = true;

break;

}

}

// Если нет места ни в одной коробке или нарушается условие, увеличиваем счетчик для данного типа

if (!placed) {

notFitElements[type]++;

}

}

}

// Записываем содержимое коробок в файл

ofstream outFile("result.txt");

// Получаем текущее время и дату

time\_t now = time(0);

tm ltm;

localtime\_s(&ltm, &now);

// Форматируем дату и время

outFile << "===================================================================================" << endl;

outFile << "Дата и время поставки: "

<< ltm.tm\_mday << "/"

<< ltm.tm\_mon + 1 << "/"

<< ltm.tm\_year + 1900 << " "

<< setfill('0') << setw(2) << ltm.tm\_hour << ":"

<< setw(2) << ltm.tm\_min << endl;

outFile << "===================================================================================" << endl;

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

outFile << i + 1 << ") \t" << "Ваза " << " размером " << boxSizes[i] << ":" << endl << "\t";

for (int j = 0; j < boxes[i].size(); j++) {

outFile << boxes[i][j] << " ";

}

outFile << endl;

}

outFile << "===================================================================================" << endl;

// Вывод информации о не влезших элементах

if (!notFitElements.empty()) {

outFile << "Не все цветы можно разместить в вазы: " << endl;

for (auto it : notFitElements) {

outFile << "\t" << "Вид цветка: " << it.first << ". Количество: " << it.second << " шт." << endl;

}

outFile << "===================================================================================" << endl;

}

outFile.close();

}

void Start();// Основное

void About();// Описание

int main()

{

RenderWindow win(VideoMode(1280, 720), "ARM");// Создание окна

win.setFramerateLimit(60);//Количество кадров в секунду

// Загрузка иконки

Image icon;

icon.loadFromFile("Image/icon1.png");

win.setIcon(1200, 1084, icon.getPixelsPtr());

win.setMouseCursorVisible(false);// Отключение видимости курсора

Music music;// Вщзпроизведение музыкальной дороожки

music.openFromFile("music.ogg");

music.play();

music.setLoop(true);

music.setVolume(2.f);

// Создание рамки

Image image;

image.loadFromFile("Image/panel.jpg");

image.createMaskFromColor(image.getPixel(0, 0));

Texture texture;

texture.loadFromImage(image);

Sprite sprite;

sprite.setTexture(texture);

sprite.setPosition(Vector2f(330, 50));

// Создание движущегося фона

Texture textureFon;

textureFon.loadFromFile("Image/fon.jpg");

RectangleShape fon(Vector2f(1280, 720));

fon.setTexture(&textureFon);

RectangleShape fon2(Vector2f(1280, 720));

fon2.setTexture(&textureFon);

fon2.setPosition(Vector2f(1280, 0));

// Загрузка шрифта

Font font;

if (!font.loadFromFile("font/myFont.otf")) return 5;

// Создание заголовка

Text Titul;

Titul.setFont(font);

TextFormat Ftext;

Ftext.size\_font = 45;

Ftext.menu\_text\_color = Color{ 20, 20, 20};

Ftext.bord = 0;

InitText(Titul, 450, 250, L"АРМ ФЛОРИСТА", Ftext);

//Основное меню

vector<String> name\_menu{ L"НАЧАТЬ", L"ОПИСАНИЕ", L"ВЫЙТИ" };// Название пунктов меню

game::MyMenu mymenu(win, 625 , 320, 35, 35, name\_menu);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu.setColorTextMenu(Color(200,200,200), Color(15,15,15), Color::Black);// Установка цвета отображения меню

mymenu.AlignMenu(2); //Выравнивание пунктов меню

Vector2f pos;// Координаты объектов

Clock clock; // Таймер

float time;

// Основной цикл

while (win.isOpen())

{

// Обработка событий

Event event;

while (win.pollEvent(event))

{

if (event.type == sf::Event::Closed) //Для закрытия через крестик

win.close();

if (event.type == Event::KeyReleased)

{

// События выбра пунктов меню

if (event.key.code == Keyboard::Up) { mymenu.MoveUp(); } // вверх

if (event.key.code == Keyboard::Down) { mymenu.MoveDown(); } // вниз

if (event.key.code == Keyboard::Return) // ввод

{

// Переходим на выбранный пункт меню

switch (mymenu.getSelectedMenuNumber())

{

case 0:Start(); break;

case 1:About(); break;

case 2:win.close(); break;

default:break;

}

}

}

}

// Обновление таймера

time = clock.getElapsedTime().asMicroseconds();

time = time / 15000;

clock.restart();

// Движение фона

fon.move(-0.2 \* time, 0);

pos = fon.getPosition();

if (pos.x < -1280) fon.setPosition(1280, pos.y);

fon2.move(-0.2 \* time, 0);

pos = fon2.getPosition();

if (pos.x < -1280) fon2.setPosition(1280, pos.y);

win.clear();// Очистка экрана

win.draw(fon);// Отрисовка элементов игры

win.draw(fon2);

win.draw(sprite);

mymenu.draw();

win.draw(Titul);

win.display();// Обновление экрана

}

return 0;

}

// Основное

void Start()

{

RenderWindow win(VideoMode(1280, 720), "ARM"); // Создание окна

win.setFramerateLimit(60);

// Загрузка иконки

Image icon;

icon.loadFromFile("Image/icon1.png");

win.setIcon(1200, 1084, icon.getPixelsPtr());

// Загрузка шрифта

Font font;

font.loadFromFile("font/myFont2.ttf");

// Цветы сбоку

Image image;

image.loadFromFile("Image/panel2.jpg");

image.createMaskFromColor(image.getPixel(30, 30));

Texture texture;

texture.loadFromImage(image);

Sprite panel;

panel.setTexture(texture);

panel.setPosition(Vector2f(595, 0));

// Создание рамки вокруг текста

Image image2;

image2.loadFromFile("Image/panel3.jpg");

image2.createMaskFromColor(image2.getPixel(0, 0));

Texture texture2;

texture2.loadFromImage(image2);

Sprite panel2;

panel2.setTexture(texture2);

panel2.setPosition(Vector2f(60, 60));

panel2.setScale(550 / panel2.getLocalBounds().width, 350 / panel2.getLocalBounds().height);

// Создание движущегося фона

Texture textureFon;

textureFon.loadFromFile("Image/fon.jpg");

RectangleShape fon(Vector2f(1280, 720));

fon.setTexture(&textureFon);

RectangleShape fon2(Vector2f(1280, 720));

fon2.setTexture(&textureFon);

fon2.setPosition(Vector2f(1280, 0));

// Создание заголовка

Text Titul;

Titul.setFont(font);

TextFormat Ftext;

Ftext.size\_font = 35;

Ftext.menu\_text\_color = Color{ 20, 20, 20 };

Ftext.bord = 0;

InitText(Titul, 123, 100, L"ВЫБЕРИТЕ НУЖНЫЙ ПУНКТ", Ftext);

//Основное меню

vector<String> name\_menu{ L"Управление цветами", L"Управление вазами", L"Получить результат", L"Назад"}; // Название пунктов меню

game::MyMenu mymenu(win, 330, 170, 35, 45, name\_menu);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu.setColorTextMenu(Color(200, 200, 200), Color(15, 15, 15), Color::Black);// Установка цвета отображения меню

mymenu.AlignMenu(2);//Выпавнивание пунктов меню

//Фон основного мею

RectangleShape button(Vector2f(470, 290));

button.setFillColor(Color::White);

button.setPosition(100, 90);

button.setOutlineThickness(1);

button.setOutlineColor(Color(100, 100, 100));

//ДЛЯ УПРАВЛЕНИЯ ЦВЕТАМИ

bool menuExpanded = false;//Окошко после нажатия на управление цветами

int x = 640;

int y = 90;

//ПЕРВОЕ ОКОШКО С ЦВЕТКОМ

RectangleShape for\_flovers1(Vector2f(330, 50));//Для конкретного цветка окошко

for\_flovers1.setFillColor(Color::White);

for\_flovers1.setOutlineThickness(1);

for\_flovers1.setOutlineColor(Color(100, 100, 100));

for\_flovers1.setPosition(x, y );

vector<String> name\_menu1{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu1(win, x+305, y, 30, 15, name\_menu1);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu1.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu1.AlignMenu(2);//Выпавнивание пунктов меню

String f1 = L"1. Гвоздика садовая";// Название конкретного цветка

Text name\_flovers1;

name\_flovers1.setFont(font);

name\_flovers1.setFillColor(Color(15,15,15));

name\_flovers1.setString(f1);

name\_flovers1.setCharacterSize(30);

name\_flovers1.setPosition(for\_flovers1.getPosition().x + 5, for\_flovers1.getPosition().y+ 5);

RectangleShape for\_zifri1(Vector2f(30, 30));//Колво цветов окошко

for\_zifri1.setFillColor(Color::White);

for\_zifri1.setPosition(for\_flovers1.getPosition().x + 260, for\_flovers1.getPosition().y + 10);

for\_zifri1.setOutlineThickness(1);

for\_zifri1.setOutlineColor(Color(100, 100, 100));

int z1 = 0;// Кол во цветов цифры

Text zifri1;

zifri1.setFont(font);

zifri1.setFillColor(Color::Black);

zifri1.setString(to\_string(z1));

zifri1.setCharacterSize(25);

zifri1.setOutlineThickness(0.1);

zifri1.setOutlineColor(Color::Black);

zifri1.setPosition(for\_zifri1.getPosition().x + 2, for\_zifri1.getPosition().y);

//ВТОРОЕ ОКОШКО С ЦВЕТКОМ

RectangleShape for\_flovers2(Vector2f(330, 50));//Для конкретного цветка окошко

for\_flovers2.setFillColor(Color::White);

for\_flovers2.setOutlineThickness(1);

for\_flovers2.setOutlineColor(Color(100, 100, 100));

for\_flovers2.setPosition(x , y = y + 50);

vector<String> name\_menu2{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu2(win, x + 305, y, 30, 15, name\_menu2);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu2.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu2.AlignMenu(2);//Выпавнивание пунктов меню

String f2 = L"2. Ландыши майские";// Название конкретного цветка

Text name\_flovers2;

name\_flovers2.setFont(font);

name\_flovers2.setFillColor(Color(15, 15, 15));

name\_flovers2.setString(f2);

name\_flovers2.setCharacterSize(30);

name\_flovers2.setPosition(for\_flovers2.getPosition().x + 5, for\_flovers2.getPosition().y + 5);

RectangleShape for\_zifri2(Vector2f(30, 30));//Колво цветов окошко

for\_zifri2.setFillColor(Color::White);

for\_zifri2.setPosition(for\_flovers2.getPosition().x + 260, for\_flovers2.getPosition().y + 10);

for\_zifri2.setOutlineThickness(1);

for\_zifri2.setOutlineColor(Color(100, 100, 100));

int z2 = 0;// Кол во цветов цифры

Text zifri2;

zifri2.setFont(font);

zifri2.setFillColor(Color::Black);

zifri2.setString(to\_string(z2));

zifri2.setCharacterSize(25);

zifri2.setOutlineThickness(0.1);

zifri2.setOutlineColor(Color::Black);

zifri2.setPosition(for\_zifri2.getPosition().x + 2, for\_zifri2.getPosition().y);

//3 ОКОШКО С ЦВЕТКОМ

RectangleShape for\_flovers3(Vector2f(330, 50));//Для конкретного цветка окошко

for\_flovers3.setFillColor(Color::White);

for\_flovers3.setOutlineThickness(1);

for\_flovers3.setOutlineColor(Color(100, 100, 100));

for\_flovers3.setPosition(x, y = y + 50);

vector<String> name\_menu3{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu3(win, x + 305, y, 30, 15, name\_menu3);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu3.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu3.AlignMenu(2);//Выпавнивание пунктов меню

String f3 = L"3. Лилия азиатская";// Название конкретного цветка

Text name\_flovers3;

name\_flovers3.setFont(font);

name\_flovers3.setFillColor(Color(15, 15, 15));

name\_flovers3.setString(f3);

name\_flovers3.setCharacterSize(30);

name\_flovers3.setPosition(for\_flovers3.getPosition().x + 5, for\_flovers3.getPosition().y + 5);

RectangleShape for\_zifri3(Vector2f(30, 30));//Колво цветов окошко

for\_zifri3.setFillColor(Color::White);

for\_zifri3.setPosition(for\_flovers3.getPosition().x + 260, for\_flovers3.getPosition().y + 10);

for\_zifri3.setOutlineThickness(1);

for\_zifri3.setOutlineColor(Color(100, 100, 100));

int z3 = 0;// Кол во цветов цифры

Text zifri3;

zifri3.setFont(font);

zifri3.setFillColor(Color::Black);

zifri3.setString(to\_string(z3));

zifri3.setCharacterSize(25);

zifri3.setOutlineThickness(0.1);

zifri3.setOutlineColor(Color::Black);

zifri3.setPosition(for\_zifri3.getPosition().x + 2, for\_zifri3.getPosition().y);

//4 ОКОШКО С ЦВЕТКОМ

RectangleShape for\_flovers4(Vector2f(330, 50));//Для конкретного цветка окошко

for\_flovers4.setFillColor(Color::White);

for\_flovers4.setOutlineThickness(1);

for\_flovers4.setOutlineColor(Color(100, 100, 100));

for\_flovers4.setPosition(x, y = y + 50);

vector<String> name\_menu4{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu4(win, x + 305, y, 30, 15, name\_menu4);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu4.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu4.AlignMenu(2);//Выпавнивание пунктов меню

String f4 = L"4. Нарциссы белые";// Название конкретного цветка

Text name\_flovers4;

name\_flovers4.setFont(font);

name\_flovers4.setFillColor(Color(15, 15, 15));

name\_flovers4.setString(f4);

name\_flovers4.setCharacterSize(30);

name\_flovers4.setPosition(for\_flovers4.getPosition().x + 5, for\_flovers4.getPosition().y + 5);

RectangleShape for\_zifri4(Vector2f(30, 30));//Колво цветов окошко

for\_zifri4.setFillColor(Color::White);

for\_zifri4.setPosition(for\_flovers4.getPosition().x + 260, for\_flovers4.getPosition().y + 10);

for\_zifri4.setOutlineThickness(1);

for\_zifri4.setOutlineColor(Color(100, 100, 100));

int z4 = 0;// Кол во цветов цифры

Text zifri4;

zifri4.setFont(font);

zifri4.setFillColor(Color::Black);

zifri4.setString(to\_string(z4));

zifri4.setCharacterSize(25);

zifri4.setOutlineThickness(0.1);

zifri4.setOutlineColor(Color::Black);

zifri4.setPosition(for\_zifri4.getPosition().x + 2, for\_zifri4.getPosition().y);

//5 ОКОШКО С ЦВЕТКОМ

RectangleShape for\_flovers5(Vector2f(330, 50));//Для конкретного цветка окошко

for\_flovers5.setFillColor(Color::White);

for\_flovers5.setOutlineThickness(1);

for\_flovers5.setOutlineColor(Color(100, 100, 100));

for\_flovers5.setPosition(x, y = y + 50);

vector<String> name\_menu5{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu5(win, x + 305, y, 30, 15, name\_menu5);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu5.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu5.AlignMenu(2);//Выпавнивание пунктов меню

String f5 = L"5. Pозы штамбовые";// Название конкретного цветка

Text name\_flovers5;

name\_flovers5.setFont(font);

name\_flovers5.setFillColor(Color(15, 15, 15));

name\_flovers5.setString(f5);

name\_flovers5.setCharacterSize(30);

name\_flovers5.setPosition(for\_flovers5.getPosition().x + 5, for\_flovers5.getPosition().y + 5);

RectangleShape for\_zifri5(Vector2f(30, 30));//Колво цветов окошко

for\_zifri5.setFillColor(Color::White);

for\_zifri5.setPosition(for\_flovers5.getPosition().x + 260, for\_flovers5.getPosition().y + 10);

for\_zifri5.setOutlineThickness(1);

for\_zifri5.setOutlineColor(Color(100, 100, 100));

int z5 = 0;// Кол во цветов цифры

Text zifri5;

zifri5.setFont(font);

zifri5.setFillColor(Color::Black);

zifri5.setString(to\_string(z5));

zifri5.setCharacterSize(25);

zifri5.setOutlineThickness(0.1);

zifri5.setOutlineColor(Color::Black);

zifri5.setPosition(for\_zifri5.getPosition().x + 2, for\_zifri5.getPosition().y);

// ДЛЯ УПРАВЛЕНИЯ ВАЗАМИ

bool menuExpanded2 = false;

x = 640;

y = 90;

//ПЕРВОЕ ОКОШКО С ВАЗАМИ

RectangleShape for\_vazi1(Vector2f(330, 50));//Для конкретного цветка окошко

for\_vazi1.setFillColor(Color::White);

for\_vazi1.setOutlineThickness(1);

for\_vazi1.setOutlineColor(Color(100, 100, 100));

for\_vazi1.setPosition(x, y);

vector<String> name\_menu\_vasi1{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu\_vasi1(win, x + 305, y, 30, 15, name\_menu\_vasi1);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu\_vasi1.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu\_vasi1.AlignMenu(2);//Выпавнивание пунктов меню

String v1 = L"Малая ваза (3 цв.)";// Название конкретного цветка

Text name\_vasi1;

name\_vasi1.setFont(font);

name\_vasi1.setFillColor(Color(15, 15, 15));

name\_vasi1.setString(v1);

name\_vasi1.setCharacterSize(30);

name\_vasi1.setPosition(for\_vazi1.getPosition().x + 5, for\_vazi1.getPosition().y + 5);

RectangleShape for\_vasi1(Vector2f(30, 30));//Колво цветов окошко

for\_vasi1.setFillColor(Color::White);

for\_vasi1.setPosition(for\_vazi1.getPosition().x + 260, for\_vazi1.getPosition().y + 10);

for\_vasi1.setOutlineThickness(1);

for\_vasi1.setOutlineColor(Color(100, 100, 100));

int zv1 = 0;// Кол во цветов цифры

Text zifri\_vazi1;

zifri\_vazi1.setFont(font);

zifri\_vazi1.setFillColor(Color::Black);

zifri\_vazi1.setString(to\_string(zv1));

zifri\_vazi1.setCharacterSize(25);

zifri\_vazi1.setOutlineThickness(0.1);

zifri\_vazi1.setOutlineColor(Color::Black);

zifri\_vazi1.setPosition(for\_vasi1.getPosition().x + 2, for\_vasi1.getPosition().y);

//2 ОКОШКО С ВАЗАМИ

RectangleShape for\_vazi2(Vector2f(330, 50));//Для конкретного цветка окошко

for\_vazi2.setFillColor(Color::White);

for\_vazi2.setOutlineThickness(1);

for\_vazi2.setOutlineColor(Color(100, 100, 100));

for\_vazi2.setPosition(x, y = y+50);

vector<String> name\_menu\_vasi2{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu\_vasi2(win, x + 305, y, 30, 15, name\_menu\_vasi2);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu\_vasi2.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu\_vasi2.AlignMenu(2);//Выпавнивание пунктов меню

String v2 = L"Средняя ваза (5 цв.)";// Название конкретного цветка

Text name\_vasi2;

name\_vasi2.setFont(font);

name\_vasi2.setFillColor(Color(15, 15, 15));

name\_vasi2.setString(v2);

name\_vasi2.setCharacterSize(30);

name\_vasi2.setPosition(for\_vazi2.getPosition().x + 5, for\_vazi2.getPosition().y + 5);

RectangleShape for\_vasi2(Vector2f(30, 30));//Колво цветов окошко

for\_vasi2.setFillColor(Color::White);

for\_vasi2.setPosition(for\_vazi2.getPosition().x + 260, for\_vazi2.getPosition().y + 10);

for\_vasi2.setOutlineThickness(1);

for\_vasi2.setOutlineColor(Color(100, 100, 100));

int zv2 = 0;// Кол во цветов цифры

Text zifri\_vazi2;

zifri\_vazi2.setFont(font);

zifri\_vazi2.setFillColor(Color::Black);

zifri\_vazi2.setString(to\_string(zv2));

zifri\_vazi2.setCharacterSize(25);

zifri\_vazi2.setOutlineThickness(0.1);

zifri\_vazi2.setOutlineColor(Color::Black);

zifri\_vazi2.setPosition(for\_vasi2.getPosition().x + 2, for\_vasi2.getPosition().y);

//3 ОКОШКО С ВАЗАМИ

RectangleShape for\_vazi3(Vector2f(330, 50));//Для конкретного цветка окошко

for\_vazi3.setFillColor(Color::White);

for\_vazi3.setOutlineThickness(1);

for\_vazi3.setOutlineColor(Color(100, 100, 100));

for\_vazi3.setPosition(x, y= y+50);

vector<String> name\_menu\_vasi3{ L"+", L"-" }; // Название пунктов меню

game::MyMenu mymenu\_vasi3(win, x + 305, y, 30, 15, name\_menu\_vasi3);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu\_vasi3.setColorTextMenu(Color::Black, Color::Black, Color::Black);// Установка цвета отображения меню

mymenu\_vasi3.AlignMenu(2);//Выпавнивание пунктов меню

String v3 = L"Большая ваза (10 цв.)";// Название конкретного цветка

Text name\_vasi3;

name\_vasi3.setFont(font);

name\_vasi3.setFillColor(Color(15, 15, 15));

name\_vasi3.setString(v3);

name\_vasi3.setCharacterSize(30);

name\_vasi3.setPosition(for\_vazi3.getPosition().x + 5, for\_vazi3.getPosition().y + 5);

RectangleShape for\_vasi3(Vector2f(30, 30));//Колво цветов окошко

for\_vasi3.setFillColor(Color::White);

for\_vasi3.setPosition(for\_vazi3.getPosition().x + 260, for\_vazi3.getPosition().y + 10);

for\_vasi3.setOutlineThickness(1);

for\_vasi3.setOutlineColor(Color(100, 100, 100));

int zv3 = 0;// Кол во цветов цифры

Text zifri\_vazi3;

zifri\_vazi3.setFont(font);

zifri\_vazi3.setFillColor(Color::Black);

zifri\_vazi3.setString(to\_string(zv3));

zifri\_vazi3.setCharacterSize(25);

zifri\_vazi3.setOutlineThickness(0.1);

zifri\_vazi3.setOutlineColor(Color::Black);

zifri\_vazi3.setPosition(for\_vasi3.getPosition().x + 2, for\_vasi3.getPosition().y);

//ОКОШКО С РЕЗУЛЬТАТАМИ

bool menuExpanded3 = false;

x = 640;

y = 90;

RectangleShape for\_rez1(Vector2f(470, 125));//Для конкретного цветка окошко

for\_rez1.setFillColor(Color::White);

for\_rez1.setOutlineThickness(1);

for\_rez1.setOutlineColor(Color(100, 100, 100));

for\_rez1.setPosition(x, y);

String r1 = L"Рекомендации по расположению цветов\nзаписаны на ваш компьютер в файл:\nresult.txt";// Название конкретного цветка

Text rez1;

rez1.setFont(font);

rez1.setFillColor(Color(15, 15, 15));

rez1.setString(r1);

rez1.setCharacterSize(30);

rez1.setPosition(for\_rez1.getPosition().x + 5, for\_rez1.getPosition().y + 5);

Vector2f pos;// Координаты объектов

Clock clock;// Таймер

float time;

// Основной цикл

while (win.isOpen())

{

// Обработка событий

Event event;

while (win.pollEvent(event))

{

if (event.type == Event::Closed) //Для закрытия через крестик

win.close();

if (event.type == Event::KeyReleased)

{

// Обработчик событий для основного меню окна (управление цветами, управление вазами, назад)

if (event.key.code == Keyboard::Up) { menuExpanded = 0; menuExpanded2 = 0; menuExpanded3 = 0; mymenu.MoveUp(); } // вверх

if (event.key.code == Keyboard::Down) { menuExpanded = 0; menuExpanded2 = 0; menuExpanded3 = 0; mymenu.MoveDown(); } // вниз

if (event.key.code == Keyboard::Return)

{

// Переходим на выбранный пункт меню

switch (mymenu.getSelectedMenuNumber())

{

case 0:

menuExpanded = true;

break;

case 1:

menuExpanded2 = true;

break;

case 2:

menuExpanded3 = true;

// Вызываем функцию расфасовки элементов

packElements(z1, z2, z3, z4, z5, zv1, zv2, zv3);

break;

case 3:

win.close();

break;

default:

break;

}

}

}

// Обработчик событий для меню Управления цветами

if (event.type == Event::MouseButtonPressed)

{

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

{

if (menuExpanded)

{

// Получаем позицию мыши

Vector2f mousePos = win.mapPixelToCoords(Mouse::getPosition(win));

int w = 50;

//ПЕРВОЕ ОКОШКО С ЦВЕТКОМ

if (for\_flovers1.getGlobalBounds().contains(mousePos))// Проверяем, был ли клик внутри области меню

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 && mousePos.y <= 115)// Проверяем, был ли клик на кнопке "+"

{

z1 = z1 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 && mousePos.y <= 130)// Проверяем, был ли клик на кнопке "-"

{

if (z1 > 0) {// Выполняем действия для кнопки "-"

z1 = z1 - 1;

}

}

zifri1.setString(to\_string(z1));// Обновляем текстовое поле с количеством

}

//ВТОРОЕ ОКОШКО С ЦВЕТКОМ

if (for\_flovers2.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w && mousePos.y <= 115 + w)// Проверяем, был ли клик на кнопке "+"

{

z2 = z2 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w && mousePos.y <= 130 + w)// Проверяем, был ли клик на кнопке "-"

{

if (z2 > 0) { // Выполняем действия для кнопки "-"

z2 = z2 - 1;

}

}

zifri2.setString(to\_string(z2));// Обновляем текстовое поле с количеством

}

//3 ОКОШКО С ЦВЕТКОМ

if (for\_flovers3.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w \* 2 && mousePos.y <= 115 + w \* 2)// Проверяем, был ли клик на кнопке "+"

{

z3 = z3 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w \* 2 && mousePos.y <= 130 + w \* 2)// Проверяем, был ли клик на кнопке "-"

{

if (z3 > 0) { // Выполняем действия для кнопки "-"

z3 = z3 - 1;

}

}

zifri3.setString(to\_string(z3));// Обновляем текстовое поле с количеством

}

//4 ОКОШКО С ЦВЕТКОМ

if (for\_flovers4.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w \* 3 && mousePos.y <= 115 + w \* 3)// Проверяем, был ли клик на кнопке "+"

{

z4 = z4 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w \* 3 && mousePos.y <= 130 + w \* 3)// Проверяем, был ли клик на кнопке "-"

{

if (z4 > 0) { // Выполняем действия для кнопки "-"

z4 = z4 - 1;

}

}

zifri4.setString(to\_string(z4));// Обновляем текстовое поле с количеством

}

//5 ОКОШКО С ЦВЕТКОМ

if (for\_flovers5.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w \* 4 && mousePos.y <= 115 + w \* 4)// Проверяем, был ли клик на кнопке "+"

{

z5 = z5 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w \* 4 && mousePos.y <= 130 + w \* 4)// Проверяем, был ли клик на кнопке "-"

{

if (z5 > 0) { // Выполняем действия для кнопки "-"

z5 = z5 - 1;

}

}

zifri5.setString(to\_string(z5));// Обновляем текстовое поле с количеством

}

}

//Обработчик событий для меню Управления вазами

if (menuExpanded2)

{

// Получаем позицию мыши

Vector2f mousePos = win.mapPixelToCoords(Mouse::getPosition(win));

int w = 50;

//ПЕРВОЕ ОКОШКО С ВАЗОЙ

if (for\_vazi1.getGlobalBounds().contains(mousePos))// Проверяем, был ли клик внутри области меню

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 && mousePos.y <= 115)// Проверяем, был ли клик на кнопке "+"

{

zv1 = zv1 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 && mousePos.y <= 130)// Проверяем, был ли клик на кнопке "-"

{

if (zv1 > 0) {// Выполняем действия для кнопки "-"

zv1 = zv1 - 1;

}

}

zifri\_vazi1.setString(to\_string(zv1));// Обновляем текстовое поле с количеством

}

//ВТОРОЕ ОКОШКО С ВАЗОЙ

if (for\_vazi2.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w && mousePos.y <= 115 + w)// Проверяем, был ли клик на кнопке "+"

{

zv2 = zv2 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w && mousePos.y <= 130 + w)// Проверяем, был ли клик на кнопке "-"

{

if (zv2 > 0) { // Выполняем действия для кнопки "-"

zv2 = zv2 - 1;

}

}

zifri\_vazi2.setString(to\_string(zv2));// Обновляем текстовое поле с количеством

}

//3 ОКОШКО С ВАЗОЙ

if (for\_vazi3.getGlobalBounds().contains(mousePos))

{

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 100 + w \* 2 && mousePos.y <= 115 + w \* 2)// Проверяем, был ли клик на кнопке "+"

{

zv3 = zv3 + 1;// Выполняем действия для кнопки "+"

}

if (mousePos.x >= 940 && mousePos.x <= 965 && mousePos.y >= 115 + w \* 2 && mousePos.y <= 130 + w \* 2)// Проверяем, был ли клик на кнопке "-"

{

if (zv3 > 0) { // Выполняем действия для кнопки "-"

zv3 = zv3 - 1;

}

}

zifri\_vazi3.setString(to\_string(zv3));// Обновляем текстовое поле с количеством

}

}

}

}

}

// Обновление таймера

time = clock.getElapsedTime().asMicroseconds();

time = time / 15000;

clock.restart();

// Движение фона

fon.move(-0.2 \* time, 0);

pos = fon.getPosition();

if (pos.x < -1280) fon.setPosition(1280, pos.y);

fon2.move(-0.2 \* time, 0);

pos = fon2.getPosition();

if (pos.x < -1280) fon2.setPosition(1280, pos.y);

win.clear();// Очистка экрана

win.draw(fon); // Отрисовка элементов

win.draw(fon2);

win.draw(panel);

win.draw(panel2);

win.draw(button);

mymenu.draw();

win.draw(Titul);

if (menuExpanded) {

win.draw(for\_flovers1); win.draw(name\_flovers1); mymenu1.draw(); win.draw(for\_zifri1); win.draw(zifri1);

win.draw(for\_flovers2); win.draw(name\_flovers2); mymenu2.draw(); win.draw(for\_zifri2); win.draw(zifri2);

win.draw(for\_flovers3); win.draw(name\_flovers3); mymenu3.draw(); win.draw(for\_zifri3); win.draw(zifri3);

win.draw(for\_flovers4); win.draw(name\_flovers4); mymenu4.draw(); win.draw(for\_zifri4); win.draw(zifri4);

win.draw(for\_flovers5); win.draw(name\_flovers5); mymenu5.draw(); win.draw(for\_zifri5); win.draw(zifri5);

}

if (menuExpanded2) {

win.draw(for\_vazi1); win.draw(name\_vasi1); mymenu\_vasi1.draw(); win.draw(for\_vasi1); win.draw(zifri\_vazi1);

win.draw(for\_vazi2); win.draw(name\_vasi2); mymenu\_vasi2.draw(); win.draw(for\_vasi2); win.draw(zifri\_vazi2);

win.draw(for\_vazi3); win.draw(name\_vasi3); mymenu\_vasi3.draw(); win.draw(for\_vasi3); win.draw(zifri\_vazi3);

}

if (menuExpanded3) {

win.draw(for\_rez1); win.draw(rez1);

}

win.display();// Обновление экрана

}

}

// Описание

void About() {

RenderWindow win(VideoMode(1280, 720), "ARM");// Создание окна

win.setFramerateLimit(60);

// Загрузка иконки

Image icon;

icon.loadFromFile("Image/icon1.png");

win.setIcon(1200, 1084, icon.getPixelsPtr());

win.setMouseCursorVisible(false);// Отключение видимости курсора

// Серый фон для текста

Image image;

image.loadFromFile("Image/panel4.jpg");

image.createMaskFromColor(image.getPixel(0, 0));

Texture texture;

texture.loadFromImage(image);

Sprite panel;

panel.setTexture(texture);

panel.setPosition(Vector2f(0, 0));

panel.setScale(1280 / panel.getLocalBounds().width, 720 / panel.getLocalBounds().height);

// Создание движущегося фона

Texture textureFon;

textureFon.loadFromFile("Image/fon2.jpg");

RectangleShape fon(Vector2f(1280, 720));

fon.setTexture(&textureFon);

RectangleShape fon2(Vector2f(1280, 720));

fon2.setTexture(&textureFon);

fon2.setPosition(Vector2f(1280, 0));

// Загрузка шрифта

Font font;

font.loadFromFile("font/myFont2.ttf");

// Создание текста с названием экрана

Text Titul;

Titul.setFont(font);

TextFormat Ftext;

Ftext.size\_font = 35;

Ftext.menu\_text\_color = Color{ 10, 10, 10 };

Ftext.bord = 0;

InitText(Titul, 400, 160, L"ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНЬЯ:", Ftext);

// Загрузка шрифта

Font font2;

font2.loadFromFile("font/myFont2.ttf");

// Создание текста с названием экрана

Text Titul2;

Titul2.setFont(font2);

TextFormat Ftext2;

Ftext2.size\_font = 4;

Ftext2.menu\_text\_color = Color( 20, 20, 20 );

Ftext2.bord = 0;

InitText(Titul2, 200, 220, L"\tДанное приложене представляет из себя автоматизированное \nрабочее место флориста.\n\n\tОно способно упостить и ускорить работу флориста во время \nпоступления крупных партий.\n\n\tС его помощью можно отслеживать дату поставки товара и \nсоответственно контролировать качество продукции.", Ftext);

//Переход назад (реализовано через меню)

vector<String> name\_menu2{ L"Нажмите Enter" };// Название пунктов меню

game::MyMenu mymenu(win, 930, 560, 20, 0, name\_menu2);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu.setColorTextMenu(Color(20, 20, 20), Color(20, 20, 20), Color(20,20,20)); // Установка цвета отображения меню

mymenu.AlignMenu(3);

Vector2f pos;// Координаты объектов

Clock clock;// Таймер

float time;

// Основной цикл

while (win.isOpen())

{

// Обработка событий

Event event;

while (win.pollEvent(event))

{

if (event.type == sf::Event::Closed) //Закрытие через крестик

win.close();

if (event.type == Event::KeyReleased)

{

// События выбра пунктов меню

if (event.key.code == Keyboard::Up) { mymenu.MoveUp(); } // вверх

if (event.key.code == Keyboard::Down) { mymenu.MoveDown(); } // вниз

if (event.key.code == Keyboard::Return) // ввод

{

// Переходим на выбранный пункт меню

switch (mymenu.getSelectedMenuNumber())

{

case 0:win.close(); break;

default:break;

}

}

}

}

// Обновление таймера

time = clock.getElapsedTime().asMicroseconds();

time = time / 8000;

clock.restart();

// Движение фона

fon.move(-0.2 \* time, 0);

pos = fon.getPosition();

if (pos.x < -1280) fon.setPosition(1280, pos.y);

fon2.move(-0.2 \* time, 0);

pos = fon2.getPosition();

if (pos.x < -1280) fon2.setPosition(1280, pos.y);

win.clear();// Очистка экрана

win.draw(fon);// Отрисовка элементов

win.draw(fon2);

win.draw(panel);

mymenu.draw();

win.draw(Titul);

win.draw(Titul2);

win.display();// Обновление экрана

}

}

### *//Mymenu.cpp*

#include "MyMenu.h"

#include <vector>

void game::MyMenu::setInitText(sf::Text& text, const sf::String& str, float xpos, float ypos) const

{

text.setFont(font);

text.setFillColor(menu\_text\_color);

text.setString(str);

text.setCharacterSize(size\_font);

text.setPosition(xpos, ypos);

text.setOutlineThickness(0.1);

text.setOutlineColor(border\_color);

}

void game::MyMenu::AlignMenu(int posx)

{

float nullx = 0;

for (int i = 0; i < max\_menu; i++) {

switch (posx)

{

case 0:

nullx = 0;

break;

case 1:

nullx = mainMenu[i].getLocalBounds().width;

break;

case 2:

nullx = mainMenu[i].getLocalBounds().width / 2;

break;

default:break;

}

mainMenu[i].setPosition(mainMenu[i].getPosition().x - nullx, mainMenu[i].getPosition().y);

}

}

game::MyMenu::MyMenu(sf::RenderWindow& window, float menux, float menuy, int sizeFont, int step, std::vector<sf::String>& name)

:menu\_X(menux), menu\_Y(menuy), menu\_Step(step), max\_menu(static\_cast<int>(name.size())), size\_font(sizeFont), mainMenu(name.size()), mywindow(window)

{

if (!font.loadFromFile("font/myFont2.ttf")) exit(32);

for (int i = 0, ypos = static\_cast<int>(menu\_Y); i < max\_menu; i++, ypos += menu\_Step)

setInitText(mainMenu[i], name[i], menu\_X, static\_cast<float>(ypos));

mainMenuSelected = 0;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

void game::MyMenu::MoveUp()

{

mainMenuSelected--;

if (mainMenuSelected >= 0) {

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

mainMenu[mainMenuSelected + 1].setFillColor(menu\_text\_color);

}

else

{

mainMenu[0].setFillColor(menu\_text\_color);

mainMenuSelected = max\_menu - 1;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

}

void game::MyMenu::MoveDown()

{

mainMenuSelected++;

if (mainMenuSelected < max\_menu) {

mainMenu[mainMenuSelected - 1].setFillColor(menu\_text\_color);

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

else

{

mainMenu[max\_menu - 1].setFillColor(menu\_text\_color);

mainMenuSelected = 0;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

}

void game::MyMenu::draw()

{

for (int i = 0; i < max\_menu; i++) mywindow.draw(mainMenu[i]);

}

void game::MyMenu::setColorTextMenu(sf::Color menColor, sf::Color ChoColor, sf::Color BordColor)

{

menu\_text\_color = menColor;

chose\_text\_color = ChoColor;

border\_color = BordColor;

for (int i = 0; i < max\_menu; i++) {

mainMenu[i].setFillColor(menu\_text\_color);

mainMenu[i].setOutlineColor(border\_color);

}

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

### *//Mymenu.h*

#pragma once

#include <SFML/Graphics.hpp>

namespace game {

class MyMenu

{

float menu\_X; // Координаты меню по X

float menu\_Y; // Координаты меню по Y

int menu\_Step; // Расстояние между пунктами меню

int max\_menu; // Максимальное количество пунктов меню

int size\_font; // Размер шрифта

int mainMenuSelected; // Номер текущего пункта меню

sf::Font font; // Шрифт меню

std::vector<sf::Text> mainMenu; // Динамический массив названий меню

sf::Color menu\_text\_color = sf::Color::White; // Цвет меню

sf::Color chose\_text\_color = sf::Color::Yellow; // Цвет выбора меню

sf::Color border\_color = sf::Color::Black; // Цвет обводки текста

// Настройка текста меню

void setInitText(sf::Text& text, const sf::String& str, float xpos, float ypos) const;

sf::RenderWindow& mywindow; // Ссылка на графическое окно

public:

MyMenu(sf::RenderWindow& window, float menux, float menuy, int sizeFont, int step, std::vector<sf::String>& name);

void draw(); // Рисуем меню

void MoveUp(); // Перемещение выбора меню вверх

void MoveDown(); // Перемещение выбора меню вниз

void setColorTextMenu(sf::Color menColor, sf::Color ChoColor, sf::Color BordColor); // Цвет элементов игрового меню

void AlignMenu(int posx); // Выравнивание положения меню (по левому по правому по центру)

int getSelectedMenuNumber() const // Возвращает номер выбраного элемента

{

return mainMenuSelected;

}

};

}

# **Решение задачи Коммивоязжёра.**

## Постановка задачи:

Создать универсальное приложения на языке С++ для решения задачи коммивоязжера методом ветвей и границ для любого набора исходных данных. Приложение должно включать в себя несколько функций такких как: перемещение вершин, добавление вершин, удаление вершин, добавление ребра, удаление ребра, создание нового графа, вывод матрицы смежности, решение задачи коммивоязжера. С помощью графиеских библиотек SFML и OpenGL реализовать визуализацию задачи.

## Анализ задачи:

Управление графом:

* Перемещение вершин.
* Добавление вершин.
* Удаление вершин.
* Добавление ребер.
* Удаление ребер.
* Создание нового графа.

Ввод данных с консоли:

* Для функций добавления и удаления ребер: количество ребер, исходная вершина, конечная вершина, вес ребра.
* Для функции создания нового графа: возможность выбора вида графа (смешанный, не оринтированных, ориентированный), ввод количества вершин и ребер, ввод исх. и кон. вершин для ребра, предусмотреть возможность ввода разних видов ребер.

Вывод данных на консоль:

* Вывод матрицы смежности.
* Вывод решения задачи Коммивоязжера.

Решение задачи:

* Решение задачи коммивояжера методом ветвей и границ.

Визуализация:

* Визуализация графа и всех функций преобразующих его с помощью SFML или OpenGL.

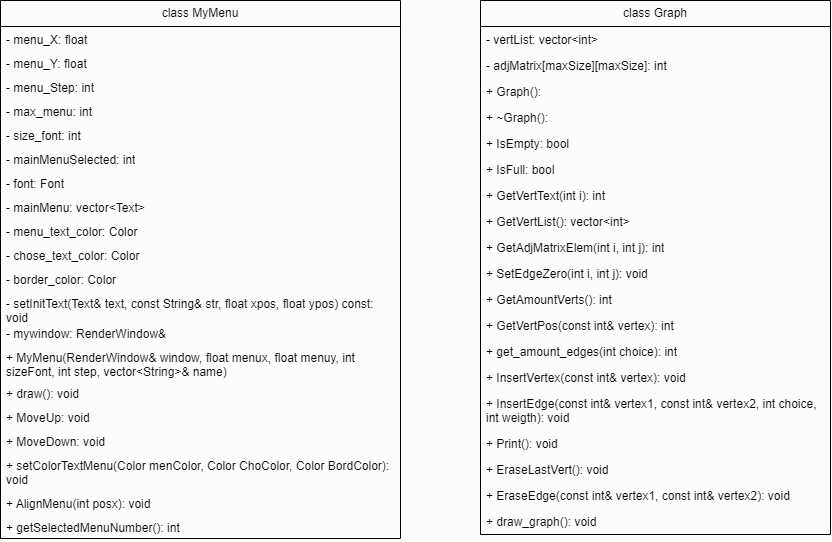
## Преимущества:

* Универсальность: возможность решения задачи для любого графа.
* Визуализация: повышение наглядности решения задачи.

## Технические требования:

* Язык программирования: C++.
* Графические библиотеки: SFML и OpenGL.
* Среда разработки: Visual Studio 2022.
* Операционная система: Windows.

## UML диаграмма:

На рисунке 2 представлена UML диаграмма алгоритма программы.

***Рисунок 2 – UML диаграмма задачи Коммивоязжера.***

## Код на С++ с комментариями:

### *//main.cpp*

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include "Graph.h"

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <iostream>

#include"MyMenu.h"

#include<Windows.h>

#include <vector>

#include <fstream>

#include <map>

#include <ctime>

#include <iomanip>

using namespace std;

using namespace sf;

int main(int argc, char\*\* argv)

{

setlocale(LC\_ALL, "ru");

RenderWindow win(VideoMode(1280, 720), L"Комивоязжер");// Создание окна

win.setFramerateLimit(60);

//Музыка на фоне

Music music;

music.openFromFile("music.ogg");

music.play();

music.setLoop(true);

music.setVolume(2.f);

// Загрузка иконки

Image icon;

icon.loadFromFile("Image/icon1.jpg");

win.setIcon(2048, 2048, icon.getPixelsPtr());

win.setMouseCursorVisible(false);// Отключение видимости курсора

// Создание движущегося фона

Texture textureFon;

textureFon.loadFromFile("Image/fon.jpg");

RectangleShape fon(Vector2f(1280, 720));

fon.setTexture(&textureFon);

RectangleShape fon2(Vector2f(1280, 720));

fon2.setTexture(&textureFon);

fon2.setPosition(Vector2f(1280, 0));

CircleShape octagon(80, 8);

octagon.setFillColor(Color(500, 500, 500));

octagon.setRadius(250);

octagon.setPosition(370, 90);

// Загрузка шрифта

Font font;

if (!font.loadFromFile("font/myFont.otf")) return 5;

// Создание текста с названием экрана

String f1 = L"КОМИВОЯЗЖЕР";// Название конкретного цветка

Text name\_flovers1;

name\_flovers1.setFont(font);

name\_flovers1.setFillColor(Color(15, 15, 15));

name\_flovers1.setString(f1);

name\_flovers1.setCharacterSize(40);

name\_flovers1.setPosition(460, 260);

//Основное меню

vector<String> name\_menu{ L"НАЧАТЬ РАБОТУ", L"ВЫЙТИ" };// Название пунктов меню

game::MyMenu mymenu(win, 625, 320, 35, 35, name\_menu);// Объект меню (..., x, y, размер текста, размер отступа между пунктами, ...)

mymenu.setColorTextMenu(Color(200, 200, 200), Color(15, 15, 15), Color::Black);// Установка цвета отображения меню

mymenu.AlignMenu(2); //Выравнивание пунктов меню

Vector2f pos;// Координаты объектов

Clock clock; // Таймер

float time;

// Основной цикл

while (win.isOpen())

{

// Обработка событий

Event event;

while (win.pollEvent(event))

{

if (event.type == sf::Event::Closed) //Для закрытия через крестик

win.close();

if (event.type == Event::KeyReleased)

{

// События выбра пунктов меню

if (event.key.code == Keyboard::Up) { mymenu.MoveUp(); } // вверх

if (event.key.code == Keyboard::Down) { mymenu.MoveDown(); } // вниз

if (event.key.code == Keyboard::Return) // ввод

{

// Переходим на выбранный пункт меню

switch (mymenu.getSelectedMenuNumber())

{

case 0:

glutInit(&argc, argv);

make\_graph\_baza();

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowSize(wight, hight);

glutCreateWindow("Graph");

//glutFullScreen(); //во весь экрвн

glutDisplayFunc(draw);

glutReshapeFunc(mashtab);

glutMouseFunc(tab\_mouse);

glutPassiveMotionFunc(dvisenir\_mous);

glutMainLoop();

break;

case 1:win.close(); break;

default:break;

}

}

}

}

// Обновление таймера

time = clock.getElapsedTime().asMicroseconds();

time = time / 15000;

clock.restart();

// Движение фона

fon.move(-0.2 \* time, 0);

pos = fon.getPosition();

if (pos.x < -1280) fon.setPosition(1280, pos.y);

fon2.move(-0.2 \* time, 0);

pos = fon2.getPosition();

if (pos.x < -1280) fon2.setPosition(1280, pos.y);

win.clear();// Очистка экрана

win.draw(fon);// Отрисовка элементов игры

win.draw(fon2);

win.draw(octagon);

mymenu.draw();

win.draw(name\_flovers1);

win.display();// Обновление экрана

}

return 0;

}

### *//Mymenu.cpp*

#include "MyMenu.h"

#include <vector>

void game::MyMenu::setInitText(sf::Text& text, const sf::String& str, float xpos, float ypos) const

{

text.setFont(font);

text.setFillColor(menu\_text\_color);

text.setString(str);

text.setCharacterSize(size\_font);

text.setPosition(xpos, ypos);

text.setOutlineThickness(0.1);

text.setOutlineColor(border\_color);

}

void game::MyMenu::AlignMenu(int posx)

{

float nullx = 0;

for (int i = 0; i < max\_menu; i++) {

switch (posx)

{

case 0:

nullx = 0;

break;

case 1:

nullx = mainMenu[i].getLocalBounds().width;

break;

case 2:

nullx = mainMenu[i].getLocalBounds().width / 2;

break;

default:break;

}

mainMenu[i].setPosition(mainMenu[i].getPosition().x - nullx, mainMenu[i].getPosition().y);

}

}

game::MyMenu::MyMenu(sf::RenderWindow& window, float menux, float menuy, int sizeFont, int step, std::vector<sf::String>& name)

:menu\_X(menux), menu\_Y(menuy), menu\_Step(step), max\_menu(static\_cast<int>(name.size())), size\_font(sizeFont), mainMenu(name.size()), mywindow(window)

{

if (!font.loadFromFile("font/myFont2.ttf")) exit(32);

for (int i = 0, ypos = static\_cast<int>(menu\_Y); i < max\_menu; i++, ypos += menu\_Step)

setInitText(mainMenu[i], name[i], menu\_X, static\_cast<float>(ypos));

mainMenuSelected = 0;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

void game::MyMenu::MoveUp()

{

mainMenuSelected--;

if (mainMenuSelected >= 0) {

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

mainMenu[mainMenuSelected + 1].setFillColor(menu\_text\_color);

}

else

{

mainMenu[0].setFillColor(menu\_text\_color);

mainMenuSelected = max\_menu - 1;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

}

void game::MyMenu::MoveDown()

{

mainMenuSelected++;

if (mainMenuSelected < max\_menu) {

mainMenu[mainMenuSelected - 1].setFillColor(menu\_text\_color);

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

else

{

mainMenu[max\_menu - 1].setFillColor(menu\_text\_color);

mainMenuSelected = 0;

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

}

void game::MyMenu::draw()

{

for (int i = 0; i < max\_menu; i++) mywindow.draw(mainMenu[i]);

}

void game::MyMenu::setColorTextMenu(sf::Color menColor, sf::Color ChoColor, sf::Color BordColor)

{

menu\_text\_color = menColor;

chose\_text\_color = ChoColor;

border\_color = BordColor;

for (int i = 0; i < max\_menu; i++) {

mainMenu[i].setFillColor(menu\_text\_color);

mainMenu[i].setOutlineColor(border\_color);

}

mainMenu[mainMenuSelected].setFillColor(chose\_text\_color);

}

### *//Mymenu.h*

#pragma once

#include <SFML/Graphics.hpp>

namespace game {

class MyMenu

{

float menu\_X; // Координаты меню по X

float menu\_Y; // Координаты меню по Y

int menu\_Step; // Расстояние между пунктами меню

int max\_menu; // Максимальное количество пунктов меню

int size\_font; // Размер шрифта

int mainMenuSelected; // Номер текущего пункта меню

sf::Font font; // Шрифт меню

std::vector<sf::Text> mainMenu; // Динамический массив названий меню

sf::Color menu\_text\_color = sf::Color::White; // Цвет меню

sf::Color chose\_text\_color = sf::Color::Yellow; // Цвет выбора меню

sf::Color border\_color = sf::Color::Black; // Цвет обводки текста

// Настройка текста меню

void setInitText(sf::Text& text, const sf::String& str, float xpos, float ypos) const;

sf::RenderWindow& mywindow; // Ссылка на графическое окно

public:

MyMenu(sf::RenderWindow& window, float menux, float menuy, int sizeFont, int step, std::vector<sf::String>& name);

void draw(); // Рисуем меню

void MoveUp(); // Перемещение выбора меню вверх

void MoveDown(); // Перемещение выбора меню вниз

void setColorTextMenu(sf::Color menColor, sf::Color ChoColor, sf::Color BordColor); // Цвет элементов игрового меню

void AlignMenu(int posx); // Выравнивание положения меню (по левому по правому по центру)

int getSelectedMenuNumber() const // Возвращает номер выбраного элемента

{

return mainMenuSelected;

}

};

}

### *//Graph.cpp*

#include "Graph.h"

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include "Graph.h"

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <iostream>

using namespace std;

using namespace sf;

int wight = 1280; // Определяем стартовые размеры окна

int hight = 720;

bool\* Mouse\_Vert = new bool[maxSize]; //массив меток (курсор в зоне вершины или нет)

bool stand\_vid = false; //отвечает за перестроение графа в стандартный вид (все вершины расположены по кругу)

bool Vert\_Move = false; //отвечает за перемещение вершин

int R; //радиус вершин графа

int Button\_Flag; //если курсор в зоне кнопки, принимает значение номера кнопки (подсветка кнопок при наведении)

int Moving\_Vertex; //номер перемещаемой вершины

int CursorX; //позиция курсора по Х

int CursorY; //позиция курсора по У

vertCoord vertC[maxSize]; //массив структур (координаты вершин), +2 чтобы не вылетало искл., при заполнении до maxSize

Graph graph;

vector<pair<int, int>> Way;

vector<int> New\_Way;

//Конструктор, заполнение матрицы смежности 0 и меток Mouse\_Vert

Graph::Graph()

{

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

{

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

{

adjMatrix[i][j] = 0;

}

Mouse\_Vert[i] = false;

}

}

//деструктор

Graph::~Graph()

{ }

//получение номера вершины в векторе вершин

int Graph::GetVertPos(const int& vertex)

{

for (size\_t i = 0; i < vertList.size(); i++)

{

if (vertList[i] == vertex)

return i;

}

return -1;

}

//проверка на пустоту

bool Graph::IsEmpty()

{

if (vertList.size() != 0)

return false;

else

return true;

}

//проверка на полноту

bool Graph::IsFull()

{

return (vertList.size() == maxSize);

}

//добавление новой вершины

void Graph::InsertVertex(const int& vertex)

{

if (!IsFull())

vertList.push\_back(vertex);

else

{

cout << "Граф уже заполнен. Невозможно добавить новую вершину. " << endl;

return;

}

}

//добавление нового ребра

void Graph::InsertEdge(const int& vertex1, const int& vertex2, int choice, int weight)

{

if (weight < 1)

{

cout << "\nДанная величина веса некорректна\n";

return;

}

if (GetVertPos(vertex1) != (-1) && GetVertPos(vertex2) != (-1))//если вершины есть в графе

{

int vertPos1 = GetVertPos(vertex1); //находим позиции вершин

int vertPos2 = GetVertPos(vertex2);

if (adjMatrix[vertPos1][vertPos2] != 0 && adjMatrix[vertPos2][vertPos1] != 0)//если между ними уже есть ребро

{

cout << "Ребро между вершинами уже есть" << endl;

return;

}

else//иначе добавляем ребро

{

adjMatrix[vertPos1][vertPos2] = weight;

if (choice == 2)

{

adjMatrix[vertPos2][vertPos1] = weight;

}

}

}

else

{

cout << "Обеих вершин (или одной из них) нет в графе " << endl;

return;

}

}

//вывод матрицы смежности на экран

void Graph::Print()

{

if (!IsEmpty())

{

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

cout << "-\t";

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

{

cout << vertList[i] << "\t";

}

cout << endl;

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

{

cout << vertList[i] << "\t";

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

{

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

}

cout << endl;

}

}

else

cout << "\nГраф пуст\n" << endl;

}

//удаление последней вершины

void Graph::EraseLastVert(){

if (IsEmpty())

{

cout << "\nГраф пуст\n";

return;

}

int n = vertList.size();

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

{

adjMatrix[n - 1][j] = 0;

adjMatrix[j][n - 1] = 0;

}

vertList.pop\_back();

}

//удаление ребра

void Graph::EraseEdge(const int& vertex1, const int& vertex2)

{

if (GetVertPos(vertex1) != (-1) && GetVertPos(vertex2) != (-1))//если обе вершины есть в графе

{

int vertPos1 = GetVertPos(vertex1);//находим позиции вершин

int vertPos2 = GetVertPos(vertex2);

if (adjMatrix[vertPos1][vertPos2] == 0) //если между вершинами ребра нет

{

cout << "Ребра между данными вершинами нет\n" << endl;

return;

}

else//иначе удаляем ребро (зануляем соответствуюшую ячейку в матрице смежности)

{

adjMatrix[vertPos1][vertPos2] = 0;

adjMatrix[vertPos2][vertPos1] = 0;

}

}

else //если хотя бы одной вершины в графе нет

{

cout << "Обеих вершин (или одной из них) нет в графе " << endl;

return;

}

}

//получение количества ребер в графе

int Graph::get\_amount\_edges(int choice)

{

int numb = 0;

if (!IsEmpty())

{

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

{

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

{

if (adjMatrix[i][j] != 0)

numb++;

}

}

}

if (choice == 1)

return numb;

else

return numb / 2;

}

//рисование графа

void Graph::draw\_graph()

{

int n = graph.GetAmountVerts();

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

{

if (!stand\_vid)

drav\_vce\_krugi(i, n);

}

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

{

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

{

int a = adjMatrix[i][j];

if (a != 0)

draw\_rebro(a, vertC[i].x, vertC[i].y, vertC[j].x, vertC[j].y);

if (a == adjMatrix[j][i] && a != 0)

draw\_rebro(a, vertC[j].x, vertC[j].y, vertC[i].x, vertC[i].y);

}

}

draw\_vertex(n);

glutPostRedisplay();

}

//создается матрица, заменяя 0 на -1

int\*\* matr\_0\_v\_1()

{

int n = graph.GetAmountVerts();

int\*\* matrix = new int\* [n];

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

matrix[i] = new int[n];

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

{

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

{

int elem = graph.GetAdjMatrixElem(i, j);

if (elem == 0 or i == j)

matrix[i][j] = -1;

else

matrix[i][j] = elem;

}

}

cout << "Начальная матрица: \n";

print\_matrix(matrix);

return matrix;

}

//поиск минимального элемента в строке матрицы

int\* min\_el(int\* line, int n)

{

int min = 1000000;

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

if (line[j] >= 0 && line[j] < min)

min = line[j];

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

if (line[j] >= 0)

line[j] -= min;

return line;

}

//вывод текущей матрицы

void print\_matrix(int\*\* matrix)

{

int n = graph.GetAmountVerts();

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

{

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

cout << setw(4) << matrix[i][j];

cout << endl;

}

}

//редуцированние матрицы

int\*\* reduct\_matrix(int\*\* oldmatrix)

{

int\*\* matrix = oldmatrix;

int n = graph.GetAmountVerts();

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

matrix[i] = min\_el(matrix[i], n);

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

{

int min = 1000000;

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

{

if (matrix[j][i] >= 0 && matrix[j][i] < min)

min = matrix[j][i];

}

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

{

if (matrix[j][i] >= 0)

matrix[j][i] -= min;

}

}

cout << "\nРедуцированная матрица: \n";

print\_matrix(matrix);

return matrix;

}

//Удаление нуля с наибольшей оценкой

int\*\* rabita\_viz\_0(int\*\* oldmatrix)

{

int n = graph.GetAmountVerts();

int\*\* matrix = reduct\_matrix(oldmatrix);

int max = -1;

int line, column;

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

{

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

{

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

{

int minLine = 1000000;

int minColumn = 1000000;

for (int k = 0; k < n; k++)

{

if (matrix[i][k] != -1 && k != j && matrix[i][k] < minLine)

minLine = matrix[i][k];

}

for (int k = 0; k < n; k++)

{

if (matrix[k][j] != -1 && k != i && matrix[k][j] < minColumn)

minColumn = matrix[k][j];

}

if (max < minColumn + minLine)

{

max = minColumn + minLine;

line = i;

column = j;

}

}

}

}

pair<int, int> p;

p.first = line + 1;

p.second = column + 1;

Way.push\_back(p);

matrix[line][column] = -1;

matrix[column][line] = -1;

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

{

matrix[line][i] = -1;

matrix[i][column] = -1;

}

cout << endl;

cout << "Матрица после удаления 0 с наибольшей оценкой: \n";

print\_matrix(matrix);

cout << "\nПромежуточные отрезки путей: ";

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

cout << Way[i].first << " -> " << Way[i].second << " ";

cout << endl;

return matrix;

}

//Вывод результатов на консоль

void print\_result()

{

int second = Way[0].second;

int i = 2;

New\_Way.push\_back(Way[0].first);

New\_Way.push\_back(Way[0].second);

while (i != graph.GetAmountVerts() + 1)

for (int j = 1; j < graph.GetAmountVerts(); j++)

if (Way[j].first == second)

{

second = Way[j].second;

New\_Way.push\_back(second);

i++;

}

cout << "Ответ: ";

for (int i = 0; i < New\_Way.size(); i++)

{

cout << New\_Way[i];

if (i != New\_Way.size() - 1)

cout << " -> ";

}

int sum = 0;

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

{

int line = Way[i].first - 1;

int column = Way[i].second - 1;

sum += graph.GetAdjMatrixElem(line, column);

}

cout << "\nS = " << sum << endl;;

}

// Рисование кнопок

void button1()

{

if (Button\_Flag == 1)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 20);

glVertex2i(50, hight - hight / 7);

glVertex2i(wight / 7, hight - hight / 7);

glVertex2i(wight / 7, hight - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 20);

glVertex2i(50, hight - hight / 7);

glVertex2i(wight / 7, hight - hight / 7);

glVertex2i(wight / 7, hight - 20);

glEnd();

string n1 = "MAKE NEW GRAPH";

glRasterPos2i(wight / 20, 0.92 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

glRasterPos2i(wight / 20, (0.92 \* hight) - 15);

}

void button2()

{

if (Button\_Flag == 2)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - hight / 7 - 20);

glVertex2i(50, hight - 2 \* (hight / 7));

glVertex2i(wight / 7, hight - 2 \* (hight / 7));

glVertex2i(wight / 7, hight - hight / 7 - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - hight / 7 - 20);

glVertex2i(50, hight - 2 \* (hight / 7));

glVertex2i(wight / 7, hight - 2 \* (hight / 7));

glVertex2i(wight / 7, hight - hight / 7 - 20);

glEnd();

string n1 = "\"SALESMAN\" TASK";

glRasterPos2i(wight / 20, 0.77 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

}

void button3()

{

if (Button\_Flag == 3)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 2 \* (hight / 7) - 20);

glVertex2i(50, hight - 3 \* (hight / 7));

glVertex2i(wight / 7, hight - 3 \* (hight / 7));

glVertex2i(wight / 7, hight - 2 \* (hight / 7) - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 2 \* (hight / 7) - 20);

glVertex2i(50, hight - 3 \* (hight / 7));

glVertex2i(wight / 7, hight - 3 \* (hight / 7));

glVertex2i(wight / 7, hight - 2 \* (hight / 7) - 20);

glEnd();

string n1 = "PRINT MATRIX";

string n2 = "IN CONSOLE";

glRasterPos2i(wight / 20, 0.63 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

glRasterPos2i(wight / 20, (0.63 \* hight) - 15);

for (int i = 0; i < n2.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n2[i]);

}

void button4()

{

if (Button\_Flag == 4)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 3 \* (hight / 7) - 20);

glVertex2i(50, hight - 4 \* (hight / 7));

glVertex2i(wight / 7, hight - 4 \* (hight / 7));

glVertex2i(wight / 7, hight - 3 \* (hight / 7) - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 3 \* (hight / 7) - 20);

glVertex2i(50, hight - 4 \* (hight / 7));

glVertex2i(wight / 7, hight - 4 \* (hight / 7));

glVertex2i(wight / 7, hight - 3 \* (hight / 7) - 20);

glEnd();

string n1 = "ADD VERTEX";

glRasterPos2i(wight / 20, 0.48 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

}

void button5()

{

if (Button\_Flag == 5)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 4 \* (hight / 7) - 20);

glVertex2i(50, hight - 5 \* (hight / 7));

glVertex2i(wight / 7, hight - 5 \* (hight / 7));

glVertex2i(wight / 7, hight - 4 \* (hight / 7) - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 4 \* (hight / 7) - 20);

glVertex2i(50, hight - 5 \* (hight / 7));

glVertex2i(wight / 7, hight - 5 \* (hight / 7));

glVertex2i(wight / 7, hight - 4 \* (hight / 7) - 20);

glEnd();

string n1 = "DELETE VERTEX";

glRasterPos2i(wight / 20, 0.34 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

}

void button6()

{

if (Button\_Flag == 6)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 5 \* (hight / 7) - 20);

glVertex2i(50, hight - 6 \* (hight / 7));

glVertex2i(wight / 7, hight - 6 \* (hight / 7));

glVertex2i(wight / 7, hight - 5 \* (hight / 7) - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 5 \* (hight / 7) - 20);

glVertex2i(50, hight - 6 \* (hight / 7));

glVertex2i(wight / 7, hight - 6 \* (hight / 7));

glVertex2i(wight / 7, hight - 5 \* (hight / 7) - 20);

glEnd();

string n1 = "ADD EDGE";

glRasterPos2i(wight / 20, 0.2 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

}

void button7()

{

if (Button\_Flag == 7)

glColor3f(0.9, 0.9, 0.9);

else

glColor3f(0.6, 0.6, 0.6);

glBegin(GL\_QUADS);

glVertex2i(50, hight - 6 \* (hight / 7) - 20);

glVertex2i(50, 10);

glVertex2i(wight / 7, 10);

glVertex2i(wight / 7, hight - 6 \* (hight / 7) - 20);

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2i(50, hight - 6 \* (hight / 7) - 20);

glVertex2i(50, 10);

glVertex2i(wight / 7, 10);

glVertex2i(wight / 7, hight - 6 \* (hight / 7) - 20);

glEnd();

string n1 = "DELETE EDGE";

glRasterPos2i(wight / 20, 0.061 \* hight);

for (int i = 0; i < n1.length(); i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, n1[i]);

}

//Рисование круга

void draw\_krug(int x, int y, int R)

{

glColor3f(0.6f,0.6f, 0.6f);

float x1, y1;

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float theta = 2.0f \* 3.1415926f \* float(i) / float(360);

y1 = R \* cos(theta) + y;

x1 = R \* sin(theta) + x;;

glVertex2f(x1, y1);

}

glEnd();

glColor3f(0.0f, 0.0f, 0.0f);

float x2, y2;

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i < 360; i++)

{

float theta = 2.0f \* 3.1415926f \* float(i) / float(360);

y2 = R \* cos(theta) + y;

x2 = R \* sin(theta) + x;

glVertex2f(x2, y2);

}

glEnd();

}

//Рисование выбранного круга

void draw\_krug2(int x, int y, int R)

{

glColor3f(0.9, 0.9, 0.9);

float x1, y1;

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float theta = 2.0f \* 3.1415926f \* float(i) / float(360);

y1 = R \* cos(theta) + y;

x1 = R \* sin(theta) + x;;

glVertex2f(x1, y1);

}

glEnd();

glColor3f(0.0, 0.392, 0.0);

float x2, y2;

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i < 360; i++)

{

float theta = 2.0f \* 3.1415926f \* float(i) / float(360);

y2 = R \* cos(theta) + y;

x2 = R \* sin(theta) + x;

glVertex2f(x2, y2);

}

glEnd();

}

//Рисование веса ребра

void draw\_zifri(int text, int x1, int y1)

{

glColor3f(0.0, 0.0, 0.0);

GLvoid\* font = GLUT\_BITMAP\_HELVETICA\_18;

string s = to\_string(text);

glRasterPos2i(x1 - 5, y1 - 5);

for (size\_t j = 0; j < s.length(); j++)

glutBitmapCharacter(font, s[j]);

}

//Рисование ребра ориентированного взвешенного графа

void draw\_rebro(int text, int x0, int y0, int x1, int y1)

{

glColor3i(0, 0, 0);

glBegin(GL\_LINES);

glVertex2i(x0, y0);

glVertex2i(x1, y1);

glEnd();

draw\_zifri(text, (x0 + x1) / 2 + 10, (y0 + y1) / 2 + 10);

float vx = x0 - x1;

float vy = y0 - y1;

float s = 1.0f / sqrt(vx \* vx + vy \* vy);

vx \*= s;

vy \*= s;

x1 = x1 + R \* vx;

y1 = y1 + R \* vy;

glColor3i(0, 0, 0);

glBegin(GL\_TRIANGLES);

glVertex2f(x1, y1);

glVertex2f(x1 + 10 \* (vx + vy), y1 + 10 \* (vy - vx));

glVertex2f(x1 + 10 \* (vx - vy), y1 + 10 \* (vy + vx));

glEnd();

}

//Pисование вершины (круг + значение)

void draw\_vertex(int n)

{

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

{

if (Mouse\_Vert[i])

draw\_krug2(vertC[i].x, vertC[i].y, R);

else

draw\_krug(vertC[i].x, vertC[i].y, R);

draw\_zifri(i + 1, vertC[i].x, vertC[i].y);

}

}

//Расстановка координат вершин по кругу радиуса R и установка радиуса вершин R

void drav\_vce\_krugi(int i, int n)

{

int R\_;

int x0 = wight / 2;

int y0 = hight / 2;

if (wight > hight)

{

R = 5 \* (hight / 13) / n;

R\_ = hight / 2 - R - 10;

}

else

{

R = 5 \* (wight / 13) / n;

R\_ = wight / 2 - R - 10;

}

float theta = 2.0f \* 3.1415926f \* i / n;

int y1 = R\_ \* cos(theta) + y0;

int x1 = R\_ \* sin(theta) + x0;

vertC[i].x = x1;

vertC[i].y = y1;

}

//Создание графа

void make\_graph()

{

stand\_vid = false;

int amountVerts, amountEdges, sourceVertex, targetVertex, edgeWeight, choice;

cout << "Какой граф? \n1) Ориентированный\n2) Неориентированный\n3) Смешанный\n";

cin >> choice;

cout << "Введите количество вершин в графе: "; cin >> amountVerts;

cout << "Введите количество ребер в графе: "; cin >> amountEdges;

cout << endl;

for (int i = 1; i <= amountVerts; i++) {

graph.InsertVertex(i);

}

if (choice == 3)

{

cout << "Сколько ребер являются неориентированными? ";

int unEdges;

cin >> unEdges;

amountEdges -= unEdges;

cout << "Вводите данные по неориентированным ребрам:\n ";

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

{

cout << "Исходная вершина: ";

cin >> sourceVertex;

cout << "Конечная вершина: ";

cin >> targetVertex;

cout << "Вес ребра: ";

cin >> edgeWeight;

cout << endl;

int choice3 = 2;

//graph.Insert(choice);

graph.InsertEdge(sourceVertex, targetVertex, choice3, edgeWeight);

}

choice = 1;

cout << "Вводите данные по ориентированным ребрам:\n ";

}

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

{

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

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

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

graph.InsertEdge(sourceVertex, targetVertex, choice, edgeWeight);

}

cout << endl;

graph.Print();

}

//сщздание базового графа

void make\_graph\_baza()

{

graph.InsertVertex(1);

graph.InsertVertex(2);

graph.InsertVertex(3);

graph.InsertVertex(4);

graph.InsertVertex(5);

graph.InsertVertex(6);

graph.InsertEdge(1, 3, 2, 7);//не ориентированные ребра

graph.InsertEdge(3, 5, 2, 12);

graph.InsertEdge(5, 6, 2, 16);

graph.InsertEdge(6, 4, 2, 14);

graph.InsertEdge(1, 5, 1, 19);//ориентированные ребра

graph.InsertEdge(5, 4, 1, 31);

graph.InsertEdge(4, 1, 1, 20);

graph.InsertEdge(2, 6, 1, 24);

graph.InsertEdge(3, 2, 1, 28);

}

//Проверка: корректна ли задача Коммивояжера

bool f\_or\_t(int\*\* matrix)

{

if (graph.IsEmpty())

return false;

for (int i = 0; i < graph.GetAmountVerts(); i++)

{

int cnt = 0;

for (int j = 0; j < graph.GetAmountVerts(); j++)

{

if (matrix[i][j] > 0)

cnt++;

}

if (cnt < 1)

return false;

}

return true;

}

//Проверка: курсор в зоне вершины или нет

int mous\_vertexs\_tab(int x, int y)

{

for (int i = 0; i < graph.GetAmountVerts(); i++)

if (pow(x - vertC[i].x, 2) + pow(y - vertC[i].y, 2) <= pow(R, 2))

return i;

return -1;

}

//Проверка: курсор в зоне кнопки или нет

void y\_n(int x, int y)

{

if (x > 50 && x < wight / 7 && y < (hight - 20) && y >(hight - hight / 7))

Button\_Flag = 1;

else if (x > 50 && x < wight / 7 && y < (hight - hight / 7 - 20) && y > hight - 2 \* (hight / 7))

Button\_Flag = 2;

else if (x > 50 && x < wight / 7 && y < hight - 2 \* (hight / 7) - 20 && y > hight - 3 \* (hight / 7))

Button\_Flag = 3;

else if (x > 50 && x < wight / 7 && y > hight - 4 \* (hight / 7) && y < hight - 3 \* (hight / 7) - 20)

Button\_Flag = 4;

else if (x > 50 && x < wight / 7 && y > hight - 5 \* (hight / 7) && y < hight - 4 \* (hight / 7) - 20)

Button\_Flag = 5;

else if (x > 50 && x < wight / 7 && y > hight - 6 \* (hight / 7) && y < hight - 5 \* (hight / 7) - 20)

Button\_Flag = 6;

else if (x > 50 && x < wight / 7 && y > 10 && y < hight - 6 \* (hight / 7) - 20)

Button\_Flag = 7;

else

Button\_Flag = 0;

}

//Отслеживание движения курсора и возможность двигать вершину

void dvisenir\_mous(int x, int y)

{

y = hight - y;

CursorX = x;

CursorY = y;

int i = mous\_vertexs\_tab(x, y);

if (i != -1)

Mouse\_Vert[i] = true;

else

for (int j = 0; j < graph.GetAmountVerts(); j++)

Mouse\_Vert[j] = false;

if (Vert\_Move)

{

vertC[Moving\_Vertex].x = CursorX;

vertC[Moving\_Vertex].y = CursorY;

}

y\_n(x, y);

glutPostRedisplay();

}

//регистрация нажатий мыши

void tab\_mouse(int button, int state, int x, int y)

{

int j = mous\_vertexs\_tab(x, hight - y);

if (Vert\_Move)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

Vert\_Move = false;

return;

}

}

if (j != -1)

{

stand\_vid = true;

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

Vert\_Move = true;

Moving\_Vertex = j;

return;

}

}

if (button == 3 || button == 4)

{

stand\_vid = true;

if (button == 3)

{

if (R != 300)

R++;

}

else if (button == 4)

{

if (R != 15)

R--;

}

}

if (x >= 50 and x <= (wight / 7) and y >= 20 and y <= (hight / 7))

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

New\_Way.clear();

Way.clear();

Graph New;

graph = New;

make\_graph();

return;

}

if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)

{

stand\_vid = false;

return;

}

}

if (x >= 50 and x <= (wight / 7) and y >= ((hight / 7) + 20) and y <= 2 \* (hight / 7))

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

New\_Way.clear();

Way.clear();

cout << "\nЗадача Коммивояжера:\n";

int\*\* matrix = matr\_0\_v\_1();

bool checker = f\_or\_t(matrix);

if (!checker)

{

cout << "\nЗадача Коммивояжера для данного графа некорректна\n\n";

return;

}

int n = graph.GetAmountVerts();

while (Way.size() < n)

matrix = rabita\_viz\_0(matrix);

cout << endl;

print\_result();

return;

}

}

if (x >= 50 and x <= (wight / 7) and y >= (2 \* (hight / 7) + 20) and y <= 3 \* (hight / 7))

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

graph.Print();

return;

}

}

if (x >= 50 && x <= wight / 7 && y <= 4 \* (hight / 7) && y >= 3 \* (hight / 7) + 20)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

int cur = graph.GetAmountVerts();

graph.InsertVertex(cur + 1);

vertC[cur].x = wight / 2;

vertC[cur].y = hight / 2;

return;

}

}

if (x >= 50 && x <= wight / 7 && y <= 5 \* (hight / 7) && y >= 4 \* (hight / 7) + 20)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

graph.EraseLastVert();

int cur = graph.GetAmountVerts();

return;

}

}

if (x >= 50 && x <= wight / 7 && y <= 6 \* (hight / 7) && y >= 5 \* (hight / 7) + 20)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

int sourceVertex, targetVertex, edgeWeight, vertNum, choice;

cout << "Количество добавляемых ребер: "; cin >> vertNum;

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

{

cout << "Какое ребро(1 - ориентированное, 2 - неориентированное)?";

cin >> choice;

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

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

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

graph.InsertEdge(sourceVertex, targetVertex, choice, edgeWeight);

}

return;

}

}

if (x > 50 && x < wight / 7 && y <= hight - 10 && y >= 6 \* (hight / 7) + 20)

{

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

{

int sourceVertex, targetVertex;

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

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

graph.EraseEdge(sourceVertex, targetVertex);

return;

}

}

}

//отвечает за масштабирование

void mashtab(int w, int h)

{

wight = w;

hight = h;

glViewport(0, 0, (GLsizei)wight, (GLsizei)hight);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, (GLdouble)wight, 0, (GLdouble)hight);

glutPostRedisplay();

}

//Изображение

void draw()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, wight, 0, hight);

glViewport(0, 0, wight, hight);

glClearColor(0.2, 0.2, 0.2, 0.2);

glClear(GL\_COLOR\_BUFFER\_BIT);

button1();

button2();

button3();

button4();

button5();

button6();

button7();

graph.draw\_graph();

glutSwapBuffers();

}

### *//Graph.h*

#pragma once

#include <GL\glew.h>

#include <GL\freeglut.h>

#include <SFML/Graphics.hpp>

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <vector>

#include <cmath>

#include <string>

using namespace std;

const int maxSize = 10; //максимальное число вершин(если больше небольшие баги в отрисовке)

extern int wight; //ширина окна (OpenGL)

extern int hight; //высота окна (OpenGL)

struct vertCoord{int x, y;}; //структура координат вершин

class Graph

{

private:

vector<int> vertList; //вектор вершин

int adjMatrix[maxSize][maxSize]; //матрица смежности

public:

Graph();

~Graph();

bool IsEmpty(); //проверка на пустоту

bool IsFull(); //проверка на полноту

int GetVertText(int i) { return vertList[i]; }

vector<int> GetVertList() { return vertList; }

int GetAdjMatrixElem(int i, int j) { return adjMatrix[i][j]; }

void SetEdgeZero(int i, int j) { adjMatrix[i][j] = 0; adjMatrix[j][i] = 0; }

int GetAmountVerts() { return vertList.size(); }

int GetVertPos(const int& vertex);

int get\_amount\_edges(int choice);

void InsertVertex(const int& vertex); //добавление новой вершины

void InsertEdge(const int& vertex1, const int& vertex2, int choice, int weigth); //добавление нового ребра

void Print();

void EraseLastVert();

void EraseEdge(const int& vertex1, const int& vertex2);

void draw\_graph();

};

extern Graph graph; //создание объекта класса Graph

// Функции для решения задачи Коммивояжера

void print\_matrix(int\*\* matrix); //вывод текущей матрицы

bool f\_or\_t(int\*\* matrix); //проверка: можно ли решить задачу Коммивояжера

int\*\* matr\_0\_v\_1(); //создается матрица, заменяя 0 на -1

int\* min\_el(int\* line, int n); //поиск минимального элемента в строке матрицы

int\*\* reduct\_matrix(int\*\* oldmatrix); //редукция матрицы

int\*\* rabita\_viz\_0(int\*\* oldmatrix); //Удаление нуля с наибольшей оценкой

void print\_result(); //вывод результата решения задачи Коммивояжера

// Рисование кнопок

void button1(); //кнопка "Make New Graph"

void button2(); //кнопка "Salesman Task"

void button3(); //кнопка "Print Matrix In Console"

void button4(); //кнопка "Add Vertex"

void button5(); //кнопка "Delete Vertex"

void button7(); //кнопка "Delete Edge"

void button6(); //кнопка "Add Edge"

// Рисование элементов графа (OpenGL)

void draw\_krug(int x, int y, int R); //рисование круга

void draw\_krug2(int x, int y, int R); //рисование круга, если курсор в зоне вершины

void draw\_zifri(int nom, int x1, int y1); //рисование веса ребра

void draw\_rebro(int text, int x0, int y0, int x1, int y1); //рисование ребра

void draw\_vertex(int n); //рисование вершины

void make\_graph(); //создание графа

void make\_graph\_baza(); //создание базового графа

void drav\_vce\_krugi(int i, int n); //установка координат для вершин графа

// Функции - проверки

int mous\_vertexs\_tab(int x, int y); //проверка: курсор в зоне вершины или нет

void y\_n(int x, int y); //проверка: курсор в зоне кнопки или нет

// Функции отслеживания поведения мыши

void dvisenir\_mous(int x, int y); //отслеживание движения курсора мыши

void tab\_mouse(int button, int state, int x, int y); //функция регистрации нажатий мыши

// Изображение

void mashtab(int w, int h); //отвечает за масштабирование

void draw(); //отвечает за изображение

# **Заключение**

В процессе изучения библиотек OpenGL и SFML я освоила базовые методы визуализации графики. Овладела основами работы с OpenGL, такими как создание буферов, текстур, шейдеров и отрисовки геометрических примитивов. SFML предоставила удобный инструментарий для работы с окнами, вводом-выводом, звуком и изображениями, что позволило мне создавать интерактивные приложения с графическим интерфейсом.

Благодаря изучению этих библиотек я приобрела навыки:

1. Создание визуальных элементов: Отрисовка геометрических фигур, работа с текстурами и цветами.
2. Управление графическим контекстом: Использование буферов, шейдеров и матриц преобразований.
3. Создание интерактивных элементов: Обработка событий, создание меню и кнопок.

Опыт работы с OpenGL и SFML расширил мои знания в области компьютерной графики и программирования. Я готова применять полученные навыки для создания более сложных и интересных проектов.

В будущем я планирую продолжить изучение этих библиотек, освоить более продвинутые техники, такие как шейдерное программирование, анимация и эффекты, для создания более реалистичной и захватывающей графики.

## Ссылка на видео отчет на YouTube:

<https://youtu.be/JNwmsi_8jxs?si=Lt3lZ8FcYMHJeIiH>

## Ссылка на GitHab:

<https://github.com/ANG555/ARM-Florista>

<https://github.com/ANG555/Kommivouzor>