Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»

ПНИПУ

Отчет по лабораторной работе

«АРМ + Решение задачи коммивояжёра и его визуализация»

Выполнили:
студенты группы ИВТ-23-2Б
Чудинов Данил Николаевич
Меновщиков Глеб Николаевич
Соловьева Екатерина Александровна

Проверила:
доцент кафедры ИТАС
О.А. Полякова

Пермь, 2024 г.

АВТОМАТИЗИРОВАННОЕ РАБОЧЕЕ МЕСТО ОФИЦИАНТА

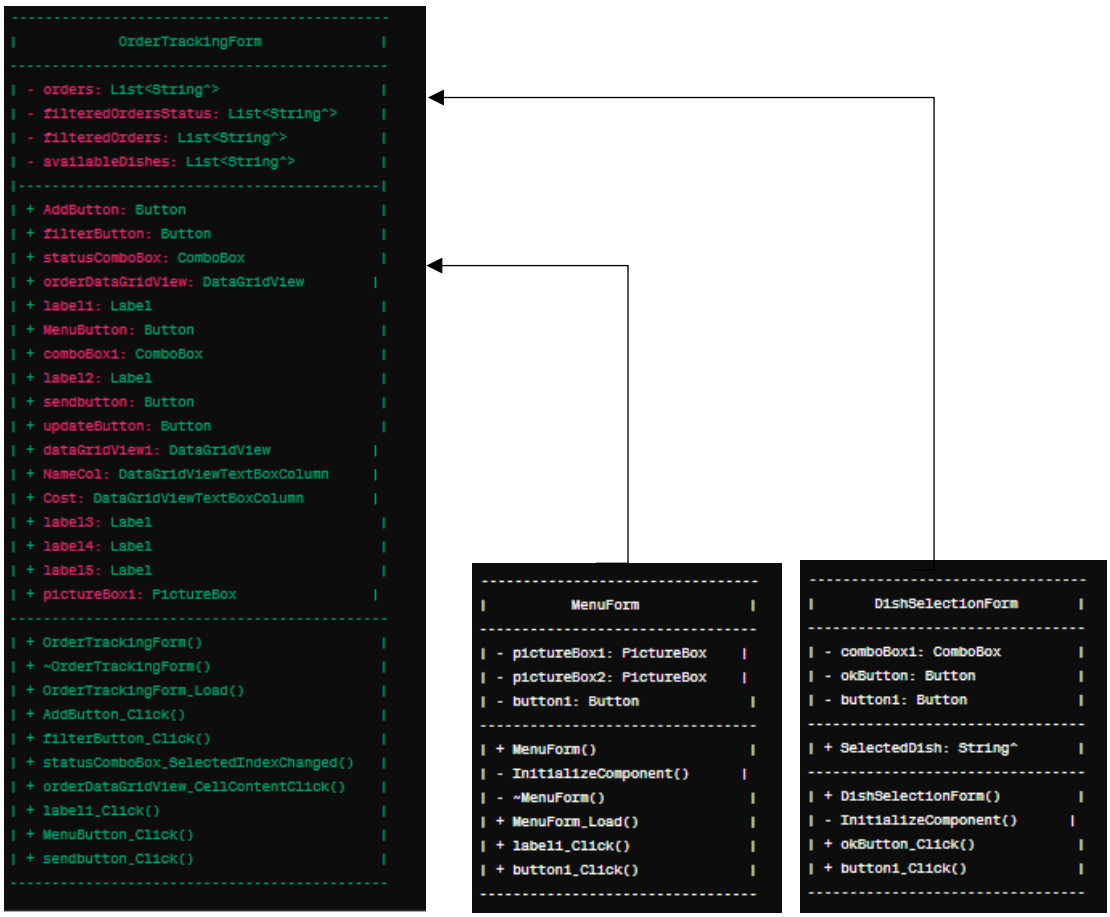
Идея:
Работа официанта требует много внимания для каждого посетителя. В свою очередь каждому посетителю хочется знать когда его заказ готов и каков итоговый чек. Наш проект объединил решения этих проблем.

ПО:
Для выполнение творческой работы выбран Windows Forms, видео снято с помощью стандартной функции записи экрана windows.

Классы:

1. OrderTrackingForm - форма где происходит взаимодействие клиента с интерфейсом программы.
2. MenuForm – форма которая демонстрирует меню
3. DishSelectionForm – форма демонстрирующая список доступных блюд и позволяет выбрать одно из них.
4. Base – для реализации базы данных
5. ClassName – вспомогательно для базы данных
6. ManageForm – форма для работы с программой со стороны персонала.

UML:



Программный код

Main.cpp

```
#include "OrderTrackingForm.h"
using namespace Project1;
int main()
{
   Application::EnableVisualStyles();
   Application::SetCompatibleTextRenderingDefault(false);

   OrderTrackingForm^ form = gcnew OrderTrackingForm();
   Application::Run(form);

   return 0;
}
```

ClassNames.cpp

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Data::SqlClient;

ref class ClassNames
{
public:
   ClassNames();

   String^ numbertable;
   property String^ NumberTable
   {
      String^ get()
      {
         return numbertable;
      }
      void set(String^ value)
      {
         numbertable = value;
      }
   }

   String^ numberorder;
   property String^ NumberOrder
   {
      String^ get()
      {
         return numberorder;
      }
      void set(String^ value)
      {
         numberorder = value;
      }
   }

   String^ namefood;
   property String^ NameFood
   {
```

```
        String^ get()
        {
            return namefood;
        }
        void set(String^ value)
        {
            namefood = value;
        }
    }

    String^ status;
    property String^ Status
    {
        String^ get()
        {
            return status;
        }
        void set(String^ value)
        {
            status = value;
        }
    }
};
```

Base.h

```
#pragma once
#include "ClassNames.h"
#include <iostream>

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Data::SqlClient;
using namespace System::Collections::Generic;

ref class Base
{
        public:
                Base();

                SqlConnection^ conn;
                SqlConnectionStringBuilder^ connStringBuilder;

                void ConnectToDB()
                {
                        connStringBuilder = gcnew SqlConnectionStringBuilder();
                        connStringBuilder->DataSource = "DESKTOP-QDGFT6I\\SQLEXPRESS";
                        connStringBuilder->InitialCatalog = "OrderTrackingDB";
                        connStringBuilder->IntegratedSecurity = true;

                        conn = gcnew SqlConnection(connStringBuilder->ConnectionString);
                }

        public:
                void InsertData(String^ Table, String^ Number, String^ Name, String^ Status)
                {
                        try
                        {
                                ConnectToDB();

                                String^ cmdText = "INSERT INTO dbo.Orders([Номер столика], [Номер заказа],
[Название блюда], [Готовность]) VALUES(@TableVstavka, @NumberVstavka, @NameVstavka, @StatusVstavka)";
```

```
                SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
                cmd->Parameters->AddWithValue("@TableVstavka", Table);
                cmd->Parameters->AddWithValue("@NumberVstavka", Number);
                cmd->Parameters->AddWithValue("@NameVstavka", Name);
                cmd->Parameters->AddWithValue("@StatusVstavka", Status);

                conn->Open();
                cmd->ExecuteNonQuery();
            }
            catch (Exception^ ex)
            {
                // Обработка исключения, например, вывод ошибки
                Console::WriteLine("Ошибка при выполнении запроса: " + ex->Message);
            }
            finally
            {
                conn->Close();
            }
        }


    public:
        void Update(String^ Number, String^ newStatus)
        {
            try
            {
                ConnectToDB();

                String^ cmdText = "UPDATE dbo.Orders SET Готовность = @Readiness WHERE [Номер заказа] = @OrderNumber";
                SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);

                cmd->Parameters->AddWithValue("@Readiness", newStatus);
                cmd->Parameters->AddWithValue("@OrderNumber", Number);

                conn->Open();
                cmd->ExecuteNonQuery();
            }
            catch (Exception^ ex)
            {
                // Обработка исключения, например, вывод ошибки
                Console::WriteLine("Ошибка при выполнении запроса: " + ex->Message);
            }
            finally
            {
                conn->Close();
            }
        }

    public:
        void DeleteData(String^ Number)
        {
            try
            {
                ConnectToDB();

                String^ cmdText = "DELETE FROM dbo.Orders WHERE [Номер заказа] = @NumberToDelete";
                SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
                cmd->Parameters->AddWithValue("@NumberToDelete", Number);

                conn->Open();
                cmd->ExecuteNonQuery();
            }
            catch (Exception^ ex)
            {
```

```
                        // Обработка исключения, например, вывод ошибки
                        Console::WriteLine("Ошибка при выполнении запроса: " + ex->Message);
                    }
                    finally
                    {
                        conn->Close();
                    }
                }

            public:
                List<ClassNames^>^ Base::FillTable()
                {
                    List<ClassNames^>^ namesList = gcnew List<ClassNames^>();
                    try
                    {
                        ConnectToDB();
                        String^ cmdText = "SELECT * FROM dbo.Orders";
                        SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
                        conn->Open();
                        SqlDataReader^ reader = cmd->ExecuteReader();
                        while (reader->Read())
                        {
                            ClassNames^ n = gcnew ClassNames();
                            n->numbertable = reader["Номер столика"]->ToString();
                            n->numberorder = reader["Номер заказа"]->ToString();
                            n->namefood = reader["Название блюда"]->ToString();
                            n->status = reader["Готовность"]->ToString();

                            namesList->Add(n);
                        }
                        return namesList;
                    }
                    finally
                    {
                        if (conn != nullptr)
                        {
                            conn->Close();
                        }
                    }
                }
};
```

DishSelectionForm.h

```
#pragma once
#include <vector>
#include <string>

namespace Project1 {
   using namespace System;
   using namespace System::ComponentModel;
   using namespace System::Collections;
   using namespace System::Windows::Forms;
   using namespace System::Data;
   using namespace System::Drawing;
   using namespace std;
   public ref class DishSelectionForm : public System::Windows::Forms::Form
   {
   public:
      property String^ SelectedDish {
         String^ get() {
            return comboBox1->SelectedItem != nullptr ? comboBox1->SelectedItem->ToString() : "";
         }
      }

      DishSelectionForm(System::Collections::Generic::List<String^>^ dishes) {
```

```
            InitializeComponent();
            for each (String ^ dish in dishes) {
                comboBox1->Items->Add(dish);
            }
            comboBox1->SelectedIndex = 0;
        }


    protected:
        ~DishSelectionForm() {
            if (components) {
                delete components;
            }
        }

    private:
        System::ComponentModel::Container^ components;
        System::Windows::Forms::ComboBox^ comboBox1;
    private: System::Windows::Forms::Button^ button1;
           System::Windows::Forms::Button^ okButton;

        void InitializeComponent(void) {
            this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
            this->okButton = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->SuspendLayout();
            //
            // comboBox1
            //
            this->comboBox1->FormattingEnabled = true;
            this->comboBox1->Location = System::Drawing::Point(18, 18);
            this->comboBox1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->comboBox1->Name = L"comboBox1";
            this->comboBox1->Size = System::Drawing::Size(280, 28);
            this->comboBox1->TabIndex = 0;
            //
            // okButton
            //
            this->okButton->BackColor =
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->okButton->Location = System::Drawing::Point(18, 60);
            this->okButton->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->okButton->Name = L"okButton";
            this->okButton->Size = System::Drawing::Size(112, 35);
            this->okButton->TabIndex = 1;
            this->okButton->Text = L"OK";
            this->okButton->UseVisualStyleBackColor = false;
            this->okButton->Click += gcnew System::EventHandler(this, &DishSelectionForm::okButton_Click);
            //
            // button1
            //
            this->button1->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(128)),
                static_cast<System::Int32>(static_cast<System::Byte>(128)));
            this->button1->Location = System::Drawing::Point(188, 60);
            this->button1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(112, 35);
            this->button1->TabIndex = 2;
            this->button1->Text = L"Отмена";
            this->button1->UseVisualStyleBackColor = false;
            this->button1->Click += gcnew System::EventHandler(this, &DishSelectionForm::button1_Click);
            //
```

```
            // DishSelectionForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(318, 109);
            this->ControlBox = false;
            this->Controls->Add(this->button1);
            this->Controls->Add(this->okButton);
            this->Controls->Add(this->comboBox1);
            this->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->Name = L"DishSelectionForm";
            this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
            this->Text = L"Выбор блюда";
            this->Load += gcnew System::EventHandler(this, &DishSelectionForm::DishSelectionForm_Load);
            this->ResumeLayout(false);

        }

        // Обработчик события для нажатия кнопки "OK"
        void okButton_Click(System::Object^ sender, System::EventArgs^ e) {
            this->DialogResult = System::Windows::Forms::DialogResult::OK;
            this->Close();
        }
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
        this->Close();
    }
    private: System::Void DishSelectionForm_Load(System::Object^ sender, System::EventArgs^ e) {
    }
    };
}
```

MenuForm.h

```
#pragma once
#include <vcclr.h>
#include <vector>
#include <string>

namespace Project1 {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    public ref class MenuForm : public System::Windows::Forms::Form
    {
    public:
        MenuForm(void)
        {
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        ~MenuForm()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::PictureBox^ pictureBox1;
```

```
    protected:
    private: System::Windows::Forms::PictureBox^ pictureBox2;
    private: System::Windows::Forms::Button^ button1;

    protected:


    private: System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(MenuForm::typeid));
            this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
            this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());
            this->button1 = (gcnew System::Windows::Forms::Button());
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->BeginInit();
            this->SuspendLayout();
            //
            // pictureBox1
            //
            this->pictureBox1->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"pictureBox1.Image")));
            this->pictureBox1->Location = System::Drawing::Point(12, 12);
            this->pictureBox1->Name = L"pictureBox1";
            this->pictureBox1->Size = System::Drawing::Size(749, 1010);
            this->pictureBox1->TabIndex = 0;
            this->pictureBox1->TabStop = false;
            //
            // pictureBox2
            //
            this->pictureBox2->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"pictureBox2.Image")));
            this->pictureBox2->Location = System::Drawing::Point(785, 12);
            this->pictureBox2->Name = L"pictureBox2";
            this->pictureBox2->Size = System::Drawing::Size(763, 1010);
            this->pictureBox2->TabIndex = 1;
            this->pictureBox2->TabStop = false;
            //
            // button1
            //
            this->button1->BackColor = System::Drawing::Color::Red;
            this->button1->ForeColor = System::Drawing::SystemColors::ButtonHighlight;
            this->button1->Location = System::Drawing::Point(23, 29);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(103, 23);
            this->button1->TabIndex = 2;
            this->button1->Text = L"Закрыть меню";
            this->button1->UseVisualStyleBackColor = false;
            this->button1->Click += gcnew System::EventHandler(this, &MenuForm::button1_Click);
            //
            // MenuForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->AutoScroll = true;
            this->ClientSize = System::Drawing::Size(541, 938);
            this->ControlBox = false;
            this->Controls->Add(this->button1);
            this->Controls->Add(this->pictureBox2);
            this->Controls->Add(this->pictureBox1);
            this->Name = L"MenuForm";
            this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
            this->Text = L"MenuForm";
```

```
            this->Load += gcnew System::EventHandler(this, &MenuForm::MenuForm_Load);
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->EndInit();
            this->ResumeLayout(false);

        }
#pragma endregion
    private: System::Void MenuForm_Load(System::Object^ sender, System::EventArgs^ e) {
        Size = System::Drawing::Size(540, 960);
    }
    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
    }
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
        this->Close();
    }
    };
}
```

OrderTrackingForm.h

```
#pragma once
#include <vcclr.h>
#include <vector>
#include <string>
#include "DishSelectionForm.h"
#include "MenuForm.h"
#include "Base.h"

namespace Project1 {

            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 OrderTrackingForm
            /// </summary>
            public ref class OrderTrackingForm : public System::Windows::Forms::Form
            {
            public:
                        OrderTrackingForm(void)
                        {
                                    InitializeComponent();
                                    //
                                    //TODO: добавьте код конструктора
                                    //
                                    orders = gcnew System::Collections::Generic::List<String^>();
                                    filteredOrdersStatus = gcnew System::Collections::Generic::List<String^>();
                                    filteredOrders = gcnew System::Collections::Generic::List<String^>();
                                    this->Load += gcnew System::EventHandler(this,
&OrderTrackingForm::OrderTrackingForm_Load);

                                    availableDishes = gcnew System::Collections::Generic::List<String^>();
                                    availableDishes->Add("Яйца невинности");
                                    availableDishes->Add("Гамбо по Луизиански");
                                    availableDishes->Add("Ширако");
                                    availableDishes->Add("Сеульский Бибимбап");
                                    availableDishes->Add("Дим-Самы");
                                    availableDishes->Add("Напиток: Кола 0,5 л");
                                    availableDishes->Add("Напиток: Чай 1 л");
                                    availableDishes->Add("Напиток: Молочный коктель 0,3 л");
                                    availableDishes->Add("Напиток: Глинтвейн 0,3 л");
                                    availableDishes->Add("Напиток: Пиво 0,5 л");
```

```
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~OrderTrackingForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^ AddButton;
	private: System::Windows::Forms::Button^ filterButton;
	private: System::Windows::Forms::ComboBox^ statusComboBox;
	private: System::Windows::Forms::DataGridView^ orderDataGridView;

	protected:
	private:
		System::Collections::Generic::List<String^>^ orders;
		System::Collections::Generic::List<String^>^ filteredOrdersStatus;
		System::Collections::Generic::List<String^>^ filteredOrders;


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

	private: System::Collections::Generic::List<String^>^ availableDishes;

	private: System::Windows::Forms::Button^ MenuButton;
	private: System::Windows::Forms::ComboBox^ comboBox1;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn1;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn2;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::Button^ sendbutton;
	private: System::Windows::Forms::Button^ updateButton;
	private: System::Windows::Forms::DataGridView^ dataGridView1;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ NameCol;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ Cost;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::PictureBox^ pictureBox1;

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container^ components;

#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)
		{
			System::ComponentModel::ComponentResourceManager^ resources = (gcnew
System::ComponentModel::ComponentResourceManager(OrderTrackingForm::typeid));
			this->AddButton = (gcnew System::Windows::Forms::Button());
			this->filterButton = (gcnew System::Windows::Forms::Button());
			this->statusComboBox = (gcnew System::Windows::Forms::ComboBox());
			this->orderDataGridView = (gcnew System::Windows::Forms::DataGridView());
			this->dataGridViewTextBoxColumn1 = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
```

```
                this->dataGridViewTextBoxColumn2 = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->MenuButton = (gcnew System::Windows::Forms::Button());
                this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->sendbutton = (gcnew System::Windows::Forms::Button());
                this->updateButton = (gcnew System::Windows::Forms::Button());
                this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
                this->NameCol = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->Cost = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->label5 = (gcnew System::Windows::Forms::Label());
                this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->orderDataGridView))-
>BeginInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))-
>BeginInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))-
>BeginInit();
                this->SuspendLayout();
                //
                // AddButton
                //
                this->AddButton->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(0)),
static_cast<System::Int32>(static_cast<System::Byte>(192)),
                        static_cast<System::Int32>(static_cast<System::Byte>(0)));
                this->AddButton->Location = System::Drawing::Point(338, 299);
                this->AddButton->Name = L"AddButton";
                this->AddButton->Size = System::Drawing::Size(141, 58);
                this->AddButton->TabIndex = 0;
                this->AddButton->Text = L"Добавить заказ";
                this->AddButton->UseVisualStyleBackColor = false;
                this->AddButton->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::AddButton_Click);
                //
                // filterButton
                //
                this->filterButton->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(192)),
                        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                this->filterButton->Location = System::Drawing::Point(260, 83);
                this->filterButton->Name = L"filterButton";
                this->filterButton->Size = System::Drawing::Size(75, 23);
                this->filterButton->TabIndex = 1;
                this->filterButton->Text = L"Фильтр";
                this->filterButton->UseVisualStyleBackColor = false;
                this->filterButton->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::filterButton_Click);
                //
                // statusComboBox
                //
                this->statusComboBox->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(192)),
                        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                this->statusComboBox->FormattingEnabled = true;
                this->statusComboBox->Items->AddRange(gcnew cli::array< System::Object^ >(3) {
L"New", L"In Progress", L"Completed" });
                this->statusComboBox->Location = System::Drawing::Point(341, 85);
                this->statusComboBox->Name = L"statusComboBox";
                this->statusComboBox->Size = System::Drawing::Size(138, 21);
```

```
                this->statusComboBox->TabIndex = 2;
                this->statusComboBox->SelectedIndexChanged += gcnew System::EventHandler(this,
&OrderTrackingForm::statusComboBox_SelectedIndexChanged);
                //
                // orderDataGridView
                //
                this->orderDataGridView->BackgroundColor = System::Drawing::Color::IndianRed;
                this->orderDataGridView->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->orderDataGridView->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^ >(3) {
                        this->dataGridViewTextBoxColumn1,
                                this->dataGridViewTextBoxColumn2, this->Column1
                });
                this->orderDataGridView->GridColor = System::Drawing::Color::DarkGoldenrod;
                this->orderDataGridView->Location = System::Drawing::Point(46, 112);
                this->orderDataGridView->Name = L"orderDataGridView";
                this->orderDataGridView->RowHeadersWidth = 62;
                this->orderDataGridView->Size = System::Drawing::Size(433, 181);
                this->orderDataGridView->TabIndex = 3;
                this->orderDataGridView->CellContentClick += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this,
&OrderTrackingForm::orderDataGridView_CellContentClick);
                //
                // dataGridViewTextBoxColumn1
                //
                this->dataGridViewTextBoxColumn1->HeaderText = L"Номер заказа";
                this->dataGridViewTextBoxColumn1->MinimumWidth = 8;
                this->dataGridViewTextBoxColumn1->Name = L"dataGridViewTextBoxColumn1";
                this->dataGridViewTextBoxColumn1->Width = 50;
                //
                // dataGridViewTextBoxColumn2
                //
                this->dataGridViewTextBoxColumn2->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                this->dataGridViewTextBoxColumn2->HeaderText = L"Выбранное блюдо";
                this->dataGridViewTextBoxColumn2->MinimumWidth = 8;
                this->dataGridViewTextBoxColumn2->Name = L"dataGridViewTextBoxColumn2";
                //
                // Column1
                //
                this->Column1->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                this->Column1->HeaderText = L"Статус";
                this->Column1->MinimumWidth = 8;
                this->Column1->Name = L"Column1";
                this->Column1->Resizable = System::Windows::Forms::DataGridViewTriState::False;
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(43, 67);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(119, 13);
                this->label1->TabIndex = 5;
                this->label1->Text = L"Выберите Ваш столик";
                this->label1->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::label1_Click);
                //
                // MenuButton
                //
                this->MenuButton->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(192)),
static_cast<System::Int32>(static_cast<System::Byte>(192)),
                        static_cast<System::Int32>(static_cast<System::Byte>(255)));
```

```
            this->MenuButton->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->MenuButton->Location = System::Drawing::Point(46, 299);
            this->MenuButton->Name = L"MenuButton";
            this->MenuButton->Size = System::Drawing::Size(98, 58);
            this->MenuButton->TabIndex = 6;
            this->MenuButton->Text = L"Меню";
            this->MenuButton->UseVisualStyleBackColor = false;
            this->MenuButton->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::MenuButton_Click);
            //
            // comboBox1
            //
            this->comboBox1->FormattingEnabled = true;
            this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(10) {
                L"1", L"2", L"3", L"4", L"5", L"6", L"7", L"8",
                    L"9", L"10"
            });
            this->comboBox1->Location = System::Drawing::Point(46, 85);
            this->comboBox1->Name = L"comboBox1";
            this->comboBox1->Size = System::Drawing::Size(121, 21);
            this->comboBox1->TabIndex = 7;
            //
            // label2
            //
            this->label2->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),
                static_cast<System::Int32>(static_cast<System::Byte>(128)));
            this->label2->Location = System::Drawing::Point(150, 308);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(182, 40);
            this->label2->TabIndex = 8;
            this->label2->Text = L"Нажмите на название заказа чтобы удалить из списка";
            //
            // sendbutton
            //
            this->sendbutton->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(0)),
static_cast<System::Int32>(static_cast<System::Byte>(192)),
                static_cast<System::Int32>(static_cast<System::Byte>(0)));
            this->sendbutton->Location = System::Drawing::Point(338, 363);
            this->sendbutton->Name = L"sendbutton";
            this->sendbutton->Size = System::Drawing::Size(141, 58);
            this->sendbutton->TabIndex = 9;
            this->sendbutton->Text = L"Отправить список повару";
            this->sendbutton->UseVisualStyleBackColor = false;
            this->sendbutton->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::sendbutton_Click);
            //
            // updateButton
            //
            this->updateButton->BackColor = System::Drawing::Color::LightCoral;
            this->updateButton->Location = System::Drawing::Point(485, 112);
            this->updateButton->Name = L"updateButton";
            this->updateButton->Size = System::Drawing::Size(33, 181);
            this->updateButton->TabIndex = 10;
            this->updateButton->Text = L"О\r\nБ\r\nН\r\nО\r\nВ\r\nИ\r\nТ\r\nЬ";
            this->updateButton->UseVisualStyleBackColor = false;
            this->updateButton->Click += gcnew System::EventHandler(this,
&OrderTrackingForm::updateButton_Click);
            //
            // dataGridView1
            //
```

```
                this->dataGridView1->BackgroundColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(192)),
                        static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)));
                this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->dataGridView1->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^ >(2) {
                        this->NameCol,
                                this->Cost
                });
                this->dataGridView1->Location = System::Drawing::Point(46, 501);
                this->dataGridView1->Name = L"dataGridView1";
                this->dataGridView1->Size = System::Drawing::Size(276, 150);
                this->dataGridView1->TabIndex = 11;
                this->dataGridView1->CellContentClick += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this,
&OrderTrackingForm::dataGridView1_CellContentClick);
                //
                // NameCol
                //
                this->NameCol->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                this->NameCol->HeaderText = L"Название блюда";
                this->NameCol->Name = L"NameCol";
                //
                // Cost
                //
                this->Cost->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                this->Cost->HeaderText = L"Цена";
                this->Cost->Name = L"Cost";
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label3->Location = System::Drawing::Point(334, 501);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(82, 20);
                this->label3->TabIndex = 12;
                this->label3->Text = L"К оплате:";
                //
                // label4
                //
                this->label4->AutoSize = true;
                this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->label4->Location = System::Drawing::Point(428, 501);
                this->label4->Name = L"label4";
                this->label4->Size = System::Drawing::Size(40, 20);
                this->label4->TabIndex = 13;
                this->label4->Text = L"0.00";
                //
                // label5
                //
                this->label5->AutoSize = true;
                this->label5->Location = System::Drawing::Point(474, 506);
                this->label5->Name = L"label5";
                this->label5->Size = System::Drawing::Size(24, 13);
                this->label5->TabIndex = 14;
                this->label5->Text = L"руб";
                //
```

```
                        // pictureBox1
                        //
                        this->pictureBox1->Image = (cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"pictureBox1.Image")));
                        this->pictureBox1->Location = System::Drawing::Point(338, 524);
                        this->pictureBox1->Name = L"pictureBox1";
                        this->pictureBox1->Size = System::Drawing::Size(160, 127);
                        this->pictureBox1->SizeMode = System::Windows::Forms::PictureBoxSizeMode::Zoom;
                        this->pictureBox1->TabIndex = 15;
                        this->pictureBox1->TabStop = false;
                        //
                        // OrderTrackingForm
                        //
                        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                        this->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),
                                static_cast<System::Int32>(static_cast<System::Byte>(128)));
                        this->ClientSize = System::Drawing::Size(524, 888);
                        this->Controls->Add(this->pictureBox1);
                        this->Controls->Add(this->label5);
                        this->Controls->Add(this->label4);
                        this->Controls->Add(this->label3);
                        this->Controls->Add(this->dataGridView1);
                        this->Controls->Add(this->updateButton);
                        this->Controls->Add(this->sendbutton);
                        this->Controls->Add(this->label2);
                        this->Controls->Add(this->comboBox1);
                        this->Controls->Add(this->MenuButton);
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->orderDataGridView);
                        this->Controls->Add(this->statusComboBox);
                        this->Controls->Add(this->filterButton);
                        this->Controls->Add(this->AddButton);
                        this->Icon = (cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
                        this->Name = L"OrderTrackingForm";
                        this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
                        this->Text = L"OrderTrackingForm";
                        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->orderDataGridView))-
>EndInit();
                        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))-
>EndInit();
                        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))-
>EndInit();
                        this->ResumeLayout(false);
                        this->PerformLayout();

                }
#pragma endregion
        private: System::Void OrderTrackingForm_Load(System::Object^ sender, System::EventArgs^ e) {
                Text = "Order Tracking";
                Size = System::Drawing::Size(540, 960);
                StartPosition = FormStartPosition::CenterScreen;
                // Установка "New" по умолчанию в ComboBox
                statusComboBox->SelectedIndex = 0;
        }

        private: System::Void AddButton_Click(System::Object^ sender, System::EventArgs^ e) {
                // Показываем форму выбора блюда
                DishSelectionForm^ dishForm = gcnew DishSelectionForm(availableDishes);
                if (dishForm->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
                        // Получаем выбранное блюдо
                        String^ selectedDish = dishForm->SelectedDish;
                        // Если блюдо было выбрано, добавляем его в таблицу заказов
                        if (!String::IsNullOrEmpty(selectedDish)) {
```

```
                                        // Добавляем строку с информацией о заказе
                                        orderDataGridView->Rows->Add(orders->Count + 1, selectedDish, "New");
                                        // Добавляем информацию о заказе в список
                                        orders->Add(selectedDish);
                                }
                        }
                }

                private: System::Void filterButton_Click(System::Object^ sender, System::EventArgs^ e) {
                        // Получаем выбранный статус из ComboBox
                        String^ selectedStatus = statusComboBox->SelectedItem->ToString();

                        // Проходим по каждой строке таблицы и делаем ее невидимой, если статус не соответствует
выбранному
                        for (int i = 0; i < orderDataGridView->RowCount; i++) {
                                // Проверяем, не является ли текущая строка новой строкой
                                if (!orderDataGridView->Rows[i]->IsNewRow) {
                                        String^ currentStatus = "";
                                        if (orderDataGridView->Rows[i]->Cells[2]->Value != nullptr) {
                                                currentStatus = safe_cast<String^>(orderDataGridView->Rows[i]-
>Cells[2]->Value);
                                        }
                                        // Если статус не совпадает с выбранным, делаем строку невидимой
                                        orderDataGridView->Rows[i]->Visible = (currentStatus == selectedStatus);
                                }
                        }
                }


                private: System::Void statusComboBox_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)
{
                }

                private: System::Void orderDataGridView_CellContentClick(System::Object^ sender,
System::Windows::Forms::DataGridViewCellEventArgs^ e) {
                        // Проверяем, что клик был на кнопке "Отменить"
                        if (e->RowIndex >= 0) {
                                // Получаем номер заказа из ячейки
                                String^ orderNumber = orderDataGridView->Rows[e->RowIndex]->Cells[0]->Value-
>ToString();

                                // Показываем диалоговое окно с вопросом
                                System::Windows::Forms::DialogResult result = MessageBox::Show("Хотите ли вы
отменить заказ №" + orderNumber + "?", "Отмена заказа", MessageBoxButtons::YesNo, MessageBoxIcon::Question);

                                // Если пользователь выбрал "Да"
                                if (result == System::Windows::Forms::DialogResult::Yes) {

                                        Base b;
                                        b.ConnectToDB();
                                        b.DeleteData(orderDataGridView->Rows[e->RowIndex]->Cells[0]->Value-
>ToString());

                                        // Удаляем заказ из списка
                                        orders->RemoveAt(e->RowIndex);
                                        // Удаляем строку из таблицы
                                        orderDataGridView->Rows->RemoveAt(e->RowIndex);

                                }
                        }
                }


                private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {
                }
```

```
private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
}

private: System::Void MenuButton_Click(System::Object^ sender, System::EventArgs^ e) {
	MenuForm^ menuForm = gcnew MenuForm();
	menuForm->Show();
}

private: System::Void sendbutton_Click(System::Object^ sender, System::EventArgs^ e) {
	Base b;
	Dictionary<String^, double> dishPrices;
	dishPrices["Яйца невинности"] = 100.0;
	dishPrices["Гамбо по Луизиански"] = 150.0;
	dishPrices["Ширако"] = 120.0;
	dishPrices["Сеульский Бибимбап"] = 180.0;
	dishPrices["Дим-Самы"] = 90.0;
	dishPrices["Напиток: Кола 0,5 л"] = 50.0;
	dishPrices["Напиток: Чай 1 л"] = 30.0;
	dishPrices["Напиток: Молочный коктель 0,3 л"] = 70.0;
	dishPrices["Напиток: Глинтвейн 0,3 л"] = 80.0;
	dishPrices["Напиток: Пиво 0,5 л"] = 60.0;

	double totalCost;

	// Перебор каждой строки в orderDataGridView
	for (int i = 0; i < orderDataGridView->RowCount; i++) {
		if (orderDataGridView->Rows[i]->Cells[1]->Value != nullptr) {
			String^ dishName = orderDataGridView->Rows[i]->Cells[1]->Value->ToString();
			double price = 0.0;
			// Проверка, существует ли название блюда в словаре dishPrices
			if (dishPrices.ContainsKey(dishName)) {
				price = dishPrices[dishName];
			}
			// Добавление названия блюда и цены в dataGridView1
			dataGridView1->Rows->Add(dishName, price);
			// Добавление цены блюда к общей сумме
			totalCost += price;
		}
	}
	label4->Text = totalCost.ToString();
	// Проходим по каждой строке таблицы заказов
	for (int i = 0; i < orderDataGridView->RowCount; ) {
		// Проверяем, что статус заказа в текущей строке равен "New"
		if (orderDataGridView->Rows[i]->Cells[2]->Value != nullptr &&
			orderDataGridView->Rows[i]->Cells[2]->Value->ToString() == "New") {
			// Меняем статус заказа на "In Progress"
			orderDataGridView->Rows[i]->Cells[2]->Value = "In Progress";

			b.InsertData(comboBox1->Text,
				orderDataGridView->Rows[i]->Cells[0]->Value->ToString(),
				orderDataGridView->Rows[i]->Cells[1]->Value->ToString(),
				orderDataGridView->Rows[i]->Cells[2]->Value->ToString());

			i++;
		}
		else {
			// Если статус не равен "New", переходим к следующей строке
			i++;
		}
	}


}


private: System::Void updateButton_Click(System::Object^ sender, System::EventArgs^ e) {
```

```
		Base b;
		List<ClassNames^>^ namesList = b.FillTable();
		orderDataGridView->Rows->Clear();
		for (int i = 0; i < namesList->Count; i++)
		{
			orderDataGridView->Rows->Add();
			orderDataGridView->Rows[i]->Cells[0]->Value = namesList[i]->NumberOrder;
			orderDataGridView->Rows[i]->Cells[1]->Value = namesList[i]->NameFood;
			orderDataGridView->Rows[i]->Cells[2]->Value = namesList[i]->Status;
		}
	}

private: System::Void dataGridView1_CellContentClick(System::Object^ sender,
System::Windows::Forms::DataGridViewCellEventArgs^ e) {
}
};
}
```

Manage.sln

main.cpp

```
#include "ManageForm.h"
using namespace Manage;
int main()
{
  Application::EnableVisualStyles();
  Application::SetCompatibleTextRenderingDefault(false);

  ManageForm^ form = gcnew ManageForm();
  Application::Run(form);

  return 0;
}
```

Base.h

```
#pragma once
#include "ClassNames.h"
#include <iostream>

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Data::SqlClient;
using namespace System::Collections::Generic;

ref class Base
{

public:
	Base();

	SqlConnection^ conn;
	SqlConnectionStringBuilder^ connStringBuilder;

	void ConnectToDB()
	{
		connStringBuilder = gcnew SqlConnectionStringBuilder();
		connStringBuilder->DataSource = "DESKTOP-QDGFT6I\\SQLEXPRESS";
		connStringBuilder->InitialCatalog = "OrderTrackingDB";
		connStringBuilder->IntegratedSecurity = true;
```

```
            conn = gcnew SqlConnection(connStringBuilder->ConnectionString);
        }


public:
        List<ClassNames^>^ Base::FillTable()
        {
            List<ClassNames^>^ namesList = gcnew List<ClassNames^>();
            try
            {
                ConnectToDB();
                String^ cmdText = "SELECT * FROM dbo.Orders";
                SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);
                conn->Open();
                SqlDataReader^ reader = cmd->ExecuteReader();
                while (reader->Read())
                {
                    ClassNames^ n = gcnew ClassNames();
                    n->numbertable = reader["Номер столика"]->ToString();
                    n->numberorder = reader["Номер заказа"]->ToString();
                    n->namefood = reader["Название блюда"]->ToString();
                    String^ readiness = reader["Готовность"]->ToString();
                    if (readiness->Equals("Completed", StringComparison::InvariantCultureIgnoreCase))
{
                        n->status = true;
                    }
                    else {
                        n->status = false;
                    }
                    namesList->Add(n);
                }
                return namesList;
            }
            finally
            {
                if (conn != nullptr)
                {
                    conn->Close();
                }
            }

        }

public:
        void UpdateStatusInDatabase(String^ numOrder, String^ newStatus)
        {
            try
            {
                ConnectToDB();
                String^ cmdText = "UPDATE dbo.Orders SET Готовность = @Readiness WHERE [Номер
заказа] = @numOrder";
                SqlCommand^ cmd = gcnew SqlCommand(cmdText, conn);

                cmd->Parameters->AddWithValue("@Readiness", newStatus);
                cmd->Parameters->AddWithValue("@numOrder", numOrder);

                conn->Open();
                cmd->ExecuteNonQuery();
            }
            finally
            {
                if (conn != nullptr)
                {
                    conn->Close();
                }
```

```
                }
            }

};
```

ClassNames.h

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Data::SqlClient;

ref class ClassNames
{
public:
    ClassNames();

    String^ numbertable;
    property String^ NumberTable
    {
        String^ get()
        {
            return numbertable;
        }
        void set(String^ value)
        {
            numbertable = value;
        }
    }

    String^ numberorder;
    property String^ NumberOrder
    {
        String^ get()
        {
            return numberorder;
        }
        void set(String^ value)
        {
            numberorder = value;
        }
    }

    String^ namefood;
    property String^ NameFood
    {
        String^ get()
        {
            return namefood;
        }
        void set(String^ value)
        {
            namefood = value;
        }
    }

    Boolean^ status;
    property Boolean^ Status
```

```
    {
        Boolean^ get()
        {
            return status;
        }
        void set(Boolean^ value)
        {
            status = value;
        }
    }
};
```

ManageForm.h

```
#pragma once
#include "Base.h"

namespace Manage {

	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 ManageForm
	/// </summary>
	public ref class ManageForm : public System::Windows::Forms::Form
	{
	public:
		ManageForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~ManageForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataGridView^ dataGridView1;



	private: System::Windows::Forms::Button^ sendButton;
	private: System::Windows::Forms::Button^ updateButton;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ tableNumber;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ OrderNumber;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ OrderName;
	private: System::Windows::Forms::DataGridViewCheckBoxColumn^ OrderStatus;
```

```
        protected:



        private:
                /// <summary>
                /// Обязательная переменная конструктора.
                /// </summary>
                System::ComponentModel::Container ^components;

#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->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
                        this->tableNumber = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->OrderNumber = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->OrderName = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                        this->OrderStatus = (gcnew System::Windows::Forms::DataGridViewCheckBoxColumn());
                        this->sendButton = (gcnew System::Windows::Forms::Button());
                        this->updateButton = (gcnew System::Windows::Forms::Button());
                        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))-
>BeginInit();
                        this->SuspendLayout();
                        //
                        // dataGridView1
                        //
                        this->dataGridView1->BackgroundColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
                                static_cast<System::Int32>(static_cast<System::Byte>(128)),
static_cast<System::Int32>(static_cast<System::Byte>(128)));
                        this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                        this->dataGridView1->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^  >(4) {
                                this->tableNumber,
                                        this->OrderNumber, this->OrderName, this->OrderStatus
                        });
                        this->dataGridView1->Location = System::Drawing::Point(12, 12);
                        this->dataGridView1->Name = L"dataGridView1";
                        this->dataGridView1->RowHeadersWidth = 62;
                        this->dataGridView1->Size = System::Drawing::Size(459, 210);
                        this->dataGridView1->TabIndex = 0;
                        this->dataGridView1->CellContentClick += gcnew
System::Windows::Forms::DataGridViewCellEventHandler(this, &ManageForm::dataGridView1_CellContentClick);
                        //
                        // tableNumber
                        //
                        this->tableNumber->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                        this->tableNumber->HeaderText = L"Номер столика";
                        this->tableNumber->MinimumWidth = 8;
                        this->tableNumber->Name = L"tableNumber";
                        //
                        // OrderNumber
                        //
                        this->OrderNumber->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                        this->OrderNumber->HeaderText = L"Номер заказа";
                        this->OrderNumber->MinimumWidth = 8;
                        this->OrderNumber->Name = L"OrderNumber";
                        //
```

```
                                // OrderName
                                //
                                this->OrderName->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                                this->OrderName->HeaderText = L"Название блюда";
                                this->OrderName->MinimumWidth = 8;
                                this->OrderName->Name = L"OrderName";
                                //
                                // OrderStatus
                                //
                                this->OrderStatus->AutoSizeMode =
System::Windows::Forms::DataGridViewAutoSizeColumnMode::Fill;
                                this->OrderStatus->HeaderText = L"Готовность";
                                this->OrderStatus->MinimumWidth = 8;
                                this->OrderStatus->Name = L"OrderStatus";
                                //
                                // sendButton
                                //
                                this->sendButton->BackColor =
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->sendButton->Location = System::Drawing::Point(489, 12);
                                this->sendButton->Name = L"sendButton";
                                this->sendButton->Size = System::Drawing::Size(124, 65);
                                this->sendButton->TabIndex = 1;
                                this->sendButton->Text = L"Отправить";
                                this->sendButton->UseVisualStyleBackColor = false;
                                this->sendButton->Click += gcnew System::EventHandler(this,
&ManageForm::sendButton_Click);
                                //
                                // updateButton
                                //
                                this->updateButton->Location = System::Drawing::Point(489, 94);
                                this->updateButton->Name = L"updateButton";
                                this->updateButton->Size = System::Drawing::Size(124, 56);
                                this->updateButton->TabIndex = 2;
                                this->updateButton->Text = L"Обновить";
                                this->updateButton->UseVisualStyleBackColor = true;
                                this->updateButton->Click += gcnew System::EventHandler(this,
&ManageForm::updateButton_Click);
                                //
                                // ManageForm
                                //
                                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                                this->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)),
static_cast<System::Int32>(static_cast<System::Byte>(255)),
                                                static_cast<System::Int32>(static_cast<System::Byte>(128)));
                                this->ClientSize = System::Drawing::Size(646, 429);
                                this->Controls->Add(this->updateButton);
                                this->Controls->Add(this->sendButton);
                                this->Controls->Add(this->dataGridView1);
                                this->Name = L"ManageForm";
                                this->Text = L"ManageForm";
                                this->Load += gcnew System::EventHandler(this, &ManageForm::ManageForm_Load);
                                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))-
>EndInit();
                                this->ResumeLayout(false);

                        }
#pragma endregion
                private: System::Void ManageForm_Load(System::Object^ sender, System::EventArgs^ e) {
                }
```

```
        private: System::Void dataGridView1_CellContentClick(System::Object^ sender,
System::Windows::Forms::DataGridViewCellEventArgs^ e) {
        }
        private: System::Void updateButton_Click(System::Object^ sender, System::EventArgs^ e)
        {
                Base b;
                List<ClassNames^>^ namesList = b.FillTable();
                dataGridView1->Rows->Clear();
                for (int i = 0; i < namesList->Count; i++)
                {
                        dataGridView1->Rows->Add();
                        dataGridView1->Rows[i]->Cells[0]->Value = namesList[i]->NumberTable;
                        dataGridView1->Rows[i]->Cells[1]->Value = namesList[i]->NumberOrder;
                        dataGridView1->Rows[i]->Cells[2]->Value = namesList[i]->NameFood;
                        dataGridView1->Rows[i]->Cells[3]->Value = namesList[i]->Status;
                }
        }
        private: System::Void sendButton_Click(System::Object^ sender, System::EventArgs^ e)
        {
                Base b;
                for (int i = 0; i < dataGridView1->RowCount; i++)
                {
                        // Приводим значение к типу bool и проверяем, равно ли оно true
                        bool readiness;
                        if (Boolean::TryParse(dataGridView1->Rows[i]->Cells[3]->Value->ToString(), readiness)
&& readiness)
                        {
                                // Изменяем значение в базе данных
                                String^ numOrder = dataGridView1->Rows[i]->Cells[1]->Value->ToString(); //
Получаем Номер заказа
                                String^ newStatus = "Completed";
                                b.UpdateStatusInDatabase(numOrder, newStatus);
                        }
                }
        }
};
}
```

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ КОММИВОЕЖОРА

Для данной лабораторной работы нужен был метод ветвей и границ.

Алгоритм решения:

1. Составление матрицы смежности;
2. Нахождение минимума по строкам;
3. Редукция строк;
4. Нахождение минимума по столбцам;
5. Редукция столбцов;
6. Нахождение оценок для нулевых элементов;
7. Редукция матрицы;
8. Выбор: Если мы еще не нашли все отрезки пути, которые позволяют вернуться Коммивояжеру в исходный город, то возвращаемся к шагу

Если все отрезки пути найдены или оставшаяся часть очевидна – переходим к заключительному шагу – соединение путей. В реалиях данной задачи необходимо перейти к шагу 2.;

9. Построение маршрута;
10. Вычисление длины пути.

Программный код

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <sstream>
#include <Windows.h>
#include <GL\glew.h>
#include <GL\freeglut.h>
#include <iostream>

using namespace std;

int n;
int** help;
int* result;
int*** mat;
int R;
int WinW;
int WinH;
const int maxSize = 20;
int amountVerts = 0;

struct vertCoord//Структура установки координат
{
	int x, y;
};
vertCoord vertC[20];

template<class T>
class Graph
{
	vector<T> vetrexList;
	vector<T> labelList;
	int size;
	bool* visitedVerts = new bool[vetrexList.size()];
public:
	vector<vector<int>> adjMatrix;
	Graph();
	Graph<T>(const int& ksize);
	~Graph();
	void DrawGraph();
	void InsertEdge(const T& vertex1, const T& vertex2, int weight); //Шаблон графа, здесь написаны прототипы функций
	inline void insertVertex(const T& vertex);
	void removeVertex(const T& vertex);
	inline int GetVertPos(const T& vertex);
	inline bool isEmpty();
	inline bool IsFull();
	inline int GetAmountVerts();
	int GetAmountEdges();
	inline int GetWeight(const T& vertex1, const T& vertex2);
	vector<T> GetNbrs(const T& vertex);
	void PrintMatrix();
	void removeEdge(const T& vertex1, const T& vertex2);
```

```
        void editEdgeWeight(const T& vertex1, const T& vertex2, int newWeight);
};

Graph<int> graph(20);
template<class T>
vector<T> Graph<T> ::GetNbrs(const T& vetrex) {//Функция для получения вектора соседей
        vector<T> nbrsList;
        int pos = this->GetVertPos(vetrex);
        if (pos != -1) {
                for (int i = 0; i < this->vetrexList.size(); i++) {
                        if (this->adjMatrix[pos][i] != 0) {
                                nbrsList.push_back(this->vetrexList[i]);
                        }
                }
        }
        return nbrsList;
}



template<class T>
inline void Graph<T>::insertVertex(const T& vert) {//Функция, которая добавляет вершину
        if (this->IsFull()) {
                cout << "Невозможно добавить вершину." << endl;
                return;
        }
        this->vetrexList.push_back(vert);
}
template<class T>
void Graph<T>::removeEdge(const T& vertex1, const T& vertex2) {//Функция, которая удаляет ребро
        int pos1 = GetVertPos(vertex1);
        int pos2 = GetVertPos(vertex2);

        if (pos1 == -1 || pos2 == -1) {
                cout << "Одной из вершин нет в графе." << endl;
                return;
        }

        if (adjMatrix[pos1][pos2] == 0) {
                cout << "Ребра между вершинами " << vertex1 << " и " << vertex2 << " нет." << endl;
                return;
        }

        adjMatrix[pos1][pos2] = 0;
        adjMatrix[pos2][pos1] = 0;

        cout << "Ребро между вершинами " << vertex1 << " и " << vertex2 << " удалено." << endl;
}

template<class T>
void Graph<T>::removeVertex(const T& vertex) {////Функция, которая удаляет вершину
        int pos = GetVertPos(vertex);
        if (pos == -1) {
                cout << "Вершины " << vertex << " нет в графе." << endl;
                return;
        }

        for (int i = 0; i < size; i++) {
                if (adjMatrix[pos][i] != 0) removeEdge(vertex, vetrexList[i]);
                if (adjMatrix[i][pos] != 0) removeEdge(vetrexList[i], vertex);
        }

        vetrexList.erase(vetrexList.begin() + pos);

        // Удаляем столбец pos из каждой строки матрицы
        for (int i = 0; i < size; i++) {
```

```
                adjMatrix[i].erase(adjMatrix[i].begin() + pos);
        }
        // Удаляем строку pos из матрицы
        adjMatrix.erase(adjMatrix.begin() + pos);

        size--;

        cout << "Вершина " << vertex << " удалена." << endl;
}
template<class T>
int Graph<T>::GetAmountEdges() {//Функция для получения количества ребер для неориентированного графа
        int amount = 0;
        if (!this->isEmpty()) {
                for (int i = 0; i < this->vetrexList.size(); i++) {
                        for (int j = 0; j < this->vetrexList.size(); j++) {
                                if (this->adjMatrix[i][j] != 0) {
                                        amount++;
                                }
                        }
                }
        }
        return amount / 2;
}

template<class T>
inline int Graph<T>::GetWeight(const T& g1, const T& g2) {//Получение веса между вершинами
        if (this->isEmpty()) {
                return 0;
        }
        int g1_p = this->GetVertPos(g1);
        int g2_p = this->GetVertPos(g2);
        if (g1_p == -1 || g2_p == -1) {
                cout << "Одного из выбранных узлов в графе не существует!";
                return 0;
        }
        return this->adjMatrix[g1_p][g2_p];
}

template<class T>
inline int Graph<T>::GetAmountVerts() {//Получение количества вершин
        return this->vetrexList.size();
}
template<class T>
inline bool Graph<T>::isEmpty() {//Проверка графа на то, что он пуст
        return this->vetrexList.size() == 0;
}
template<class T>//Проверка графа на то, что он заполнен
inline bool Graph<T>::IsFull() {
        return (vetrexList.size() == maxSize);
}

template <class T>
inline int Graph<T>::GetVertPos(const T& g) {//Получение индекса вершин
        for (int i = 0; i < vetrexList.size(); i++) {
                if (this->vetrexList[i] == g) {
                        return i;
                }
        }
        return -1;
}
template<class T>
Graph<T>::Graph() {
        size = maxSize;
        labelList.resize(size, 1000000);
        adjMatrix.resize(size, vector<int>(size, 0));
        visitedVerts = new bool[size];
```

```
}

template<class T>
Graph<T>::Graph(const int& ksize) {
	size = ksize;
	labelList.resize(size, 1000000);
	adjMatrix.resize(size, vector<int>(size, 0));
	visitedVerts = new bool[size];
}

template<class T>
Graph<T>::~Graph() {//Деструктор графа

}

template<class T>
void Graph<T>::InsertEdge(const T& vetrex1, const T& vetrex2, int weight) {//Вставка ребра для неориентированного графа
	if (GetVertPos(vetrex1) != (-1) && this->GetVertPos(vetrex2) != (-1)) {
		int vertPos1 = GetVertPos(vetrex1);
		int vertPos2 = GetVertPos(vetrex2);
		if (this->adjMatrix[vertPos1][vertPos2] != 0 && this->adjMatrix[vertPos2][vertPos1] != 0) {
			cout << "Ребро между вершинами уже есть" << endl;
			return;
		}
		else {
			this->adjMatrix[vertPos1][vertPos2] = weight;
			this->adjMatrix[vertPos2][vertPos1] = weight;
		}
	}
	else {
		cout << "Какой-либо вершины нет в графе" << endl;
		return;
	}
}

template<class T>
void Graph<T>::PrintMatrix() {//Печать матрицы смежности графа
	if (!this->isEmpty()) {
		cout << "Матрица смежности: " << endl;
		cout << "- ";
		for (int i = 0; i < vetrexList.size(); i++) {
			cout << " " << vetrexList[i] << " ";
		}
		cout << endl;
		for (int i = 0; i < this->vetrexList.size(); i++) {
			cout << this->vetrexList[i] << " ";
			for (int j = 0; j < this->vetrexList.size(); j++) {
				cout << " " << this->adjMatrix[i][j] << " ";
			}
			cout << endl;
		}
	}
	else {
		cout << "Граф пуст" << endl;
	}
}
template<class T>
void Graph<T>::editEdgeWeight(const T& vertex1, const T& vertex2, int newWeight) {//Функция, которая меняет вес ребра между вершинами
	int pos1 = GetVertPos(vertex1);
	int pos2 = GetVertPos(vertex2);

	if (pos1 == -1 || pos2 == -1) {
		cout << "Одной из вершин нет в графе." << endl;
		return;
```

```
	}

	if (adjMatrix[pos1][pos2] == 0) {
		cout << "Ребра между вершинами " << vertex1 << " и " << vertex2 << " нет." << endl;
		return;
	}

	adjMatrix[pos1][pos2] = newWeight;
	adjMatrix[pos2][pos1] = newWeight;

	cout << "Вес ребра между вершинами " << vertex1 << " и " << vertex2 << " изменен на " << newWeight <<
"." << endl;
}

void answer(int*** mat, int n, int** help, int* path)//Эта функция реализует алгоритм решения задачи коммивояжера,
используя Венгерский алгоритм.
{
	for (int l = 0; l < n; l++)
	{
		for (int i = 0; i < n; i++)
		{
			int min = 1000000;
			for (int j = 0; j < n; j++)
			{
				if (mat[i][j] && min > *mat[i][j])
				{
					min = *mat[i][j];
				}
			}
			for (int j = 0; j < n; j++)
			{
				if (mat[i][j])
				{
					*mat[i][j] -= min;
				}
			}
		}
		for (int j = 0; j < n; j++)
		{
			int min = 1000000;
			for (int i = 0; i < n; i++)
			{
				if (mat[i][j] && min > *mat[i][j])
				{
					min = *mat[i][j];
				}
			}
			for (int i = 0; i < n; i++)
			{
				if (mat[i][j])
				{
					*mat[i][j] -= min;
				}
			}
		}

		for (int i = 0; i < n; i++)
		{
			for (int j = 0; j < n; j++)
			{
				help[i][j] = 0;
			}
		}

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

```
                        for (int j = 0; j < n; j++)
                        {
                                if (mat[i][j] && !*mat[i][j])
                                {
                                        int hmin = 1000000;
                                        int vmin = 1000000;

                                        for (int l = 0; l < n; l++)
                                        {
                                                if (l != i && mat[l][j] && hmin > *mat[l][j])
                                                {
                                                        hmin = *mat[l][j];
                                                }
                                        }
                                        for (int l = 0; l < n; l++)
                                        {
                                                if (l != j && mat[i][l] && hmin > *mat[i][l])
                                                {
                                                        vmin = *mat[i][l];
                                                }
                                        }
                                        help[i][j] = hmin + vmin;
                                }
                        }
                }
                int mcost = 0, mi = 0, mj = 0;
                for (int i = 0; i < n; i++)
                {
                        for (int j = 0; j < n; j++)
                        {
                                if (mat[i][j] && mcost < help[i][j])
                                {
                                        mcost = help[i][j];
                                        mi = i;
                                        mj = j;
                                }
                        }
                }
                path[mi] = mj;

                for (int i = 0; i < n; i++)
                {
                        mat[i][mj] = nullptr;
                }
                for (int i = 0; i < n; i++)
                {
                        mat[mi][i] = nullptr;
                }

                mat[mj][mi] = nullptr;
        }
}

void preparation(int***& mat, int& n, int**& help, int*& result)// Эта функция подготавливает данные для алгоритма TSP (коммивояжера)
{
        n = amountVerts;// Присваиваем количество вершин из графа
        // Выделяем память под вспомогательные матрицы help и result
        help = new int* [n];
        result = new int[n];
        // Выделяем память под трехмерную матрицу mat, которая будет хранить матрицу смежности графа
        mat = new int** [n];
        // Инициализируем help
        for (int i = 0; i <= n; i++)
        {
                help[i] = new int[n];
```

```
		}
		// Заполняем mat значениями из матрицы смежности графа
		for (int i = 0; i <= n; i++)
		{
			mat[i] = new int* [n];
			for (int j = 0; j < n; j++)
			{
				if (graph.adjMatrix[i][j] == 0)
				{
					mat[i][j] = nullptr;
					continue;
				}
				mat[i][j] = new int(graph.adjMatrix[i][j]);
			}
		}
}

void TSP(int*** mat, int n, int** help, int* result)// Эта функция является точкой входа для решения задачи коммивояжера (TSP).
{
		preparation(mat, n, help, result);
		int s = 0;
		answer(mat, n, help, result);
		cout << endl << "Отрезки путей: ";
		for (int i = 0, j = 0; i < n; i++)
		{
			j = result[i];
			cout << i + 1 << " -> " << j + 1 << '\t';
			s += graph.adjMatrix[i][j];
		}
		cout << endl;
		cout << endl << "Кратчайший путь: ";
		int tmp = 0;
		for (int l = 0; l < n;)
		{
			for (int i = 0, j = 0; i < n; i++)
			{
				if (tmp == 0 || i + 1 == tmp)
				{
					if (tmp == 0)
					{
						cout << i + 1;
					}
					j = result[i];
					tmp = j + 1;
					if (tmp > 0)
					{
						cout << " -> " << tmp;
					}
					l++;
				}
			}
		}
		cout << endl << "Минимальное расстояние: " << s;
		cout << endl;
}

void setCoord(int i, int n)
{
		int R_;

		int x0 = WinW / 2;
		int y0 = WinH / 2;
		if (WinW > WinH)
		{
			R = 5 * (WinH / 13) / n;
```

```
                R_ = WinH / 2 - R - 10;
        }
        else {
                R = 5 * (WinW / 13) / n;
                R_ = WinW / 2 - R - 10;
        }
        float theta = 2.0f * 3.1415926f * float(i) / float(n);
        float y1 = R_ * cos(theta) + y0;
        float x1 = R_ * sin(theta) + x0;

        vertC[i].x = x1;
        vertC[i].y = y1;
}

void drawCircle(int x, int y, int R)//Функция, предназначенная для рисования круга
{
        glColor3f(1.0, 0.0, 0.0);
        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();
}

void drawText(int nom, int x1, int y1)//Отрисовка текста в вершине
{
        GLvoid* font = GLUT_BITMAP_TIMES_ROMAN_24;
        string s = to_string(nom);
        glRasterPos2i(x1 - 5, y1 - 5);
        for (int j = 0; j < s.length(); j++)
                glutBitmapCharacter(font, s[j]);
}

void drawVertex(int n)//Отрисовка вершины, текста в ней
{
        for (int i = 0; i < n; i++) {
                drawCircle(vertC[i].x, vertC[i].y, R);
                drawText(i + 1, vertC[i].x, vertC[i].y);
        }
}

void drawLine(int text, int x0, int y0, int x1, int y1) {//Отрисовка ребра, и текста на ребре
        glColor3f(0.0, 0.0, 0.0);
        glBegin(GL_LINES);
        glVertex2i(x0, y0);
        glVertex2i(x1, y1);
        glEnd();

        glColor4f(1.0f, 1.0f, 1.0f, 0.0f);
```

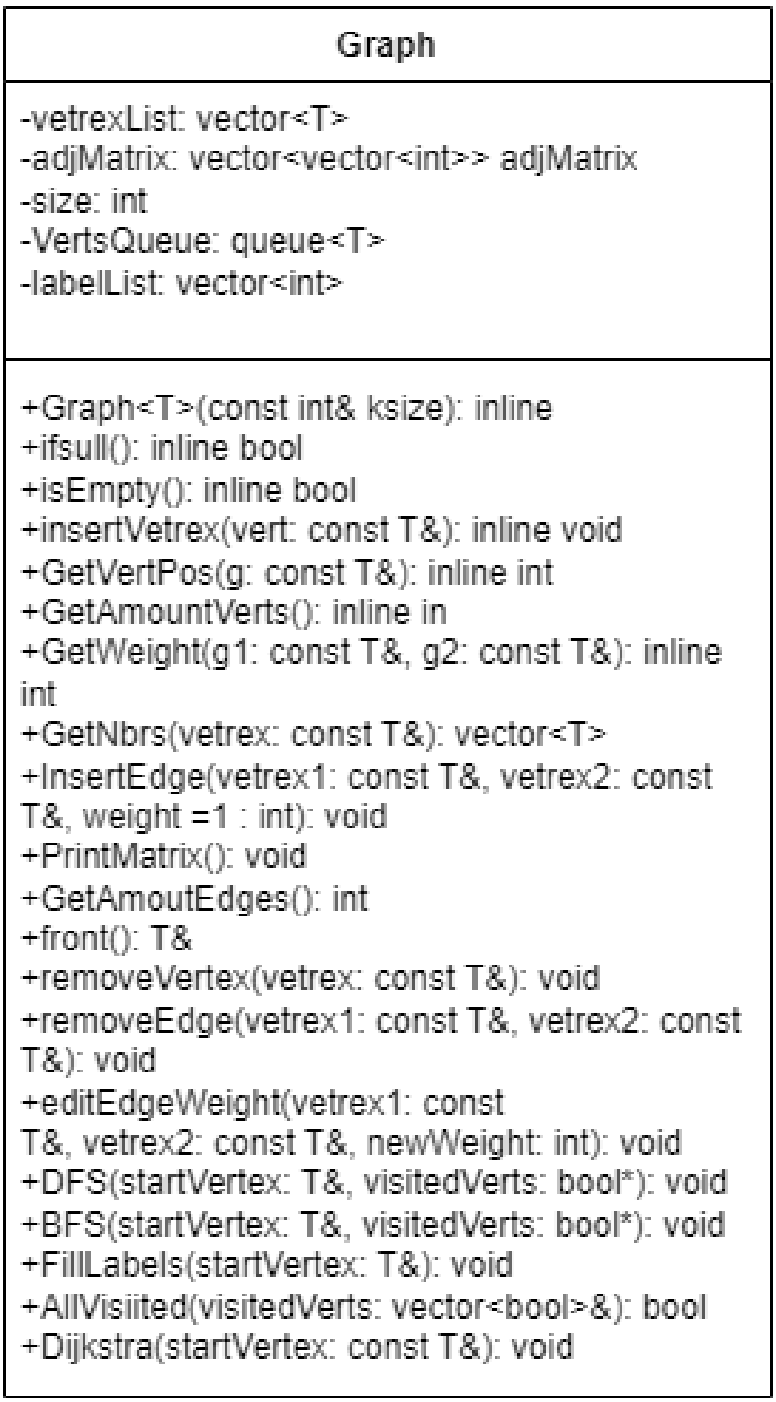
```
	drawText(text, (x0 + x1) / 2 + 11, (y0 + y1) / 2 + 11);
}



template<class T>
void Graph<T>::DrawGraph()//Главная функция, которая рисует сам граф
{
	int n = vetrexList.size();
	for (int i = 0; i < n; i++)
	{
		setCoord(i, n);
	}
	for (int i = 0; i < n; i++)
	{
		for (int j = i + 1; j < n; j++)
		{
			int a = adjMatrix[i][j];
			if (a != 0)
			{
				drawLine(a, vertC[i].x, vertC[i].y, vertC[j].x, vertC[j].y);
			}
		}
	}
	drawVertex(n);
}

void reshape(int w, int h)//Функция отвечающая за изменение размера вершин
{
	WinW = w;
	WinH = h;
	glViewport(0, 0, (GLsizei)WinW, (GLsizei)WinH);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, (GLdouble)WinW, 0, (GLdouble)WinH);
	glutPostRedisplay();
}

void drawMenuText(string text, int x1, int y1)//Функция для текста и его шрифта в менюшке
{
	GLvoid* font = GLUT_BITMAP_9_BY_15;
	string s = text;
	glRasterPos2i(x1 + 5, y1 - 20);
	for (int j = 0; j < s.length(); j++)
		glutBitmapCharacter(font, s[j]);
}
void drawMenu()//Рисуется меню с соответсвующими функциями
{
	int shift = 60;
	int height = 730;

	glColor3d(0.0, 0.0, 0.0);
	glBegin(GL_QUADS);
	glVertex2i(shift, height - shift - 30);
	glVertex2i(shift + 135, height - shift - 30);
	glVertex2i(shift + 135, height - shift);
	glVertex2i(shift, height - shift);
	glEnd();
	glColor3d(1, 1, 1);
	drawMenuText("insertVertex", shift, height - shift - 2);

	glColor3d(0.0, 0.0, 0.0);
	glBegin(GL_QUADS);
	glVertex2i(shift, height - shift - 70);
	glVertex2i(shift + 135, height - shift - 70);
	glVertex2i(shift + 135, height - shift - 40);
```

```
		glVertex2i(shift, height - shift - 40);
		glEnd();
		glColor3d(1, 1, 1);
		drawMenuText("DeleteVertex", shift, height - shift - 42);

		glColor3d(0.0, 0.0, 0.0);
		glBegin(GL_QUADS);
		glVertex2i(shift, height - shift - 110);
		glVertex2i(shift + 135, height - shift - 110);
		glVertex2i(shift + 135, height - shift - 80);
		glVertex2i(shift, height - shift - 80);
		glEnd();
		glColor3d(1, 1, 1);
		drawMenuText("PrintMatrix", shift, height - shift - 82);

		glColor3d(0.0, 0.0, 0.0);
		glBegin(GL_QUADS);
		glVertex2i(shift, height - shift - 150);
		glVertex2i(shift + 135, height - shift - 150);
		glVertex2i(shift + 135, height - shift - 120);
		glVertex2i(shift, height - shift - 120);
		glEnd();
		glColor3d(1, 1, 1);
		drawMenuText("TSP", shift, height - shift - 122);

		glColor3d(0.0, 0.0, 0.0);
		glBegin(GL_QUADS);
		glVertex2i(shift, height - shift - 190);
		glVertex2i(shift + 135, height - shift - 190);
		glVertex2i(shift + 135, height - shift - 160);
		glVertex2i(shift, height - shift - 160);
		glEnd();
		glColor3d(1, 1, 1);
		drawMenuText("editEdgeWeight", shift, height - shift - 162);
}
void mouseClick(int btn, int stat, int x, int y) {//Функция, которая позваляет взаимодействовать с кодом через визуализацию, изменять, удалять и т.д.
		int shift = 60;
		int height = 730;

		if (stat == GLUT_DOWN) {

			if (x > shift && x < shift + 135 && y >  shift && y < shift + 30)
			{
				int vertex;
				int sourceVertex;
				int targetVetrex;
				int edgeWeight;
				int Weight;
				int g, k;
				cout << "Введите кол-во вершин, которые вы хотите добавить: ";
				cin >> g;
				cout << "Введите кол-во ребёр, которые хотите добавить: ";
				cin >> k;
				for (int i = 0; i < g; i++) {
					cout << "Вершина: ";
					cin >> vertex;
					graph.insertVertex(vertex);
					amountVerts++;
					cout << endl;
				}
				for (int i = 0; i < k; i++) {
					cout << "Исходная вершина: ";
					cin >> sourceVertex;
					cout << endl;
					cout << "Конечная вершина: ";
```

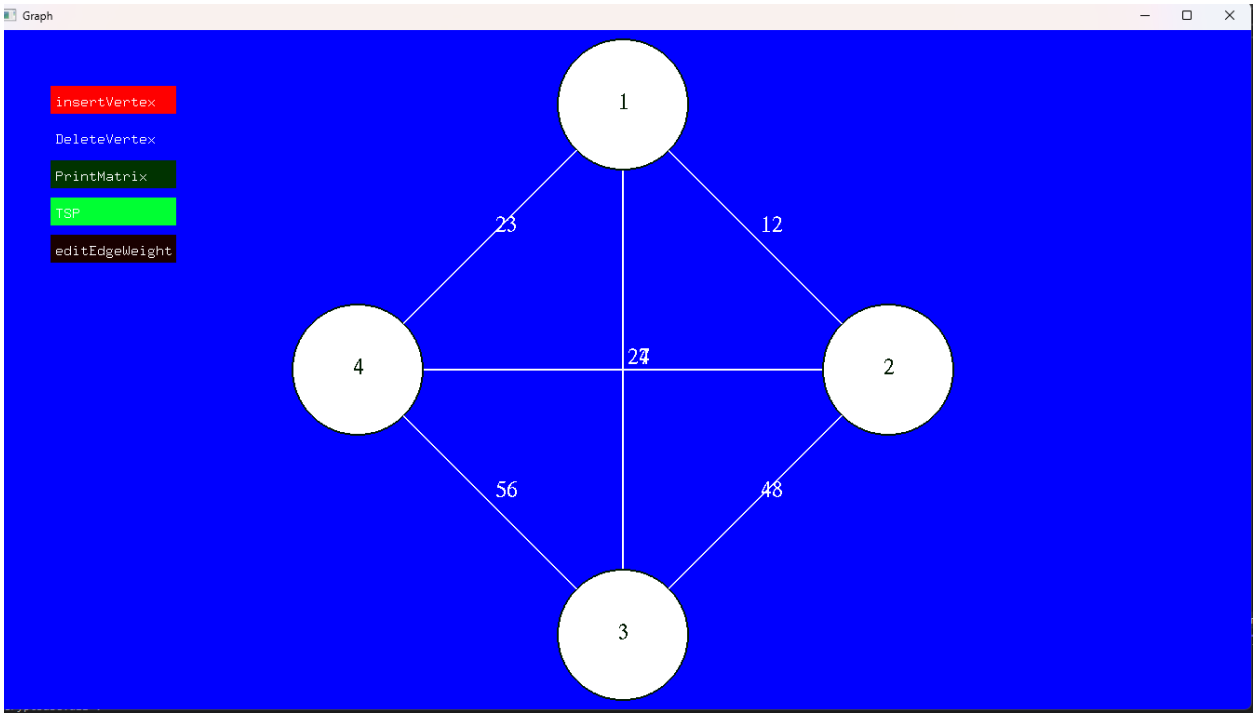
```
                                        cin >> targetVetrex;
                                        cout << endl;
                                        cout << "Вес ребра: ";
                                        cin >> Weight;
                                        cout << endl;
                                        int* targetVerPtr = &targetVetrex;
                                        graph.InsertEdge(sourceVertex, targetVetrex, Weight);
                                }


                        }
                        if (x > shift && x < shift + 135 && y > shift + 40 && y < shift + 70)
                        {
                                int sourceVertex;
                                int targetVertex;
                                int edgeWeight;

                                cout << "Удалить вершину >> "; cin >> sourceVertex; cout << endl;

                                graph.removeVertex(sourceVertex);
                                amountVerts--;


                        }
                        if (x > shift && x < shift + 135 && y >  shift + 80 && y < shift + 100)
                        {
                                graph.PrintMatrix();
                        }
                        if (x > shift && x < shift + 135 && y >  shift + 120 && y < shift + 140)
                        {
                                TSP(mat, n, help, result);
                        }
                        if (x > shift && x < shift + 135 && y >  shift + 160 && y < shift + 180)
                        {
                                int vertex, Weight, vertex1;
                                cout << "Введите номера вершин, между которыми нужно изменить вес ребра: ";
                                cin >> vertex;
                                cin >> vertex1;
                                cout << endl << endl;
                                cout << "Введите нужный вес: ";
                                cin >> Weight;
                                graph.editEdgeWeight(vertex, vertex1, Weight);
                        }
                }
                glutPostRedisplay();
}

void display()//Фунция вызова экрана и вызова функции отрисовки графа
{
                glShadeModel(GL_SMOOTH);
                glMatrixMode(GL_PROJECTION);
                glLoadIdentity();
                gluOrtho2D(0, WinW, 0, WinH);
                glViewport(0, 0, WinW, WinH);
                glClearColor(0.0, 0.0, 1.0, 0.0);
                glClear(GL_COLOR_BUFFER_BIT);
                graph.DrawGraph();
                drawMenu();
                glutSwapBuffers();
}

int main(int argc, char* argv[])
{
                setlocale(LC_ALL, "rus");
                system("chcp 1251>NULL");
                glutInit(&argc, argv);
```

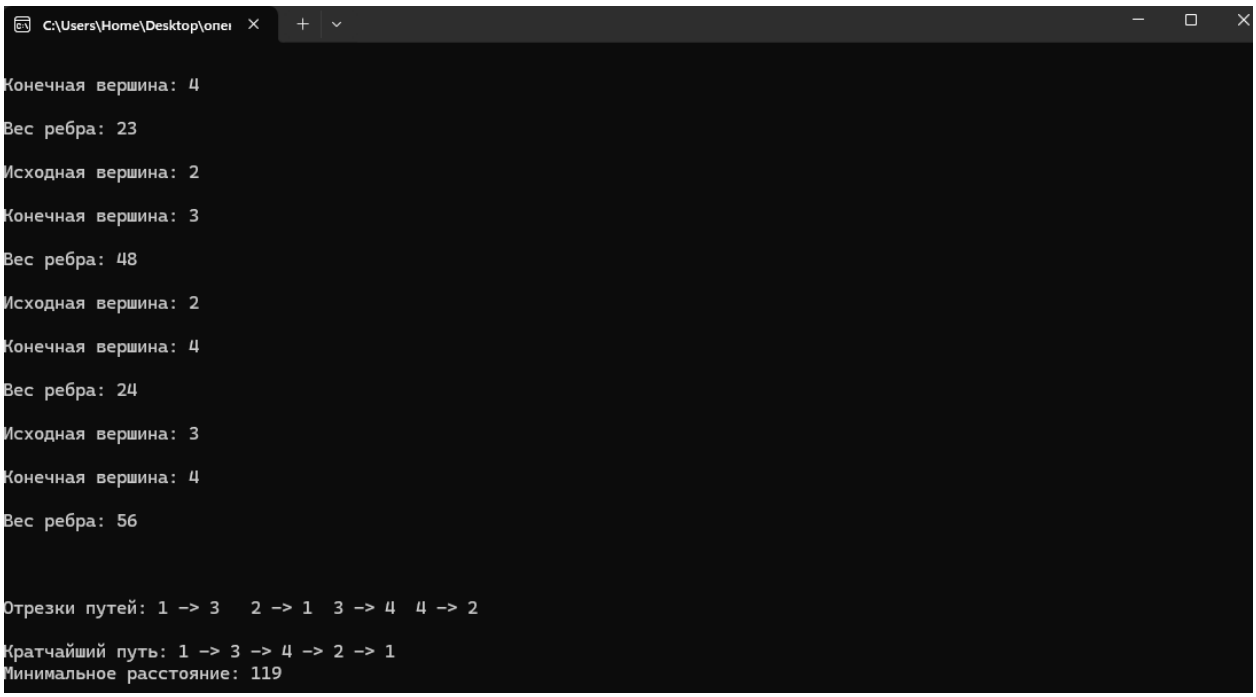
```
	int Verts, Edges, vertex, sourceVertex, targetVetrex, Weight;
	cout << "Введите количество вершин: " << endl;
	cin >> Verts;
	cout << "Введите количество ребер графа: " << endl;
	cin >> Edges;
	cout << endl;
	for (int i = 0; i < Verts; i++) {
		cout << "Вершина: ";
		cin >> vertex;
		graph.insertVertex(vertex);
		amountVerts++;
		cout << endl;
	}
	for (int i = 0; i < Edges; i++) {
		cout << "Исходная вершина: ";
		cin >> sourceVertex;
		cout << endl;
		cout << "Конечная вершина: ";
		cin >> targetVetrex;
		cout << endl;
		cout << "Вес ребра: ";
		cin >> Weight;
		cout << endl;
		int* targetVerPtr = &targetVetrex;
		graph.InsertEdge(sourceVertex, targetVetrex, Weight);
	}
	cout << endl;
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
	glutInitWindowSize(1350, 730);
	glutCreateWindow("Graph");
	WinW = glutGet(GLUT_WINDOW_WIDTH);
	WinH = glutGet(GLUT_WINDOW_HEIGHT);
	glLineWidth(2);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMouseFunc(mouseClick);
	glutMainLoop();
	return 0;
}
```

UML



Визуализация





Видеосопровождение

https://disk.yandex.ru/i/2CC9Z0l6WdC6Zw