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.

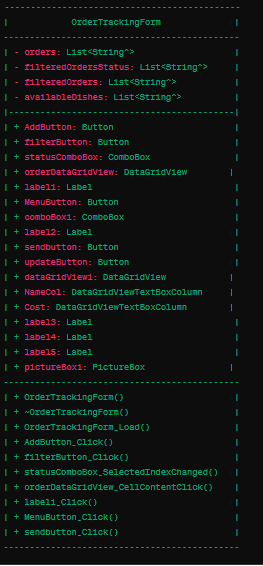
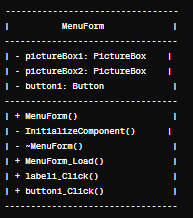
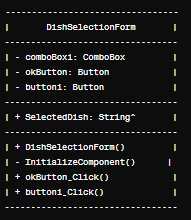
ПО:

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:

# **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.

**Код для коммивояжёра**

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;

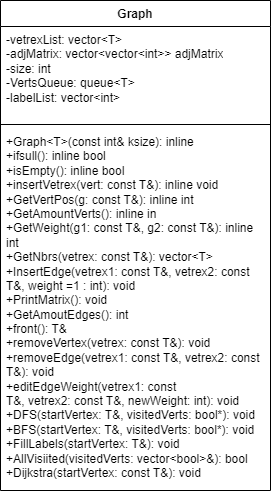
cout << endl;

}

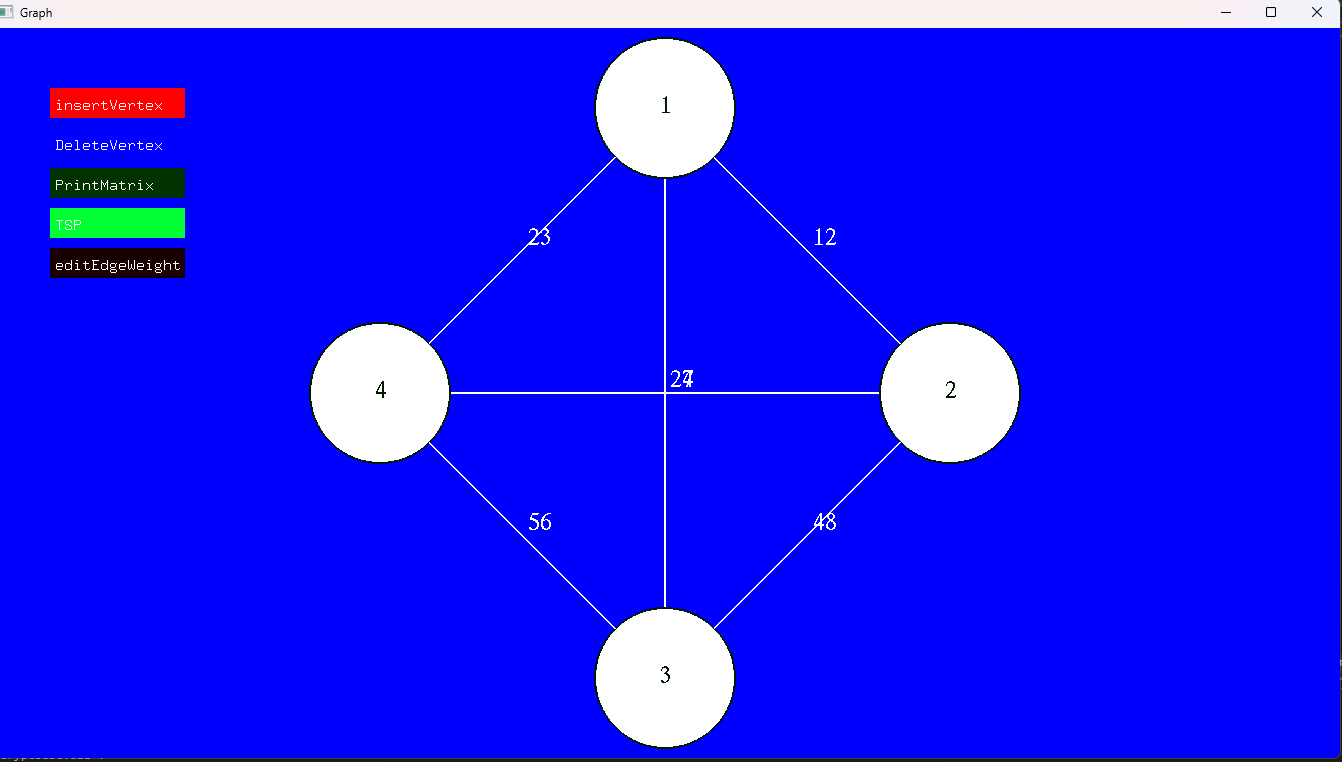
**Полный код**

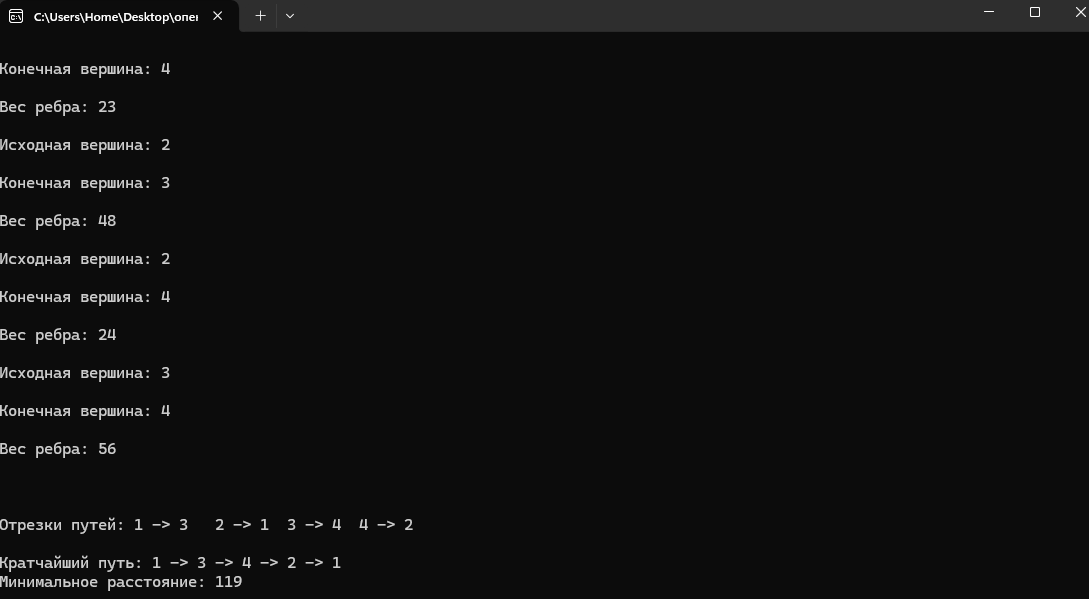
**https://yadi.sk/i/iBV0qT5GCAEa-g**

UML



**Визуализация**

****



**Видеосопровождение**

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