Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

По курсу “Логика и основы алгоритмизации в инженерных задачах”

на тему “Решение задачи коммивояжера”

Выполнил студент группы 20ВВ3:

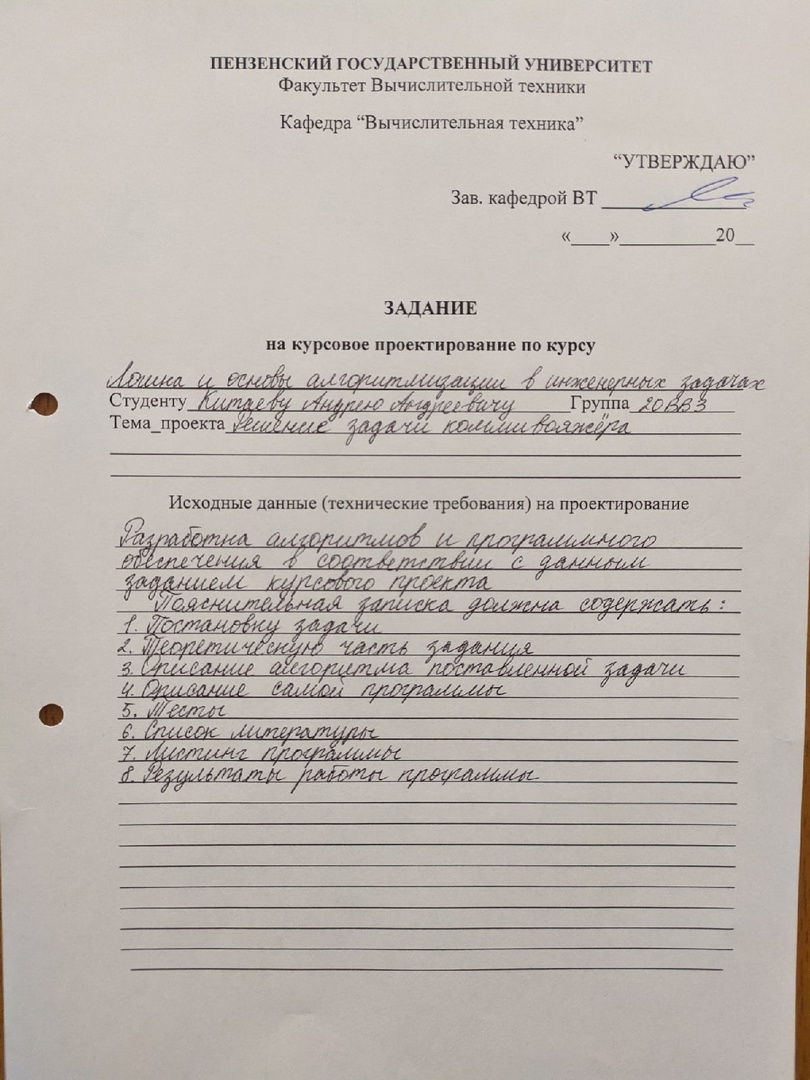
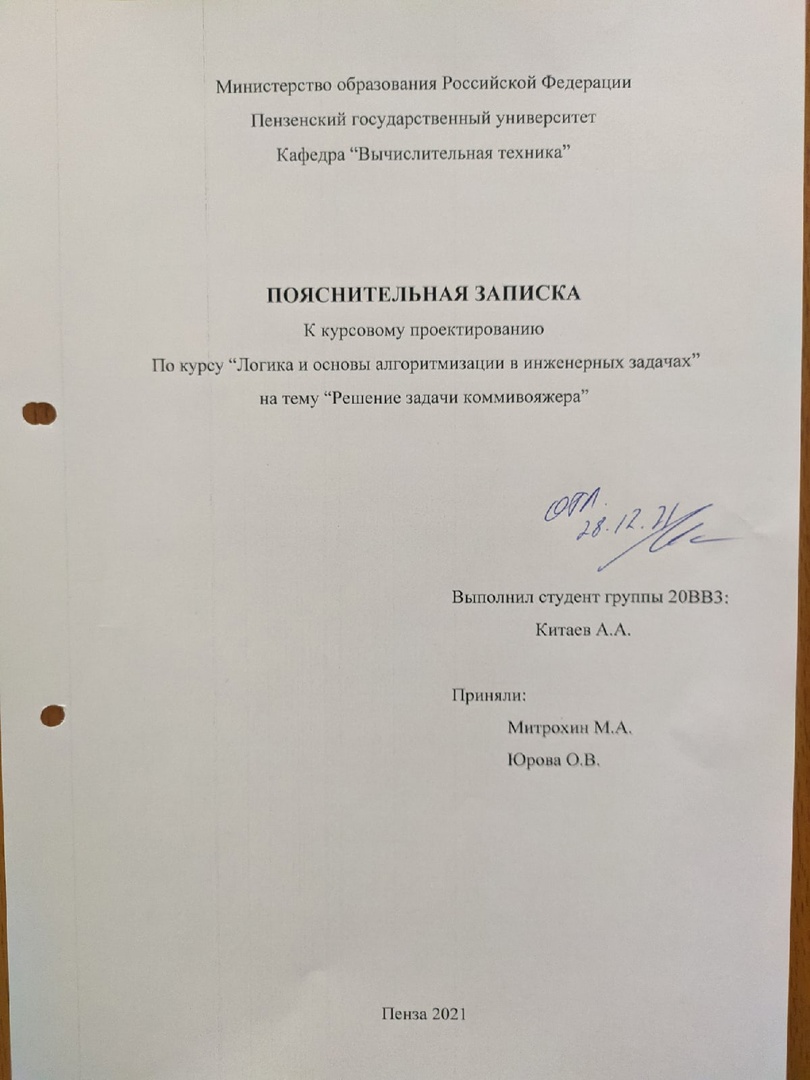
Китаев А.А.

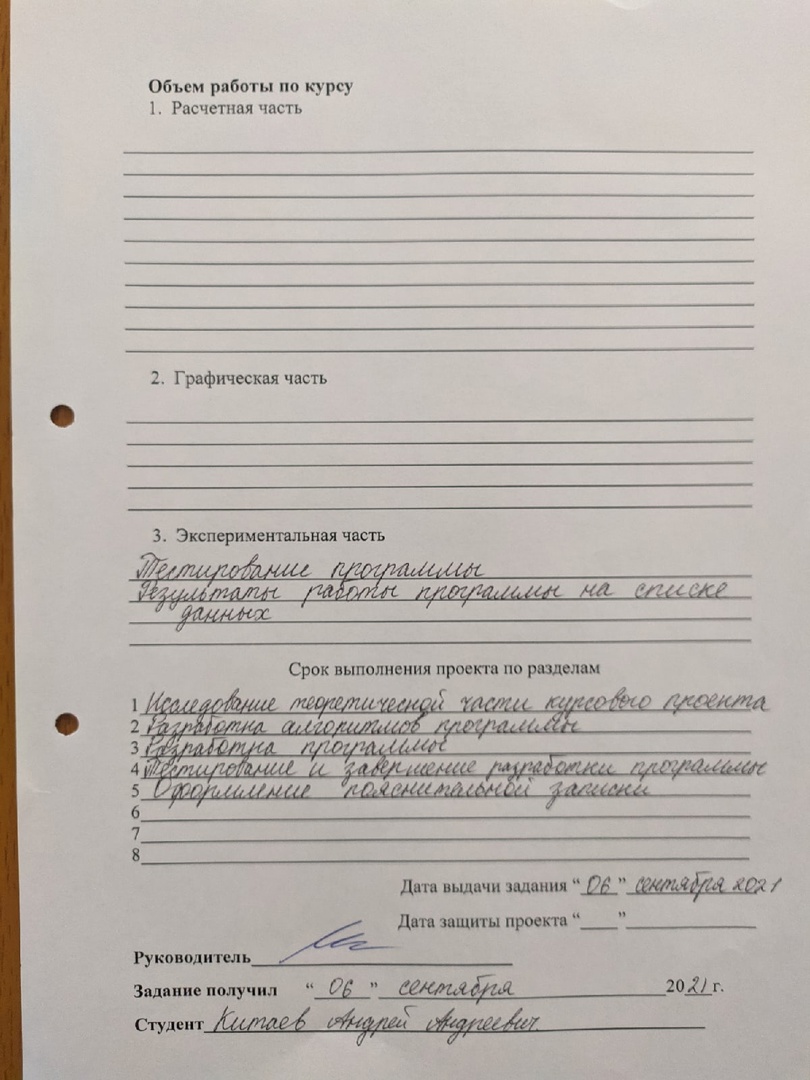
Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021

****



**Содержание**

Реферат 5

Введение 6

1.Постановка задачи 7

2.Теоретическая часть 8

3.Описание алгоритма программы 9

4. Описание программы 10

5. Тестирование программы 14

6. Заключение 15

Список литературы 16

Листинг 17

**Реферат**

Отчет 25 страниц ,8 рисунков

Задача коммивояжера. Поиск кратчайшего пути между городами.

Цель исследования – разработка программы, способной найти кратчайший путь между городами.

В работе используется метод ближайшего соседа, с его помощью можно найти кратчайший путь.

**Введение**

Задача [коммивояжёра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80)  — одна из самых известных задач [комбинаторной оптимизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), заключающаяся в поиске самого выгодного [маршрута](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)&action=edit&redlink=1), проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город.

В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешёвый, совокупный критерий и тому подобное) и соответствующие матрицы расстояний, стоимости и тому подобного. Как правило, указывается, что маршрут должен проходить через каждый город только один раз — в таком случае выбор осуществляется среди [гамильтоновых циклов](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%B8%D0%BB%D1%8C%D1%82%D0%BE%D0%BD%D0%BE%D0%B2_%D1%86%D0%B8%D0%BA%D0%BB). Существует несколько частных случаев общей постановки задачи, в частности, [геометрическая задача коммивояжёра](https://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0&action=edit&redlink=1) (также называемая планарной или евклидовой, когда матрица расстояний отражает расстояния между точками на плоскости), [метрическая задача коммивояжёра](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0&action=edit&redlink=1) (когда на матрице стоимостей выполняется [неравенство треугольника](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%80%D0%B0%D0%B2%D0%B5%D0%BD%D1%81%D1%82%D0%B2%D0%BE_%D1%82%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%B0)), [симметричная](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0&action=edit&redlink=1) и [асимметричная задачи коммивояжёра](https://ru.wikipedia.org/w/index.php?title=%D0%90%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0&action=edit&redlink=1). Также существует обобщение задачи, так называемая [обобщённая задача коммивояжёра](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D0%B2%D0%BE%D1%8F%D0%B6%D1%91%D1%80%D0%B0).

Метод Ближайшего Соседа в решении задачи Коммивояжера привлекает простотой алгоритма. Метод ближайшего соседа в исходной формулировке заключается в нахождении замкнутой кривой минимальной длины, соединяющей заданный набор точек на плоскости.

**1.Постановка задачи**

Требуется разработать программу, которая будет обходить все города по кратчайшему пути, и возвращаться обратно в город из которого начинался обход.

Программа должна содержать в себе:

-Текстовое или графическое меню

-Чтение данных из файла

-Алгоритм обхода городов

-Возможность сохранения результатов работы программы

**2.Теоретическая часть**

Первое что требуется для работы программы – чтение данных городов из файла. Для этого нужно открыть файл и считывать данные из него в динамический массив, пока не достигнем конца файла.

После этого, так как операцию обхода проводят 2 элемента, не пересекаясь друг с другом, то необходимо каким-то образом разделить список городов на 2. Мы решили разделить их по принципу: западнее или восточнее определённого меридиана. Для разделения списка достаточно сравнить значение долготы со значением разделения в цикле, сохраняя соответствующие элементы в одном файле, а остальные – в другом. Затем при помощи простого цикла мы нашли меридиан, который разделяет города с наименьшим суммарным расстоянием путей

Сначала нужно найти кратчайший путь для первого списка, затем для второго. Алгоритм будет для обоих случаев одинаковым. Вписываем первым городом тот город, из которого мы хотим начать обход. После чего сравниваем все расстояния путей, которые могут быть из этой точки в другую. Для вычисления расстояния между точками на поверхности земного шара можно использовать формулу, известную в сферической геометрии и геодезии:

S = 6371×arccos(sin (φ1×180°/ π) × sin (φ2×180°/ π) + cos (φ1×180°/ π) × cos (φ2×180°/ π) × cos (L2×180°/ π -L1×180°/ π)),

где S - расстояние, км;φ1 и L1 - широта и долгота точки 1 (для северной широты и восточной долготы со знаком плюс, для южной широты и западной долготы со знаком минус), градусы;φ2 и L2 - широта и долгота точки 2, градусы; константа 6371 – длина радиуса Земли, км.

Городам которые были уже посещены – присваиваем метку, чтобы больше к ним не возвращаться. В самом конце снова записываем город, из которого начался обход. Повторяем алгоритм для второго файла и сохраняем.

**3.Описание алгоритма программы**

Программа в самом начале считывает данные с файла, пока не дойдет до конца. Следующий шаг – поиск и удаление города “Великий Устюг”. Далее программа перебирает все города и сортирует их по двум файлам – на восток и запад.

Массив очищается и открывается первый файл. Вписывает в самое начало «Великий Устюг», перебирает все города и находит расстояние между Предыдущим городом. Если расстояние кратчайшее – расстояние сохраняется, а городу присваивается метка посещённости. Данный цикл повторяется, пока не закончатся все города в списке. После чего, в самый конец списка снова вводится «Великий Устюг».

В самом конце программа суммирует все расстояния и выводит результат.

**4.Описание программы**

Для написания данной программы использован язык программирования Си. Язык программирования Си - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

При запуске программы пользователь должен ввести название файла, в котором находятся данные городов. После чего происходит заполнение массива с указанного файла.

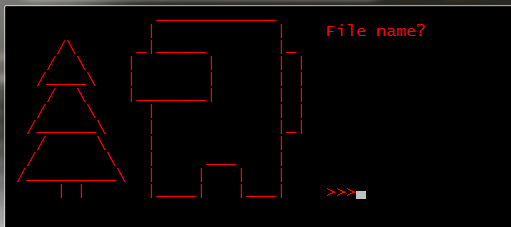


Рисунок 1-Ввод названия файла.

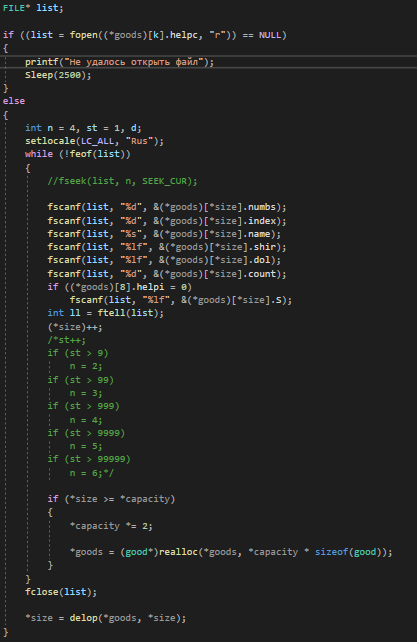


Рисунок 2- Код ввода названия файла.

Следующий этап – разделение данных на 2 части.

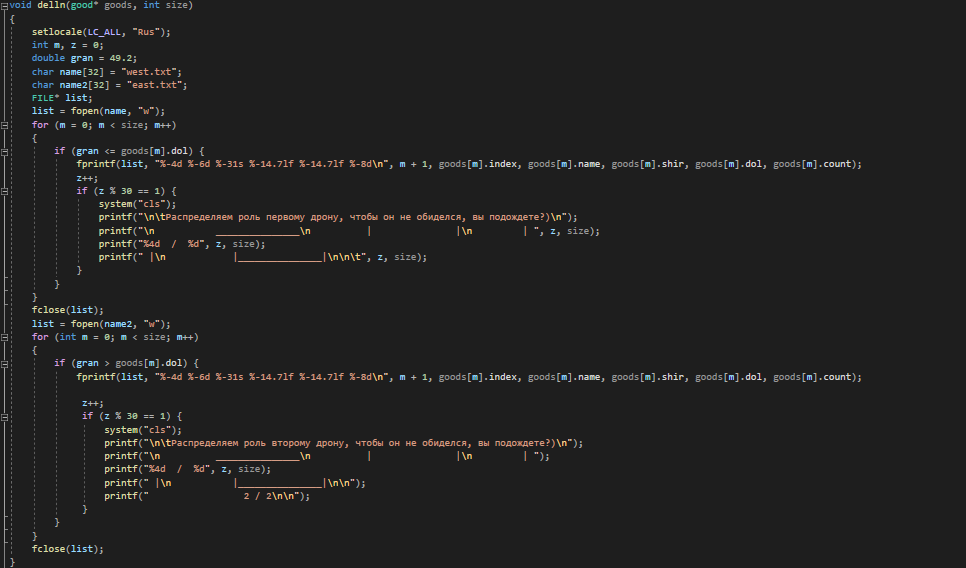


Рисунок 3- Код разделения списка на 2 части.

Далее – идет поиск кратчайших путей методом ближайшего соседа.

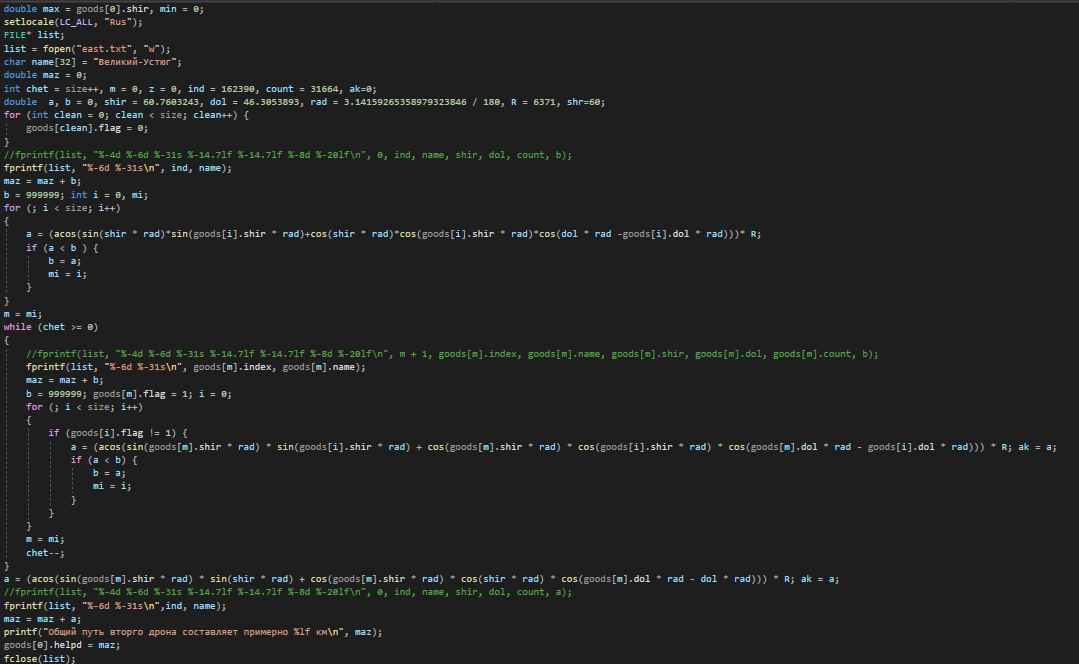


Рисунок 4-Поиск расстояний.

Вывод расстояния на экран и в файлы.

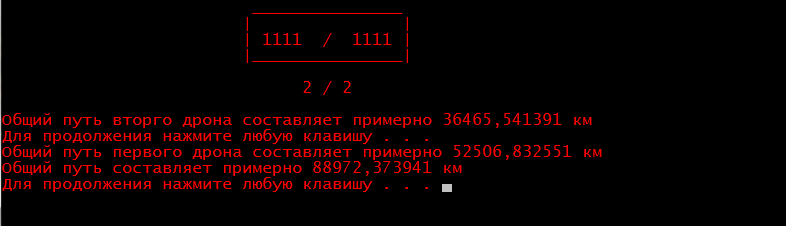


Рисунок 5- Вывод на экран.

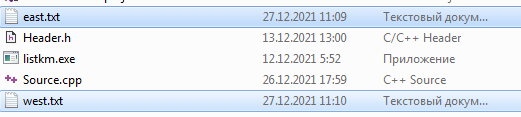


Рисунок 6- Вывод в файлы.

**5.Тестирование программы**

Среда разработки MicrosoftVisualStudio 2010 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки,

после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы:

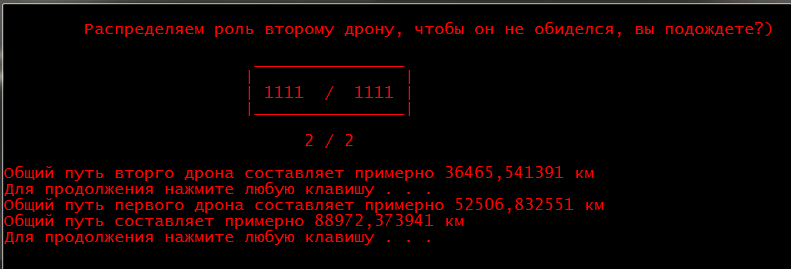


Рисунок 7 -Тестирование программы.

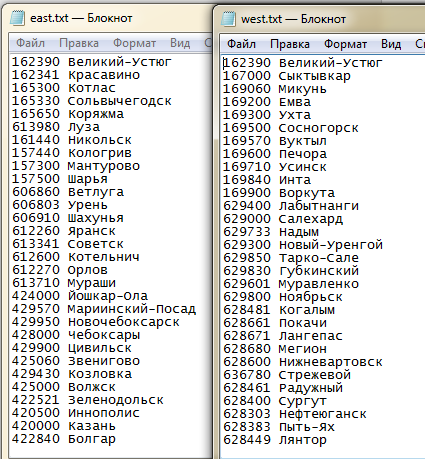


Рисунок 8-Результаты программы.

**Заключение**

Таким образом, в процессе создания данного проекта разработана программа, производящая решение задачи Коммивояжера методом ближайшего соседа.

При выполнении данной курсовой работы были получены навыки разработки программ и решения задач, связанных с нахождением путей между точками. Приобретены навыки по осуществлению алгоритма поиска кратчайших путей. Углублены знания языка программирования Cи.

Недостатком разработанной программы является примитивный

пользовательский интерфейс. Потому что программа работает в консольном

режиме, не добавляющем к сложности языка сложность программного

оконного интерфейса.

Программа имеет небольшой, но достаточный для использования

функционал возможностей.

**Список литературы**

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд — М.: Издательский дом «Вильямс», 2009. — ISBN 978-5-8459-0857-5.
2. Романовский И. В. Дискретный анализ: Учебное пособие для студентов, специализирующихся по прикладной математике и информатике. Изд. 3-е. — СПб.: Невский диалект, 2003. — 320 с. — ISBN 5-7940-0114-3.

**Листинг**

**Header.h**

#define \_CRT\_SECURE\_NO\_WARNINGS

#ifndef HEADER\_H

#define HEADER\_H

#include <string>

#include <iostream>

#include <vector>

#include <limits>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <iostream>

#include <string.h>

#include <conio.h>

#include <ctype.h>

#include <Windows.h>

#include <iostream>

#include <fstream>

#include "math.h"

#define CHARS ".,?\" \n"

typedef struct good {

int index, count, numbs, helpi, flag; // индексы

char name[32], helpc[20]; // название-описание товара

double shir, dol, S, helpd, helpdd; // цена товара

} good;

void open(good\*\* goods, int\* size, int\* capacity);

void openwest(good\*\* goods, int\* size, int\* capacity);

void openeast(good\*\* goods, int\* size, int\* capacity);

void searh(good\* goods, int size);

void sortwest(good\* goods, int size);

void sorteast(good\* goods, int size);

void dell(good\* goods, int size);

void delln(good\* goods, int size);

void print\_menuadm();

int delete\_goods(good\* goods, int size);

int dels(good\* goods, int size);

void zst();

#endif HEADER\_H

**Листинг**

**Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

int main()

{

setlocale(LC\_ALL, "Rus");

int size = 0; // количество элементов массива товаров

int capacity = 1; // ёмкость массива товаров

zst();

good\* goods = (good\*)malloc(capacity \* sizeof(good));

system("cls");

print\_menuadm(); // выводим меню на экран

int i = 0;

scanf("%s.txt", &goods[0].helpc);

//strcpy(goods[0].helpc, "city.txt");

size = dels(goods, size);

goods[10].helpi = 0;

open(&goods, &size, &capacity);

goods[10].helpi = 0;

searh(goods, size);

size = delete\_goods(goods, size);

delln(goods, size);

size = dels(goods, size);

openeast(&goods, &size, &capacity);

sorteast(goods, size);

size = dels(goods, size);

system("pause");

openwest(&goods, &size, &capacity);

sortwest(goods, size);

size = dels(goods, size);

system("pause");

}

void searh(good\* goods, int size)

{

int sear = 0;

setlocale(LC\_ALL, "Rus");

system("cls"); // очищаем экран

int sl = 162390;

int chet = 0;

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

{

if (goods[i].index == sl)

{

goods[11].helpi = i;

goods[11].helpd = goods[i].dol;

}

}

}

void delln(good\* goods, int size)

{

setlocale(LC\_ALL, "Rus");

int m, z = 0;

double gran = 49.2;

char name[32] = "west.txt";

char name2[32] = "east.txt";

FILE\* list;

list = fopen(name, "w");

for (m = 0; m < size; m++)

{

if (gran <= goods[m].dol) {

fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d\n", m + 1, goods[m].index, goods[m].name, goods[m].shir, goods[m].dol, goods[m].count);

z++;

if (z % 30 == 1) {

system("cls");

printf("\n\tРаспределяем роль первому дрону, чтобы он не обиделся, вы подождете?)\n");

printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n | |\n | ", z, size);

printf("%4d / %d", z, size);

printf(" |\n |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\n\t", z, size);

}

}

}

fclose(list);

list = fopen(name2, "w");

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

{

if (gran > goods[m].dol) {

fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d\n", m + 1, goods[m].index, goods[m].name, goods[m].shir, goods[m].dol, goods[m].count);

z++;

if (z % 30 == 1) {

system("cls");

printf("\n\tРаспределяем роль второму дрону, чтобы он не обиделся, вы подождете?)\n");

printf("\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n | |\n | ");

printf("%4d / %d", z, size);

printf(" |\n |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n\n");

printf(" 2 / 2\n\n");

}

}

}

fclose(list);

}

void sorteast(good\* goods, int size)

{

double max = goods[0].shir, min = 0;

setlocale(LC\_ALL, "Rus");

FILE\* list;

list = fopen("east.txt", "w");

char name[32] = "Великий-Устюг";

double maz = 0;

int chet = size++, m = 0, z = 0, ind = 162390, count = 31664, ak=0;

double a, b = 0, shir = 60.7603243, dol = 46.3053893, rad = 3.14159265358979323846 / 180, R = 6371, shr=60;

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

goods[clean].flag = 0;

}

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", 0, ind, name, shir, dol, count, b);

fprintf(list, "%-6d %-31s\n", ind, name);

maz = maz + b;

b = 999999; int i = 0, mi;

for (; i < size; i++)

{

a = (acos(sin(shir \* rad)\*sin(goods[i].shir \* rad)+cos(shir \* rad)\*cos(goods[i].shir \* rad)\*cos(dol \* rad -goods[i].dol \* rad)))\* R;

if (a < b ) {

b = a;

mi = i;

}

}

m = mi;

while (chet >= 0)

{

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", m + 1, goods[m].index, goods[m].name, goods[m].shir, goods[m].dol, goods[m].count, b);

fprintf(list, "%-6d %-31s\n", goods[m].index, goods[m].name);

maz = maz + b;

b = 999999; goods[m].flag = 1; i = 0;

for (; i < size; i++)

{

if (goods[i].flag != 1) {

a = (acos(sin(goods[m].shir \* rad) \* sin(goods[i].shir \* rad) + cos(goods[m].shir \* rad) \* cos(goods[i].shir \* rad) \* cos(goods[m].dol \* rad - goods[i].dol \* rad))) \* R; ak = a;

if (a < b) {

b = a;

mi = i;

}

}

}

m = mi;

chet--;

}

a = (acos(sin(goods[m].shir \* rad) \* sin(shir \* rad) + cos(goods[m].shir \* rad) \* cos(shir \* rad) \* cos(goods[m].dol \* rad - dol \* rad))) \* R; ak = a;

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", 0, ind, name, shir, dol, count, a);

fprintf(list, "%-6d %-31s\n",ind, name);

maz = maz + a;

printf("Общий путь вторго дрона составляет примерно %lf км\n", maz);

goods[0].helpd = maz;

fclose(list);

}

void sortwest(good\* goods, int size)

{

double max = goods[0].shir, min = 0;

setlocale(LC\_ALL, "Rus");

FILE\* list;

list = fopen("west.txt", "w");

char name[32] = "Великий-Устюг";

double maz = 0;

int chet = size++, m = 0, z = 0, ind = 162390, count = 31664, ak=0;

double a, b = 0, shir = 60.7603243, dol = 46.3053893, rad = 3.14159265358979323846 / 180, R = 6371, shr = 60;

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

goods[clean].flag = 0;

}

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", 0, ind, name, shir, dol, count, b);

fprintf(list,"%-6d %-31s\n", ind, name);

maz = maz + b;

b = 999999;

int i = 0, mi;

for (; i < size; i++)

{

a = (acos(sin(shir \* rad) \* sin(goods[i].shir \* rad) + cos(shir \* rad) \* cos(goods[i].shir \* rad) \* cos(dol \* rad - goods[i].dol \* rad))) \* R;

if (a < b) {

b = a;

mi = i;

}

}

m = mi;

while (chet >= 0)

{

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", m + 1, goods[m].index, goods[m].name, goods[m].shir, goods[m].dol, goods[m].count, b);

fprintf(list, "%-6d %-31s\n", goods[m].index, goods[m].name);

maz = maz + b;

b = 999999; goods[m].flag = 1; i = 0;

for (; i < size; i++)

{

if (goods[i].flag != 1) {

a = (acos(sin(goods[m].shir \* rad) \* sin(goods[i].shir \* rad) + cos(goods[m].shir \* rad) \* cos(goods[i].shir \* rad) \* cos(goods[m].dol \* rad - goods[i].dol \* rad))) \* R;

if (a < b) {

b = a;

mi = i;

}

}

}

m = mi;

chet--;

}

a = (acos(sin(goods[m].shir \* rad) \* sin(shir \* rad) + cos(goods[m].shir \* rad) \* cos(shir \* rad) \* cos(goods[m].dol \* rad - dol \* rad))) \* R;

//fprintf(list, "%-4d %-6d %-31s %-14.7lf %-14.7lf %-8d %-20lf\n", 0, ind, name, shir, dol, count, a);

fprintf(list, "%-6d %-31s\n", ind, name);

maz = maz + a;

goods[0].helpd = goods[0].helpd + maz;

printf("Общий путь первого дрона составляет примерно %lf км\n", maz);

printf("Общий путь составляет примерно %lf км\n", goods[0].helpd);

fclose(list);

}

void zst()

{

system("cls");

printf(" \_ \_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_ \_ \n");

printf(" | | | | | \_ | | \_\_ \\ | \_\_ \\ \\ \\ / / \n");

printf(" | |\_| | | |\_| | | |\_\_| | | |\_\_| | \\ \\/ / \n");

printf(" | \_ | | \_ | | \_\_\_/ | \_\_\_/ \\ / \n");

printf(" | | | | | | | | | | | | / / \n");

printf(" |\_| |\_| |\_| |\_| |\_| |\_| /\_/ \n"); Sleep(450);

printf(" \_ \_ \_\_\_\_\_\_ \_ \_ \n");

printf(" | \\ | | | \_\_\_\_| | | | | \n");

printf(" | \\| | | |\_\_\_\_ | | \_ | | \n");

printf(" | |\\ | | \_\_\_\_| | |\_/ \\\_| | \n");

printf(" | | \\ | | |\_\_\_\_ | \_ | \n");

printf(" |\_| \\\_| |\_\_\_\_\_\_| \\\_\_/ \\\_\_/ \n"); Sleep(450);

printf(" \_ \_ \_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \n");

printf(" \\ \\ / / | \_\_\_\_| | \_ | | \_\_ \\ \n");

printf(" \\ \\/ / | |\_\_\_\_ | |\_| | | |\_\_| | \n");

printf(" \\ / | \_\_\_\_| | \_ | | \_ / \n");

printf(" / / | |\_\_\_\_ | | | | | | \\ \\ \n");

printf(" /\_/ |\_\_\_\_\_\_| |\_| |\_| |\_| |\_| \n"); Sleep(550);

}

void print\_menuadm() {

system("cls"); // очищаем экран

printf(" \_\_\_\_\_\_\_\_\_\_\_\_ \n");

printf(" | | File name? \n");

printf(" /\\ \_|\_\_\_\_\_ |\_ \n");

printf(" / \\ | | | | \n");

printf(" /\_\_\_\_\\ | | | | \n");

printf(" / \\ |\_\_\_\_\_\_\_| | | \n");

printf(" / \\ | | | \n");

printf(" /\_\_\_\_\_\_\\ | |\_| \n");

printf(" / \\ | | \n");

printf(" / \\ | \_\_\_ | \n");

printf(" /\_\_\_\_\_\_\_\_\_\\ | | | | \n");

printf(" | | |\_\_\_\_| |\_\_\_| >>>");

}

int delete\_goods(good\* goods, int size)

{

setlocale(LC\_ALL, "Rus");

if (goods[10].helpi != 0) {

if (size == 0)

printf("| ERROR! NO Goods |\n");

else {

int bruh;

printf("Товар который вы хотите удалить: ");

scanf("%d", &bruh);

printf("\n");

for (int i = bruh - 1; i + 1 < size; i++)

{

memcpy(goods[i].name, goods[i + 1].name, 20);

goods[i].shir = goods[i + 1].shir;

goods[i].dol = goods[i + 1].dol;

goods[i].count = goods[i + 1].count;

goods[i].index = goods[i + 1].index;

}

size--;

printf("Товар |%d| успешно удален!", bruh); Sleep(1650);

}

}

else {

for (int i = goods[11].helpi; i + 1 < size; i++)

{

memcpy(goods[i].name, goods[i + 1].name, 32);

goods[i].shir = goods[i + 1].shir;

goods[i].dol = goods[i + 1].dol;

goods[i].count = goods[i + 1].count;

goods[i].index = goods[i + 1].index;

}

size--;

}

return(size);

}

int dels(good\* goods, int size)

{

setlocale(LC\_ALL, "Rus");

while (size != 0)

{

int bruh = 1;

for (int i = bruh - 1; i + 1 < size; i++)

{

memcpy(goods[i].name, goods[i + 1].name, 20);

goods[i].index = goods[i + 1].index;

goods[i].shir = goods[i + 1].shir;

goods[i].dol = goods[i + 1].dol;

goods[i].count = goods[i + 1].count;

}

size--;

}

return(size);

}

int delop(good\* goods, int size) {

int bruh = size;

for (int i = bruh - 1; i + 1 < size; i++)

{

memcpy(goods[i].name, goods[i + 1].name, 20);

goods[i].shir = goods[i + 1].shir;

goods[i].dol = goods[i + 1].dol;

goods[i].count = goods[i + 1].count;

}

size--;

return(size);

printf("Файл загружен!");

Sleep(1000); system("cls");

}

void open(good\*\* goods, int\* size, int\* capacity)

{

setlocale(LC\_ALL, "Rus");

int k = 0;

if ((\*goods)[10].helpi != 0 && (\*goods)[10].helpi != 1) {

printf("\nВведите названия файла, который хотите открыть\n>>>>");

scanf("%s.txt", &(\*goods)[0].helpc);

}

else

k = (\*goods)[10].helpi;

FILE\* list;

if ((list = fopen((\*goods)[k].helpc, "r")) == NULL)

{

printf("Не удалось открыть файл");

Sleep(2500);

}

else

{

int n = 4, st = 1, d;

setlocale(LC\_ALL, "Rus");

while (!feof(list))

{

//fseek(list, n, SEEK\_CUR);

fscanf(list, "%d", &(\*goods)[\*size].numbs);

fscanf(list, "%d", &(\*goods)[\*size].index);

fscanf(list, "%s", &(\*goods)[\*size].name);

fscanf(list, "%lf", &(\*goods)[\*size].shir);

fscanf(list, "%lf", &(\*goods)[\*size].dol);

fscanf(list, "%d", &(\*goods)[\*size].count);

if ((\*goods)[8].helpi = 0)

fscanf(list, "%lf", &(\*goods)[\*size].S);

int ll = ftell(list);

(\*size)++;

/\*st++;

if (st > 9)

n = 2;

if (st > 99)

n = 3;

if (st > 999)

n = 4;

if (st > 9999)

n = 5;

if (st > 99999)

n = 6;\*/

if (\*size >= \*capacity)

{

\*capacity \*= 2;

\*goods = (good\*)realloc(\*goods, \*capacity \* sizeof(good));

}

}

fclose(list);

\*size = delop(\*goods, \*size);

}

}

void openwest(good\*\* goods, int\* size, int\* capacity)

{

setlocale(LC\_ALL, "Rus");

int k = 0;

FILE\* list;

char name[32] = "west.txt";

if ((list = fopen(name, "r")) == NULL)

{

printf("Не удалось открыть файл");

Sleep(2500);

}

else

{

int n = 4, st = 1, d;

setlocale(LC\_ALL, "Rus");

while (!feof(list))

{

//fseek(list, n, SEEK\_CUR);

fscanf(list, "%d", &(\*goods)[\*size].numbs);

fscanf(list, "%d", &(\*goods)[\*size].index);

fscanf(list, "%s", &(\*goods)[\*size].name);

fscanf(list, "%lf", &(\*goods)[\*size].shir);

fscanf(list, "%lf", &(\*goods)[\*size].dol);

fscanf(list, "%d", &(\*goods)[\*size].count);

//fscanf(list, "%lf", &(\*goods)[\*size].S);

int ll = ftell(list);

(\*size)++;

if (\*size >= \*capacity)

{

\*capacity \*= 2;

\*goods = (good\*)realloc(\*goods, \*capacity \* sizeof(good));

}

}

fclose(list);

\*size = delop(\*goods, \*size);

}

}

void openeast(good\*\* goods, int\* size, int\* capacity)

{

setlocale(LC\_ALL, "Rus");

int k = 0;

FILE\* list;

char name[32] = "east.txt";

if ((list = fopen(name, "r")) == NULL)

{

printf("Не удалось открыть файл");

Sleep(2500);

}

else

{

int n = 4, st = 1, d;

setlocale(LC\_ALL, "Rus");

while (!feof(list))

{

//fseek(list, n, SEEK\_CUR);

fscanf(list, "%d", &(\*goods)[\*size].numbs);

fscanf(list, "%d", &(\*goods)[\*size].index);

fscanf(list, "%s", &(\*goods)[\*size].name);

fscanf(list, "%lf", &(\*goods)[\*size].shir);

fscanf(list, "%lf", &(\*goods)[\*size].dol);

fscanf(list, "%d", &(\*goods)[\*size].count);

//fscanf(list, "%lf", &(\*goods)[\*size].S);

int ll = ftell(list);

(\*size)++;

if (\*size >= \*capacity)

{

\*capacity \*= 2;

\*goods = (good\*)realloc(\*goods, \*capacity \* sizeof(good));

}

}

fclose(list);

\*size = delop(\*goods, \*size);

}

}