**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе работе №9 на тему:   
**«Нахождение кратчайшего маршрута»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-114,Константинов А.Д.

«\_\_» \_\_\_\_\_\_\_\_ 2022 г. \_\_\_\_\_\_\_\_\_\_\_/А.Д. Константинов/

Принял: к.ф.-м.н., доцент, И.А. Моисеев

« » ноября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание:**

[Цель работы: 3](#_Toc114692552)

[Листинг кода: 3](#_Toc114692553)

[Результат работы программы: 8](#_Toc114692554)

[Вывод: 8](#_Toc114692555)

[Приложение: 22](#_Toc114692556)

Цель работы:

Найти кратчайший маршрут.

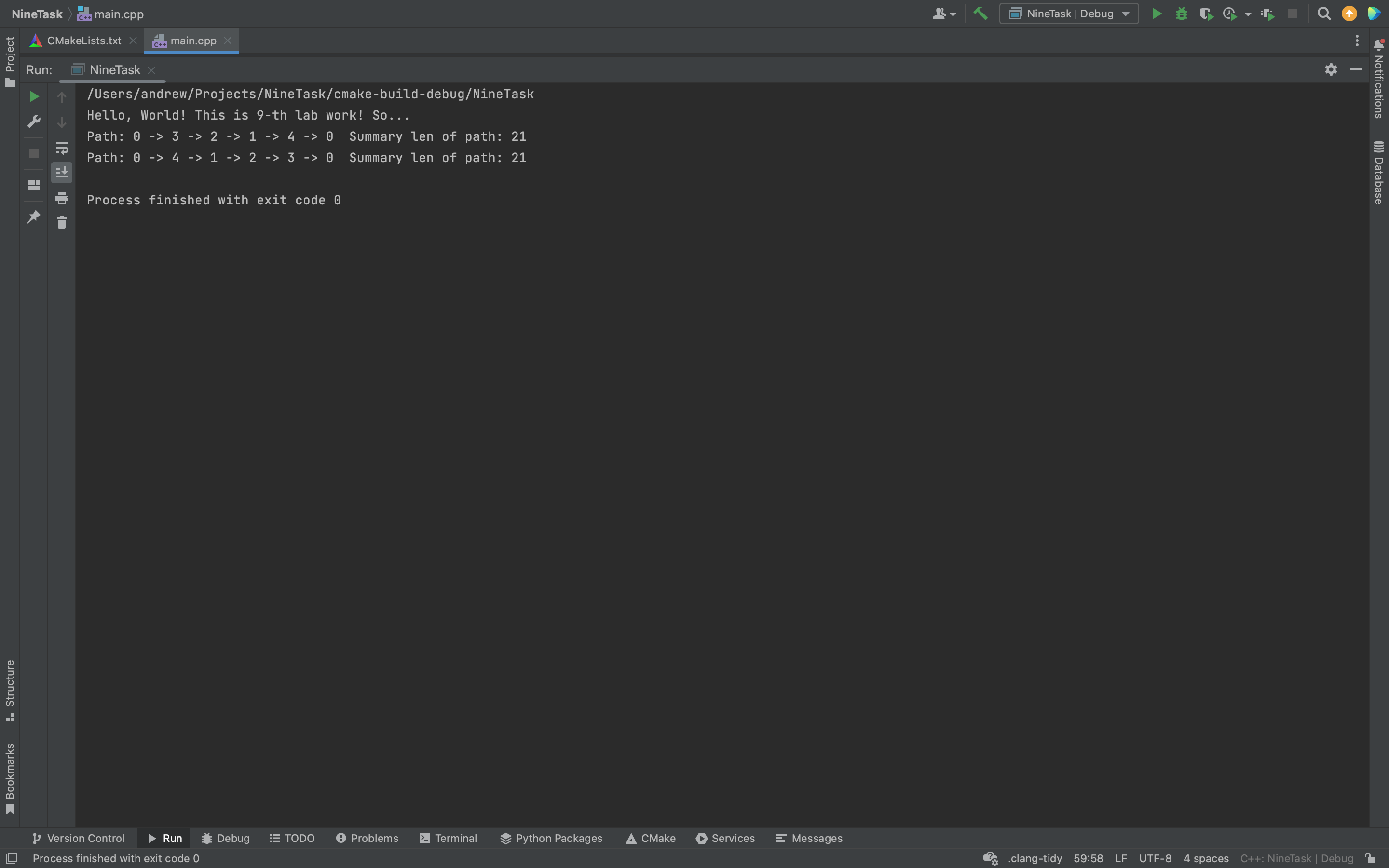
**Задание:**

# Известны расстояния от почтамта до районных отделений связи и расстояния между отделениями связи. Почта поступает сначала на почтамт, а потом развозится по отделениям связи. Составьте маршрут машины, развозящей почту от почтамта до каждого отделения связи, чтобы ее путь был наименьшим, при этом она должна вернуться на почтамт.

# Листинг кода:

#include <iostream>  
#include "vector"  
using namespace std;  
struct Path{  
 int len;  
 string path="";  
};  
void Deikstra(vector<vector<int>>\* graph, vector<int> used,int len, vector<Path>\* path){  
 if(used.size() == 5)  
 {  
 int outPoint = used[used.size()-1];  
 auto array = (\*graph)[outPoint];  
 len+=array[0];  
 used.push\_back(0);  
 Path\* p = new Path;  
 p->len = len;  
  
 string path\_h = "";  
 for (int i = 0; i < 6; ++i) {  
 path\_h+=(to\_string(used[i]));  
 if(i!=5) path\_h+=" -> ";  
 }  
 p->path = path\_h;  
 path->push\_back(\*p);  
 return;  
 }  
 int outPoint = used[used.size()-1];  
 auto array = (\*graph)[outPoint];  
 for (int i = 0; i < 5; ++i) {  
 bool propusk = false;  
 for (int j = 0; j < used.size(); ++j) {  
 if(used[j] == i)  
 {propusk=true;break;}  
 }  
 if(array[i] != 0 && !propusk)  
 {  
 auto add = used;  
 add.push\_back(i);  
 Deikstra(graph,add,len+array[i],path);  
 }  
 }  
}  
  
int main() {  
 vector<vector<int>>graph;  
 vector<int> l0 = **{**0,5,3,4,4**}**;  
 vector<int> l1 = **{**5,0,3,5,6**}**;  
 vector<int> l2 = **{**3,3,0,4,7**}**;  
 vector<int> l3 = **{**4,5,4,0,7**}**;  
 vector<int> l4 = **{**4,6,7,7,0**}**;  
 graph.push\_back(l0);  
 graph.push\_back(l1);  
 graph.push\_back(l2);  
 graph.push\_back(l3);  
 graph.push\_back(l4);  
 vector<Path> paths;  
  
 Deikstra(&graph,vector<int> **{**0**}**,0,&paths);  
 std::cout << "Hello, World! This is 9-th lab work! So..." << std::endl;  
 int minLen = 99999;  
  
 for (int i = 0; i < paths.size(); ++i) {  
 if(paths[i].len < minLen) minLen = paths[i].len;  
 }  
 for (int i = 0; i < paths.size(); ++i) {  
 if(paths[i].len == minLen)  
 cout << "Path: " << paths[i].path << " Summary len of path: " << paths[i].len << endl;  
 }  
  
 graph.clear();  
 return 0;  
}

# Результат работы программы:



# Вывод:

В ходе выполнения практической работы была написана программа, находящая кратчайший путь.