|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий |
| Кафедра вычислительной техники |
|  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3** | |
| **по дисциплине** | |
| *«Структуры и алгоритмы обработки данных»* | |
| Выполнил студент группы *ИКБО-02-17* | *Политов А.Ю* |
| Принял | *Асадова Ю.С* |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Подпись студента* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г.. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Подпись руководителя* |

Москва 2019

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт информационных технологий

Кафедра вычислительной техники

Выполнено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/А.Ю. Политов/

Зачтено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ю.С. Асадова/

**Задание на лабораторную работу**

**Дисциплина**: **«**Структуры и алгоритмы обработки данных**»**

**Студент** Политов А.Ю. **Шифр** 17И0325 **Группа** ИКБО-02-17

**1. Тема**: «Графовые структуры данных»

**2. Срок сдачи студентом законченной работы:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г.

**3. Исходные данные:** граф (числа, разделенные пробелами)

**4. Задание:** По системе двусторонних дорог определить, можно ли, построив какие-нибудь три дороги, чтоб расстояние между городами было меньше 100км.

**5. Содержание отчета:**

* титульный лист;
* задание;
* оглавление;
* введение;
* основные разделы отчета;
* заключение;
* список использованных источников;

Руководитель работы Ю.С. Асадова \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2018г.

подпись

Задание принял к исполнению А.Ю Политов \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018г.

подпись

*Оглавление*

Введение…………………………………………………………………………… 4

1. Теоретическое введение …………………………………………………… 5
2. Постановка задачи………………………………………………………….. 6
3. Определение входных и выходных данных………………………………. 6
4. Краткое описание программы …………………………………………….. 7
5. Разработка программы (код) ……………………………………………… 7
6. Разработка тестов …………………………………………………………. 9

Заключение ……………………………………………………………………….. 9

Список литературы ……………………………………………………………….10

*Введение*

Целью данной лабораторной работы является ознакомление с графовыми структурами данных, получение умений реализации типовых операций над ними, выработка навыков применения графовых структур в задачах. Лабораторная работа была выполнена в среде разработки MS Visual Studio на языке С++.

1. ***Теоретическое введение.***

Граф, или неориентированный граф G — это упорядоченная пара {G: = (V, E)}, где V— непустое множество вершин или узлов, а E — множество пар (в случае неориентированного графа — неупорядоченных) вершин, называемых рёбрами.

V (а значит и, E, иначе оно было бы мультимножеством) обычно считаются конечными множествами. Многие результаты, полученные для конечных графов, неверны (или каким-либо образом отличаются) для бесконечных графов, поскольку не все утверждения, имеющие место для конечных совокупностей, выполняются в случае бесконечных множеств.

Ориентированный граф (сокращённо орграф) G — это упорядоченная пара {G: = (V, A)} V — непустое множество вершин или узлов, и A — множество (упорядоченных) пар различных вершин, называемых дугами, или ориентированными рёбрами.

Пусть (v, w) — это дуга. Тогда вершину v называют её началом, а w — концом. Можно сказать, что дуга v к w ведёт от вершины v к вершине w.

Маршрутом в графе называют конечную последовательность вершин, в которой каждая вершина (кроме последней) соединена со следующей в последовательности вершиной ребром. Цепью называется маршрут без повторяющихся рёбер. Простой цепью называется маршрут без повторяющихся вершин (откуда следует, что в простой цепи нет повторяющихся рёбер).

Ориентированным маршрутом (или путём) в орграфе называют конечную последовательность вершин и дуг, в которой каждый элемент инцидентен предыдущему и последующему.

Циклом называют цепь, в которой первая и последняя вершины совпадают.

Путь (или цикл) называют простым, если рёбра в нём не повторяются; элементарным, если он простой и вершины в нём не повторяются.

1. ***Постановка задачи.***

По системе двусторонних дорог определить, можно ли, построив какие-нибудь три дороги, чтоб расстояние между городами было меньше 100км.

**Дано:** граф (числа, разделенные пробелами)

**Ограничения на данные:** вес ребра неотрицательное число

**Ограничения на решаемую задачу**: нет

1. ***Определение входных и выходных данных.***

**Входные данные** – числа, вводится с клавиатуры.

**Результат** – кратчайший пути (числа), каркас найденного пути (массив).

1. ***Краткое описание программы.***

На вход программы подается количество вершин дерева. Далее пользователь вводит расстояния между городами. После алгоритм пытается оптимизировать полученный граф, а далее программа выводит результат.

1. ***Разработка программы (код).***

#include "iostream"

#include "Windows.h"

#include <vector>

#include "string"

using namespace std;

bool tryParse(int& num) {

num = 0;

string str;

getline(cin, str);

if (str.length() > 7)

return false;

else {

if (str[0] != '-' && (str[0] < '0' || str[0] > '9')) return false;

for (size\_t index = 1; index < str.length(); index++)

if (str[index] < '0' || str[index] > '9')

return false;

num = atoi(str.c\_str());

return true;

}

}

void addPoint(vector<vector<int>> &graph, vector<int> &link) {

vector<int> temp;

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

temp.push\_back(link.at(i));

}

graph.push\_back(temp);

for (int i = 0; i < graph.size() - 1; i++) {

graph.at(i).push\_back(link.at(i));

}

graph.at(graph.size() - 1).push\_back(0);

}

void show(vector<vector<int>> graph) {

cout << "Граф: " << "\t";

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

cout << i + 1 << ":\t";

}

cout << endl << endl;

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

cout << i + 1 << ":\t";

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

cout << graph.at(i).at(j) << "\t";

}

cout << endl<< endl;

}

}

void optiFine(vector<vector<int>> &graph) {

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

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

for (int k = 0; k < graph.size(); k++) {

if (graph.at(i).at(k) == -1 && graph.at(i).at(j) > 0 && graph.at(j).at(k) > 0)

graph.at(i).at(k) = graph.at(i).at(j) + graph.at(j).at(k);

if (graph.at(i).at(j) > 0 && graph.at(j).at(k) > 0 && graph.at(i).at(k) > 0 && graph.at(i).at(j) + graph.at(j).at(k) < graph.at(i).at(k))

graph.at(i).at(k) = graph.at(i).at(j) + graph.at(j).at(k);

}

}

}

}

void check(vector<vector<int>> &graph) {

int temp = 0;

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

for (int j = i + 1; j < graph.size(); j++) {

if (graph.at(i).at(j) > 100)

temp++;

}

}

cout << ((temp > 3) ? "Нельзя оптимизировать дороги" : "Можно оптимизировать дороги");

cout << endl;

}

void add(vector<vector<int>> &graph) {

vector<int> link;

link.clear();

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

cout << i + 1 << ": ";

int temp;

if (!tryParse(temp)) i--;

else link.push\_back(temp);

}

addPoint(graph, link);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<vector<int>> graph;

add(graph);

int comand = 10;

cout << "1 - Добавление города" << endl;

cout << "2 - Вывод графа" << endl;

cout << "3 - Проверка на возможность оптимизации" << endl;

cout << "0 - Выход из программы" << endl;

while (comand != 0) {

cout << "Введите команду: ";

while (!tryParse(comand));

switch (comand) {

case 1:

add(graph);

break;

case 2:

show(graph);

break;

case 3:

optiFine(graph);

check(graph);

show(graph);

break;

case 0:

break;

}

}

system("pause");

}

1. ***Разработка тестов***

В таблице 2 представленный сведения о тестировании программы.

Таблица 2 – Тестирование программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  теста | Входные данные | Эталон результата | Результат программы | Результат анализа  теста |
| word |  |  |  |
| 1 | 0 4 23 -1  4 0 104 23  23 104 0 105  -1 23 105 0 | можно | Можно | + |

На рисунке 1 показана работа программы для данных, указанных в таблице 2.

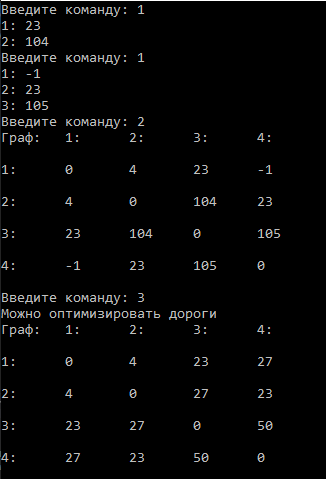


Рисунок 1 - Результат выполнения теста

***Заключение***

В ходе выполнения работы были получены практические навыки при работе с графовыми структурами данных, получены умения реализации типовых операций с ними, приобретены навыки применения графовых структур в задачах на языке С++.

***Список литературы***

1. «qaru.site» — информационный сервис для разработчиков [Электронный ресурс]. URL: <http://qaru.site> (дата обращения: 1.05.2019)
2. «cppstudio» — информационный сервис для разработчиков [Электронный ресурс]. URL: <http://cppstudio.com> (дата обращения: 1.05.2019)
3. [Кей Хорстманн, Гари Корнелл «С++. Библиотека профессионала. Том 1».10-е издание (2016](https://t.me/progbook/168)) (дата обращения: 1.05.2019)