Изображение выглядит как символ, эмблема, герб, нашивка

Контент, сгенерированный ИИ, может содержать ошибки.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Отчет по выполнению практического задания №7.2

**Тема: «**Графы»

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Жижикин Л.С.

группа ИКБО-50-24

**Москва 2025**

Содержание

[Цель 3](#_Toc211006873)

[Формулировка задачи 3](#_Toc211006874)

[Ход работы 3](#_Toc211006875)

[Код работы 3](#_Toc211006876)

[Файл Graph.hpp 3](#_Toc211006877)

[Структура ребра 3](#_Toc211006878)

[Тестирование 4](#_Toc211006879)

[Вывод 4](#_Toc211006880)

[Список литературы 4](#_Toc211006881)

# Цель

Освоить приёмы реализации графов.

# Формулировка задачи

Составить программу создания графа и реализовать процедуру для работы с графом, определенную индивидуальным вариантом задания.

Вариант: Нахождение кратчайшего пути методом Беллмана-Форда

# Ход работы

## Код работы

### Файл Graph.hpp

#### Структура ребра

1. **typedef** size\_t Vertex;

2. **struct** Edge {

3.     **union** { Vertex u; Vertex **from**; Vertex source; };

4.     **union** { Vertex v; Vertex to; Vertex dest;};

5.     **union** { i64 w; i64 weight; };

6.

7.     Edge() {}

8.     Edge(Vertex u\_, Vertex v\_, i64 w\_) {

9.         u = u\_;

10.         v = v\_;

11.         w = w\_;

12.     }

13. };

14.

#### Структура графа и конструктор

1. **struct** Graph {

2.     size\_t size;

3.     std::vector<Edge> edges;

4.

5.     Graph(size\_t size, std::vector<string> edges={}) {

6.         **this**->size = size;

7.         size\_t u, v;

8.         i64 w;

9.         **for** (**string** edge : edges) {

10.             std::istringstream iss(edge);

11.             iss >> u >> v >> w;

12.             **if** (u <= size && v <= size && u != v)

13.                 **this**->add\_edge(u, v, w);

14.         }

15.     }

16. }

17.

#### Методы добавления вершин и ребер

1. **void** add\_vertex() { std::cout << "Created new vertex #" << size++ << std::endl; }

2.

3. **void** add\_edge(Vertex u, Vertex v, i64 w) {

4.     **if** (u-- == v--)

5.         std::cout << "Trying to create a self-loop\n";

6.     **else** **if** (u < size && v < size)

7.         edges.push\_back(Edge(u, v, w));

8.     **else**

9.         std::cout << "One of the vertices doesn't exist\n";

10. }

11. **void** add\_edge(**string** edge){

12.     size\_t u, v;

13.     i64 w;

14.     std::istringstream iss(edge);

15.     iss >> u >> v >> w;

16.     **if** (u <= size && v <= size && u != v)

17.         **this**->add\_edge(u, v, w);

18. }

19.

#### Метод добавления ненаправленного ребра

1. **void** add\_non\_directed\_edge(**string** edge) {

2.     size\_t u, v;

3.     i64 w;

4.     std::istringstream iss(edge);

5.     iss >> u >> v >> w;

6.     **if** (u <= size && v <= size && u != v)

7.         **this**->add\_non\_directed\_edge(u, v, w);

8. }

9.

10. **void** add\_non\_directed\_edge(Vertex u, Vertex v, i64 w) {

11.     **if** (u-- == v--)

12.         std::cout << "Trying to create a self-loop\n";

13.     **else** **if** (u < size && v < size) {

14.         edges.push\_back(Edge(u, v, w));

15.         edges.push\_back(Edge(v, u, w));

16.     }

17.     **else**

18.         std::cout << "One of the vertices doesn't exist\n";

19. }

20.

#### Алгоритм Беллмана-Форда

1. std::vector<i64> bellman\_ford(Vertex source) {

2.     std::vector<i64> distance(size, INT\_MAX);

3.     distance[--source] = 0;

4.     //  construct distance vector for each vertex

5.     **for** (size\_t i = 0; i < size - 1; ++i)

6.         **for** (**const** Edge e : **this**->edges)

7.             **if** (distance[e.**from**] != INT\_MAX && distance[e.**from**] + e.weight < distance[e.to])

8.                 distance[e.to] = distance[e.**from**] + e.weight;

9.

10.     //  check for negative-weight cycles

11.     **for** (**const** Edge e : **this**->edges) {

12.         **if** (distance[e.**from**] != INT\_MAX &&

13.             distance[e.**from**] + e.weight < distance[e.to]) {

14.             std::cout << "Graph contains negative weight cycle\n" << "Distance to any node is infinitely small\n";

15.             std::fill(distance.**begin**(), distance.**end**(), INT\_MIN);

16.             **return** distance;

17.         }

18.     }

19.

20.     //  result

21.     std::cout << "Vertex distances from " << source + 1 << "\n";

22.     **for** (size\_t i = 0; i < size; ++i) {

23.         **if** (distance[i] == INT\_MAX)

24.             std::cout << source + 1 << " -> " << i + 1 << "\t\t" << "UNREACHABLE\n";

25.         **else**

26.             std::cout << source + 1 << " -> " << i + 1 << "\t\t" << distance[i] << "\n";

27.     }

28.     **return** distance;

29. }

30.

### Файл main.cpp

#### Функция пользовательского создания графа

1. Graph user\_input() {

2.     size\_t size;

3.     std::vector<string> edges;

4.     std::cout << "Enter amount of Vertices:\t"; std::cin >> size;

5.     std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

6.     std::cout << "Start entering your Edges, empty to exit:\n";

7.     **string** s="00 00 00";

8.     **while** (**true**) {

9.         std::getline(std::cin, s);

10.         **if** (s == "\0")

11.             **break**;

12.         edges.push\_back(s);

13.     }

14.     **return** Graph(size, edges);

15. }

#### Функция main

1. **int** main() {

2.     **const** std::vector<string> edges =

3.     {

4.         "1 2 23", "1 3 12",

5.         "2 3 25", "2 5 22", "2 8 35",

6.         "3 4 18",

7.         "5 6 23",  "5 7 14",

8.         "6 7 24",

9.         "7 8 16"

10.     };

11.     Graph g(8);

12.     **for** (**const** **string** edge : edges) {

13.         g.add\_non\_directed\_edge(edge);

14.     }

15.     // g.print\_edges();

16.     g.bellman\_ford(2);

17. }

## Тестирование

#### Граф из условия:

A black screen with blue text

AI-generated content may be incorrect.

#### Для пользовательского ввода

1. **int** main() {

2.     Graph g = user\_input();

3.     g.print\_edges();

4.     g.bellman\_ford(1);

5. }

A screen shot of a computer

AI-generated content may be incorrect.

# Вывод

В результате данной работы я освоил приёмы реализации графов.

# Список литературы

1. Документация по языку С++ [Электронный ресурс]. URL: https://cppreference. com/ (дата обращения: 11.09.2025).

2. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения: 11.09.2025).

3. Керниган Б., Ритчи Д. Язык программирования Си / Керниган Б., Ритчи Д., 3- е изд., 1988.

4. Страуструп Б. Программирование. Принципы и практика с использованием C+ + / Страуструп Б., 2-е изд., 2016.