Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная работа № 1**

Язык программирования Си

по дисциплине «Низкоуровневое программирование»

Выполнил

студент гр. 3530901/90004

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кривицкий В.В.

(подпись)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург   
2021

**Задача**

В соответствие с выбранным мною вариантом – 7, требуется реализовать неориентированный граф, представленный смежными вершинами.

Реализуем данную структуру данных на языке Си, в качестве компилятора используется GCC.

**Решение**

**Структура:**

* main.c – точка входа
* graph.c – реализация методов графа
* graph.h – заголовочный файл для graph.c
* math.c – реализация математических методов
* math.h – заголовочный файл для math.c
* test.c – реализация методов теста
* test.h – заголовочный файл для test.c

В заголовочном файле graph.h обозначим все структуры и методы, которые будут использоваться в реализации структуры данных

Изображение выглядит как текст

Автоматически созданное описание

Рис.1 Заголовочный файл graph.h

* createGraph– создать граф.

*args*:

amountOfNode – максимальное кол-во вершин в графе

*return*: struct Graph\* - ссылка на структуру графа.

* createNode – создать вершину.

*args*:

graph – ссылка на граф

number – значение графа

*return*: struct Node\* - ссылка на вершину

* removeNode – удалить вершину.

метод автоматически удаляет ребра с данной вершиной, сдвигает список из вершин.

*args*:

graph – ссылка на граф

number – значение графа

*return*: void

* printAllNodes – вывести в консоль все вершины

*args*:

graph – ссылка на граф

*return*: void

* createEdge – создать ребро

*args*:

graph – ссылка на граф

firstNode – ссылка на первую вершину

secondNode – ссылка на вторую вершину

int weight – вес ребра

*return*: struct Edge\* - ссылка на ребро

* removeEdge – создать ребро

*args*:

graph – ссылка на граф

firstNode – ссылка на первую вершину

secondNode – ссылка на вторую вершину

*return*: struct Edge\* - ссылка на ребро

* printAllEdges – вывести в консоль все ребра

*args*:

graph – ссылка на граф

*return*: void

* changeValueOfNode – изменить значение вершины

*args*:

graph – ссылка на граф

fromValue – старое значение

toValue – новое значение

*return*: void

* changeWeightOfNode – изменить вес ребра

*args*:

graph – ссылка на граф

edge – ссылка на ребро

newWeight – новый вес

*return*: void

Все операции реализованной структуры данных были протестированы при помощи модульных тестов, находящихся в файле test.c

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 Результат работы тестов

При помощи CMake была собрана статическая библиотека из классов реализующих граф.

Также была разработана демонстрационная программа – консольное приложение, обеспечивающее ввод входных данных из файла и вывод результата работы в файл.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3 Входные данные

9 – максимальное число вершин

Список ребер в формате d1:d2-d3 где,

d1 – первое ребро

d2 – второе ребро

d3 – вес ребра

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 Выходные данные

На рис 4. представлены выходные данные после работы. В данном случае, из графа была удалена вершина 3 и изменен вес графа [1, 2].

**Вывод**

В данной лабораторной работе мною был реализован неориентированный граф. Результаты работы полностью соответствуют ожиданиям