МГТУ им. Н. Э. Баумана

**Лабораторная работа №7**

**Графы**

Выполнил:

Коротыч Михаил

ИУ7-35Б

*2020 г.*

**Цель работы**Реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.  
 **Условие задачи**

Обработать графовую структуру в соответствии с заданным вариантом. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных осуществить на усмотрение программиста. Результат выдать в графической форме.  
  
Определить, является ли связным заданный граф

**Техническое задание**

**Входные данные**

1. Целое число, представляющее собой количество вершин в рассматриваемом графе: целое положительное число.
2. Пара целых чисел, представляющая собой связи между вершинами: целые неотрицательные числа в диапазоне от 0 до V - 1, где V – количество вершин графа.

**Выходные данные**

1. Графическая визуализация полученного графа
2. Строка, показывающая является ли заданный граф связным

**Функция программы**

Программа находит решение изложенной в условии задачи и визуализирует найденное решение при помощи **GraphViz**.

**Обращение к программе**

Собирается и запускается при помощи makefile

**Аварийные ситуации**

1. Некорректный ввод количества вершин.  
   На входе: неположительное целое число, нецелое число или буква.   
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку»
2. Некорректный ввод связей вершин в графе.  
   На входе: целое число, выходящее за диапазон **[0; V - 1],** нецелое число, буква или попытка провести путь из вершины в саму себя.  
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»  
     
    **Структуры данных**

Один из видов представления графов – это матрица смежности B(n \* n). В этой матрице элемент b[i, j] = 1, если ребро, связывающее вершины Vi и Vj, существует и b[i, j] = 0, если ребра нет. У неориентированных графов матрица смежности всегда симметрична

*Матрица смежности:*

typedef struct

{

int size;

int \*\*matrix;

} matrix\_t;

int size – количество вершин в графе;

int \*\*matrix – указатель на массив указателей;

*Массив посещённых вершин:*

int \*visited = calloc(*matrix*.size, sizeof(int));

**Алгоритм**

После ввода элементов графа граф проверяется на связность. Для проверки его на связность достаточно выполнить рекурсивный поиск в глубину для любой вершины. Если по окончании поиска была посещена каждая вершина графа, граф является связным

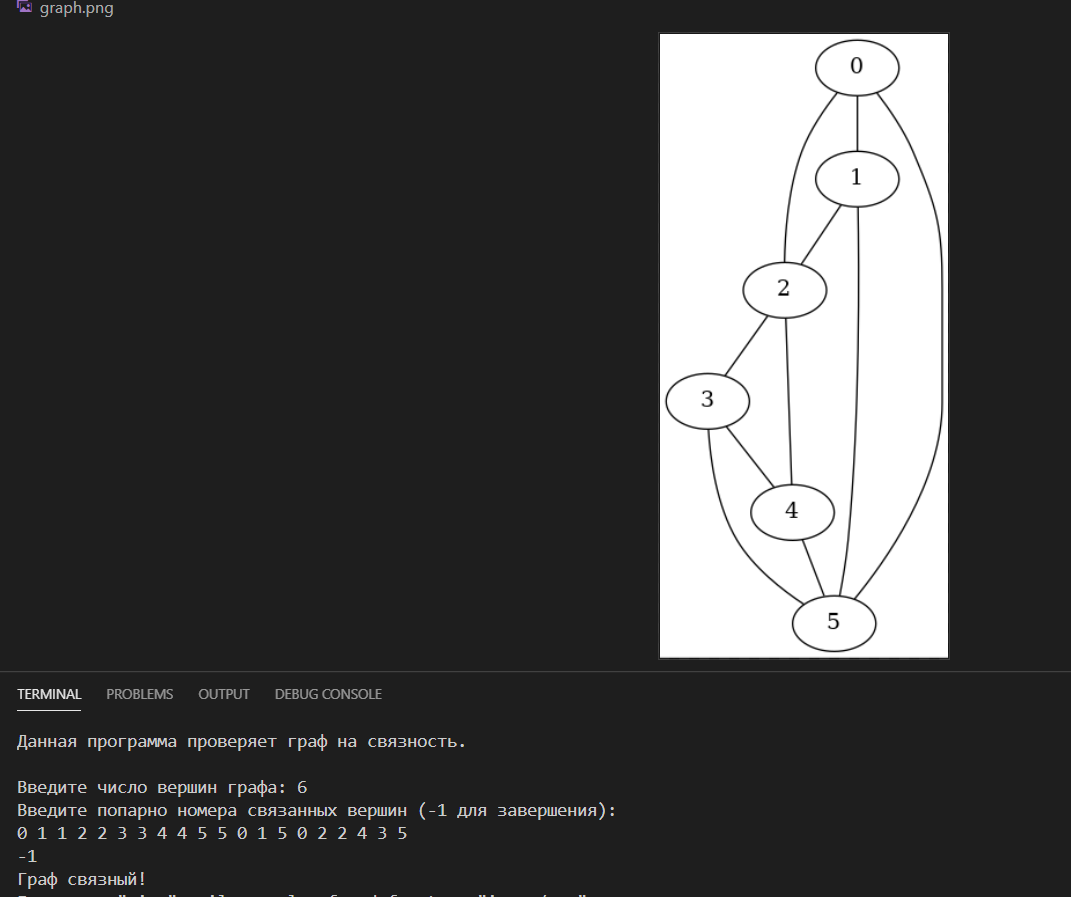
После того, как была произведена проверка на связность, выводится результат программы и изображение графа.

**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Тест | Пользовательский ввод | Результат |
| 1 | Некорректный ввод количества вершин | 0 | Неверное значение! Повторите попытку. |
| 2 | Некорректный ввод количества вершин | A | Неверное значение! Повторите попытку. |
| 3 | Некорректный ввод номера вершины | 5 (при количестве вершин больше или равном 5) | Неверное значение! Повторите попытку. |
| 4 | Некорректный ввод связи между вершинами | 1 1 | Неправильно введён путь! |
| 5 | Корректный ввод характеристик | 3; 0 1 | Граф не является связным!  Вывод изображения в отдельном окне |
| 6 | Корректный ввод характеристик | 3; 0 1 / 1 2 / 2 0 | Граф является связным!  Вывод изображения в отдельном окне |

**Пример работы**

С использованием утилиты Graphviz



**Контрольные вопросы**

**1. Что такое граф?**

Граф – конечное множество вершин и соединяющих их рёбер; **G = <V, E>**. Если пары **Е** (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным.

**2. Как представляются графы в памяти?**

С помощью матрицы смежности или списков смежности.

**3. Какие операции возможны над графами?**

Обход вершин, поиск различных путей, исключение и включение вершин.

**4. Какие способы обхода графов существуют?**

Обход в ширину **(BFS – Breadth First Search)**, обход в глубину **(DFS – Depth First Search)**.

**5. Где используются графовые структуры?**

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

**6. Какие пути в графе Вы знаете?**

Эйлеров путь, простой путь, сложный путь, гамильтонов путь.

**7. Что такое каркасы графа?**

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.

**Вывод**

В данной реализации алгоритм состоит из поиска в глубину. Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин. Если **V** – количество вершин графа, а **E** – количество рёбер, то алгоритм поиска в глубину имеет асимптотическую сложность **O(V + E)**.

Задача может применяться в ситуациях, когда необходимо проверить возможность построения сложного маршрута поездки.