МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА» - Российский технологический университет»

РТУ МИРЭА

Отчёт по выполнению практического задания № 7.2
Тема:
«Создание, алгоритмы обхода, важные задачи теории графов»

Выполнил студент: Данов Арсений Иванович

Группа: ИКБО-10-23

Москва 2024

Вариант:

Номер: *6*

Алгоритм: Нахождение кратчайшего пути методом Флойда

На рисунке 1 представлен граф, определенный в индивидуальном варианте.

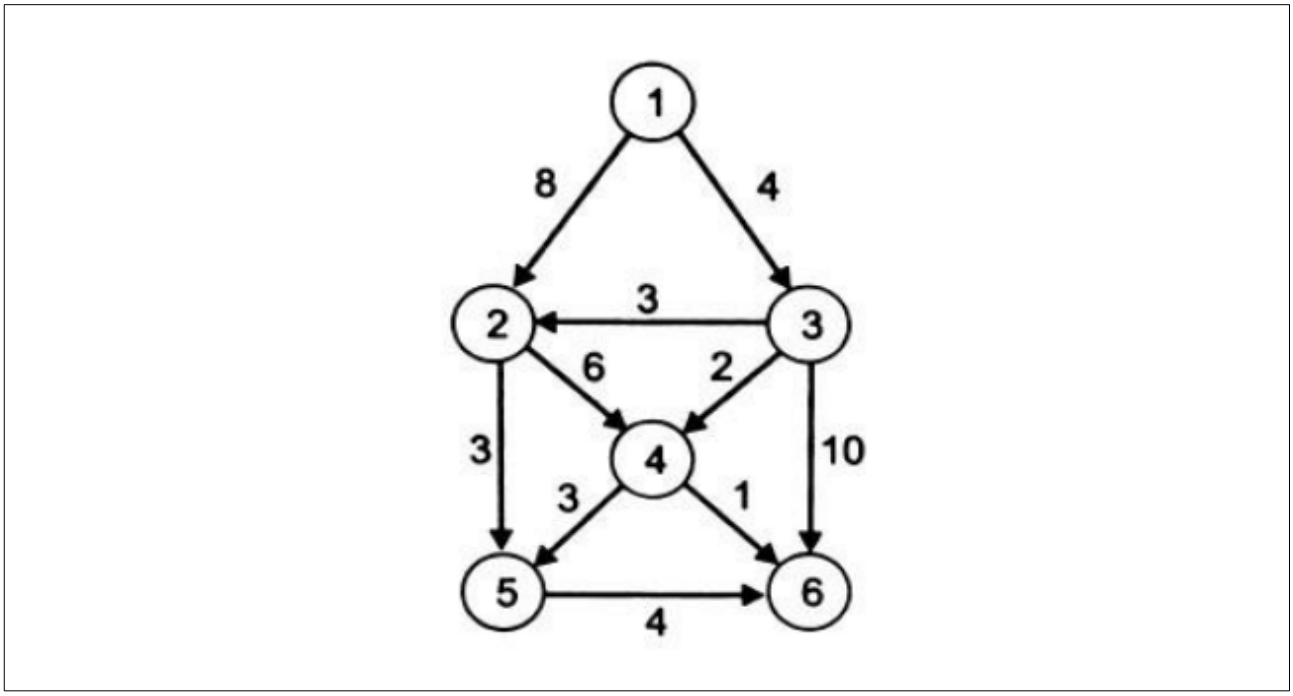


Рисунок 1 — Предложенный граф

1.1 Формулировка задачи

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

Самостоятельно выбрать и реализовать способ представления графа в памяти.

В программе предусмотреть ввод с клавиатуры произвольного графа. В вариантах построения остовного дерева также разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы на предложенном в индивидуальном варианте задания графе. Результаты тестирования в виде скриншотов экранов включить в отчет по выполненной работе.

Сделать выводы о проделанной работе, основанные на полученных результатах.

Оформить отчет с подробным описанием рассматриваемого графа, принципов программной реализации алгоритмов работы с графом, описанием текста исходного кода и проведенного тестирования программы.

1.2 Ход решения

В индивидуальном варианте №6 предоставленный граф является помеченным, взвешенным и ориентированным, но не является связным. Граф помеченный - это значит, что у каждой вершины графа есть метка (в данном случае числовая). Граф взвешенный — это значит что ребра (дуги) графа имеют веса, обозначающие чаще всего расстояние между вершинами. Граф ориентированный — значит направление обхода и перемещения между двумя вершинами существенно. Граф не связный, так как из-за направленности ребер нельзя любые его две вершины соединить маршрутом в обоих направлениях, а некоторые нельзя соединить и в одном направлении.

В программе граф размера n представляется матрицей смежности graph[n][n], где элементу graph[i][j] соответствует вес дуги, выходящей из вершины i и заканчивающейся в вершине j. Если в классической матрице смежности элементы расположены зеркально относительно главной диагонали матрицы, так как матрица составляется для неориентированного графа, в использованном мной представлении симметрия нарушена ввиду ориентированности дуг относительно вершин.

Алгоритм Флойда-Уоршелла представляет собой на деле довольно простой алгоритм полного перебора всех возможных путей. Сначала происходит адоптация матрицы смежности, в ходе которой все нули, означающие отсутсвующую дугу между вершинами, заменяют на очень большое число, теперь обозначающее бесконечное расстояние между вершинами.

Далее начинается перебор трёх индексов — i, j и k. Так сравнивается уже известное расстояние между i и j с суммарным расстоянием между i и k, а также k и j, что позволяет определить наикротчайший путь, ведь мы рассматриваем все возможные промежуточные вершины между вершинами i и j.

После завершения работы алгоритма получается матрица кратчайших расстояний graph[n][n], в которой элемент graph[i][j] обозначает кратчайший из вершины i в вершину j.

1.3 Исходный код программ

На рисунках 2-6 представлен исходный коды программы, коды необходимых функций, а также функция main.

```
vector<vector<int>> enter_graph(int n)
{
    int x;
    vector<vector<int>> graph(n, vector<int>(n, 0));
    for (int i = 0; i < n; ++i)
    {
        cout << "Введите веса дуг, исходящих из вершины " << i << ": ";
        for (int j = 0; j < n; ++j)
        {
            cin >> x;
            graph[i][j] = x;
        }
    }
    return graph;
}

vector<vector<int>> read_graph()
{
    ifstream file("graph.txt");
    int x, n;
    file >> n;
    vector<vector<int>> graph(n, vector<int>(n, 0));
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            file >> x;
            graph[i][j] = x;
        }
    }
    file.close();
    return graph;
}
```

Рисунок 2 — Исходный код программы

```
43  void build_graph(vector<vector<int>> &graph)
44  {
45      int n = graph.size();
46      int inf = 999;
47      for (int i = 0; i < n; ++i)
48      {
49          for (int j = 0; j < n; ++j)
50          {
51              if (graph[i][j] == 0 && i != j)
52              {
53                  graph[i][j] = inf;
54              }
55          }
56      }
57      for (int k = 0; k < n; ++k)
58      {
59          for (int i = 0; i < n; ++i)
60          {
61              for (int j = 0; j < n; ++j)
62              {
63                  if (graph[i][k] + graph[k][j] < graph[i][j])
64                  {
65                      graph[i][j] = graph[i][k] + graph[k][j];
66                  }
67              }
68          }
69      }
70  }
```

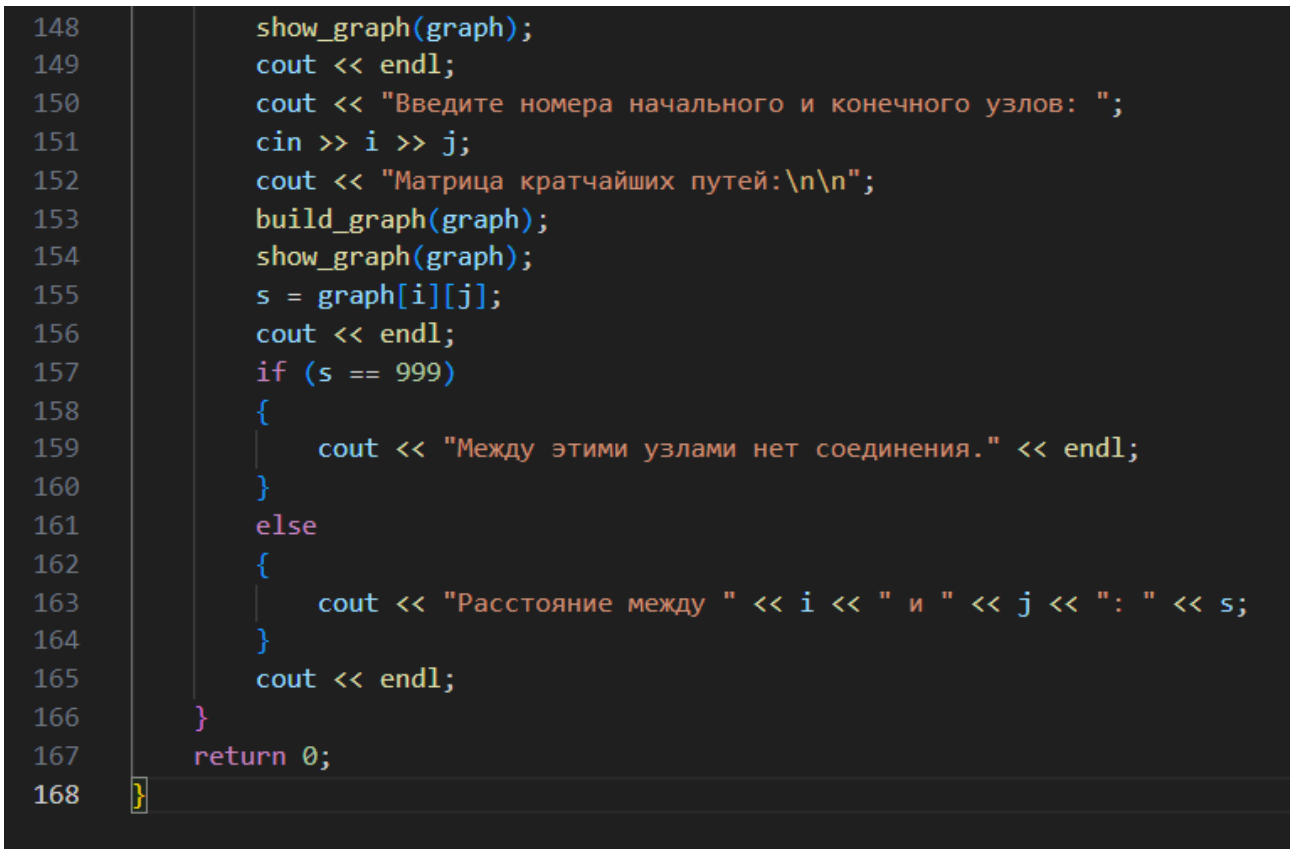
Рисунок 3 — Исходный код программы

```
72  void show_graph(vector<vector<int>> &graph)
73  {
74      int n = graph.size();
75      cout << "  \\ ";
76      int offset = 0;
77      for (int i = 0; i < n; ++i)
78      {
79          offset = (int)log10(i);
80          offset = offset < 0 ? 0 : offset;
81          cout << string(3 - offset, ' ') << i;
82      }
83      cout << endl
84           << "   \\" << string(n * 4 + 1, '-') << endl;
85      int x;
86      for (int i = 0; i < n; ++i)
87      {
88          offset = (int)log10(i);
89          offset = offset < 0 ? 0 : offset;
90          cout << string(2 - offset, ' ') << i << "|";
91          for (int j = 0; j < n; ++j)
92          {
93              x = graph[i][j];
94              if (x != 999)
95              {
96                  offset = (int)log10(x);
97                  offset = offset < 0 ? 0 : offset;
98                  cout << string(3 - offset, ' ') << x;
99              }
100             else
101             {
102                 cout << "   -";
103             }
104         }
105         cout << endl;
106     }
107 }
```

Рисунок 4 — Исходный код программы

```
109  int main()
110  {
111      vector<vector<int>> graph;
112      int n, i, j, s;
113      while (true)
114      {
115          int choise;
116          cout << endl
117               << "Что вы хотите сделать?" << endl
118               << "1 - Ввод графа с клавиатуры" << endl
119               << "2 - Ввод графа из файла" << endl
120               << "0 - Выход" << endl
121               << "Номер действия: ";
122          cin >> choise;
123          cout << endl;
124          if (choise == 1)
125          {
126              cout << "Введите количество вершин графа: ";
127              cin >> n;
128              graph = enter_graph(n);
129              cout << endl
130                   << "Граф считан:\n\n";
131          }
132          else if (choise == 2)
133          {
134              cout << "Граф считан из файла:\n\n";
135              graph = read_graph();
136          }
137          else if (choise == 0)
138          {
139              break;
140          }
141          else
142          {
143              cout << "Неизвестная команда.";
144              cout << endl;
145              continue;
146          }
```

Рисунок 5 — Исходный код программы



```
148         show_graph(graph);
149         cout << endl;
150         cout << "Введите номера начального и конечного узлов: ";
151         cin >> i >> j;
152         cout << "Матрица кратчайших путей:\n\n";
153         build_graph(graph);
154         show_graph(graph);
155         s = graph[i][j];
156         cout << endl;
157         if (s == 999)
158         {
159             cout << "Между этими узлами нет соединения." << endl;
160         }
161         else
162         {
163             cout << "Расстояние между " << i << " и " << j << ": " << s;
164         }
165         cout << endl;
166     }
167     return 0;
168 }
```

Рисунок 6 — Исходный код программы

1.4 Результаты тестирования

На рисунках 7-8 представлены результаты тестирования программы — все возможные обходы, а также дейтсвия над узлами.

```
Что вы хотите сделать?
1 - Ввод графа с клавиатуры
2 - Ввод графа из файла
0 - Выход
Номер действия: 2

Граф считан из файла:

  \    0   1   2   3   4   5
   \------------------------
  0|   0   8   4   0   0   0
  1|   0   0   0   6   3   0
  2|   0   3   0   2   0  10
  3|   0   0   0   0   3   1
  4|   0   0   0   0   0   4
  5|   0   0   0   0   0   0

Введите номера начального и конечного узлов: 0 5
Матрица кратчайших путей:

  \    0   1   2   3   4   5
   \------------------------
  0|   0   7   4   6   9   7
  1|   -   0   -   6   3   7
  2|   -   3   0   2   5   3
  3|   -   -   -   0   3   1
  4|   -   -   -   -   0   4
  5|   -   -   -   -   -   0

Расстояние между 0 и 5: 7
```

Рисунок 7 — Результаты тестирования программы

```
Что вы хотите сделать?
1 - Ввод графа с клавиатуры
2 - Ввод графа из файла
0 - Выход
Номер действия: 1

Введите количество вершин графа: 4
Введите веса дуг, исходящих из вершины 0: 1 2 5 8
Введите веса дуг, исходящих из вершины 1: 4 3 43 5
Введите веса дуг, исходящих из вершины 2: 0 31 0 12
Введите веса дуг, исходящих из вершины 3: 0 0 0 0

Граф считан:

  \    0   1   2   3
   \-----------------
  0|   1   2   5   8
  1|   4   3  43   5
  2|   0  31   0  12
  3|   0   0   0   0

Введите номера начального и конечного узлов: 0 3
Матрица кратчайших путей:

  \    0   1   2   3
   \-----------------
  0|   1   2   5   7
  1|   4   3   9   5
  2|  35  31   0  12
  3|   -   -   -   0

Расстояние между 0 и 3: 7
```

Рисунок 8 — Результаты тестирования программы

2.1 Выводы

Графы — очень многофункциональный инструмент (структура данных), позволяющий выражать сложные нелинейные отношения между элементами предметной области. Очень многие задачи являются нелинейными, а следовательно, нерешаемыми (или непредставляемыми) в виде линейных структур или деревьев, благодаря чему графы являются неотъемлимой частью решения таких задач.