Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Граф»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**Постановка задачи**

1. Сформировать идеально сбалансированное бинарное дерево, тип информационного поля указан в варианте.

2. Распечатать полученное дерево.

3. Выполнить обработку дерева в соответствии с заданием, вывести полученный результат.

4. Преобразовать идеально сбалансированное дерево в дерево поиска.

5. Распечатать полученное дерево.

**Вариант 10**

Найти кратчайшие пути из выбранной вершины – Кратчайшие пути из вершины 6.

**Код программы**

#include <iostream>

#include <cmath>

const int SIZE = 6;

using namespace std;

void print\_matr(int a[6][6])

{

    cout << "Матрица смежности: " << endl;

    // Вывод матрицы смежности

    for (int i = 0; i < SIZE; i++)

    {

        for (int j = 0; j < SIZE; j++)

        {

            cout << a[i][j] << " ";

        }

        cout << endl;

    }

}

int main()

{

    setlocale(LC\_ALL, "RUS");

    // Матрица смежности

    int a[SIZE][SIZE] = {

        {0,  7, 14, 33,  5, 12},

        {7,  0, 19, 23,  3,  0},

        {14,19,  0,  2, 15,  0},

        {33,23,  2,  0, 28,  9},

        {5,  3, 15, 28,  0, 0},

        {12, 0,  0,  9, 0,  0} };

    int min\_dist[SIZE]; // массив минимального расстояния

    int visit[SIZE]; // визит в веришины

    int tmp;

    int min\_index, min;

    int begin\_index = 5;//идём от 6-ой вершины

    print\_matr(a);

    //Предварительное создание вершин и веса рёбер

    for (int i = 0; i < SIZE; i++)

    {

        min\_dist[i] = INT\_MAX;

        visit[i] = 1;

    }

    min\_dist[begin\_index] = 0;

    do {

        min\_index = INT\_MAX;

        min = INT\_MAX;

        for (int i = 0; i < SIZE; i++)

        {

            if ((visit[i] == 1) && (min\_dist[i] < min))// Если вершину ещё не посещали и вес меньше min

            {

                min = min\_dist[i]; // Переприсваиваем значение

                min\_index = i; // Переприсваиваем значение

            }

        }

        if (min\_index != INT\_MAX)

        {

            for (int i = 0; i < SIZE; i++)

            {

                if (a[min\_index][i] > 0)

                {

                    tmp = min + a[min\_index][i];// Добавляем найденный минимальный вес к текущему весу вершины

                    if (tmp < min\_dist[i])//Сравниваем с текущим минимальным весом вершины

                    {

                        min\_dist[i] = tmp;

                    }

                }

            }

            visit[min\_index] = 0;

        }

    } while (min\_index < INT\_MAX);

    // Вывод кратчайших расстояний до вершин

    cout << endl << "Кратчайшие расстояния от 6-ой вершины до вершин остальных: " << endl;

    for (int i = 0; i < SIZE; i++)

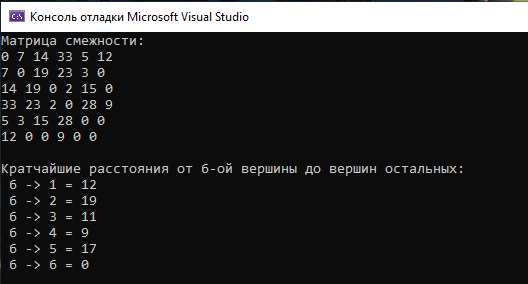
    {

        cout << " 6 -> " << i + 1 << " = " << min\_dist[i] << endl;

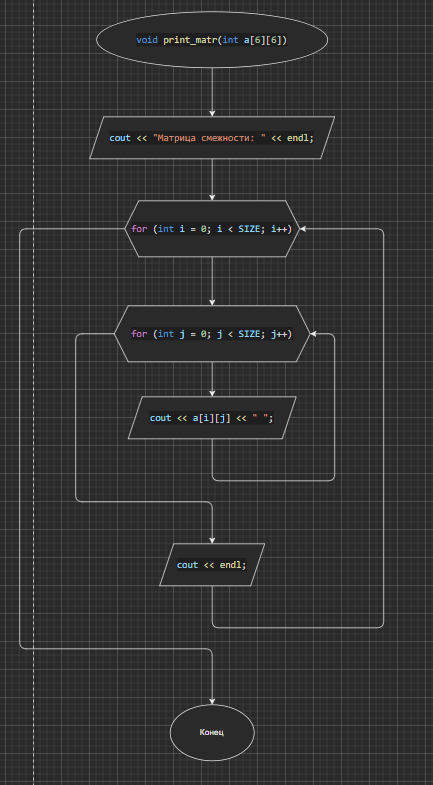
    }

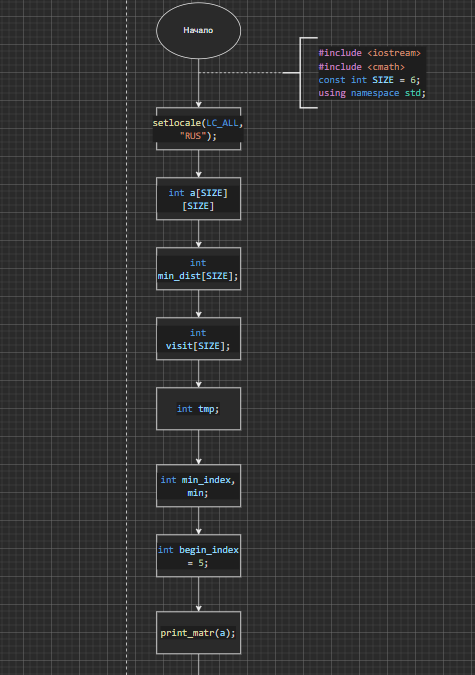
}

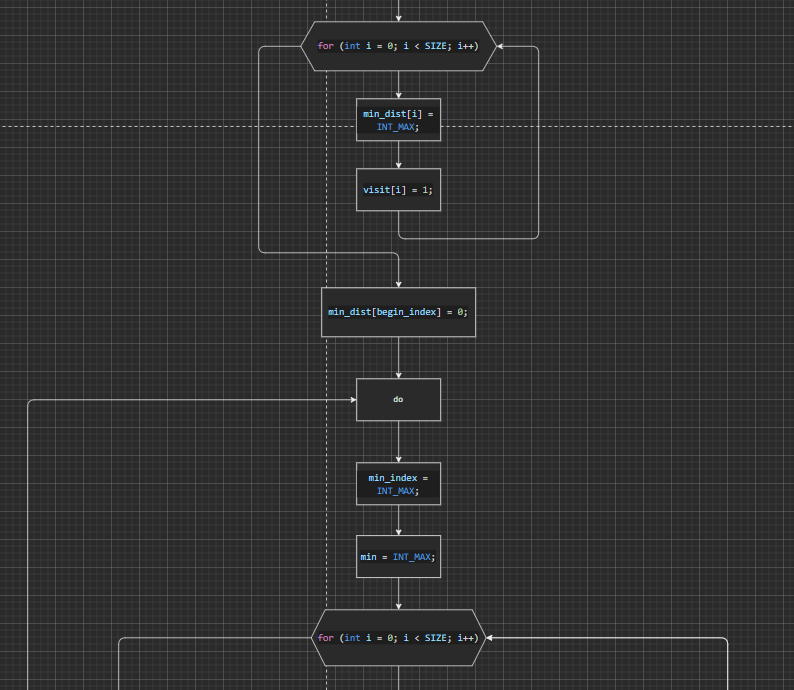
**Результат работы программы**

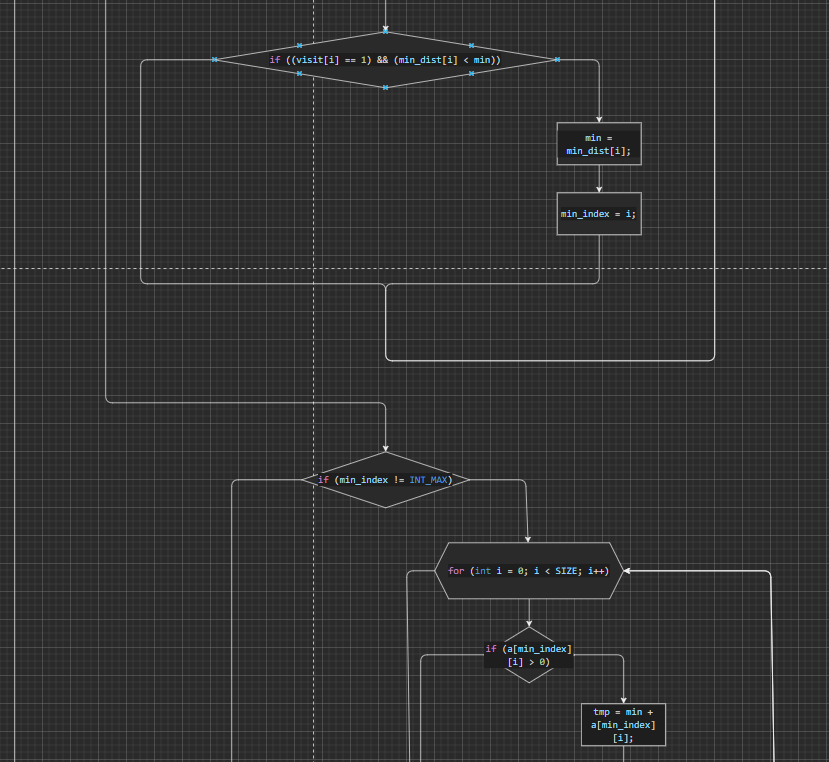
****

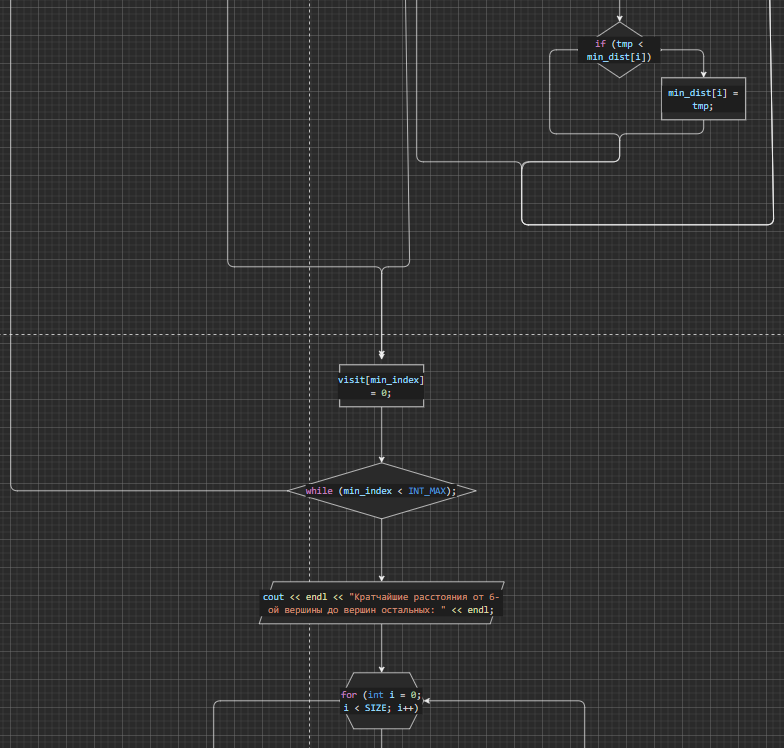
**Блок-схема**

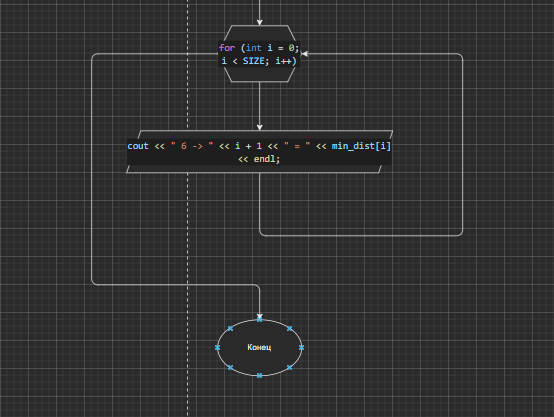
****

****

****

****

****

****