

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №7

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Обход графа в глубину"

Выполнили:

Студенты группы 24ВВВЗ:

Плотников И.А.

Виноградов Б.С.

Приняли:

Деев М.В.

Юрова О. В.

Пенза 2025

Цель

Изучение обхода графа в глубину.

Лабораторное задание

Задание 1

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру обхода в глубину, реализованную в соответствии с приведенным выше описанием.
3. *Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.

Задание 2*

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

Пояснительный текст к программам

Эта программа реализует алгоритм обхода графа в глубину (DFS). Пользователь задаёт количество вершин графа, после чего программа автоматически генерирует симметричную матрицу смежности, где:

- 0 означает отсутствие ребра между вершинами
- 1 означает наличие ребра

Главная диагональ матрицы заполняется нулями (исключаются петли). После отображения матрицы пользователь выбирает стартовую вершину, и программа выполняет обход в глубину, выводя порядок посещения вершин. По завершении вся выделенная память освобождается.

Псевдокод:

Вход: G – матрица смежности графа.

Выход: номера вершин в порядке их прохождения на экране.

Алгоритм ПОГ

- 1.1. для всех i положим $NUM[i] = \text{False}$ пометим как "не посещенную";
- 1.2. **ПОКА** существует "новая" вершина v
- 1.3. **ВЫПОЛНЯТЬ** DFS (v).

Алгоритм DFS(v):

- 2.1. пометить v как "посещенную" $NUM[v] = \text{True}$;
- 2.2. вывести на экран v ;
- 2.3. **ДЛЯ** $i = 1$ **ДО** $\text{size_}G$ **ВЫПОЛНЯТЬ**
- 2.4. **ЕСЛИ** $G(v,i) = 1$ **И** $NUM[i] = \text{False}$
- 2.5. **ТО**
- 2.6. {
- 2.7. DFS(i);
- 2.8. }

Например, пусть дан граф (рисунок 1), заданный в виде матрицы смежности:

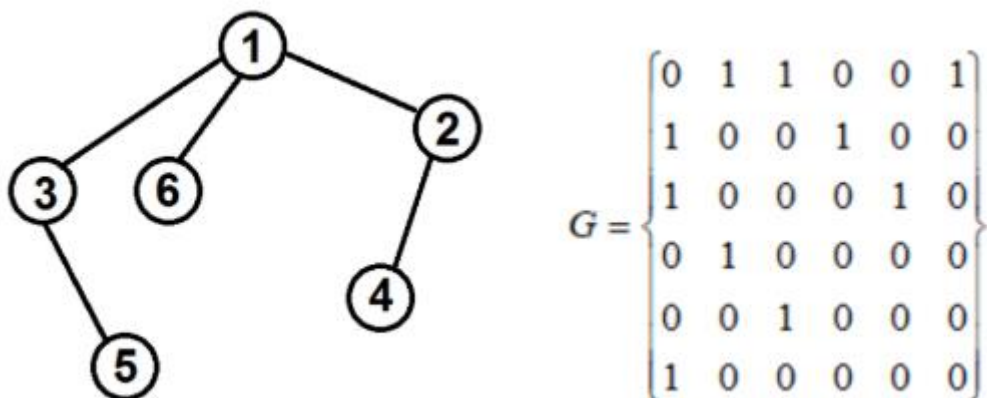
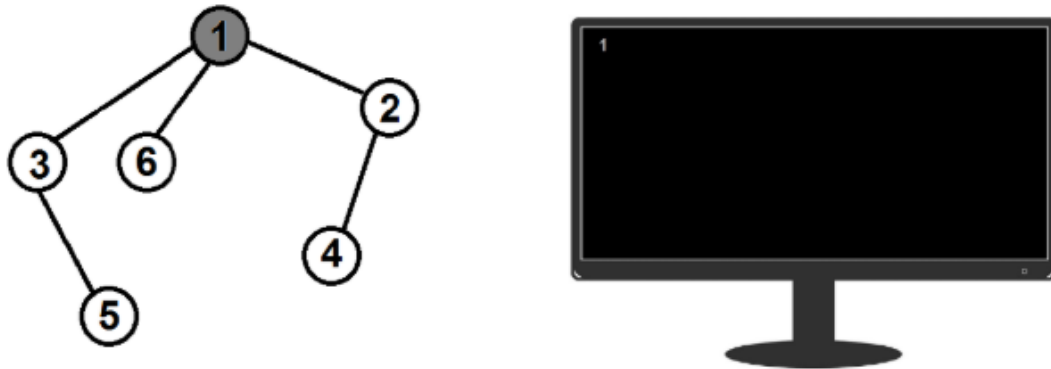


Рисунок 1 – Граф

Тогда, если мы начнем обход из первой вершины, то на шаге 2.1 она будет помечена как посещенная ($NUM[1] = True$), на экран будет выведена единица.



$NUM = \{True \ False \ False \ False \ False \ False\}$

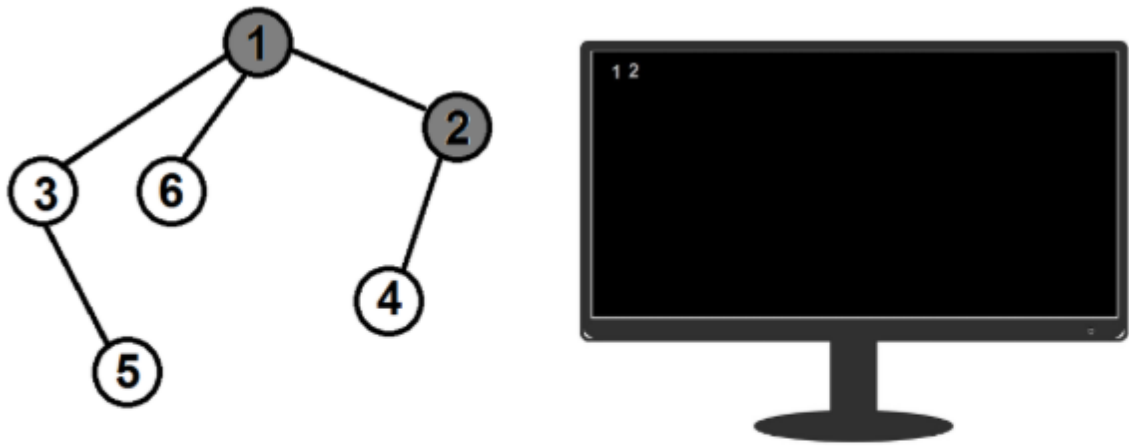
Рисунок 2 – Вызов DFS(1)

При просмотре 1-й строки матрицы смежности

$$G = \begin{Bmatrix} 0 & \boxed{1} & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

будет найдена смежная вершина с индексом 2 ($G(1,2) = 1$), которая не посещена ($NUM[2] = False$) и будет вызвана процедура обхода уже для нее - DFS(2).

На следующем вызове на шаге 2.1 вершина 2 будет помечена как посещенная ($NUM[2] = True$), на экран будет выведена двойка.



$NUM = \{True \ True \ False \ False \ False \ False\}$

Рисунок 3 – Вызов DFS(2)

И алгоритм перейдет к просмотру второй строки матрицы смежности. Первая смежная с вершиной 2 - вершина с индексом 1 ($G(2,1) = 1$),

$$G = \begin{Bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

которая к настоящему моменту уже посещена ($NUM[1] = True$) и процедура обхода для нее вызвана не будет. Цикл 2.3 продолжит просмотр матрицы смежности.

Следующая найденная вершина, смежная со второй, будет иметь индекс 4 ($G(2,4) = 1$), она не посещена ($NUM[4] = False$) и для нее будет вызвана процедура обхода - DFS(4).

$$G = \begin{Bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

Вершина 4 будет помечена как посещенная ($NUM[4] = True$), на экран будет выведена четверка.

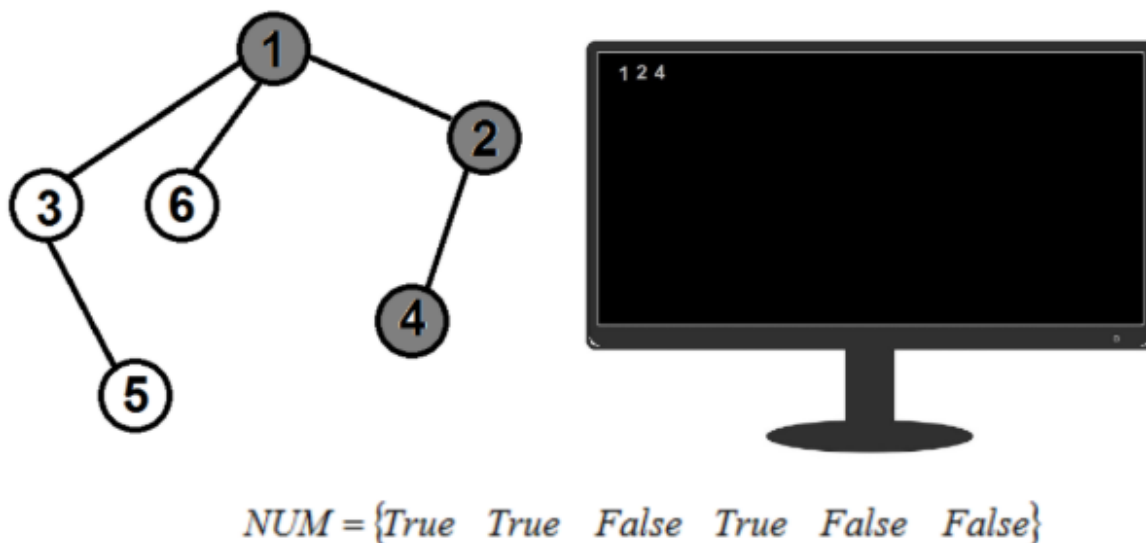


Рисунок 4 – Вызов DFS(4)

При просмотре 4-й строки матрицы будет найдена вершина 2, но она уже посещена ($NUM[2] = True$), поэтому процедура обхода вызвана не будет.

$$G = \begin{Bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

Цикл 2.3 завершится и для текущего вызова DFS(4) процедура закончит свою работу, вернувшись к точке вызова, т.е. к моменту просмотра циклом 2.3 строки с индексом 2 для вызова DFS(2).

В вызове DFS(2) цикл 2.3 продолжит просмотр строки 2 в матрице смежности, и, пройдя её до конца завершится. Вместе с этим завершится и вызов процедуры DFS(2), вернувшись к точке вызова - просмотру циклом 2.3 строки с индексом 1 для вызова DFS(1).

При просмотре строки 1 циклом 2.3 в матрице смежности будет найдена следующая не посещенная, смежная с 1-й, вершина с индексом 3 ($G(1,2) = 1$

и $NUM[3] == False$) и для нее будет вызвана $DFS(3)$.

Вершина 3 будет помечена как посещенная ($NUM[3] = True$), на экран будет выведена тройка.

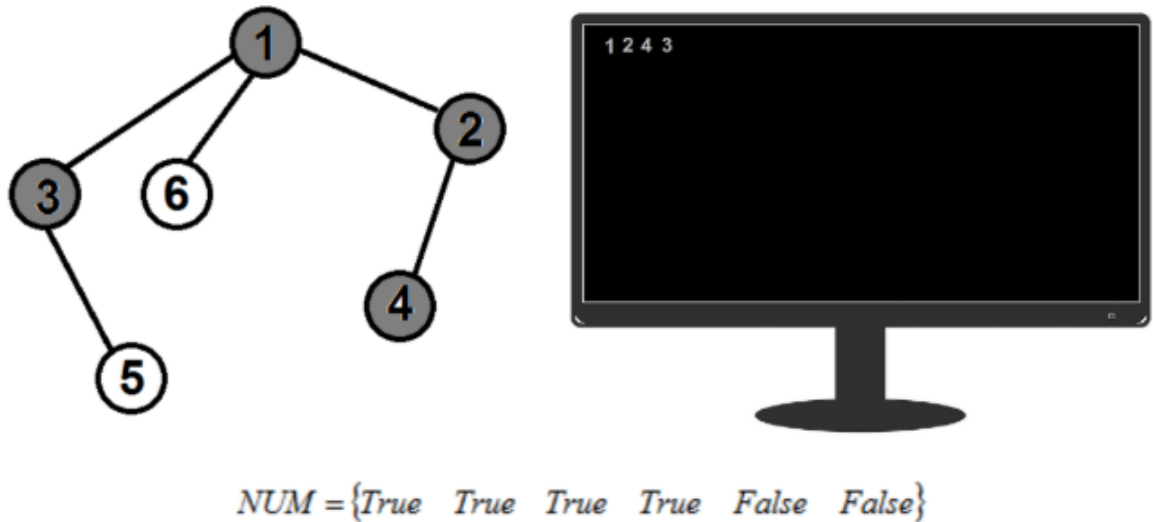


Рисунок 4 – Вызов $DFS(3)$

Работа алгоритма будет продолжаться до тех пор, пока будут оставаться не посещенные вершины, т.е. для которых $NUM[i] == False$.

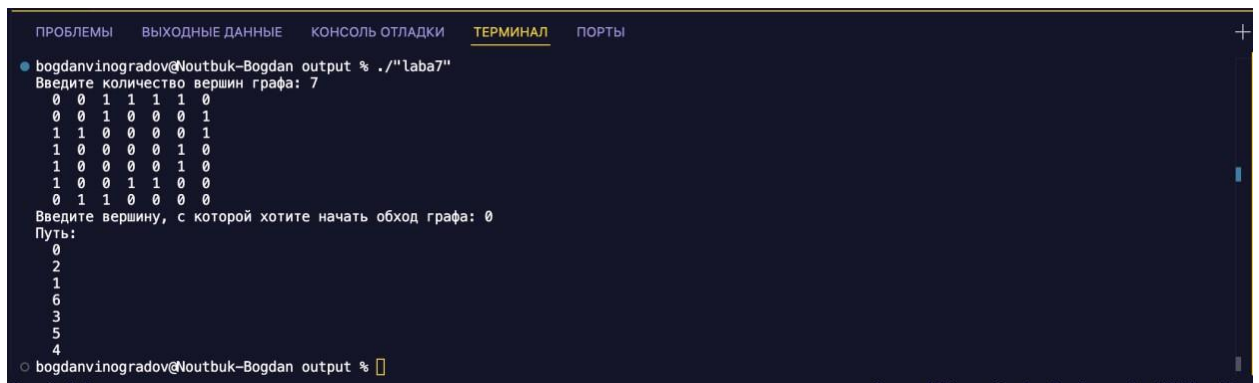
В конце работы алгоритма все вершины будут посещены. А на экран будут выведены номера вершин в порядке их посещения алгоритмом.



Рисунок 5 – Результат работы обхода

Результаты программ

Рисунок 6 - Результат работы **laba7.cpp**



```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ
● bogdanvinogradov@Noutbuk-Bogdan output % ./"laba7"
Введите количество вершин графа: 7
0 0 1 1 1 1 0
0 0 1 0 0 0 1
1 1 0 0 0 0 1
1 0 0 0 0 1 0
1 0 0 0 0 1 0
1 0 0 1 1 0 0
0 1 1 0 0 0 0
Введите вершину, с которой хотите начать обход графа: 0
Путь:
0
2
1
6
3
5
4
○ bogdanvinogradov@Noutbuk-Bogdan output %
```

Вывод: в ходе выполнения лабораторной работы была разработана программа для выполнения заданий Лабораторной работы №7 – обход графа в глубину.

Приложение А

Листинг

Файл laba7.cpp

```
// обход в глубину
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <locale>
#include <limits>
#include <iomanip>

void clearScreen();
int isInteger(const std::string& message);
void dfs(int** G, int numG, int* visited, int s);

int main() {
    setlocale(LC_ALL, "Rus");
    clearScreen();
    srand(time(NULL));
    int** G;
    int numG, current;
    int* visited;

    numG = isInteger("Введите количество вершин графа: ");
    while (numG <= 0) {
        std::cout << "Ошибка! Количество вершин должно быть
положительным\n";
        numG = isInteger("Введите количество вершин графа: ");
    }

    G = (int**)malloc(sizeof(int*) * numG);
    visited = (int*)malloc(numG * sizeof(int));

    for (int i = 0; i < numG; i++) {
        G[i] = (int*)malloc(numG * sizeof(int));
    }

    for (int i = 0; i < numG; i++) {
        visited[i] = 0;
        for (int j = 0; j < numG; j++) {
            G[i][j] = G[j][i] = (i == j ? 0 : rand() % 2);
        }
    }

    for (int i = 0; i < numG; i++) {
        for (int j = 0; j < numG; j++) {
            std::cout << std::setw(3) << G[i][j];
        }
        std::cout << "\n";
    }
}
```

```

        current = isInteger("Введите вершину, с которой хотите
начать обход графа: ");
        while (current < 0) {
            std::cout << "Ошибка! Вершина не может быть
отрицательной\n";
            current = isInteger("Введите вершину, с которой хотите
начать обход графа: ");
        }

        std::cout << "Путь: \n";

        dfs(G, numG, visited, current);

        free(visited);
        for (int i = 0; i < numG; i++){
            free(G[i]);
        }
        free(G);

        return 0;
    }

void clearScreen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

int isInteger(const std::string& message) {
    int value;
    while (true) {
        std::cout << message;
        if (!(std::cin >> value)) {
            std::cout << "Ошибка: введено не число.\n";
            std::cin.clear();

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

            continue;
        }
        if (std::cin.peek() != '\n') {
            std::cout << "Ошибка: введено не целое число.\n";
            std::cin.clear();

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

            continue;
        }
        return value;
    }
}

```

```
}

void dfs(int** G, int numG, int* visited, int s) {
    visited[s] = 1;
    std::cout << std::setw(3) << s << "\n";

    for (int i = 0; i < numG; i++) {
        if (G[s][i] == 1 && visited[i] == 0) {
            dfs(G, numG, visited, i);
        }
    }
}
```