Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в ширину»

Выполнили:
студенты группы 24ВВВ3
Агапов И.А.
Любченко В.К.

Приняли:
к.т.н., доцент Юрова О.В.
к.т.н., Деев М.В.

Пенза 2025

Цель работы – Изучение и практическая реализация алгоритма обхода графа в ширину (BFS) на примере различных структур данных для представления графа (матрица смежности, списки смежности).

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 ширину, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки C++.
3.* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

Задание 2*

1. Для матричной формы представления графов реализуйте алгоритм обхода в ширину с использованием очереди, построенной на основе структуры данных «список», самостоятельно созданной в лабораторной работе № 3.
2. Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс **queue** и использующего очередь, реализованную самостоятельно) для графов разных порядков.

Код программы

C#

```
using System;
using System.Collections.Generic;
using System.Diagnostics;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== СРАВНЕНИЕ ОБХОДА В ШИРИНУ ДЛЯ МАТРИЦЫ И СПИСКА
СМЕЖНОСТИ ===\n");

        int n = 6;
```

```
    int[,] adjacencyMatrix = GenerateAdjacencyMatrix(n);

    Console.WriteLine("Матрица смежности:");
    PrintMatrix(adjacencyMatrix);
    Console.WriteLine();

    List<List<int>> adjacencyList = ConvertToAdjacencyList(adjacencyMatrix);

    Console.WriteLine("Список смежности:");
    PrintAdjacencyList(adjacencyList);
    Console.WriteLine();

    Console.WriteLine("=== ОБХОД В ШИРИНУ ДЛЯ МАТРИЦЫ СМЕЖНОСТИ ===");
    BFS_Matrix(adjacencyMatrix);
    Console.WriteLine();

    Console.WriteLine("=== ОБХОД В ШИРИНУ ДЛЯ СПИСКА СМЕЖНОСТИ ===");
    BFS_AdjacencyList(adjacencyList);
    Console.WriteLine();
}

static int[,] GenerateAdjacencyMatrix(int size)
{
    Random rand = new Random();
    int[,] matrix = new int[size, size];

    for (int i = 0; i < size; i++)
    {
        for (int j = i + 1; j < size; j++)
        {
            if (rand.Next(2) == 1)
            {
                matrix[i, j] = 1;
                matrix[j, i] = 1;
            }
        }
    }
    return matrix;
}


static void PrintMatrix(int[,] matrix)
{
    int size = matrix.GetLength(0);


    Console.Write("   ");
    for (int i = 0; i < size; i++)
    {
        Console.Write($"{i + 1} ");
    }
    Console.WriteLine();

    for (int i = 0; i < size; i++)
    {
        Console.Write($"{i + 1}: ");
        for (int j = 0; j < size; j++)
        {
            Console.Write($"{matrix[i, j]} ");
        }
        Console.WriteLine();
    }
}

static List<List<int>> ConvertToAdjacencyList(int[,] matrix)
```

```
{
    int size = matrix.GetLength(0);
    List<List<int>> adjacencyList = new List<List<int>>();

    for (int i = 0; i < size; i++)
    {
        List<int> neighbors = new List<int>();

        for (int j = 0; j < size; j++)
        {
            if (matrix[i, j] == 1)
            {
                neighbors.Add(j);
            }
        }

        adjacencyList.Add(neighbors);
    }

    return adjacencyList;
}

static void PrintAdjacencyList(List<List<int>> adjacencyList)
{
    for (int i = 0; i < adjacencyList.Count; i++)
    {
        Console.Write($"Вершина {i + 1}: ");

        foreach (int neighbor in adjacencyList[i])
        {
            Console.Write($"{neighbor + 1} ");
        }

        Console.WriteLine();
    }
}

static void BFS_Matrix(int[,] graph)
{
    int size = graph.GetLength(0);
    bool[] visited = new bool[size];

    for (int i = 0; i < size; i++)
    {
        visited[i] = false;
    }

    Console.Write("Порядок обхода: ");

    for (int i = 0; i < size; i++)
    {
        if (!visited[i])
        {
            BFS_Matrix_Traversal(graph, i, visited);
        }
    }
    Console.WriteLine();
}

static void BFS_Matrix_Traversal(int[,] graph, int startVertex, bool[] visited)
{
    Queue<int> queue = new Queue<int>();

    queue.Enqueue(startVertex);
    visited[startVertex] = true;
```

```
        while (queue.Count > 0)
        {
            int currentVertex = queue.Dequeue();
            Console.Write($"{currentVertex + 1} ");

            for (int i = 0; i < graph.GetLength(0); i++)
            {
                if (graph[currentVertex, i] == 1 && !visited[i])
                {
                    queue.Enqueue(i);
                    visited[i] = true;
                }
            }
        }
    }


    static void BFS_AdjacencyList(List<List<int>> adjacencyList)
    {
        int size = adjacencyList.Count;
        bool[] visited = new bool[size];

        for (int i = 0; i < size; i++)
        {
            visited[i] = false;
        }

        Console.Write("Порядок обхода: ");

        for (int i = 0; i < size; i++)
        {
            if (!visited[i])
            {
                BFS_AdjacencyList_Traversal(adjacencyList, i, visited);
            }
        }
        Console.WriteLine();
    }

    static void BFS_AdjacencyList_Traversal(List<List<int>> adjacencyList, int
startVertex, bool[] visited)
    {
        Queue<int> queue = new Queue<int>();

        queue.Enqueue(startVertex);
        visited[startVertex] = true;

        while (queue.Count > 0)
        {
            int currentVertex = queue.Dequeue();
            Console.Write($"{currentVertex + 1} ");

            foreach (int neighbor in adjacencyList[currentVertex])
            {
                if (!visited[neighbor])
                {
                    queue.Enqueue(neighbor);
                    visited[neighbor] = true;
                }
            }
        }
    }
}
```

Результаты работы программы

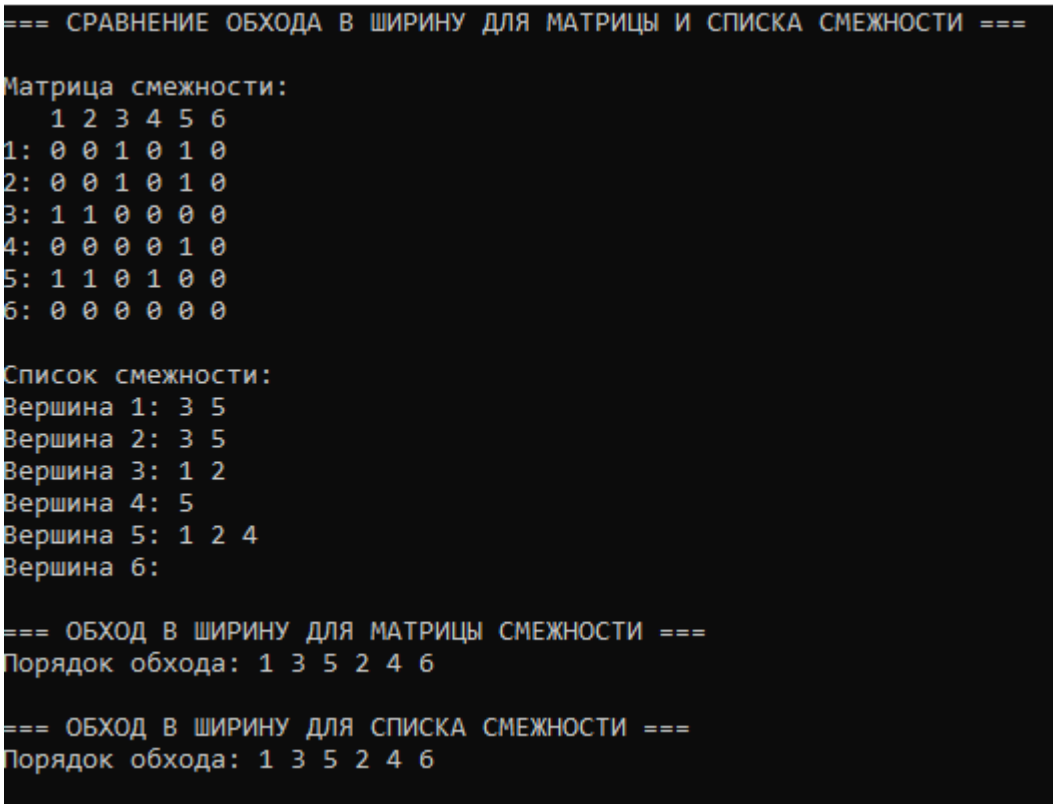

Рисунок 1 - Результат работы программы

Вывод: В ходе выполнения лабораторной работы был успешно изучен и реализован алгоритм обхода графа в ширину для различных структур данных: матрицы смежности и списков смежности.