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 22ВВВ2:

Юртаев Д. Ю.

Шатин Д. Д.

Харитонов А. А.

Приняли:

Акифьев И. В.

Митрохин М. А.

Пенза 2023

**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 С++.

**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.

**Задание №1.1.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#include <clocale>

#define MAX\_VERTICES 6 // Максимальное количество вершин

#define MAX\_QUEUE\_SIZE 100

// Функция для генерации случайного числа от 0 до 1

double randomProbability() {

return (double)rand() / RAND\_MAX;

}

// Генерация случайной матрицы смежности

void generateRandomAdjacencyMatrix(int matrix[MAX\_VERTICES][MAX\_VERTICES], int numVertices, int numEdges) {

int i, j;

for (i = 0; i < numVertices; i++) {

for (j = 0; j < numVertices; j++) {

matrix[i][j] = 0;

}

}

int edgeCount = 0;

while (edgeCount < numEdges) {

int vertex1 = rand() % numVertices;

int vertex2 = rand() % numVertices;

if (vertex1 != vertex2 && matrix[vertex1][vertex2] == 0) {

matrix[vertex1][vertex2] = 1;

matrix[vertex2][vertex1] = 1;

edgeCount++;

}

}

}

// Функция для обхода в ширину

void breadthFirstSearch(int graph[MAX\_VERTICES][MAX\_VERTICES], int startVertex, int numVertices) {

bool visited[MAX\_VERTICES] = { 0 };

int queue[MAX\_QUEUE\_SIZE];

int front = 0, rear = 0;

visited[startVertex] = true;

printf("Посещенная вершина: %d\n", startVertex);

queue[rear++] = startVertex;

while (front != rear) {

int currentVertex = queue[front++];

for (int neighbor = 0; neighbor < numVertices; neighbor++) {

if (graph[currentVertex][neighbor] == 1 && !visited[neighbor]) {

visited[neighbor] = true;

printf("Посещенная вершина: %d\n", neighbor);

queue[rear++] = neighbor;

}

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int numVertices = 6; // Задайте желаемое количество вершин

int numEdges = 8; // Задайте желаемое количество ребер

int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES];

generateRandomAdjacencyMatrix(adjacencyMatrix, numVertices, numEdges);

printf("Матрица смежности:\n");

for (int i = 0; i < numVertices; i++) {

for (int j = 0; j < numVertices; j++) {

printf("%d ", adjacencyMatrix[i][j]);

}

printf("\n");

}

int startVertex;

printf("Введите начальную вершину для обхода в ширину (от 0 до %d): ", numVertices - 1);

scanf("%d", &startVertex);

if (startVertex < 0 || startVertex >= numVertices) {

printf("Недопустимая начальная вершина. Пожалуйста, введите вершину между 0 и %d.\n", numVertices - 1);

return 1; // Выход с кодом ошибки

}

printf("Обход в ширину, начиная с вершины %d:\n", startVertex);

breadthFirstSearch(adjacencyMatrix, startVertex, numVertices);

return 0;

}

**Задание №1.3**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <clocale>

#define MAX\_VERTICES 6

// Функция для добавления ребра между вершинами u и v

void addEdge(int adjacencyList[MAX\_VERTICES][MAX\_VERTICES], int u, int v) {

adjacencyList[u][v] = 1;

adjacencyList[v][u] = 1; // Для ненаправленного графа

}

// Процедура для вывода списка смежности графа

void printAdjacencyList(int adjacencyList[MAX\_VERTICES][MAX\_VERTICES], int numVertices) {

printf("Список смежности графа:\n");

for (int i = 0; i < numVertices; i++) {

printf("Вершина %d: ", i);

for (int j = 0; j < numVertices; j++) {

if (adjacencyList[i][j] == 1) {

printf("%d ", j);

}

}

printf("\n");

}

}

// Процедура обхода в ширину

void breadthFirstSearch(int adjacencyList[MAX\_VERTICES][MAX\_VERTICES], int startVertex, int numVertices) {

bool visited[MAX\_VERTICES] = { false };

int queue[MAX\_VERTICES];

int front = 0, rear = 0;

visited[startVertex] = true;

printf("Посещенная вершина: %d\n", startVertex);

queue[rear++] = startVertex;

while (front < rear) {

int currentVertex = queue[front++];

for (int neighbor = 0; neighbor < numVertices; neighbor++) {

if (adjacencyList[currentVertex][neighbor] == 1 && !visited[neighbor]) {

visited[neighbor] = true;

printf("Посещенная вершина: %d\n", neighbor);

queue[rear++] = neighbor;

}

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int numVertices = 6;

int adjacencyList[MAX\_VERTICES][MAX\_VERTICES] = { 0 };

// Добавление ребер между вершинами (пример)

addEdge(adjacencyList, 0, 1);

addEdge(adjacencyList, 0, 2);

addEdge(adjacencyList, 1, 3);

addEdge(adjacencyList, 2, 3);

addEdge(adjacencyList, 3, 4);

addEdge(adjacencyList, 4, 5);

printAdjacencyList(adjacencyList, numVertices);

int startVertex;

printf("Введите начальную вершину для обхода в ширину (от 0 до %d): ", numVertices - 1);

scanf("%d", &startVertex);

if (startVertex < 0 || startVertex >= numVertices) {

printf("Недопустимая начальная вершина. Пожалуйста, введите вершину между 0 и %d.\n", numVertices - 1);

return 1; // Выход с кодом ошибки

}

printf("Обход в ширину, начиная с вершины %d:\n", startVertex);

breadthFirstSearch(adjacencyList, startVertex, numVertices);

return 0;

}

**Задание 2.**

**Результаты работы программ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 1 - Результат работы программы lab8.12**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Рисунок 2 - Результат работы программы lab8.1.3**

**Вывод**

В результате выполнения лабораторной работы были успешно реализованы алгоритмы построения матрицы смежности, усвоены и практически проверены навыки в алгоритме реализации обхода графа в ширину.