**Лабораторная работа №8**

**Обход графа в ширину**

**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.

Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс **queue** и использующего очередь, реализованную самостоятельно) для графов разных порядков.

**Ответ задание :**

#include <iostream>

#include <vector>

#include <queue>

#include <ctime>

#include <cstdlib>

#include <chrono>

#include <windows.h>

using namespace std;

// Функция для генерации случайного графа

vector<vector<int>> generateRandomGraph(int n) {

vector<vector<int>> graph(n, vector<int>(n, 0));

srand(time(0));

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

for (int j = i + 1; j < n; ++j) {

// Добавляем ребра случайным образом

graph[i][j] = graph[j][i] = rand() % 2;

}

}

return graph;

}

// Функция для обхода в ширину с использованием queue из стандартной библиотеки

void BFSUsingQueue(const vector<vector<int>>& graph, int start) {

int n = graph.size();

vector<bool> visited(n, false);

queue<int> q;

q.push(start);

visited[start] = true;

while (!q.empty()) {

int node = q.front();

q.pop();

cout << node << " ";

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

if (graph[node][i] && !visited[i]) {

q.push(i);

visited[i] = true;

}

}

}

}

// Функция для обхода в ширину с использованием самостоятельной реализации очереди

void BFSUsingCustomQueue(const vector<vector<int>>& graph, int start) {

// Ваш код для реализации обхода в ширину с использованием своей очереди

}

int main() {

// Задайте размер графа

int n = 5;

// Генерация и вывод матрицы смежности

vector<vector<int>> graph = generateRandomGraph(n);

setlocale(LC\_ALL, "rus");

cout << "Матрица смежности:" << endl;

for (const auto& row : graph) {

for (int value : row) {

cout << value << " ";

}

cout << endl;

}

// Обход в ширину с использованием queue

cout << "BFS использует очередь, начиная с узла 0: ";

BFSUsingQueue(graph, 0);

cout << endl;

// Обход в ширину с использованием своей очереди

// cout << "BFS using custom queue starting from node 0: ";

// BFSUsingCustomQueue(graph, 0);

// cout << endl;

// Ваш код для оценки времени выполнения

auto start = chrono::high\_resolution\_clock::now();

BFSUsingQueue(graph, 0);

auto stop = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::microseconds>(stop - start);

cout << "Время, затраченное BFS при использовании очереди: " << duration.count() << " микросекунды" << endl;

return 0;

}

