Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск в ширину»

**Выполнили студенты группы 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**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.

**Листинг программы:**

**Задание 1:**

#include "StdAfx.h"

#include <iostream>

#include <queue>

#include "time.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

//const int size = 7;

queue<int> Queue;

int size;

cout << "Введите размер матрицы: ";

cin >> size;

int\*\* mas = (int\*\*)malloc(sizeof(int\*) \* size);

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

mas[i] = (int\*)malloc(sizeof(int) \* size);

}

srand(time(NULL));

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

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

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

}

}

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

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

if (i == j) {

mas[i][j] = 0;

}

if (mas[i][j] = mas[j][i]) {

mas[i][j] = mas[j][i];

}

}

}

for (size\_t i = 0; i < size; ++i) {

for (size\_t j = 0; j < size; ++j)

cout << mas[i][j] << ' ';

cout << endl;

}

int \*nodes = new int [size];

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

nodes[i] = 0;

int start = clock();

Queue.push(0);

cout << "Обход в ширину: ";

while (!Queue.empty())

{

int node = Queue.front();

Queue.pop();

nodes[node] = 2;

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

{

if (mas[node][j] == 1 && nodes[j] == 0)

{ // если вершина смежная и не обнаружена

Queue.push(j);

nodes[j] = 1;

}

}

cout << node + 1 << " ";

}

int end = clock();

double time = (end-start) / 1000.0;

printf("\nTime: %lf", time);

cin.get();

return 0;

}

**Задание 2:**

#include "StdAfx.h"

#include "time.h"

#include <iostream>

using namespace std;

struct node

{

int inf; // полезная информация

struct node \*next; // ссылка на следующий элемент

};

//void spstore(int j);

struct node \*get\_struct(int priem);

struct node \*head = NULL, \*last = NULL, \*f = NULL; // указатели на первый и последний элементы списка

void spstore(int j)

{

//int priem = j;

struct node \*p = NULL;

p = get\_struct(j);

if (head == NULL && p != NULL) // если списка нет, то устанавливаем голову списка

{

head = p;

last = p;

}

else if (head != NULL && p != NULL) // список уже есть, то вставляем в конец

{

last->next = p;

last = p;

}

return;

}

struct node \*get\_struct(int priem)

{

struct node \*p = NULL;

if ((p = (node\*)malloc(sizeof(struct node))) == NULL) // выделяем память под новый элемент списка

{

printf("Ошибка при распределении памяти\n");

exit(1);

}

p->inf = priem;

p->next = NULL;

return p; // возвращаем указатель на созданный элемент

}

int remove(){

struct node \*struc = head; // указатель, проходящий по списку установлен на начало списка

struct node \*prev;// указатель на предшествующий удаляемому элемент

if (head->next != NULL) // если найденный элемент последний в списке

{

//printf("\nПоследний элемент - %d, \n", struc->inf);

int vozvr = struc->inf;

head=head->next; // обнуляем указатель предшествующего элемента

free(struc);

return vozvr;

}

else

{

int vozvr = struc->inf;

head = NULL;

free(struc);

return vozvr;

}

return 0;

}

int first(){

struct node \*struc = head; // указатель, проходящий по списку установлен на начало списка

struct node \*prev;// указатель на предшествующий удаляемому элемент

if (head->next != NULL) // если найденный элемент последний в списке

{

printf("Последний элемент - %s, \n", struc->inf);

int vozvr = struc->inf;

return vozvr;

}

return 0;

}

int main()

{

struct node \*struc = head;

setlocale(LC\_ALL, "Russian");

int size;

//int\*\* mas;

//queue<int> Queue;

//int nodes[size];

cout << "Введите размер матрицы: ";

cin >> size;

int\*\* mas = (int\*\*)malloc(sizeof(int\*) \* size);

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

mas[i] = (int\*)malloc(sizeof(int) \* size);

}

srand(time(NULL));

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

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

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

}

}

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

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

if (i == j) {

mas[i][j] = 0;

}

if (mas[i][j] = mas[j][i]) {

mas[i][j] = mas[j][i];

}

}

}

for (size\_t i = 0; i < size; ++i) {

for (size\_t j = 0; j < size; ++j)

cout << mas[i][j] << ' ';

cout << endl;

}

int \*nodes = new int [size];

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

nodes[i] = 0;

int start = clock();

spstore(0);

cout << "\nОбход в ширину: ";

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

if(nodes[i] == 0){

nodes[i] = 1;

// cout << i+1 << " ";

}

while (!struc)

{

int node = remove();

nodes[node] = 2;

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

{

if (mas[node][j] == 1 && nodes[j] == 0)

{ // если вершина смежная и не обнаружена

// cout << j << " ";

spstore(j);

nodes[j] = 1;

}

}

cout << node + 1 << " ";

}

}

int end = clock();

double time = (end-start) / 1000.0;

printf("Time: %lf", time);

// cin.get();

return 0;

}

**Ход работы:**

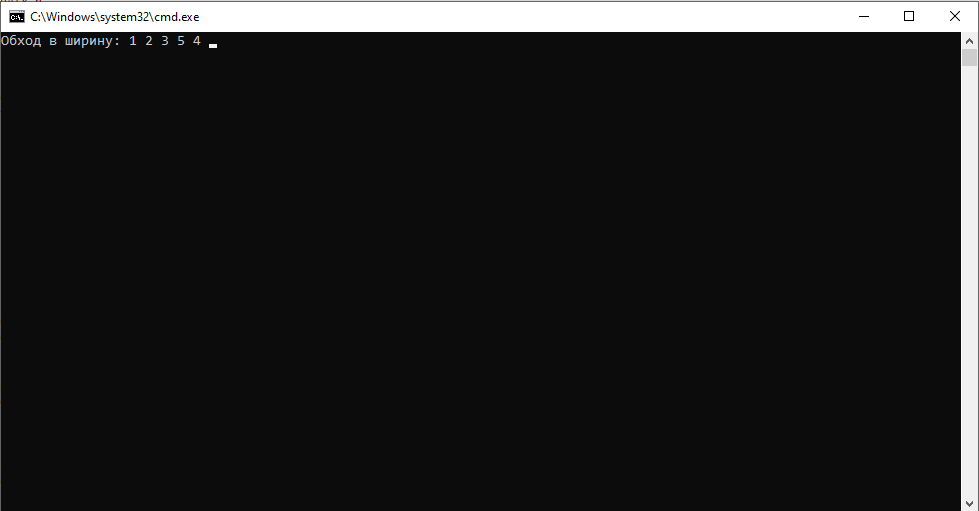
****

Рисунок 1 – результат работы программы.

**Вывод:** в ходе выполнения лабораторной работы были разработана программа, выполняющая обход графа в ширину. В результате сравнения быстродействия работы алгоритмов очереди, реализованной на структуре данных, и очереди, реализованной на встроенном классе Queue, выяснилось, что их быстродействие примерно одинаково на разных диапазонах входных данных.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с применением поиска в глубину на языке Си.