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»

**Выполнил студент группы 21вв1:**

Вартанов А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**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++.

**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:**

Задание 1:

#include <stdlib.h>

#include <stdio.h>

#include <queue>

#include "time.h"

#include <locale.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

queue<int> Queue;

int size;

printf("Введите размер матрицы: ");

scanf("%d", &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)

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

printf("\n");

}

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

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

nodes[i] = 0;

int start = clock();

Queue.push(0);

printf("Обход в ширину:\n");

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;

}

}

printf(">%d ", node + 1);

}

int end = clock();

double time = (end-start) / 1000.0;

printf("\nПройденное время: %lf \n", time);

system("Pause");

return 0;

}

Задание 2:

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include "time.h"

#include <iostream>

using namespace std;

struct node

{

int inf; // полезная информация

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

};

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)

{

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

{

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 main()

{

struct node \*struc = head;

setlocale(LC\_ALL, "Russian");

int size;

printf("Введите размер матрицы: ");

scanf("%d",&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){

printf("\n");

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

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

}

}

int \*nodes = new int [size];

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

nodes[i] = 0;

int verch;

printf("\nВведите стартовую вершину: ");

scanf("%d", &verch);

spstore(verch);

int start = clock();

printf("\nОбход в ширину: ");

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

if(nodes[i] == 0){

nodes[i] = 1;

}

while (head)

{

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

spstore(j);

nodes[j] = 1;

}

}

printf(">%d",node + 1);

}

}

int end = clock();

double time = (end-start) / 1000.0;

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

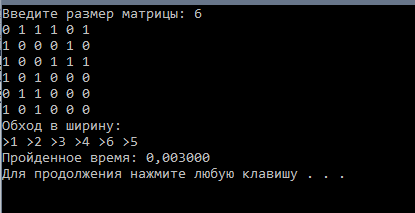
system("Pause");

return 0;

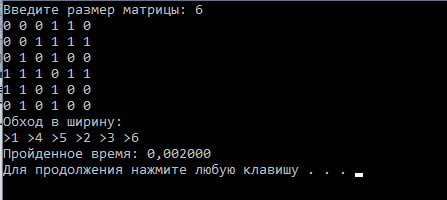
}

**Результат работы программы:**

Задание 1:



Задание 2:



Алгоритм генерирует матрицу и полностью выполняет поиск в ширину. Программа выполняет поставленную задачу.

**Вывод:**

Я научился создавать функцию поиска в ширину, реализовал алгоритмы обхода в ширину, использующего стандартный класс **queue** и очередь.