Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №5  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему: «Обход графа в ширину»

**Выполнил студент группы 19ВВ1:**

Балалаев А.А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

к.т.н. Юрова О.В.

Пенза 2020

**Цель работы:** изучить обход в ширину, научится реализовывать его, познакомится с очередью, реализованной библиотекой queue.

**Теория**

Обход графа в ширину – еще один распространенный способ обхода

графов.

Таким образом, можно предложить следующую реализацию алгоритма

обхода в ширину.

Вход: G – матрица смежности графа.

Выход: номера вершин в порядке их прохождения на экране.

Алгоритм ПОШ

1.1. для всех i положим NUM[i] = False пометим как &quot;не посещенную&quot;;

1.2. ПОКА существует &quot;новая&quot; вершина v

1.3. ВЫПОЛНЯТЬ BFS (v).

Алгоритм BFS(v):

2.1. Создать пустую очередь Q = {};

2.2. Поместить v в очередь Q.push(v);

2.3. пометить v как &quot;посещенную&quot; NUM[v] = True;

2.4. ПОКА Q != Æ очередь не пуста ВЫПОЛНЯТЬ

2.5. v = Q.front() установить текущую вершину;

2.6. Удалить первый элемент из очереди Q.pop();

2.7. вывести на экран v;

2.8. ДЛЯ i = 1 ДО size\_G ВЫПОЛНЯТЬ

2.9. ЕСЛИ G(v,i) = = 1И NUM[i] = = False

2.10. ТО

2.11. Поместить i в очередь Q.push(i);

2.12. пометить v как &quot;посещенную&quot; NUM[v] = True;

## Методы очереди

**Метод** — это та же самая функция, но она работает только с контейнерами STL. Например, очередь и [стек](https://codelessons.ru/cplusplus/realizaciya-steka-stack-v-c.html).

Для работы с очередью вам понадобится знать [функции](https://codelessons.ru/cplusplus/funkcii-function-v-c-peregruzki-i-prototipy-funkcij.html): push(), pop(), front(), back(), empty(). Кстати, если хотите узнать, как в C++ работают функции и как их правильно использовать в проекте, то можете узнать все это[здесь](https://codelessons.ru/cplusplus/funkcii-function-v-c-peregruzki-i-prototipy-funkcij.html).

1. Для **добавления в очередь нового элемента**нужно воспользоваться функцией — push(). В круглых скобках должно находится значение, которое мы хотим добавить.
2. Если нам понадобилось **удалить первый элемент** нужно оперировать функцией pop(). В круглых скобках уже не чего не нужно указывать, но по правилам они в обязательном порядке должны присутствовать! Эти функции тоже не нуждаются в указании аргумента: empty(), back() и front().
3. Если вам понадобилось **обратиться к первому элементу**очереди, то вам понадобится функция front().
4. Чтобы **обратиться к последнему элементу** в очередивам поможет функция back().
5. Чтобы узнать **пуста ли очередь** нужно воспользоваться функцией empty().
   * Если ваша очередь пуста — возвратит true.
   * Если же в ней что-то есть — возвратит false.

**Код программы**

// la4.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "windows.h"

#include "stdio.h"

#include "time.h"

#include "conio.h"

#include <queue>

#include <locale.h>

#define SIZE 5

using namespace std;

void rand\_Zap(int\* mat, int n)

{

srand(time(NULL));

printf("G\n");

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

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

{

if (i == j)

{

mat[i \* n + j] = 0;

}

if (i < j) {

mat[i \* n + j] = rand() % 2;

mat[j \* n + i] = mat[i \* n + j];

}

}

}

void print\_G(int\* mat, int n)

{

printf(" ");

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

printf("%3d", i + 1);

printf("\n\n");

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

{

printf("%2d", i + 1);

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

{

printf("%3d", mat[i \* n + j]);

}

printf("\n");

}

printf("\n");

}

void BFS(int s, int\* G, int n, int\* vis)

{

queue <int> Q;

Q.push(s);

vis[s] = 1;

while (!Q.empty()) {

s = Q.front();

printf("%d ", s + 1);

Q.pop();

for (int i = 0; i < 5; i++) {

if ((G[s \* n + i] == 1) && !vis[i]) {

Q.push(i);

vis[i] = 1;

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

int N = 1;

scanf("%d", &N);

int\*\* G = (int\*\*)malloc(N \* sizeof(int\*));

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

G[i] = (int\*)malloc(N \* sizeof(int));

}

int\* vis;

vis = (int\*)malloc(N \* sizeof(int));

int s, i = 0;

rand\_Zap(&G[0][0], N);

print\_G(&G[0][0], N);

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

{

vis[i] = 0;

}

printf("Введите вершину для начала обхода\n");

scanf("%d", &s);

while ((s < 1) || (s > N))

{

printf("Ошибка.\n");

scanf("%d", &s);

}

printf("\n");

s--;

printf("Результат:\n");

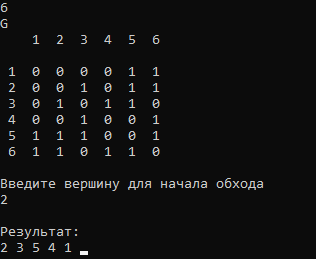
BFS(s, &G[0][0], N, &vis[0]);

getchar();

getchar();

}

**Результат работы кода**



**Вывод:** изучил обход в ширину, научился реализовывать его, познакомился с очередью, реализованной библиотекой queue.