Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №5

по дисциплине: «Логика и основы оптимизации в инженерных задачах»

## на тему «Обход графа в ширину»

Выполнили студенты группы 19ВВ2:

Муромский Д.А.

Кобзев М.И.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** освоить алгоритм обхода графа в ширину на матрицах и списках смежности. Научиться реализовывать алгоритм обхода графа без использования рекурсии (с помощью очереди). Закрепить работу со списками путём реализации очереди и сравнить время выполнения стандартной функции очереди со runtime своей функции.

**Практическая часть**

1. **Задание 1.1** Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.

srand(time(NULL));

//Ввод

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

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

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

a[j][i] = a[i][j];

}

}

//Обнуление диагонали

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

a[i][i] = 0;

}

//Одномерные массивы

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

vis[i] = 0;

}

//Вывод

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

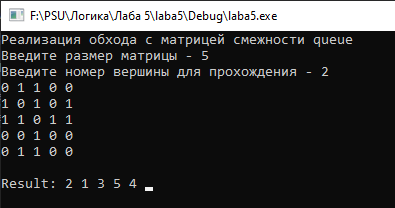
for(int j = 0; j < n; j++){

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

}

printf("\n");

}



1. **Задание 1.2** 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++.

void dfs\_norec(int start, int \*vis, int \*\*a, int n) {

queue<int> Steck;

vis[start] = 1;

Steck.push(start);

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

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

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

if (a[v][i] && (!vis[i])) {

Steck.push(i);

vis[i] = 1;

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

}

}

}

}

void zad1(){

setlocale(LC\_ALL, "Rus");

int n, v;

printf("Реализация обхода с матрицей смежности queue\n");

printf("Введите размер матрицы - ");

scanf("%d", &n);

printf("Введите номер вершины для прохождения - ");

scanf("%d", &v);

int \*\*a = dynamic\_array(n); //матрица смежности

int \*vis = dynamic\_array2(n); //проверка на прохождение

srand(time(NULL));

//Ввод

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

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

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

a[j][i] = a[i][j];

}

}

//Обнуление диагонали

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

a[i][i] = 0;

}

//Одномерные массивы

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

vis[i] = 0;

}

//Вывод

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

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

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

}

printf("\n");

}

printf("\nResult: ");

v--;

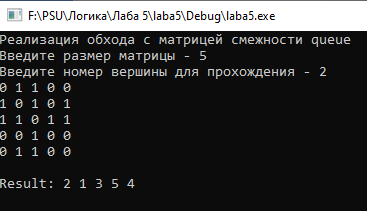
dfs\_norec(v, vis, a, n);

dynamic\_array\_free(a, n);

dynamic\_array\_free1(vis, n);

\_getch();

}



**Задание 1.3** Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

void dfslist(int start, int \*vis, List \*a, int n) {

queue<int> Steck;

vis[start] = 1;

Steck.push(start);

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

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

for (int i = 0; i < a[v].GetSize(); i++) {

if (!vis[a[v][i]]) {

Steck.push(a[v][i]);

vis[a[v][i]] = 1;

printf("%d ", a[v][i] + 1);

}

}

}

}

void zad2() {

setlocale(LC\_ALL, "Rus");

int n, v;

printf("\n\nРеализация обхода со списком смежности queue\n");

printf("Введите размер матрицы - ");

scanf("%d", &n);

printf("Введите номер вершины для прохождения - ");

scanf("%d", &v);

List \*relist = new List[n];

int \*\*a = dynamic\_array(n); //матрица смежности

int \*vis = dynamic\_array2(n); //проверка на прохождение

srand(time(NULL));

//Ввод

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

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

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

a[j][i] = a[i][j];

}

}

//Обнуление диагонали

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

a[i][i] = 0;

vis[i] = 0;

}

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

printf("V%d:", i + 1);

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

if (a[i][j]) {

if (relist[i].GetSize() == 0) {

relist[i].push\_front(j);

printf(" v%d", j + 1);

}

else {

relist[i].push\_back(j);

printf(" v%d", j + 1);

}

}

}

printf("\n");

}

printf("\nResult: ");

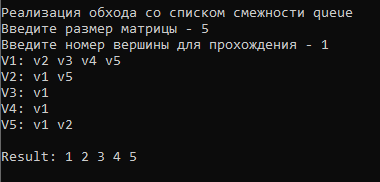
v--;

dfslist(v, vis, relist, n);

delete[] relist;

\_getch();

}



1. **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.

void dfs\_norec\_list(int start, int \*vis, int \*\*a, int n) {

List Stack;

vis[start] = 1;

Stack.push\_front(start);

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

int v;

while (Stack.GetSize()) {

v = Stack[0];

Stack.pop\_front();

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

if (a[v][i] && (!vis[i])) {

Stack.push\_back(i);

vis[i] = 1;

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

}

}

}

}

void zad3() {

setlocale(LC\_ALL, "Rus");

int n, v;

clock\_t start, end;

printf("\n\nРеализация обхода с матрицей смежности list\n");

printf("Введите размер матрицы - ");

scanf("%d", &n);

printf("Введите номер вершины для прохождения - ");

scanf("%d", &v);

int \*\*a = dynamic\_array(n); //матрица смежности

int \*vis = dynamic\_array2(n); //проверка на прохождение

srand(time(NULL));

//Ввод

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

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

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

a[j][i] = a[i][j];

}

}

//Обнуление диагонали

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

a[i][i] = 0;

}

//Одномерные массивы

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

vis[i] = 0;

//ok[i] = 0;

}

//Вывод

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

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

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

}

printf("\n");

}

printf("\nResult: ");

v--;

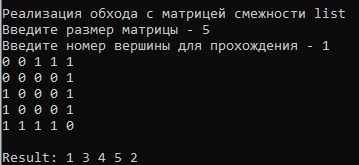
dfs\_norec\_list(v, vis, a, n);

dynamic\_array\_free(a, n);

dynamic\_array\_free1(vis, n);

\_getch();

}



**Задание 2.2\*** Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс queue и использующего очередь, реализованную самостоятельно) для графов разных порядков.

void dfs\_norec\_no(int start, int \*vis, int \*\*a, int n) {

List Stack;

vis[start] = 1;

Stack.push\_front(start);

int v;

while (Stack.GetSize()) {

v = Stack[0];

Stack.pop\_front();

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

if (a[v][i] && (!vis[i])) {

Stack.push\_back(i);

vis[i] = 1;

}

}

}

}

void dfs\_norec\_no\_lib(int start, int \*vis, int \*\*a, int n) {

queue<int> Steck;

vis[start] = 1;

Steck.push(start);

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

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

if (a[v][i] && (!vis[i])) {

Steck.push(i);

vis[i] = 1;

}

}

}

void zvezdochki() {

setlocale(LC\_ALL, "Rus");

int n, v;

clock\_t start, end;

printf("\n\nСравнение времени обхода матрицы смежности list и queue\n");

printf("Введите размер матрицы - ");

scanf("%d", &n);

printf("Введите номер вершины для прохождения - ");

scanf("%d", &v);

int \*\*a = dynamic\_array(n); //матрица смежности

int \*vis = dynamic\_array2(n); //проверка на прохождение

srand(time(NULL));

//Ввод

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

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

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

a[j][i] = a[i][j];

}

}

//Обнуление диагонали

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

a[i][i] = 0;

}

//Одномерные массивы

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

vis[i] = 0;

//ok[i] = 0;

}

printf("\nResult: ");

v--;

start = clock();

dfs\_norec\_no(v, vis, a, n);

end = clock();

printf("\nUse %.2f msecond.\n", difftime(end, start));

printf("\nResult\_Lib: ");

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

vis[i] = 0;

}

start = clock();

dfs\_norec\_no\_lib(v, vis, a, n);

end = clock();

printf("\nUse %.2f msecond.\n", difftime(end, start));

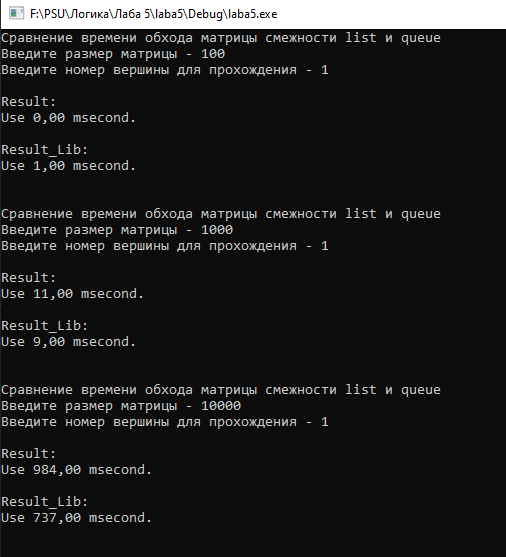
dynamic\_array\_free(a, n);

dynamic\_array\_free1(vis, n);

\_getch();

}

Результаты тестов:



**Вывод:** освоили алгоритм обхода графа в ширину на матрицах и списках смежности. Научились реализовывать алгоритм обхода графа без использования рекурсии (с помощью очереди). Закрепили работу со списками путём реализации очереди и сравнили время выполнения стандартной функции очереди со runtime своей функции. В результате сравнения выяснилось , что стандартная функция реализации очереди работает быстрее, чем написанная вручную.