Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №7

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

на тему «Поиск расстояний во взвешенном графе»

Выполнили студенты группы 19ВВ2:

Муромский Д.А.

Кобзев М.И.

Приняли:

Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** закрепить работу с алгоритмом поиска расстояний в графе в ширину. Научится создавать различные комбинации графов (ориентированный взвешенный, ориентированный не взвешенный, неориентированный взвешенный, неориентированный не взвешенный). Научиться запускать программу с параметрами командной строки.

**Практическая часть**

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

srand(time(NULL));

//Взвешенный неориентированный

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

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

if (rand() % 2) {

a[i][j] = 1 + rand() % 9;

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

}

else {

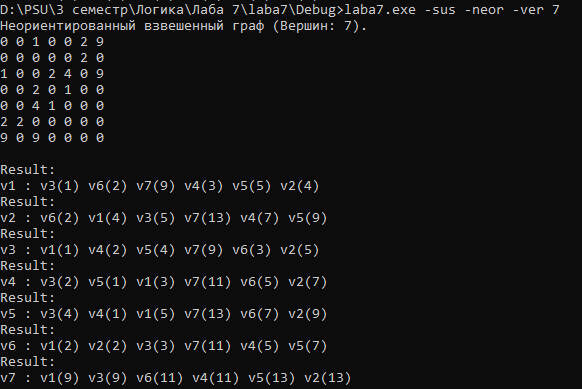
a[i][j] = 0;

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

}

}

}



**Задание 1.2** Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

void BFSD(int start, int\* dist, int\*\* a, int n) {

queue<int> Steck;

Steck.push(start);

dist[start] = 0;

printf("\nResult:\n");

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

int v;

while (!Steck.empty()) {

v = Steck.front();

Steck.pop();

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

if (a[v][i] && (dist[i] == -1)) {

Steck.push(i);

dist[i] = dist[v] + a[v][i];

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

}

}

}

}

void zad(bool orflag, bool susflag, int n){

setlocale(LC\_ALL, "Rus");

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

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

if (orflag) { // ориентированный

if (susflag) { // взвешенный

srand(time(NULL));

//Ввод

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

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

if (rand() % 2) {

a[i][j] = 1 + rand() % 9;

}

else {

a[i][j] = 0;

}

}

}

}

else { // не взвешенный

srand(time(NULL));

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

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

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

}

}

}

}

else { // неориентированный

if (susflag) { // взвешенный

srand(time(NULL));

//Взвешенный неориентированный

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

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

if (rand() % 2) {

a[i][j] = 1 + rand() % 9;

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

}

else {

a[i][j] = 0;

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

}

}

}

}

else { // не взвешенный

srand(time(NULL));

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] = -1;

}

//Вывод

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

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

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

}

printf("\n");

}

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

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

vis[j] = -1;

}

BFSD(i, vis, a, n);

}

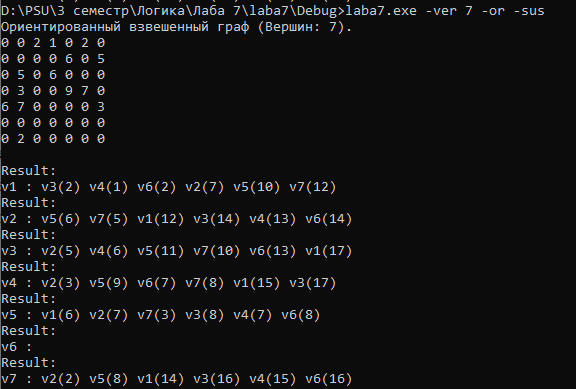
dynamic\_array\_free(a, n);

dynamic\_array\_free1(vis, n);

\_getch();

}

**Задание 1.3\*** Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.



**Задание 2.1\*** Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки (см. описание ниже). В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

int main(int argc, char\* argv[]){

Spirit(argc, argv);

}

void Spirit(int argc, char\* argv[]){

setlocale(LC\_ALL, "Rus");

bool orflag(false);

bool susflag(false);

int N = 6;

if (argc > 1)

{

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

{

if (strcmp(argv[i], "-or") == 0) {

orflag = true;

}

if (strcmp(argv[i], "-neor") == 0) {

orflag = false;

}

if (strcmp(argv[i], "-sus") == 0) {

susflag = true;

}

if (strcmp(argv[i], "-nesus") == 0) {

susflag = false;

}

if (strcmp(argv[i], "-ver") == 0) {

stringstream convert(argv[i + 1]); // создаем переменную stringstream с именем convert, инициализируя её значением argv[1]

int myint;

if (!(convert >> myint)) // выполняем конвертацию

myint = 6;

N = myint;

}

}

}

else

{

cout << "Аргументы будут использоваться по умолчанию." << endl;

}

if (orflag) {

cout << "Ориентированный ";

}else{

cout << "Неориентированный ";

}

if (susflag) {

cout << "взвешенный граф ";

}else{

cout << "не взвешенный граф ";

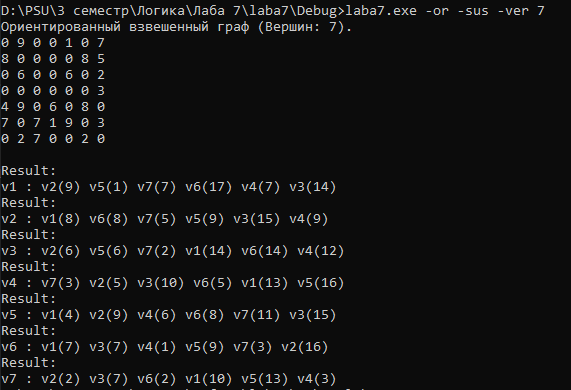
}

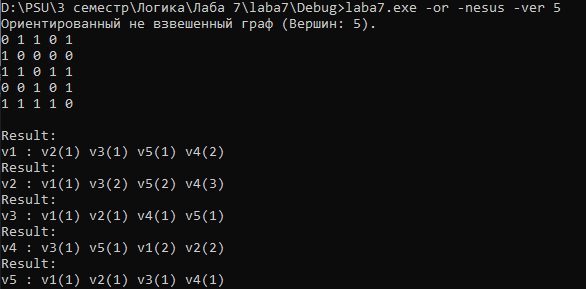
cout << "(Вершин: " << N << ")." << endl;

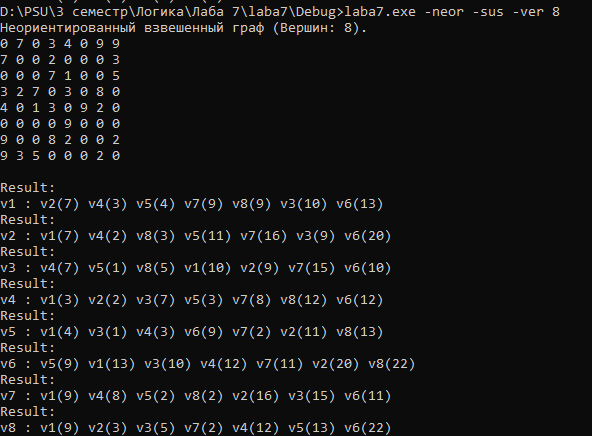
zad(orflag, susflag, N);

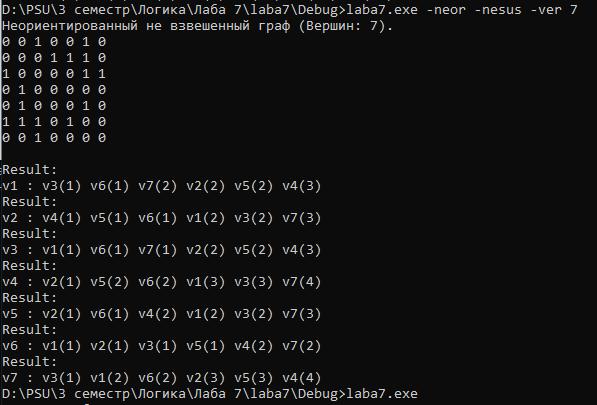
\_getch();

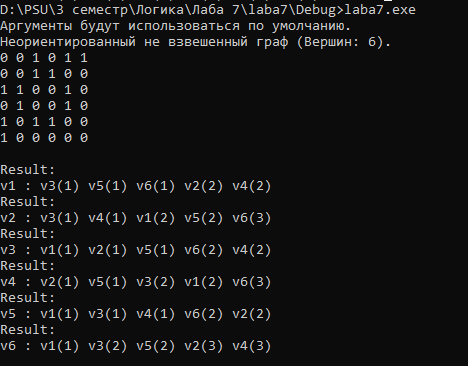
}











**Вывод:** **Цель работы:** закрепили работу с алгоритмом поиска расстояний в графе в ширину. Научились создавать различные комбинации графов (ориентированный взвешенный, ориентированный не взвешенный, неориентированный взвешенный, неориентированный не взвешенный). Научились запускать программу с параметрами командной строки.