**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №4.1

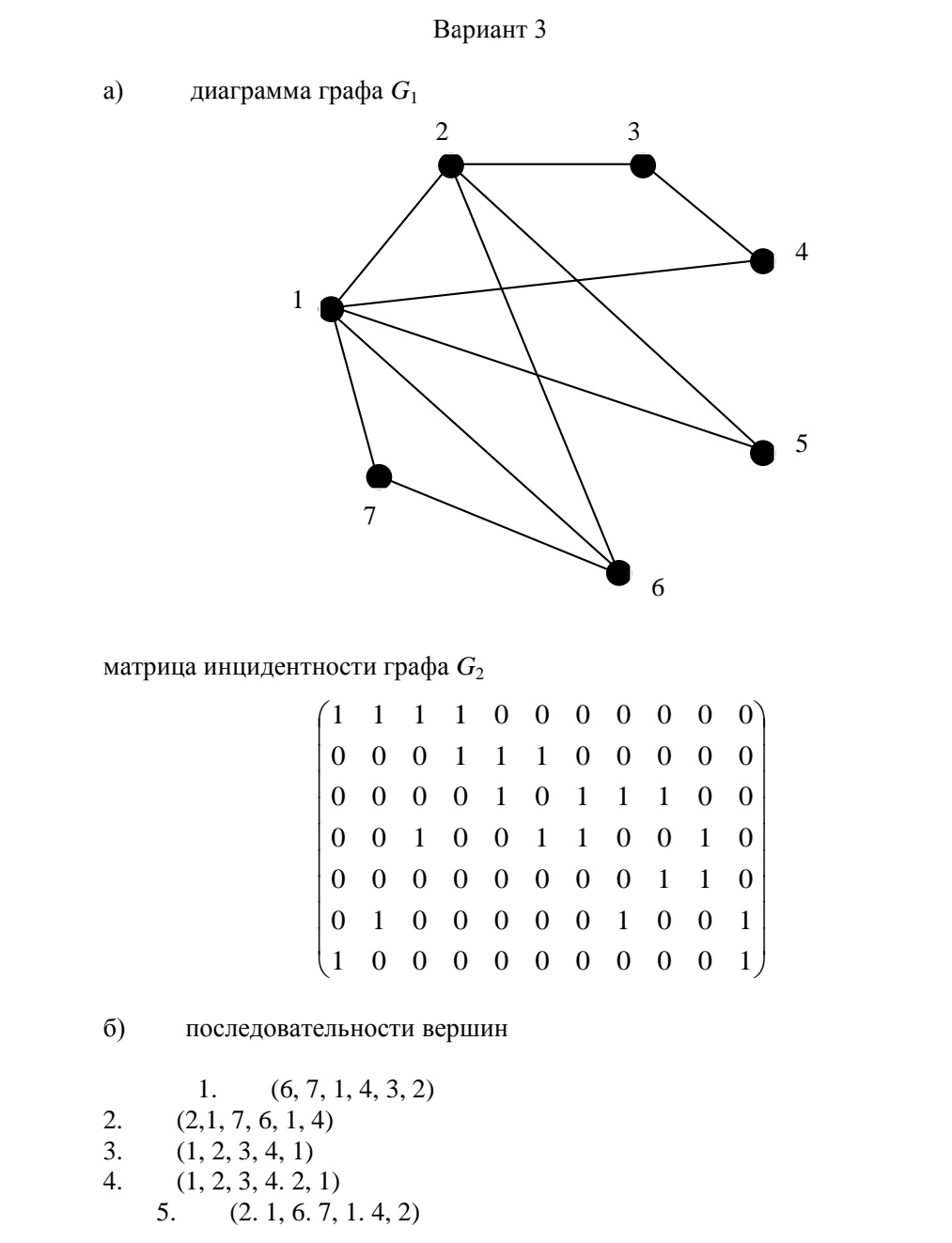
тема: «Маршруты»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ПВ-21  Проверил: Рязанов Ю.Д. |

Белгород

2019

**Цель занятия**: изучить основные понятия теории графов, способы задания графов, научиться программно реализовывать алгоритмы получения и анализа маршрутов в графах.

****

**Задание 2**

**2.** Определить, являются ли последовательности вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

1. (6, 7, 1, 4, 3, 2)

Для G1: маршрут, цепь, простая цепь

Для G2: маршрут, цепь, простая цепь

1. (2, 1, 7, 6, 1, 4)

Для G1: маршрут, цепь, цикл

Для G2: маршрут, цепь, цикл

1. (1, 2, 3, 4, 1)

Для G1: маршрут, цикл, цепь, простой цикл

Для G2: маршрут, цикл, цепь, простой цикл

1. (1, 2, 3, 4, 2, 1)

Для G1: не является маршрутом

Для G2: маршрут, цепь, цикл

1. (2, 1, 6, 7, 1, 4, 2)

Для G1: не является маршрутом

Для G2: маршрут, цепь, цикл

**Задание 3**

#include <stdio.h>

#include <locale.h>

#define N 15

int n;// Длинна последовательности

int m;// Мощность квадратной матрицы смежности графа

int\* init\_posl(int);

int\*\* init\_graf(int);

int is\_marshrut(int \*, int \*\*);

int is\_cep(int \*);

int is\_easy\_cep(int \*);

int is\_easy\_cikl(int \*);

int is\_cikl(int \*);

int main() {

setlocale(LC\_ALL, "Rus");

int \*a; // массив последовательности

int \*\*graf; // Матрица графа

printf("\nВведите длинну последовательности: ");

scanf("%d", &n);

a = init\_posl(n);

printf("\nВведите мощность множества элементов графа: ");

scanf("%d", &m);

graf = init\_graf(m);

if (is\_marshrut(a, graf)) {

printf("\n Данная последовательность для данного графа является маршрутом!");

if (is\_cep(a)) {

printf("\n Данная последовательность для данного графа является цепью!");

if (is\_easy\_cep(a))

printf("\n Данная последовательность для данного графа является простой цепью!");

if (is\_easy\_cikl(a))

printf("\n Данная последовательность для данного графа является простым циклом!");

}

else {

printf("\n Данная последовательность для данного графа не является цепью, а значит и пр. цепью и пр. циклом!");

}

if (is\_cikl(a))

printf("\n Данная последовательность для данного графа является циклом!");

}

else {

printf("\n Данная последовательность для данного графа не является маршрутом! Следовательно и ничем больше.");

}

system("pause");

return 0;

}

// Является ли данная последовательность pos маршрутом для графа graf

int is\_marshrut(int \*pos, int \*\*graf) {

int flag = 1, i = 0;

while (flag && (i < (n - 1))) {

if ((graf[pos[i] - 1][pos[i + 1] - 1]) == 0)

flag = 0;

i++;

}

return flag;

}

// Является ли данная последовательность pos цепью для графа graf

int is\_cep(int \*pos) {

int a[N] = { 0 };// Логическое множество для ребер

int i = 0;

int flag = 1;

while (flag && (i < (n - 1))) {

if (a[pos[i]] == pos[i + 1]) {

flag = 0;

}

a[pos[i]] = pos[i + 1];

i++;

}

return flag;

}

// Является ли данная последовательность pos простой цепью для графа graf

int is\_easy\_cep(int \*pos) {

int i;

int \*a;

int flag = 1;

a = (int\*)malloc((m + 1) \* sizeof(int));

for (i = 0; i < (m + 1); i++) {

a[i] = 0;

}

i = 0;

while (flag && i < (n - 1)) {

if (a[pos[i]] == 1)

flag = 0;

a[pos[i]] = 1;

i++;

}

return flag;

}

// Является ли данная последовательность pos циклом для графа graf

int is\_cikl(int \*pos) {

if (pos[0] == pos[n - 1]) {

return 1;

}

return 0;

}

int is\_easy\_cikl(int \*pos) {

if (pos[0] == pos[n - 1]) {

int i;

int \*a;

int flag = 1;

a = (int\*)malloc((m) \* sizeof(int));

for (i = 1; i < m; i++) {

a[i] = 0;

}

i = 0;

while (flag && i < (n - 1)) {

if (a[pos[i]] == 1)

flag = 0;

a[pos[i]] = 1;

i++;

}

return flag;

}

return 0;

}

//Инициализация графа а ввиде матрицы смежности n\*n

int\*\* init\_graf(int n) {

int i, j;

int \*\*a;

// Выделение памяти под указатели на строки

a = (int\*\*)malloc(n \* sizeof(int\*));

printf("\nВведите граф ввиде матрицы смежности %d: \n", n);

// Ввод элементов графа

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

// Выделение памяти под хранение строк

a[i] = (int\*)malloc(n \* sizeof(int));

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

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

}

}

return a;

}

int\* init\_posl(int n) {

int i;

int \*a;

// Выделение памяти

a = (int\*)malloc(n \* sizeof(int));

printf("\nВведите последовательность длинны %d: ", n);

// Ввод элементов последовательности

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

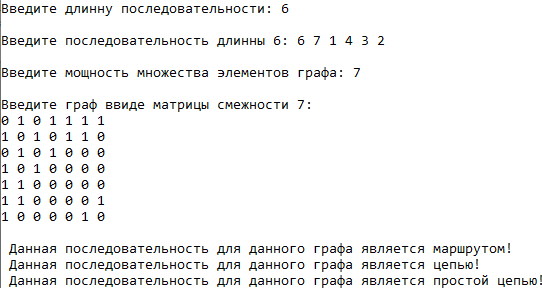
scanf("%d", &a[i]);

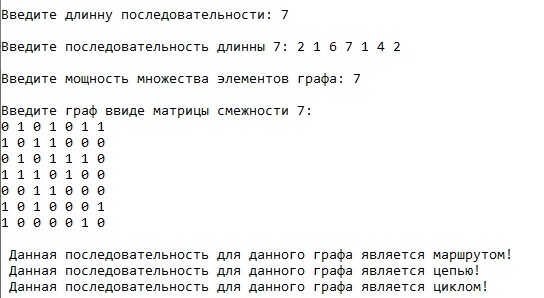
}

return a;

}

**Результат выполнения программы:**

****

****

**Задание 4**

#include <stdio.h>

#include <locale.h>

#define N 15

int \*a; // массив маршрута

int l;// Длинна маршрута

int n;// Мощность квадратной матрицы смежности графа

int \*\*graf; // Матрица графа

int \* init\_posl(int);

int \*\* init\_graf(int);

void output\_marsh(int \*);

void marshruti(int);

int main()

{

setlocale(LC\_ALL, "Rus");

printf("\nВведите длину маршрута: ");

scanf("%d", &l);

a = init\_posl((l + 1));

printf("\nВведите мощность множества элементов графа: ");

scanf("%d", &n);

graf = init\_graf(n);

printf("\nВведите вершину для которой необходимо найти маршруты: ");

scanf("%d", &a[0]);

marshruti(1);

system("pause");

return 0;

}

void marshruti(int i) {

int x;

for (x = 1; x <= n; x++) {

if ((graf[a[i - 1]][x] == 1) && (x != a[i - 1])) {

a[i] = x;

if (i == l)

output\_marsh(a);

else

marshruti(i + 1);

}

}

return;

}

void output\_marsh(int \*a) {

int i;

for (i = 0; i < (l + 1); i++)

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

printf("\n");

}

//Инициализация графа а ввиде матрицы смежности n\*n

int\*\* init\_graf(int n) {

int i, j;

int \*\*a;

// Выделение памяти под указатели на строки

a = (int\*\*)malloc((n + 1) \* sizeof(int\*));

printf("\nВведите граф ввиде матрицы смежности %d: \n", n);

// Ввод элементов графа

for (i = 1; i <= n; i++) {

// Выделение памяти под хранение строк

a[i] = (int\*)malloc((n + 1) \* sizeof(int));

for (j = 1; j <= n; j++) {

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

}

}

return a;

}

int\* init\_posl(int n) {

int i;

int \*a;

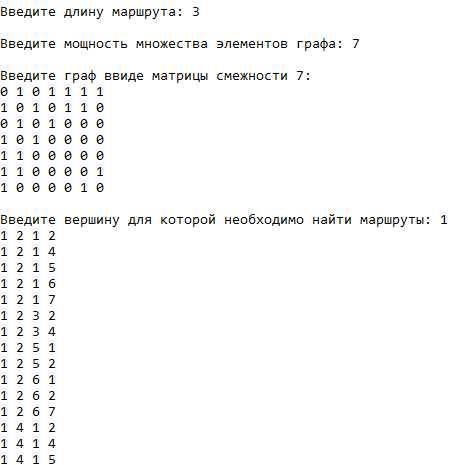
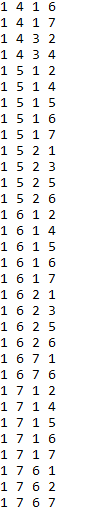
// Выделение памяти

a = (int\*)malloc(n \* sizeof(int));

return a;

}

**Результат работы программы:**

** **

**Задание 5**

#include "windows.h"

#include <stdio.h>

#include <locale.h>

#define N 15

int \*a; // массив маршрута

int \*\*r;

int l;// Длинна маршрута

int n;// Мощность квадратной матрицы смежности графа

int \*\*graf; // Матрица графа

int\* init\_posl(int);

int\*\* init\_graf(int);

void marshruti(int);

int main()

{

setlocale(LC\_ALL, "Rus");

int i, j;

printf("\nВведите длину маршрута: ");

scanf("%d", &l);

a = init\_posl((l + 1));

printf("\nВведите мощность множества элементов графа: ");

scanf("%d", &n);

graf = init\_graf(n);

r = (int\*\*)malloc((n + 1) \* sizeof(int\*));

// Ввод элементов графа

for (i = 1; i <= n; i++) {

// Выделение памяти под хранение строк

r[i] = (int\*)malloc((n + 1) \* sizeof(int));

for (j = 1; j <= n; j++) {

r[i][j] = 0;

}

}

int v = 1;

while (v <= n) {

a[0] = v;

marshruti(1);

v++;

}

j = 1;

for (i = (j + 1); i <= (n - 1); i++) {

for (j = 1; j <= n; j++) {

printf("Между вершинами %d и %d - %d маршрутов.\n", i, j, r[i][j]);

}

}

system("pause");

return 0;

}

void marshruti(int i) {

int x;

for (x = 1; x <= n; x++) {

if ((graf[a[i - 1]][x] == 1) && (x != a[i - 1])) {

a[i] = x;

if (i == l) {

r[a[0]][a[l]]++;

}

else

marshruti(i + 1);

}

}

return;

}

//Инициализация графа а ввиде матрицы смежности n\*n

int\*\* init\_graf(int n) {

int i, j;

int \*\*a;

// Выделение памяти под указатели на строки

a = (int\*\*)malloc((n + 1) \* sizeof(int\*));

printf("\nВведите граф ввиде матрицы смежности %d: \n", n);

// Ввод элементов графа

for (i = 1; i <= n; i++) {

// Выделение памяти под хранение строк

a[i] = (int\*)malloc((n + 1) \* sizeof(int));

for (j = 1; j <= n; j++) {

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

}

}

return a;

}

int\* init\_posl(int n) {

int i;

int \*a;

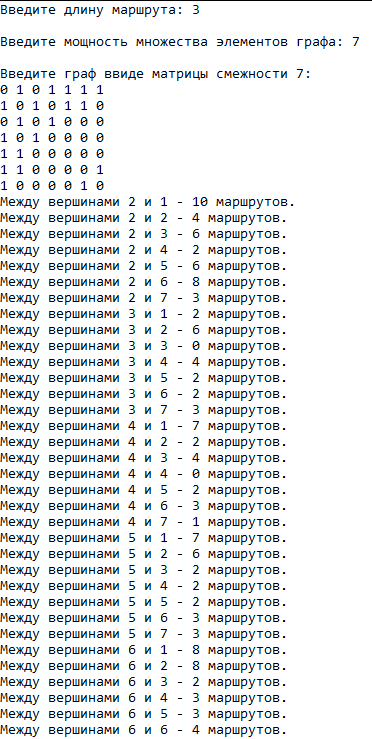
// Выделение памяти

a = (int\*)malloc(n \* sizeof(int));

return a;

}

**Результат работы программы:**

****

**Задание 6**

#include "windows.h"

#include <stdio.h>

#include <locale.h>

int \*a; // массив маршрута

int l;// Длинна маршрута

int n;// Мощность квадратной матрицы смежности графа

int \*\*graf; // Матрица графа

int \* init\_posl(int);

int \*\* init\_graf(int);

void output\_marsh(int \*);

void marshruti(int);

int main()

{

setlocale(LC\_ALL, "Rus");

printf("\nВведите длину маршрута: ");

scanf("%d", &l);

a = init\_posl((l + 1));

printf("\nВведите мощность множества элементов графа: ");

scanf("%d", &n);

graf = init\_graf(n);

printf("\nВведите 2 вершины между которыми необходимо найти маршруты: ");

scanf("%d %d", &a[0], &a[l]);

marshruti(1);

system("pause");

return 0;

}

void marshruti(int i) {

int x;

for (x = 1; x <= n; x++) {

if ((graf[a[i - 1]][x] == 1) && (x != a[i - 1]) && (x != a[l])) {

a[i] = x;

if (i == (l - 1)) {

if (graf[a[i]][a[l]] == 1)

output\_marsh(a);

}

else

marshruti(i + 1);

}

}

return;

}

void output\_marsh(int \*a) {

int i;

for (i = 0; i < (l + 1); i++)

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

printf("\n");

}

//Инициализация графа а ввиде матрицы смежности n\*n

int\*\* init\_graf(int n) {

int i, j;

int \*\*a;

// Выделение памяти под указатели на строки

a = (int\*\*)malloc((n + 1) \* sizeof(int\*));

printf("\nВведите граф ввиде матрицы смежности %d: \n", n);

// Ввод элементов графа

for (i = 1; i <= n; i++) {

// Выделение памяти под хранение строк

a[i] = (int\*)malloc((n + 1) \* sizeof(int));

for (j = 1; j <= n; j++) {

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

}

}

return a;

}

int\* init\_posl(int n) {

int i;

int \*a;

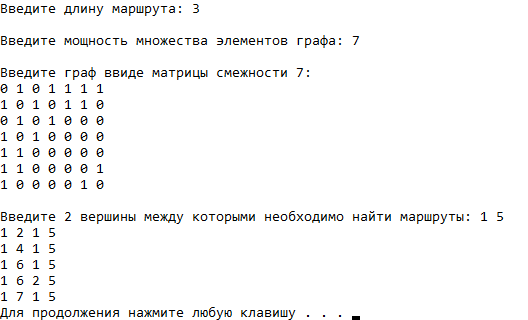
// Выделение памяти

a = (int\*)malloc(n \* sizeof(int));

return a;

}

**Результат работы программы:**



**Задание 7**

#include <locale.h>

#include <stdio.h>

#include <malloc.h>

int counter=0;

// вернет 0 если это цепь

char find(int \*w, int n)

{

for (int i = 0; i < n-1; ++i)

{

for (int j = i+1; j < n; ++j)

{

if(w[i]==w[j])

{

return 1;

}

}

}

return 0;

}

int proverka(int \* w, int l, int k)

{

for (int j=0; j<l-1; j++)

{

if (w[j]==k)

return 1;

}

return 0;

}

int check (int \*\*a, int\* w, int l, int n)

{

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

{

if (a[w[l-1]][j])

{

int k=j;

if (!proverka(w,l,k))//\*возвращает 1, если вершины смежные с последней входят в цепь

return 0;

}

}

return 1;

}

void chains(int \*\*a, int \*w, int n, int l, int i)

{

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

{

if(a[w[i-1]][j])

{

w[i] = j;//формируем маршрут

if(i==l-1)//если нашли маршрут заданной длины

{

if((!find(w,l)) && (check(a,w,l,n)))//если маршрут - это максимальная простая цепь

{

counter++;//увеличить счетчик маршрутов

for (int k = 0; k < l; ++k)

printf("%i ", w[k] + 1);//вывод вершин маршрута

puts("");

}

}

else

{

chains(a,w,n,l,i+1);

}

}

}

}

int main()

{

setlocale(LC\_ALL,"Russian");

int n,l;

printf("Введите размер матрицы : ");

scanf("%i", &n);

int \*\*a = malloc(n\* sizeof(int\*));

int \*w = malloc(l\*sizeof(int));

printf("Введите вершину начала: ");

scanf("%i",&w[0]);

w[0]--;

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

a[i] = malloc(n\*sizeof(int));

printf("Введите матрицу: \n");

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

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

scanf("%i",&a[i][j]);

puts("");

for(l=2; l<=n; l++)

chains(a,w,n,l,1);

printf("\nНайдено %i максимальных простых цепей выходящих из заданной вершины графа\n", counter);

return 0;

}

**Результат работы программы:**

