|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА – Российский технологический университет"**  **РТУ МИРЭА** |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

**ОТЧЕТ   
о выполнении лабораторной работы №3**

**«Реализация сортировки линейных структур данных»**

**по дисциплине   
«Программная реализация нелинейных структур»**

**Вариант № 37**

Выполнил: студент 2 курса

группы БИСО-03-19

шифр 19Б0637

Юсупов Константин Максимович  
*(фио студента)*

Проверил:

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2020 г

**Задание на лабораторную работу № 3.**

В рамках лабораторной работы №3 требуется программно реализовать граф, на основе архитектуре, соответствующей варианту задания.

Используя разработанный граф необходимо реализовать заданный в варианте задания алгоритм, а также дополнительный набор функций:

1. FIRST(v) – возвращает индекс первой вершины, смежной с вершиной v. Если вершина не имеет смежных вершин, то возвращается “нулевая” вершина A.
2. NEXT(v,i) – возвращает индекс вершины, смежной с вершиной v, следующий за индексом i . Если i – это индекс последней вершины, смежной с вершиной v, то возвращается A.
3. VERTEX(v,i) – возвращает вершину (её адрес) с индексом i из множества вершин, смежных с v.
4. ADD\_V – добавить узел
5. ADD\_E(v,w,c) – добавить дугу от v до w весом c
6. DEL\_V – удалить узел
7. DEL\_E – удалить дугу
8. EDIT\_V – изменить метку узла
9. EDIT\_E – изменить вес дуги.

**Вариант 37.**

Способ реализации графа – матрица смежности

Алгоритм обхода – поиск циклов обходом в ширину, вывод кол-ва циклов.

**Листинг программы**

**Makefile**

all: graph

graph: graph.c main.c

gcc -o graph -g main.c graph.c

clean:

rm graph

edit:

nano graph.c graph.h main.c

**graph.h**

#ifndef GRAPH\_H

#define GRAPH\_H

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#define A 18446744073709551615

struct graph{

unsigned long long \*\*arr;

char \*\*marks;

unsigned long long nnodes;

};

struct graph \*graph\_new(unsigned long nnodes);

void graph\_generate\_random(struct graph \*g);

void graph\_out(struct graph \*g);

void graph\_delete(struct graph \*g);

void graph\_find\_loops(struct graph \*g);

unsigned long long first(struct graph \*g, unsigned long long v);

unsigned long long next(struct graph \*g, unsigned long long v, unsigned long long j);

char \*vertex(struct graph \*g, unsigned long long v, unsigned long long j);

unsigned long long add\_v(struct graph \*g, char \*mark);

void add\_e(struct graph \*g, unsigned long long from, unsigned long long to, unsigned long long weight);

void del\_v(struct graph \*g, unsigned long long v);

void del\_e(struct graph \*g, unsigned long long from, unsigned long long to);

void edit\_v(struct graph \*g, unsigned long long index, char \*mark);

void edit\_e(struct graph \*g, unsigned long long from, unsigned long long to, unsigned long long weight);

#endif

**graph.c**

#include "graph.h"

struct graph \*graph\_new(unsigned long nnodes){

struct graph \*t=malloc(sizeof(struct graph));

t->nnodes=nnodes;

t->arr=malloc(sizeof(unsigned long long\*)\*nnodes);

t->marks=malloc(sizeof(char\*)\*nnodes);

for(unsigned long long i=0;i<nnodes;i++){

t->arr[i]=malloc(sizeof(unsigned long long)\*nnodes);

for(unsigned long long j=0;j<nnodes;j++){

t->arr[i][j]=0;

}

}

return t;

}

void graph\_generate\_random(struct graph \*g){

time\_t t;

srand((unsigned) time(&t));

for(unsigned long long i=0;i< (g->nnodes);i++)

for(unsigned long long j=0;j< (g->nnodes);j++){

g->arr[i][j]=(unsigned long long)(rand()%100);

if(rand()%2-1)g->arr[i][j]=0;

}

unsigned long long i=0;

while((i< g->nnodes)&&(i< g->nnodes)){

g->arr[i][i]=0;

i+=1;

}

}

static unsigned long long rec(struct graph \*g, unsigned long long \*path, unsigned long long npath){

unsigned long long loops=0;

if(npath>1)

if(path[0]==path[npath-1]){

for(unsigned long long i=0;i<npath;i++){

printf("%llu ",path[i]);

}

printf("\n");

free(path);

return 1;

}

for(unsigned long long i=1;i<npath-1;i++){

if(path[i]==path[npath-1])return 0;

}

for(unsigned long long i=0;i< g->nnodes;i++){

if(!(g->arr[path[npath-1]][i]))continue;

unsigned long long \*path\_copy=malloc(sizeof(unsigned long long)\*(npath+1));

memcpy(path\_copy,path,sizeof(unsigned long long)\*npath);

path\_copy[npath]=i;

loops+=rec(g,path\_copy,npath+1);

}

free(path);

return loops;

}

void graph\_find\_loops(struct graph \*g){

printf("Loops:\n");

unsigned long long totalloops=0;

for(unsigned long long i=0;i< (g->nnodes);i++){

{

unsigned long long \*path=malloc(sizeof(unsigned long long));

path[0]=i;

totalloops+=rec(g,path,1);

}

}

printf("Total loops: %llu\n",totalloops);

}

void graph\_out(struct graph \*g){

for(unsigned long long i=0;i< g->nnodes;i++){

printf("%s\n",(!g->marks[i])?"null":g->marks[i]);

}

for(unsigned long long i=0;i< g->nnodes;i++){

for(unsigned long long j=0;j< g->nnodes;j++)

printf("%3llu",g->arr[i][j]);

printf("\n");

}

printf("---------------------------\n");

for(unsigned long long i=0;i< g->nnodes;i++){

char flag;

flag=0;

for(unsigned long long j=0;j< g->nnodes;j++){

if(g->arr[i][j])

if(flag)

printf(" +--->%d\n",j);

else{

printf("%2d-T--->%d\n",i,j);

flag=1;

}

}

}

}

void graph\_delete(struct graph \*g){

for(unsigned long long i=0;i< g->nnodes;i++)

free(g->arr[i]);

free(g->arr);

g->nnodes=0;

}

unsigned long long first(struct graph \*g, unsigned long long v){

for(unsigned long long i=0;i< g->nnodes; i++){

if(g->arr[i][v])return i;

}

return A;

}

unsigned long long next(struct graph \*g, unsigned long long v, unsigned long long j){

for(unsigned long long i=j+1;i< g->nnodes; i++){

if(g->arr[i][v])return i;

}

return A;

}

char \*vertex(struct graph \*g, unsigned long long v, unsigned long long j){

unsigned long long count=0;

for(unsigned long long i=0;i< g->nnodes;i++){

if(g->arr[i][v])count+=1;

if(!(count-j))return &(g->arr[i][v]);

}

}

unsigned long long add\_v(struct graph \*g, char \*mark){

for(unsigned long long i=0;i< g->nnodes;i++){

g->arr[i]=realloc(g->arr[i],sizeof(unsigned long long)\*(g->nnodes+1));

g->arr[i][g->nnodes]=0;

}

g->arr=realloc(g->arr,sizeof(unsigned long long\*)\*(g->nnodes+1));

g->arr[g->nnodes]=calloc(sizeof(unsigned long long)\*(g->nnodes+1),1);

g->nnodes+=1;

g->marks[g->nnodes-1]=strdup(mark);

return g->nnodes-1;

}

void add\_e(struct graph \*g, unsigned long long from, unsigned long long to, unsigned long long weight){

g->arr[from][to]=weight;

}

void del\_v(struct graph \*g, unsigned long long v){

free(g->arr[v]);

if(g->marks[v]){

g->marks[v]=NULL;

}

for(unsigned long long i=v;i< g->nnodes -1;i++){

g->arr[i]=g->arr[i+1];

g->marks[i]=g->marks[i+1];

}

for(unsigned long long i=0;i< g->nnodes-1;i++){

for(unsigned long long j=v;j< g->nnodes; j++){

g->arr[i][j]=g->arr[i][j+1];

}

g->arr[i]=realloc(g->arr[i],sizeof(unsigned long long)\*(g->nnodes-1));

}

g->arr=realloc(g->arr,sizeof(unsigned long long\*)\*(g->nnodes-1));

g->nnodes-=1;

}

void del\_e(struct graph \*g, unsigned long long from, unsigned long long to){

g->arr[from][to]=0;

}

void edit\_v(struct graph \*g, unsigned long long index, char \*mark){

free(g->marks[index]);

g->marks[index]=mark;

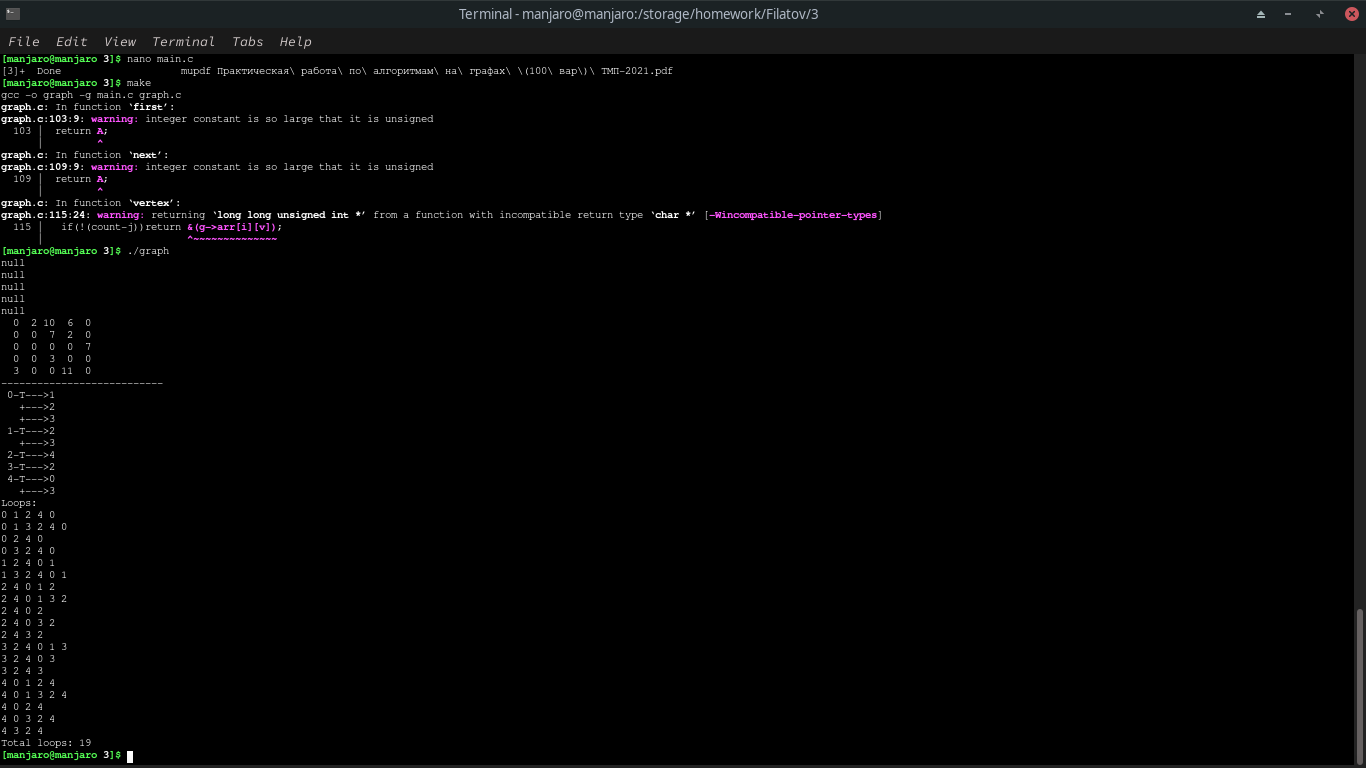
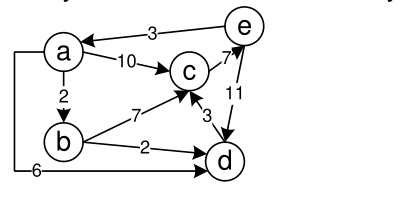
}

void edit\_e(struct graph \*g, unsigned long long from, unsigned long long to, unsigned long long weight){

g->arr[from][to]=weight;

}

**Контрольный пример 1**

**main.c для данного примера:**

#include "graph.h"

int main(){

struct graph \*g=graph\_new(5);

int l1[5]={0,2,10,6,0};

int l2[5]={0,0,7,2,0};

int l3[5]={0,0,0,0,7};

int l4[5]={0,0,3,0,0};

int l5[5]={3,0,0,11,0};

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

g->arr[0][i]=l1[i];

}

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

g->arr[1][i]=l2[i];

}

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

g->arr[2][i]=l3[i];

}

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

g->arr[3][i]=l4[i];

}

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

g->arr[4][i]=l5[i];

}

graph\_out(g);

graph\_find\_loops(g);

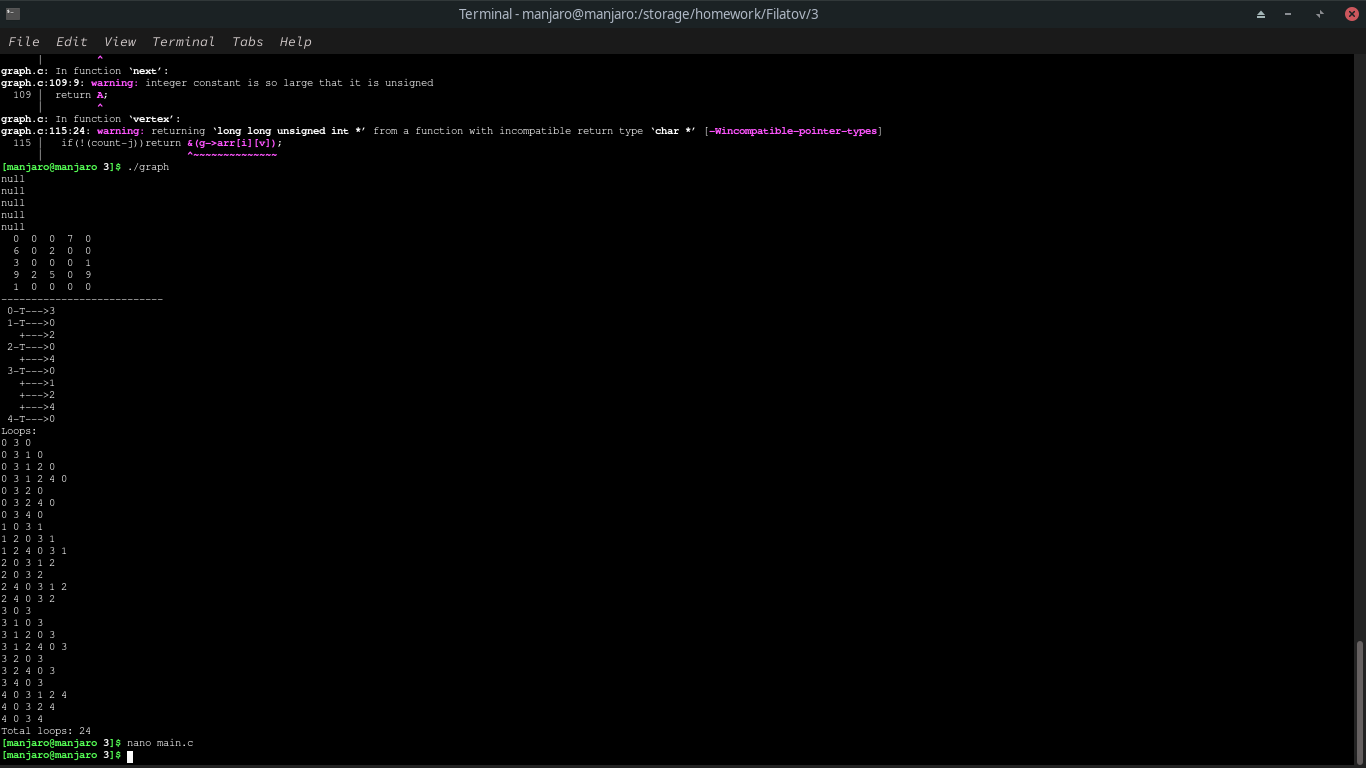
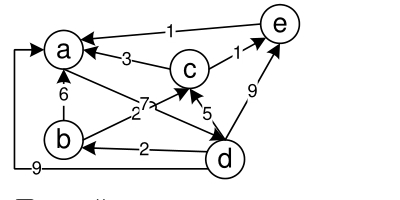
graph\_delete(g);

g=NULL;

return 0;

}

**Контрольный пример 2**

**main.c для данного примера:**

#include "graph.h"

int main(){

struct graph \*g=graph\_new(5);

int l1[5]={0,0,0,7,0};

int l2[5]={6,0,2,0,0};

int l3[5]={3,0,0,0,1};

int l4[5]={9,2,5,0,9};

int l5[5]={1,0,0,0,0};

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

g->arr[0][i]=l1[i];

}

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

g->arr[1][i]=l2[i];

}

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

g->arr[2][i]=l3[i];

}

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

g->arr[3][i]=l4[i];

}

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

g->arr[4][i]=l5[i];

}

graph\_out(g);

graph\_find\_loops(g);

graph\_delete(g);

g=NULL;

return 0;

}