##### Введение

1. Постановка задачи

2. Использование динамических структур при работе с графами

2.1. Способы представления графов

2.2. Операции над графами

2.3. Описание программной реализации

2.3.1. Описание процедур и функций языка

2.3.2. Описание функций работы с динамической памятью, графами

Выводы

Приложение А Экранные формы

Приложение Б Листинг программы

1 ПОСТАНОВКА ЗАДАЧИ

Задача.

**Найти все источники ориентированного графа.**

Исходные данные:

- номер вершины (цел типа), вводимый пользователем;

- дуга графа, задается двумя вершинами источником и стоком, вводимая пользователем.

Промежуточные данные:

Head:TUk – указатель на голову списка смежности графа;

n,m:цел – номера вершин;

c:сим – клавиша события.

Результаты:

V:массив байт – массив вершин источников;

Ограничения:

max=10 – максимальное количество вершин;

V:массив [1..max\*max].

2. Использование динамических структур при работе с графами

2.1 Способы представления графов

Способы задания графов:

1. матрица смежности;
2. матрица инцидентности;
3. список смежности;
4. список дуг (ребер);
5. и др.

Список смежности – для вершины **v** есть список концов дуг, исходящих из вершины **v,** в случае орграфа, или список смежных с **v** вершин, в случае неориентированного графа.

Список дуг – это список, в котом каждой дуге ставится в соответствие пара **<x,y>**, где **x** – начало дуги, а **y** – ее конец. Для нагруженных графов – тройка **<x,y,z>**,где **x** – начало дуги, **y** – конец дуги, **z** – вес дуги.

Матрица смежности – это квадратная матрица, строки и столбцы которой соответствуют вершинам графа. Элемент матрицы (i,j) равен 1, если вершина i связана с вершиной j ребром, иначе элемент матрицы равен 0.



Для неориентированных графов матрица смежности является симметричной относительно главной диагонали. Т.е. для получения информации о графе достаточно знать верхнюю или нижнюю треугольную матрицу смежности. Для ориентированных графов матрица смежности не является симметричной.

Матрица инцидентности – это матрица, строки которой – список вершин, а столбцы – список ребер. Элемент матрицы инциденций (i,j) равен 1, если вершина **i** инцидентна соответствующему ребру.

Для неориентированных графов:



Для ориентированных графов:



Например, дан граф G (см.рис. .1)

1

2

3

4

8

3

6

5

x2

x1

x3

x4

x5

x6

Рисунок 2.1 – Граф G

Представление графа списком смежности отображено на рисунке 2.2

Рисунок 2.2 – Список смежности графа G

х3

x2

x1

x2

x3

x4

x5

x6

x2

x4

x5

x6

x2

NilL

NilL

NilL

NilL

NilL

NilL

x5

Nil

Представление графа с помощью списка дуг имеет вид отображено на рисунке .3

Nil

x1 x2

x1 x3

x3 x2

x3 x4

x3 x5

x4 x5

x5 x6

x6 x4

Рисунок 2.3 – Список дуг графа G

Представление графа с помощью матрицы смежности показано в таблице 2.1

Таблица 2.1 – Матрица смежности

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **x1** | **x2** | **x3** | **x4** | **x5** | **x6** |
| **x1** | 0 | 1 | 1 | 0 | 0 | 0 |
| **x2** | 0 | 0 | 0 | 0 | 0 | 0 |
| **x3** | 0 | 1 | 0 | 1 | 1 | 0 |
| **x4** | 0 | 0 | 0 | 0 | 1 | 0 |
| **x5** | 0 | 0 | 0 | 0 | 0 | 1 |
| **x6** | 0 | 0 | 0 | 1 | 0 | 0 |

Представление графа с помощью матрицы инцидентности показано в таблице 2.2

Таблица 2.2 – Матрица инцидентности

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **x1x2** | **x1x3** | **x3x2** | **x3x4** | **x3x5** | **x4x5** | **x5x6** | **x6x4** |
| **x1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **x2** | -1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| **x3** | 0 | -1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **x4** | 0 | 0 | 0 | -1 | 0 | 1 | 0 | -1 |
| **x5** | 0 | 0 | 0 | 0 | -1 | -1 | 1 | 0 |
| **x6** | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 |

2.2 Операции над графами

При решении поставленной задачи для представления графа был выбран список смежности.

type TUk1=^TEl1; //элемент списка вершин

TEl1=record

Next:TUk1;

Inf:integer;

end;

TUk=^TEl;

TEl=record //элемент списка смежности

Left:TUk1;

Inf:integer;

Down:TUk;

end;

Head:TUk; //указатель на начало списка

Схематичное представление списка смежности ориентированного графа представлено на рисунке 2.1

Down

Left

Inf

Down

Left

Inf

Inf

Next

Inf

Next

Head

…

…



Рисунок 2.3 – Схема списка смежности

Алгоритм нахождения источников ориентированного графа представлен на рисунке 2.4.

В графе нет ни одной вершины

Вывод

если Head=Nil

то

иначе

k:=0; p:=Head;

пока p<>Nil

если p^.Left<>Nil

то

f:=true; p1:=Head;

пока p1<>Nil

p2:=p1^.Left;

пока (f) И (p2<>Nil)

если p2^.Inf=p^.Inf

то

f:=false

p2:=p2^.Next

p1:=p1^.Down

если f=true

то

inc(k); Is[k]:=p^.Inf;

p:=p^.Down;

Рисунок 2.4 - Алгоритм нахождения источников орграфа

2.3 Описание программной реализации

2.3.1 Описание процедур и функций языка

Процедура New***–***резервирует фрагмент кучи для размещения переменной; формат обращения

New(TP)

TP – типизированный указатель. За одно обращение к процедуре можно зарезервировать не бо­лее 65521 байт динами­ческой памяти. Если нет свободной памяти требуемого размера, возникает ошибка периода исполнения. Если память не фраг­ментирована, последовательные обращения к про­цедуре будут резервировать последовательные участки памяти, так что начало следующего будет располагаться сразу за концом предыдущего.

Процедура Dispose – возвращает в кучу фрагмент динамической памяти, который ранее был зарезервирован за типизированным указателем; формат обращения

Dispose (TP)

TP – типизированный указатель. При повторном использовании процедуры применительно к уже освобожденному фрагменту возникает ошибка периода исполнения.

2.3.2 Описание функций работы с динамической памятью, графами

Для решения поставленной задачи были разработаны следующие процедуры и функции:

1) procedure AddVer(n:integer); - добавление в список смежности вершины с номером n;

2) procedure AddDug(n,m:integer); - добавление в список дуги (n,m). n и m номера вершин источника дуги и стока соответственно;

3) procedure DelList(p:TUk1); - удаление списка смежных вершин. p – указатель на начало списка;

4) procedure DelVer(n:integer); - удаление вершины с номером n;

5) procedure DelDug(n,m:integer); - удаление дуги (n,m). n и m номера вершин источника дуги и стока соответственно;

6) procedure PrintGraph; - вывод списка смежности на экран в виде матрицы смежности;

7) procedure FindIstok; - поиск и вывод на экран всех источников орграфа.

ВЫВОДЫ

Задание, выданное на летнюю практику, поставило определенные задачи:

1) научится создавать связные структуры данных, используя указатели;

2) научится создавать и манипулировать динамическими структурами данных, такими как связные списки, очереди и стеки;

3) понять работу различных приложений со связными структурами данных.

Решение выданного задания было реализовано с помощью языка программирования Паскаль.

Во время прохождения учебной практики был закреплен теоретический материал по разработке динамических структур данных, были получены практические навыки реализации динамических структур данных на языке высокого уровня для представления и обработки графов.

Приложение А

ЭКРАННЫЕ ФОРМЫ

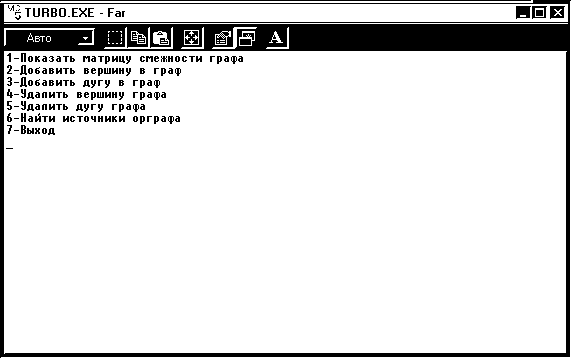


Рисунок А.1 – Главное меню программы

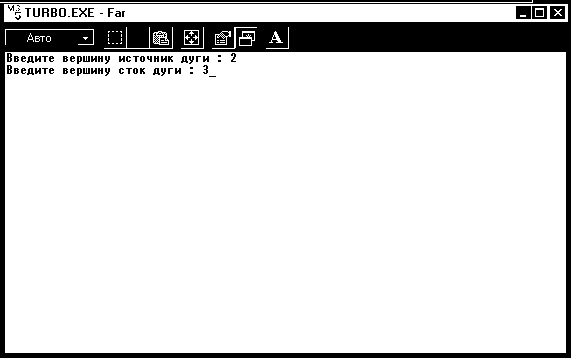


Рисунок А.2 – Добавление дуги орграфа

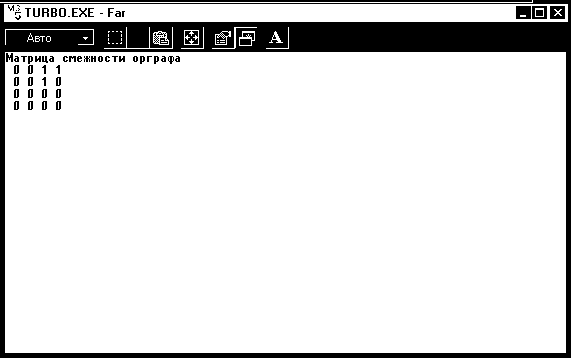


Рисунок А.3 – Матрица смежности орграфа

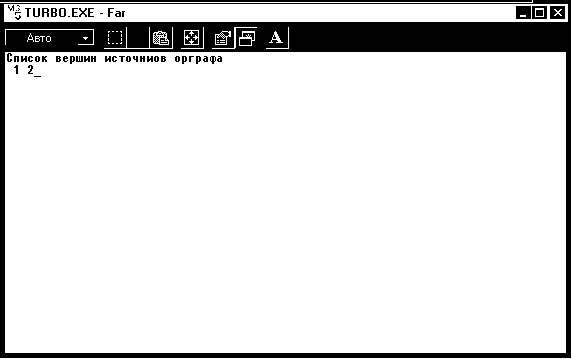


Рисунок А.4 – Список вершин источников орграфа

Приложение Б

ЛИСТИНГ ПРОГРАММЫ

program OrGraph;

uses Crt;

type TUk1=^TEl1;

TEl1=record

Next:TUk1;

Inf:integer;

end;

TUk=^TEl;

TEl=record

Left:TUk1;

Inf:integer;

Down:TUk;

end;

const

max=10;

var

Head:TUk;

c:char;

n,m:integer;

V:array[1..max] of byte;

{---------добавляем вершину в граф-------------}

procedure AddVer(n:integer);

var

p:TUk;

begin

if (Head=Nil) then

begin

New(Head);

Head^.Inf:=n;

Head^.Left:=Nil;

Head^.Down:=Nil;

end else begin

p:=Head;

while ((p^.Down<>Nil)and(p^.Inf<>n)) do p:=p^.Down;

if (p^.Inf=n) then WriteLn('Такая вершина уже есть!!!')

else begin

New(p^.Down);

p^.Down^.Inf:=n;

p^.Down^.Left:=Nil;

p^.Down^.Down:=Nil;

end;

end;

end;

{-------добавляем дугу в граф----------------------}

procedure AddDug(n,m:integer);

var

p,p1:TUk;

p2:TUk1;

begin

if (Head=Nil) then WriteLn('В графе нет ни одной вершины!!!')

else begin

p:=Head;

while ((p<>Nil)and(p^.Inf<>n)) do p:=p^.Down;

if (p=Nil) then WriteLn('В графе отсутствует указанная вершина источник!!!')

else begin

p1:=Head;

while ((p1<>Nil)and(p1^.Inf<>m)) do p1:=p1^.Down;

if (p1=Nil) then WriteLn('В графе отсутствует указанная вершина сток!!!')

else begin

if (p^.Left=Nil) then

begin

New(p^.Left);

p^.Left^.Inf:=m;

p^.Left^.Next:=Nil;

end else begin

p2:=p^.Left;

while ((p2^.Next<>Nil)and(p2^.Inf<>m)) do p2:=p2^.Next;

if (p2^.Inf=m) then WriteLn('Указанная дуга уже существует!!!')

else begin

New(p2^.Next);

p2^.Next^.Inf:=m;

p2^.Next^.Next:=Nil;

WriteLn('Дуга добавлена!!!');

end;

end;

end;

end;

end

end;

{--удаляем список дуг--}

procedure DelList(p:TUk1);

var p1:TUk1;

begin

while (p<>Nil) do

begin

p1:=p;

p:=p^.Next;

Dispose(p1);

end;

end;

{---------удаляем вершину из графа---------}

procedure DelVer(n:integer);

var

p,p1:TUk;

p2,p3:TUk1;

begin

if (Head=Nil) then WriteLn('В графе нет ни одной вершины!!!')

else begin

p:=Head;

if (p^.Inf=n) then

begin

Head:=Head^.Down;

DelList(p^.Left);

Dispose(p);

end else begin

while ((p^.Down^.Inf<>n)and(p^.Down<>Nil)) do p:=p^.Down;

if (p^.Down=Nil) then WriteLn('В графе нет указанной вершины!!!')

else begin

DelList(p^.Down^.Left);

p1:=p^.Down;

p^.Down:=p^.Down^.Down;

Dispose(p1);

end;

end;

p:=Head;

while (p<>Nil) do

begin

if (p^.Left^.Inf=n) then

begin

p2:=p^.Left;

p^.Left:=p^.Left^.Next;

Dispose(p2);

end else begin

p2:=p^.Left;

while ((p2^.Next<>Nil)and(p2^.Next^.Inf=n)) do p2:=p2^.Next;

if(p2^.Next^.Inf=n) then

begin

p3:=p2^.Next;

p2^.Next:=p2^.Next^.Next;

Dispose(p3);

end;

end;

p:=p^.Down;

end;

end;

end;

{------удаляем дугу графа--------}

procedure DelDug(n,m:integer);

var

p,p1:TUk;

p2,p3:TUk1;

begin

if (Head=Nil) then WriteLn('В графе нет ни одной вершины!!!')

else begin

p:=Head;

while ((p^.Inf<>n)and(p<>Nil)) do p:=p^.Down;

if (p=Nil) then WriteLn('В графе отсутствует указанная вершина источник')

else begin

p1:=Head;

while ((p1<>Nil)and(p1^.Inf<>m)) do p1:=p1^.Down;

if (p1=Nil) then WriteLn('В графе отсутствует указанная вершина сток!!!')

else begin

p2:=p^.Left;

if (p^.Left^.Inf=m) then

begin

p3:=p^.Left;

p^.Left:=p^.Left^.Next;

Dispose(p3);

end else begin

while ((p2^.Next^.Inf<>m)and(p2^.Next<>Nil)) do p2:=p2^.Next;

if (p2=Nil) then WriteLn('Указанного ребра нет в графе!!!')

else begin

p3:=p2^.Next;

p2^.Next:=p2^.Next^.Next;

Dispose(p3);

end;

end;

end;

end;

end;

end;

{---Вывод графа в виде матрицы смежности------}

procedure PrintGraph;

var

i,j,n:integer;

M:array [1..max,1..max] of byte;

p:TUk;

p2:TUk1;

begin

for i:=1 to max do

for j:=1 to max do M[i,j]:=0;

n:=0;

if (Head=Nil) then WriteLn('В графе нет ни одной вершины!!!')

else begin

p:=Head;

while (p<>Nil) do

begin

inc(n);

p2:=p^.Left;

while (p2<>Nil) do

begin

M[p^.Inf,p2^.Inf]:=1;

p2:=p2^.Next;

end;

p:=p^.Down;

end;

end;

for i:=1 to n do

begin

for j:=1 to n do Write(M[i,j]:2);

WriteLn;

end;

end;

{-----находим все источники орграфа----}

procedure FindIstok;

var

f:boolean;

i,k:integer;

Is:array[1..max\*max] of byte;

p,p1:TUk;

p2:TUk1;

begin

for i:=1 to max\*max do Is[i]:=0;

if (Head=Nil) then WriteLn('В графе нет ни одной вершины!!!')

else begin

k:=0;

p:=Head;

while (p<>Nil) do

begin

if (p^.Left<>Nil) then

begin

f:=true;

p1:=Head;

while (p1<>Nil) do

begin

p2:=p1^.Left;

while ((f)and(p2<>Nil)) do

begin

if p2^.Inf=p^.Inf then f:=false;

p2:=p2^.Next;

end;

p1:=p1^.Down;

end;

if (f=true) then

begin

inc(k);

Is[k]:=p^.Inf;

end;

end;

p:=p^.Down;

end;

end;

for i:=1 to k do Write(Is[i]:2);

end;

procedure Menu;

begin

WriteLn('1-Показать матрицу смежности графа');

WriteLn('2-Добавить вершину в граф');

WriteLn('3-Добавить дугу в граф');

WriteLn('4-Удалить вершину графа');

WriteLn('5-Удалить дугу графа');

WriteLn('6-Найти источники орграфа');

WriteLn('7-Выход');

end;

{--------основная программа--------}

begin

ClrScr;

repeat

clrscr;

Menu;

c:=ReadKey;

case c of

'1': begin

ClrScr; PrintGraph; ReadKey;

end;

'2': begin

ClrScr;

Write('Введите добавляемую вершину : ');

ReadLn(n); AddVer(n);

end;

'3': begin

ClrScr;

Write('Введите вершину источник дуги : ');

ReadLn(n);

Write('Введите вершину сток дуги : ');

ReadLn(m); AddDug(n,m);

end;

'4': begin

ClrScr;

Write('Введите удаляемую вершину : ');

ReadLn(n); DelVer(n);

end;

'5': begin

ClrScr;

Write('Введите вершину источник удаляемой дуги : ');

ReadLn(n);

Write('Введите вершину сток удаляемой дуги : ');

ReadLn(m); DelDug(n,m);

end;

'6': begin

ClrScr; FindIstok; ReadKey;

end;

'7': begin

halt;

end;

end;

until ord(c)=27;

end.