Лабораторная работа №5

ГРАФЫ. РЕШЕНИЕ ПРАКТИЧЕСКИХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ГРАФОВ (C++)

Теоретические сведения.

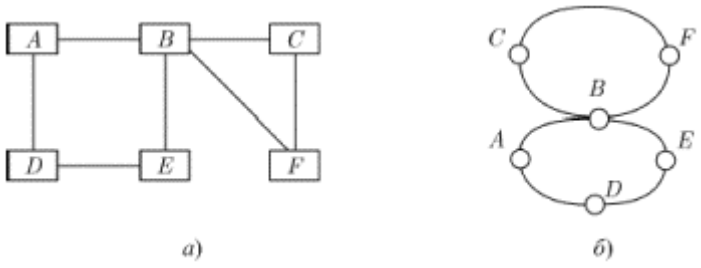
Начальные понятия теории графов

Графы являются существенным элементом математических моделей в самых разнообразных областях науки и практики. Они помогают наглядно представить взаимоотношения между объектами или событиями в сложных системах. Многие алгоритмические задачи дискретной математики могут быть сформулированы как задачи, так или иначе связанные с графами, например задачи, в которых требуется выяснить какие-либо особенности устройства графа, или найти в графе часть, удовлетворяющую некоторым требованиям, или построить *граф* с заданными свойствами.

Цель этой и двух следующих лекций - дать краткое введение в теорию графов. В них приводится *минимум* понятий, необходимый для того, чтобы можно было начать какую-либо содержательную работу с графами или приступить к более глубокому изучению теории графов. Доказательства приводятся только в тех случаях, когда их сложность не превышает некоторого интуитивно ощущаемого порога. Поэтому, например, такие важные факты, как теорема Кирхгофа или теорема Понтрягина-Куратовского, сообщаются без доказательств.

Определение графа

Для описания строения различных систем, состоящих из связанных между собой элементов, часто используют графические схемы, изображая элементы точками (кружками, прямоугольниками и т.д.), а связи между ними - линиями или стрелками, соединяющими элементы. При этом получаются диаграммы вроде тех, что показаны на рис. 1.1.



а) б)

Рис. 1.1.

На таких диаграммах часто ни способ изображения элементов, ни форма или *длина* линий не имеют значения - важно лишь, какие именно пары элементов соединены линиями. Если посмотреть внимательно, то можно заметить, что рисунки 1.1а и 1.1 бизображают одну и ту же структуру связей между элементами A, B, C, D, E, F. Эту же структуру можно описать, не прибегая к графическому изображению, а просто перечислив пары связанных между собой элементов: (A, B), (A, D), (B, C), (B, E), (B, F), (C, F)

, (D, E). Таким образом, когда мы отвлекаемся от всех несущественных подробностей, у нас остаются два списка: *список* элементов и *список* пар элементов. Вместе они составляют то, что математики называют графом. Из этого примера видно, что понятие графа само по себе не связано напрямую с геометрией или графикой. Тем не менее, возможность нарисовать *граф* - одна из привлекательных черт этого математического объекта.

Термин "*граф*" неоднозначен, это легко заметить, сравнивая приводимые в разных книгах определения. Однако во всех этих определениях есть кое-что общее. В любом случае *граф* состоит из двух множеств - *множества вершин* и *множества ребер*, причем для каждого ребра указана пара вершин, которые это *ребро соединяет*. Вершины и ребра называются *элементами* графа. Здесь будут рассматриваться только *конечные* графы, то есть такие, у которых оба *множества* конечны. Чтобы получить законченное *определение* графа того или иного типа, необходимо уточнить еще три момента.

1. Ориентированный или неориентированный?

   Прежде всего, нужно договориться, считаем ли мы пары (a, b) и (b, a) различными. Если да, то говорят, что рассматриваются упорядоченные пары (порядок элементов в паре важен), если нет - неупорядоченные. Если ребро e соединяет вершину a с вершиной b и пара (a, b) считается упорядоченной, то это ребро называется *ориентированным*, вершина a - его *началом*, вершина b - *концом*. Если же эта пара считается неупорядоченной, то ребро называется *неориентированным*, а обе вершины - его *концами*. Чаще всего рассматривают графы, в которых все ребра имеют один тип - либо ориентированные, либо неориентированные. Соответственно и весь граф называют ориентированным или неориентированным. На рисунках ориентацию ребра (направление от начала к концу) указывают стрелкой. На рис. 1.1показаны неориентированные графы, а на рис. 1.2 - ориентированные.

2. *Кратные ребра*.

   Следующий пункт, требующий уточнения, - могут ли разные ребра иметь одинаковые начала и концы? Если да, то говорят, что в графе допускаются *кратные ребра*. Граф с *кратными ребрами* называют также *мультиграфом*. На рис. 1.2 изображены два графа, левый является ориентированным *мультиграфом*, а правый - ориентированным графом без кратных ребер.

3. Петли.

   Ребро, которому поставлена в соответствие пара вида (a, a), то есть ребро, соединяющее вершину a с нею же самой, называется ***петлей***. Если такие ребра не допускаются, то говорят, что рассматриваются *графы без петель*.

Рис. 1.2.

Комбинируя эти три признака, можно получить разные варианты определения понятия графа. Особенно часто встречаются неориентированные графы без петель и кратных ребер. Такие графы называют *обыкновенными*. Если в графе нет кратных ребер, то можно просто отождествить ребра с соответствующими парами вершин - считать, что *ребро* это и есть пара вершин. Чтобы исключить петли, достаточно оговорить, что вершины, образующие *ребро*, должны быть различны. Это приводит к следующему определению *обыкновенного графа*.

Определение. ***Обыкновенным графом*** называется пара $G = (V, E)$, где V - конечное множество, E - множество неупорядоченных пар различных элементов из V. Элементы *множества* V называются ***вершинами*** графа, элементы *множества* E - его *ребрами*.

Слегка модифицируя это *определение*, можно получить определения других типов графов без кратных ребер: если заменить *слово* "неупорядоченных" словом "упорядоченных", получится *определение* ориентированного графа без петель, если убрать *слово* "различных", получится *определение* графа с петлями. *Ориентированный граф* часто называют *орграфом*.

В дальнейшем термин "*граф*" мы будем употреблять в смысле "*обыкновенный граф*", а рассматривая другие типы графов, будем специально это оговаривать.

Множество вершин графа G будем обозначать через VG, множество ребер - EG, число вершин - $n(G)$, число ребер - $m(G)$.

Из определения видно, что для задания *обыкновенного графа* достаточно перечислить его вершины и ребра, причем каждое *ребро* должно быть парой вершин. Положим, например, $VG = \{a, b, c, d, e, f\}$, $EG = \{(a, e), (a, f), (b, c), (c, d), (d, f)\}$. Тем самым задан *граф* G с $n(G) = 6$, $m(G) = 5$. Если *граф* не слишком велик, то более наглядно представить его можно с помощью рисунка, на котором вершины изображаются кружками или иными значками, а ребра - линиями, соединяющими вершины. Заданный выше *граф* G показан на рисунке 1.3. Мы будем часто пользоваться именно этим способом представления графа, при этом обозначения вершин иногда будут помещаться внутри кружков, изображающих вершины, иногда рядом с ними, а иногда, когда имена вершин несущественны, и вовсе опускаться.

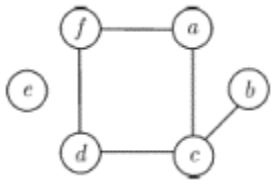


Рис. 1.3.

Некоторые специальные графы

Рассмотрим некоторые особенно часто встречающиеся графы.

Пустой граф - *граф*, не содержащий ни одного ребра. Пустой *граф* с множеством вершин $\{1, 2, \ldots, n\}$ обозначается через O_n.

Полный граф - *граф*, в котором каждые две вершины смежны. Полный *граф* с множеством вершин $\{1, 2, \ldots, n\}$ обозначается через K_n.

Граф K_1, в частности, имеет одну вершину и ни одного ребра. Очевидно, $K_1 = O_1$. Будем считать также, что существует *граф* K_0, у которого $VG = EG = \varnothing$.

Цепь (*путь*) P_n - *граф* с множеством вершин $\{1, 2, \ldots, n\}$ и множеством ребер $\{(1, 2), (2, 3) \ldots (n-1, n)\}$.

Цикл C_n - *граф*, который получается из графа P_n добавлением ребра $(1, n)$.

Все эти графы при $n = 4$ показаны на рис. 1.6

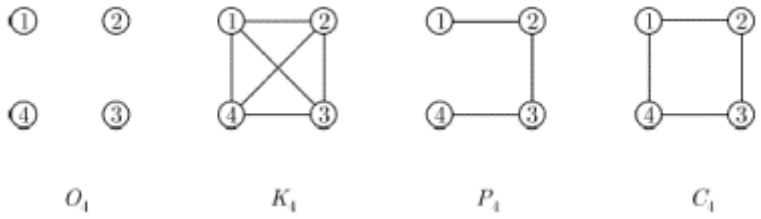


Рис. 1.6.

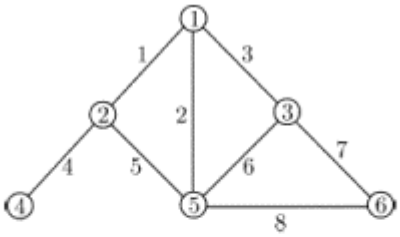
Графы и матрицы

Пусть G - *граф* с n вершинами, причем $VG = \{1, 2, \ldots, n\}$. Построим квадратную матрицу A порядка n, в которой элемент A_{ij}, стоящий на пересечении строки с номером i и столбца с номером j, определяется следующим образом:

$$A_{ij} : \begin{cases} 1, & \text{если } (i, j) \in EG, \\ 0, & \text{если } (i, j) \notin EG. \end{cases}$$

Она называется *матрицей смежности* графа. Матрицу смежности можно построить и для ориентированного графа, и для неориентированного, и для графа с петлями. Для *обыкновенного графа* она обладает двумя особенностями: из-за отсутствия петель на главной диагонали стоят нули, а так как *граф* неориентированный, то *матрица* симметрична относительно главной диагонали. Обратно, каждой квадратной матрице порядка n, составленной из нулей и единиц и обладающей двумя указанными свойствами, соответствует *обыкновенный граф* с множеством вершин $\{1, 2, \ldots, n\}$.

Другая *матрица*, ассоциированная с графом - это *матрица инцидентности*. Для ее построения занумеруем *вершины графа* числами от 1 до n, а ребра - числами от 1 до m. *Матрица* инцидентности I имеет n строк и m столбцов, а ее элемент I_{ij} равен 1, если *вершина* с номером i инцидентна ребру с номером j, в противном случае он равен нулю. На рис. 1.7 показан *граф* с занумерованными вершинами и ребрами и его матрицы смежности и инцидентности.



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad I = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Рис. 1.7.

Для ориентированного графа *матрица инцидентности* определяется несколько иначе: ее элемент I_{ij} равен 1, если *вершина* i является началом ребра j, и равен -1, если она является концом этого ребра, и он равен 0, если эта *вершина* и это *ребро* не инцидентны друг другу.

Операции над графами

Для получения новых графов можно использовать разнообразные *операции* над графами - *локальные*, при которых заменяются, удаляются или добавляются отдельные элементы графа, и *алгебраические*, когда новый *граф* строится по определенным правилам из нескольких имеющихся.

Локальные операции над графами

Простейшая операция - *удаление ребра*. При *удалении ребра* сохраняются все *вершины графа* и все его ребра, кроме удаляемого. Обратная операция - *добавление ребра*.

При *удалении вершины* вместе с вершиной удаляются и все инцидентные ей ребра. *Граф*, получаемый из графа G удалением вершины a, обозначают $G - a$. При *добавлении вершины* к графу добавляется новая *изолированная вершина*. С помощью операций добавления вершин и ребер можно "из ничего", то есть из графа K_0, построить любой *граф*.

Операция *стягивания ребра* (a, b) определяется следующим образом. Вершины a и b удаляются из графа, к нему добавляется новая *вершина* c и она соединяется ребром с каждой вершиной, с которой была смежна хотя бы одна из вершин a, b.

Операция *подразбиения ребра* (a, b) действует следующим образом. Из графа удаляется это *ребро*, к нему добавляется новая *вершина* c и два новых ребра (a, c) и (b, c). На рис. 1.10 изображены исходный *граф* G, *граф* G', полученный из него стягиванием ребра $(3, 1)$ и G'', полученный подразбиением того же ребра. В обоих случаях вновь добавленная *вершина* обозначена цифрой 1.

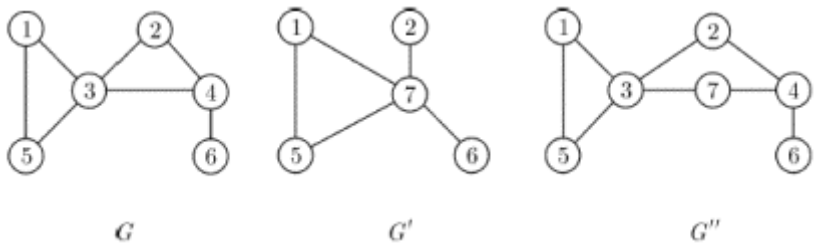


Рис. 1.10.

Способы представления графов в компьютере

Конструирование структур данных для представления в программе объектов математической модели – это основа искусства практического программирования. Далее приводится четыре различных базовых представления графов. Выбор наилучшего представления определяется требованиями конкретной задачи. Более того, при решении конкретных задач используются, как правило, некоторые комбинации или модификации указанных представлений, общее число которых необозримо. Но все они так или иначе основаны на тех базовых идеях, которые описаны в этом разделе.

Требования к представлению графов

Известны различные способы представления графов в памяти компьютера, которые различаются объемом занимаемой памяти и скоростью выполнения операций над графами. Представление выбирается, исходя из потребностей конкретной задачи. Далее приведены четыре наиболее часто используемых представления с указанием характеристики n(p,q) – объема памяти для каждого представления. Здесь p – число вершин, а q – число ребер.

Матрица смежности

Представление графа с помощью квадратной булевой матрицы M, отражающей смежность вершин, называется *матрицей смежности,* где

Для матрицы смежности $n(p,q) = O(p^2)$.

Замечание

Матрица смежности неориентированного графа симметрична относительно главной диагонали, поэтому достаточно хранить только верхнюю (или нижнюю) треугольную матрицу.

Матрица инциденций

Представление графа с помощью матрицы H, отражающей инцидентность вершин и ребер, называется *матрицей инциденций*, где для неориентированного графа

а для орграфа

Для матрицы инциденций n(p,q) = O(pq).

Обзор задач теории графов

Далее перечислим некоторые типовые задачи теории графов и их приложения:

- Задача о кратчайшей цепи

замена оборудования

составление расписания движения транспортных средств

размещение пунктов скорой помощи

размещение телефонных станций

- Задача о максимальном потоке

анализ пропускной способности коммуникационной сети

организация движения в динамической сети

оптимальный подбор интенсивностей выполнения работ

синтез двухполюсной сети с заданной структурной надежностью

задача о распределении работ

- Задача об упаковках и покрытиях

оптимизация структуры ПЗУ

размещение диспетчерских пунктов городской транспортной сети

- Раскраска в графах

распределение памяти в ЭВМ

проектирование сетей телевизионного вещания

- Связность графов и сетей

проектирование кратчайшей коммуникационной сети

синтез структурно-надежной сети циркуляционной связи

анализ надежности стохастических сетей связи

- Изоморфизм графов и сетей

структурный синтез линейных избирательных цепей

автоматизация контроля при проектировании БИС

- Изоморфное вхождение и пересечение графов

локализация неисправности с помощью алгоритмов поиска МИПГ

покрытие схемы заданным набором типовых подсхем

- Автоморфизм графов

конструктивное перечисление структурных изомеров для производных органических соединений

синтез тестов цифровых устройств

Заключение

Особенно часто в практическом программировании возникают вопросы о построении кратчайшего остова графа и нахождении максимального паросочетания.

Развитие теории графов в основном обязано большому числу всевозможных приложений. По-видимому, из всех математических объектов графы занимают одно из первых мест в качестве формальных моделей реальных систем.

Графы нашли применение практически во всех отраслях научных знаний: физике, биологии, химии, математике, истории, лингвистике, социальных науках, технике и т.п. Наибольшей популярностью теоретико-графовые модели используются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

Примеры

Пример 1

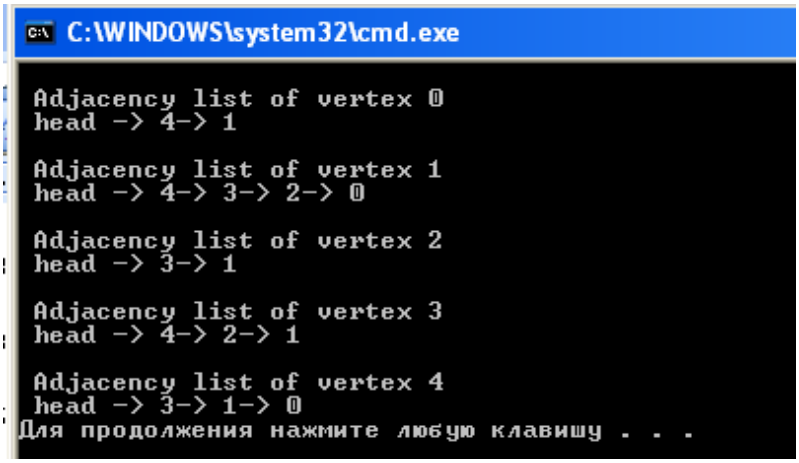
Построить неориентированный граф с занумерованными вершинами и ребрами , используя матрицу смежности.

```
#include <cstdlib>
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
// Структура для представления списка смежности узла
struct AdjListNode
{
    int dest;
    struct AdjListNode* next;
};

// Структура для представления списка смежности
struct AdjList
{
    struct AdjListNode *head;  // указатель на головной узел списка
};

// Структура для представления графа. Граф представляет собой массив из
списков смежности.
// Размер массива будет V (число вершин в графе)
struct Graph
{
    int V;
    struct AdjList* array;
};

// вспомогательная функция для создания узла нового списка смежности
struct AdjListNode* newAdjListNode(int dest)
{
    struct AdjListNode* newNode =
            (struct AdjListNode*) malloc(sizeof(struct AdjListNode));
    newNode->dest = dest;
    newNode->next = NULL;
    return newNode;
```

```
}

// Служебная функция, которая создает граф с V вершинами
struct Graph* createGraph(int V)
{
    struct Graph* graph = (struct Graph*) malloc(sizeof(struct Graph));
    graph->V = V;

    // Создание массива списков смежности. Размер массива будет V
    graph->array = (struct AdjList*) malloc(V * sizeof(struct AdjList));

     // Инициализация каждого списка смежности как пустого, сделав начало=
NULL
    int i;
    for (i = 0; i < V; ++i)
        graph->array[i].head = NULL;

    return graph;
}

//Добавляет ребро неориентированного графа
void addEdge(struct Graph* graph, int src, int dest)
{
    // Добавляет ребро из src в построение. Новый узел добавляется в список
смежности
    //  Список  src.Узел добавляется в начале
    struct AdjListNode* newNode = newAdjListNode(dest);
    newNode->next = graph->array[src].head;
    graph->array[src].head = newNode;

    // Поскольку граф неориентированный, также добавим ребро от Dest в SRC
    newNode = newAdjListNode(src);
    newNode->next = graph->array[dest].head;
    graph->array[dest].head = newNode;
}


void printGraph(struct Graph* graph)
{
    int v;
    for (v = 0; v < graph->V; ++v)
    {
        struct AdjListNode* pCrawl = graph->array[v].head;
        printf("\n Adjacency list of vertex %d\n head ", v);
        while (pCrawl)
        {
            printf("-> %d", pCrawl->dest);
            pCrawl = pCrawl->next;
        }
        printf("\n");
    }
}

// вызов фунцкий в main
int main()
{
    // создать  граф, согласно сазданнй выше фигуре
    int V = 5;
    struct Graph* graph = createGraph(V);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 4);
    addEdge(graph, 1, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 1, 4);
```

```
    addEdge(graph, 2, 3);
    addEdge(graph, 3, 4);

    // представление списка смежности вышеуказанного графа
    printGraph(graph);

    return 0;

}
```

В результате получаем следующее представление графа согласно заданным условиям:



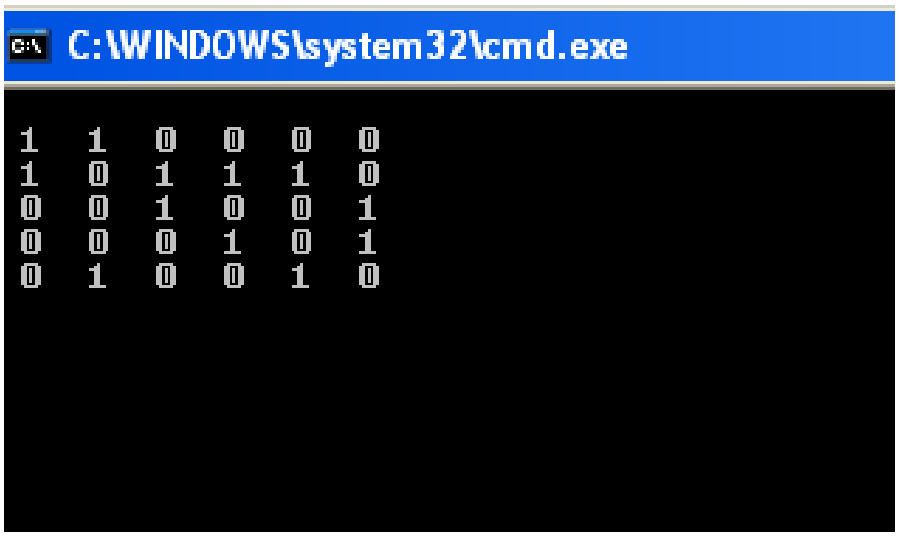
Пример 2

Есть массив MatrS[n][n] в котором записана матрица смежности графа. Написать: функцию которая создает матрицу инцидентности b[n][m] на основе даной матрицы смежности a[n][n].

```
#include<iostream>
#include <math.h>
#include <conio.h>
#include <cstdlib>
#include <stdlib.h>
using namespace std;

int find_cols_Minc(int*ar, int n)
{ int s=0; int cols;
for ( int i=0; i<n; i++)
{for (int j=0; j<n; j++)
if (ar[i*n+j]==1)  s++;
}
cols=s/2;
return cols;}

int*matr_incid(int*ar, int n, int col)
{int*ar2, j_ar2=0, i;
ar2= new int[n*col];
for(i=0; i<n*col; i++)
ar2[i]=0;
for (i=0; i<n; i++)
{for (int j=i+1; j<n; j++)
if (ar[i*n+j]==1)
```

```
{
    ar2[i*col+j_ar2]=ar2[j*col+j_ar2]=1; j_ar2++;
}
}
return ar2;
}
 void outMatrIncid (int*ar5, int n, int col)
{cout<<"\n";
for (int i=0; i<n; i++)
{for (int j=0; j<col; j++)
cout<<" "<<ar5[i*col+j]<<" ";
cout<<"\n";}
}
int main()
{int*ar3,*ar4; int n=5;
ar3 =new int [n*n]; int col_inc;
ar3[0*n+0]=0; ar3[0*n+1]=1; ar3[0*n+2]=0; ar3[0*n+3]=0; ar3[0*n+4]=1;
ar3[1*n+0]=1; ar3[1*n+1]=0; ar3[1*n+2]=1; ar3[1*n+3]=1; ar3[1*n+4]=1;
ar3[2*n+0]=0; ar3[2*n+1]=1; ar3[2*n+2]=0; ar3[2*n+3]=1; ar3[2*n+4]=0;
ar3[3*n+0]=0; ar3[3*n+1]=1; ar3[3*n+2]=1; ar3[3*n+3]=0; ar3[3*n+4]=0;
ar3[4*n+0]=1; ar3[4*n+1]=1; ar3[4*n+2]=0; ar3[4*n+3]=0; ar3[4*n+4]=0;
col_inc=find_cols_Minc(ar3,n);// теперь в col_inc кол-во столбцов для
создания матрицы инцидентности

ar4=matr_incid(ar3, n, col_inc);
outMatrIncid (ar4, n, col_inc);
getch();
}
```

В результате работы программы выводится матрица инцидентности.



Варианты индивидуальных заданий.
Задание 1

1) Построить граф, используя язык C, согласно данной схеме на рис.1.
2) По запросу пользователя должны удаляться:

- все рёбра с номером 3,
- вершина №6,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

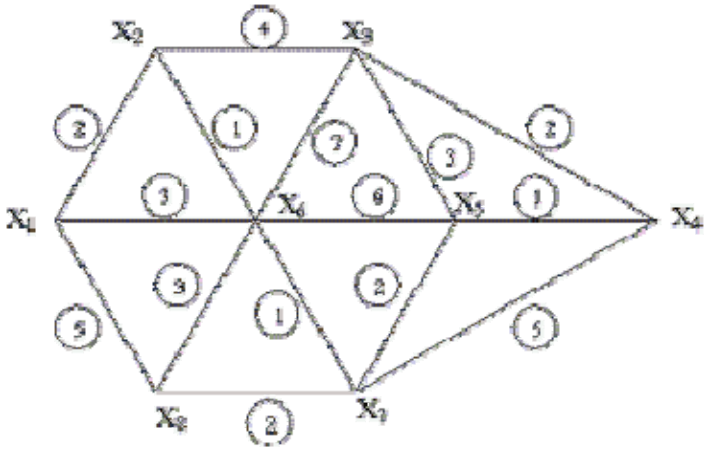


Рис. 1

Задание 2

1) Построить граф, используя язык C, согласно данной схеме на рис.1.
2) По запросу пользователя должны удаляться:

- все рёбра с номером 2,
- вершина №7,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

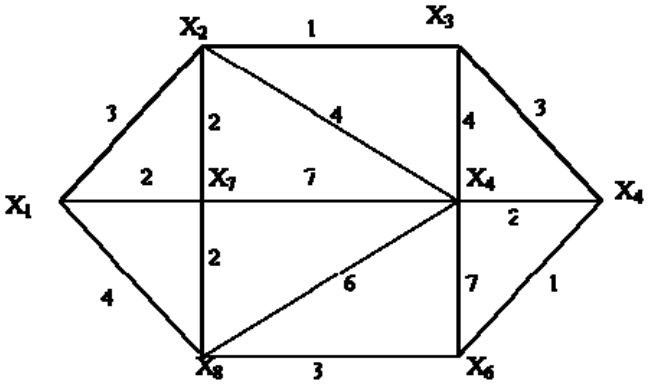


Рис. 1

Задание 3

1) Построить граф, используя язык C, согласно данной схеме на рис.1.
2) По запросу пользователя должны удаляться:

- все рёбра с номером 3,
- вершина №6,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

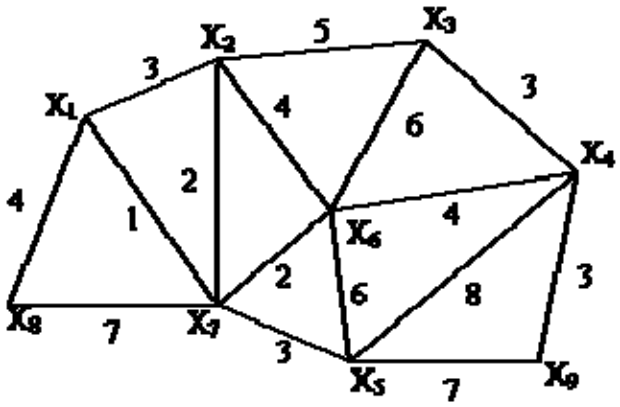


Рис. 1

Задание 4

1) Построить граф, используя язык C, согласно данной схеме на рис.1.
2) По запросу пользователя должны удаляться:

- все рёбра с номером 8,
- вершина №5,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

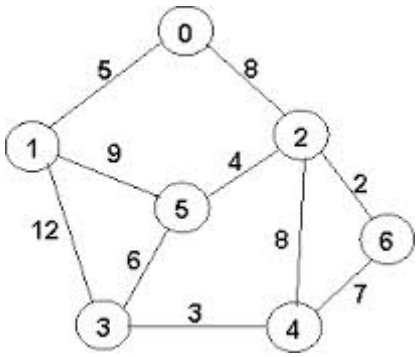


Рис. 1

Задание 5

1) Построить граф, используя язык С, согласно данной схеме на рис.1.
2) По запросу пользователя должны удаляться:
- все рёбра с номером 4,
- вершина №5,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

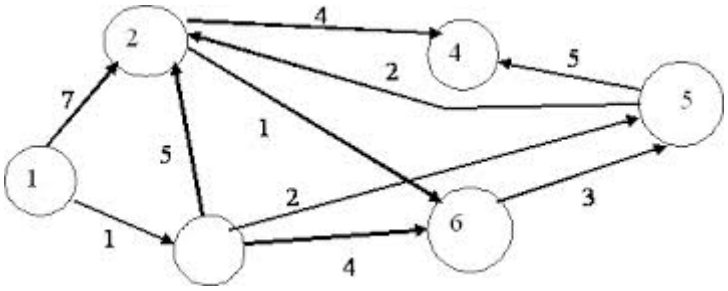


Рис. 1

Задание 6

3) Построить граф, используя язык С, согласно данной схеме на рис.1.
4) По запросу пользователя должны удаляться:
- все рёбра с номером 1,
- вершина №3,
- вывести на экран общий вид получившегося в результате графа.

3) Разработать алгоритм построения матрицы смежности для данного графа.

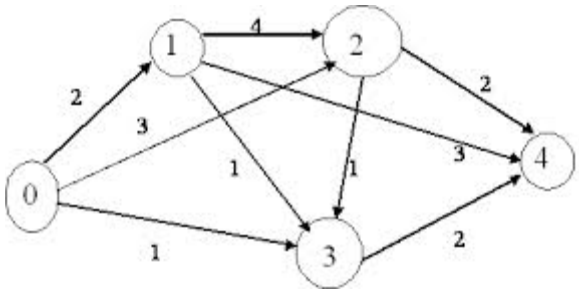


Рис. 1

Задание 7

Для неориентированного графа, представленного матрицей смежности построить все цепи, исходящие из заданной вершины. Программа должна реализовывать следующие функции:

- Функцию, позволяющую записывать данные о графе в текстовой файл.
- Функцию, позволяющую считывать данные о графе из текстового файла.
- Функцию, позволяющую заполнять граф случайно, запросив у пользователя количество вершин и ребер (данные об инцидентности задаются тоже случайным образом).
- Функцию, позволяющую по данной матрице смежности (данные задаются с консоли) построить матрицу инцидентности.
- Функцию добавления вершины. Данные об инцидентных рёбрах задаются с консоли.

Задание 8

Для неориентированного графа, представленного матрицей инцидентности построить все цепи, исходящие из заданной вершины. Программа должна реализовывать следующие функции:

- Функцию, позволяющую записывать данные о графе в текстовой файл.
- Функцию, позволяющую считывать данные о графе из текстового файла.
- Функцию, позволяющую заполнять граф случайно, запросив у пользователя количество вершин и ребер (данные об инцидентности задаются тоже случайным образом).
- Функцию, позволяющую по данной матрице инцидентности (данные задаются с консоли) построить матрицу смежности.
- Функцию добавления ребра. Данные об инцидентных вершинах задаются с консоли.