

**Олимпиадная работа по ИКТ:  
“Нахождение кратчайшего пути  
с использованием графов и алгоритма  
Дейкстры”**

Ученика **10** “В” класса гимназии  
г. Обнинска  
Кашкарова Михаила

**(Объектно-ориентированное  
программирование - Delphi)**

# Содержание:

- Графы: определения и примеры
- Ориентированные графы
- Путь в орграфе
- Матрица смежности
- Иерархический список
- Алгоритм Дейкстры
- Программа “ProGraph”
- **Описание работы программы**
- **Достоинства программы**
- **Список литературы**

# Графы: определения и примеры

---

- Говоря простым языком, *граф* - это множество точек (для удобства изображения - на плоскости) и попарно соединяющих их линий (не обязательно прямых). В *графе* важен только факт наличия связи между двумя *вершинами*. От способа изображения этой связи структура *графа* не зависит.

# Например, три *графа* на рис. 1 совпадают

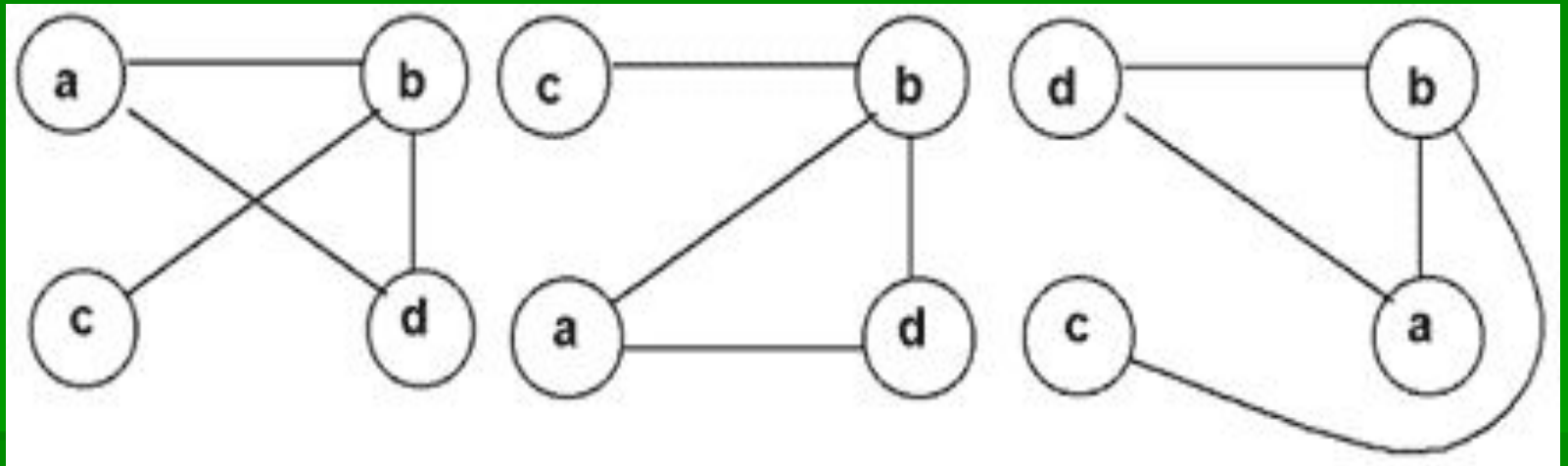


Рис. 1. Три способа изображения одного графа

# А два графа на рис. 2 - различны

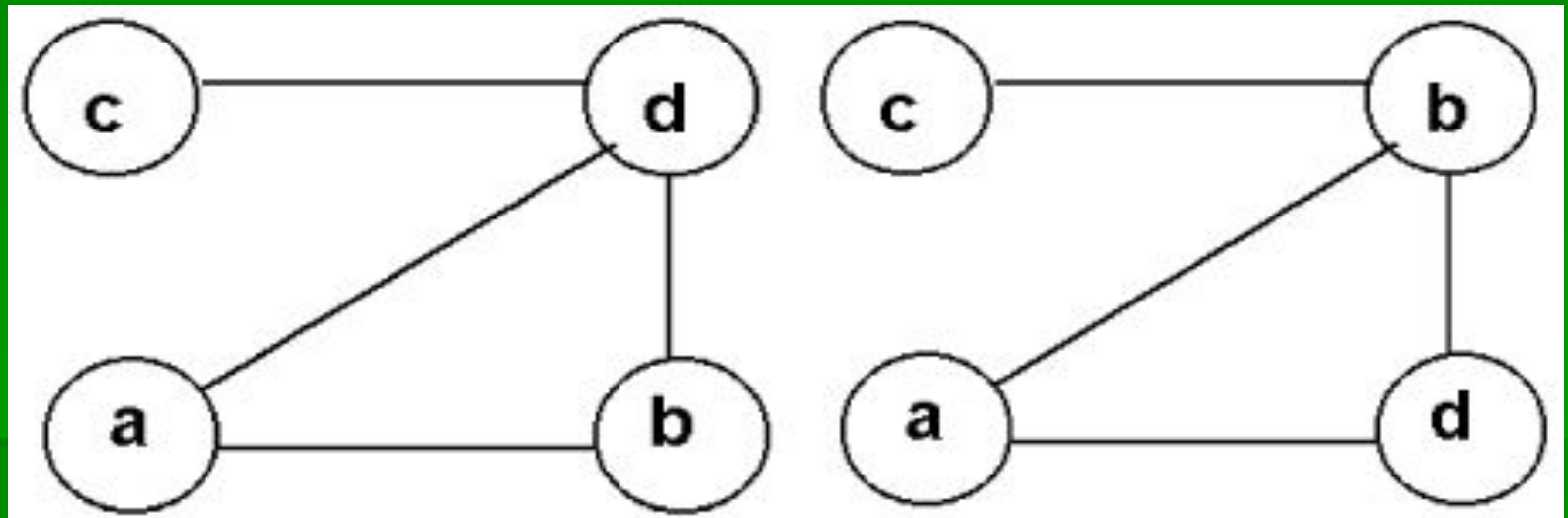


Рис. 2. Пример двух разных графов

# *Степень вершины*

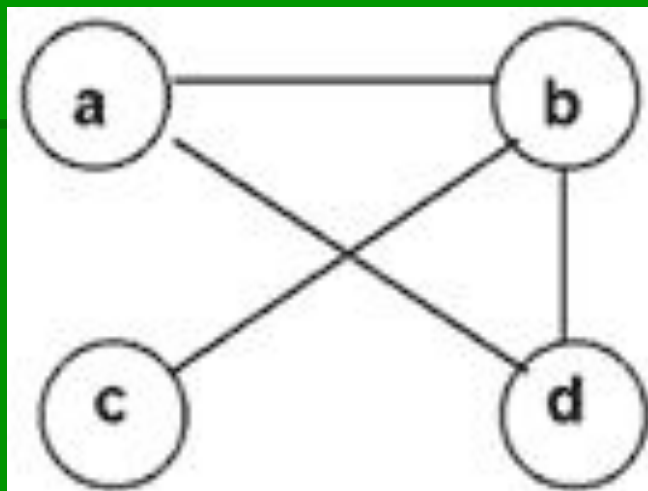
- Любому ребру соответствует ровно две вершины, а вот вершине может соответствовать произвольное количество ребер, это количество и определяет **степень вершины**.  
*Изолированная вершина вообще не имеет ребер (ее степень равна 0).*

# Смежные вершины и рёбра

- *Две вершины* называются **смежными**, если они являются разными концами одного *ребра*.
- *Два ребра* называются **смежными**, если они **выходят** из одной *вершины*.

# Путь в графе

- *Путь в графе* - это последовательность вершин (без повторений), в которой любые две соседние вершины смежны. Например, в изображенном графе, есть два различных пути из вершины *a* в вершину *c*: *adbс* и *abc*.





# Достижимость

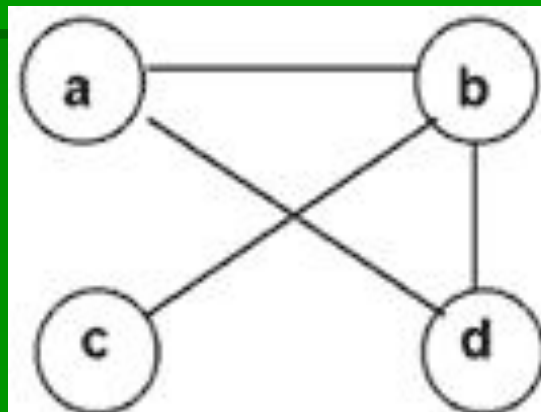
- *Вершина  $v$  достижима из вершины  $u$ , если существует путь, начинающийся в  $u$  и заканчивающийся в  $v$ .*
- *Граф называется **связным**, если все его вершины взаимно достижимы.*

# Длина пути

**Длина пути** - количество ребер, из которых этот *путь* состоит. Например, *длина* уже упомянутых *путей* *adbc* и *abc* - 3 и 2 соответственно.

**Расстояние** между между вершинами *u* и *v* - это *длина* кратчайшего *пути* от *u* до *v*. Из этого определения видно, что *расстояние* между вершинами *a* и *c* в *графе* на рис. равно 2.

- **Цикл** - это замкнутый *путь*. Все *вершины* в *цикле*, кроме первой и последней, должны быть различными. Например, *циклом* является *путь* *abda* в *графе* на рис.



# Примеры неориентированных графов

<b>Граф</b>	<b>Вершины</b>	<b>Ребра</b>
Семья	Люди	Родственные связи
Город	Перекрестки	Улицы
Сеть	Компьютеры	Кабели
Домино	Костяшки	Возможность
Дом	Квартиры	Соседство
Лабиринт	Развилки и тупики	Переходы
Метро	Станции	Пересадки
Листок в клеточку	Клеточки	Наличие общей границы

# Ориентированные графы

- **Орграф** - это граф, все ребра которого имеют направление. Такие направленные ребра называются **дугами**. На рисунках дуги изображаются стрелочками ( рис.3)

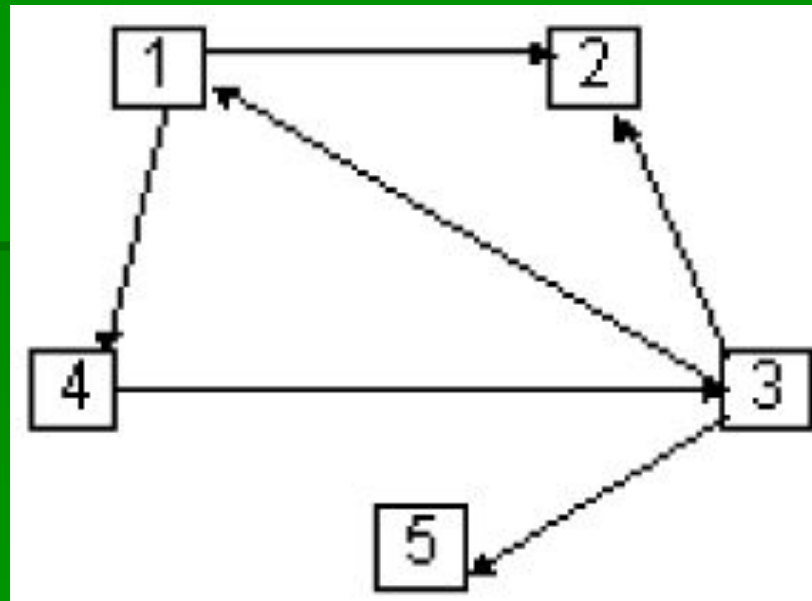


Рис. 3. Орграф

# Смешанный граф

- В отличие от ребер, *дуги* соединяют две неравноправные *вершины*: одна из них называется **началом дуги** (*дуга* из нее **исходит**), вторая - **концом дуги** (*дуга* в нее **входит**). Можно сказать, что любое *ребро* - это пара *дуг*, направленных навстречу друг другу.
- Если в *графе* присутствуют и *ребра*, и *дуги*, то его называют **смешанным**

# Путь в орграфе

- *Путь в орграфе* - это последовательность вершин (без повторений), в которой любые две соседние вершины смежны, причем каждая вершина является одновременно концом одной дуги и началом следующей дуги. Например, в орграфе на рис. 3 нет пути, ведущего из вершины 2 в вершину 5. "Двигаться" по орграфу можно только в направлениях, заданных стрелками.

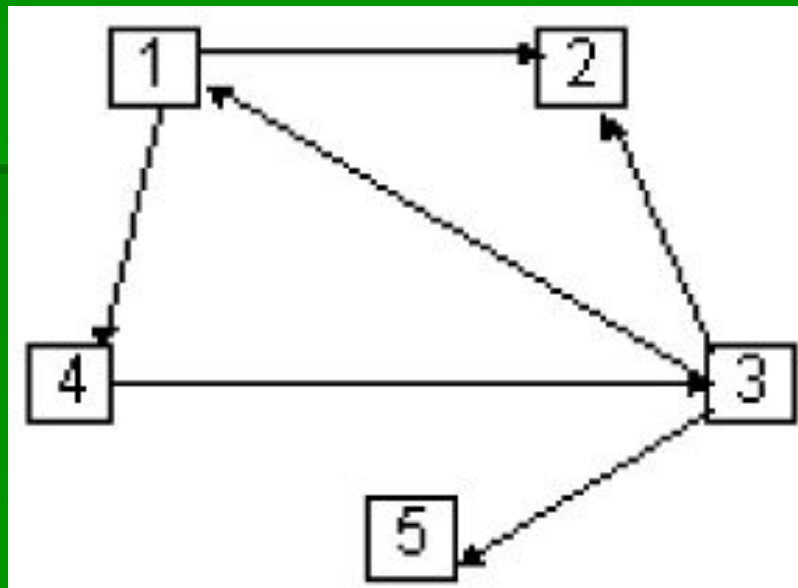


Рис. 3. Орграф

# Примеры ориентированных графов

<b>Орграф</b>	<b>Вершины</b>	<b>Дуги</b>
Чайнворд	Слова	Совпадение последней и первой букв (возможность связать два слова в цепочку)
Стройка	Работы	Необходимое предшествование (например, стены нужно построить раньше, чем крышу, т. п.)
Обучение	Курсы	Необходимое предшествование (например, курс по языку Pascal полезно изучить прежде, чем курс по Delphi, и т.п.)
Одевание ребенка	Предметы гардероба	Необходимое предшествование (например, носки должны быть надеты раньше, чем ботинки, и т.п.)
Европейский город	Перекресток	Узкие улицы с односторонним движением

# Взвешенные графы

- **Взвешенный** (другое название: **размеченный**) *граф* (или *орграф*) - это *граф* (*орграф*), некоторым элементам которого (*вершинам, ребрам* или *дугам*) сопоставлены числа. Наиболее часто встречаются *графы* с помеченными *ребрами*. Числа-пометки носят различные названия: **вес, длина, стоимость**.

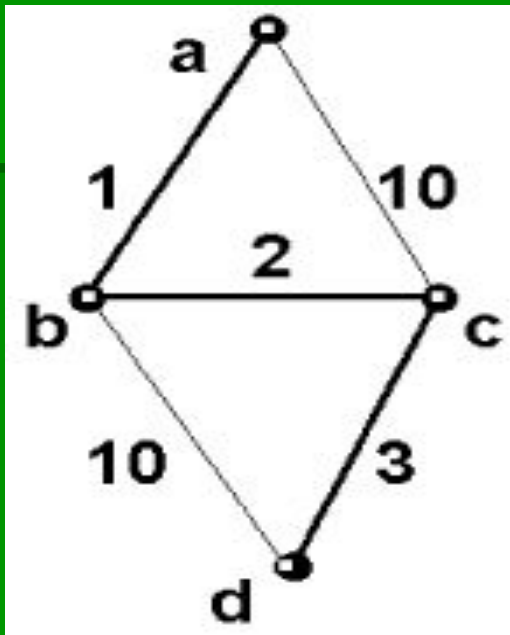


Рис. 4 Взвешенный граф



# Длина пути во взвешенном графе

- *Длина пути* во взвешенном (связном) графе - это сумма длин (весов) тех ребер, из которых состоит *путь*. *Расстояние* между вершинами - это, как и прежде, *длина кратчайшего пути*. Например, *расстояние* от вершины *a* до вершины *d* во взвешенном графе, изображенном на рис. 4, равно 6.

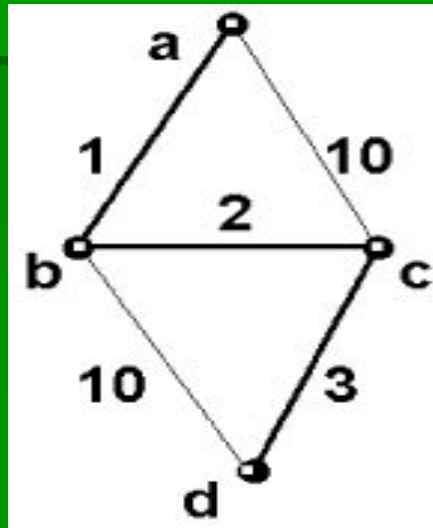


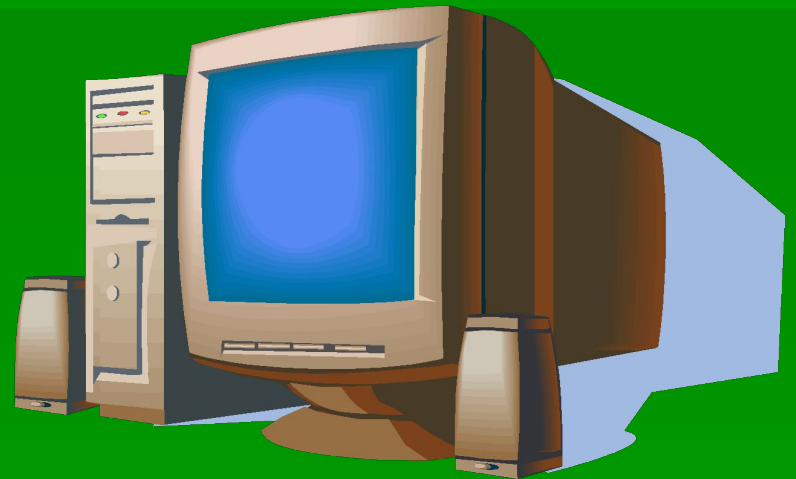
Рис. 4 Взвешенный граф

# Примеры взвешенных графов

<b>Граф</b>	<b>Вершины</b>	<b>Вес вершины</b>	<b>Ребра (дуги)</b>	<b>Вес ребра (дуги)</b>
Таможни	Государства	Площадь территории	Наличие наземной границы	Стоимость получения визы
Переезды	Города	Стоимость ночевки в гостинице	Дороги	Длина дороги
Супер-чайнворд	Слова	-	Совпадение конца и начала слов (возможность "сцепить" слова)	Длина пересекающихся частей
Карта	Государства	Цвет на карте	Наличие общей границы	-
Сеть	Компьютеры	-	Сетевой кабель	Стоимость кабеля

# Способы представления графов

- Существует довольно большое число разнообразных способов представления *графов*. Однако мы изложим здесь только самые полезные с точки зрения программирования.



# Матрица смежности

- **Матрица смежности**  $S_m$  - это квадратная матрица размером  $N \times N$  ( $N$  - количество вершин в графе), заполненная по следующему правилу:
  - Если в графе имеется ребро  $e$ , соединяющее вершины  $u$  и  $v$ , то  $S_m[u,v] = V_{es}(e)$ , в противном случае  $S_m[u,v] = "-"$ .

# Пример матрицы смежности

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
<b>a</b>	0	1	10	-
<b>b</b>	1	0	2	10
<b>c</b>	10	2	0	3
<b>d</b>	-	10	3	0

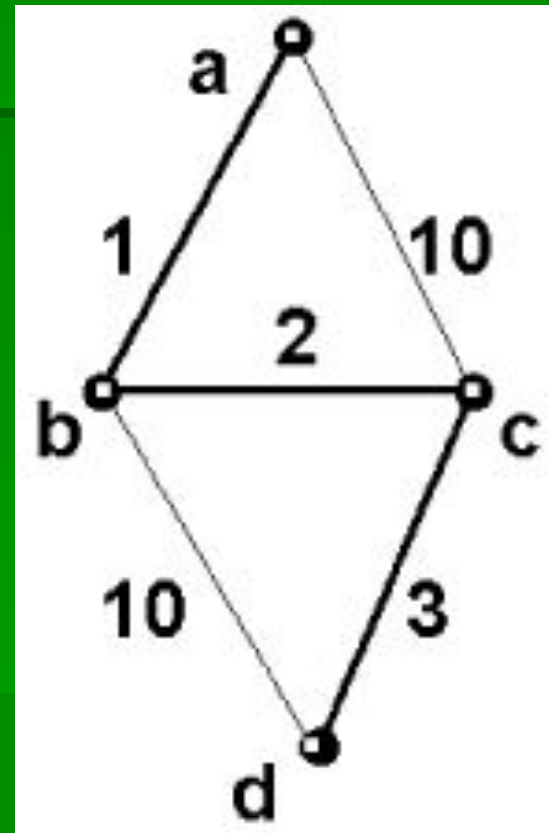


Рис. 4 Взвешенный граф

# Преимущества матрицы смежности

- Удобство *матрицы смежности* состоит в наглядности и прозрачности алгоритмов, основанных на ее использовании. А неудобство - в несколько завышенном требовании к памяти: если *граф* далек от полного, то в массиве, хранящем *матрицу смежности*, оказывается много "пустых мест" (нулей). Кроме того, для "общения" с пользователем этот способ представления *графов* не слишком удобен: его лучше применять только для внутреннего представления данных.

# Иерархический список

- В одном линейном списке содержатся номера "начальных вершин", а в остальных - номера смежных *вершин* или указатели на эти *вершины*. В качестве примера приведем *иерархический список*, задающий *орграф*, изображенный на рис.3

# Пример иерархического списка

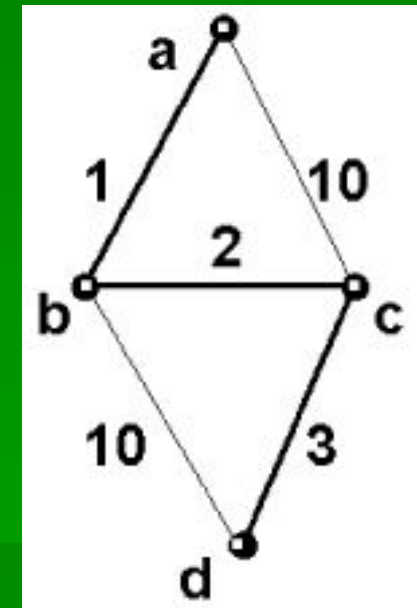


Рис. 5 Пример иерархического списка

Рис. 3 Ограф



# Преимущества иерархического списка

Вершина = запись

- Номер: Число;
- Имя: Строка;
- След Вершина: указатель на Вершина;
- Список Дуг: Дуга;
- end;

Дуга = запись

- Стоимость: Число;
- Конец Дуги: Вершина;
- След Дуга: Дуга;
- end;

Очевидное преимущество такого способа представления *графов* заключается в экономичном использовании памяти. И даже небольшая избыточность данных, к которой приходится прибегать в случае неориентированного *графа*, задавая каждое *ребро* как две *дуги*, искупается гибкостью всей структуры, что особенно удобно при необходимости частых перестроений в процессе работы программы.

# Программа “ProGraph”

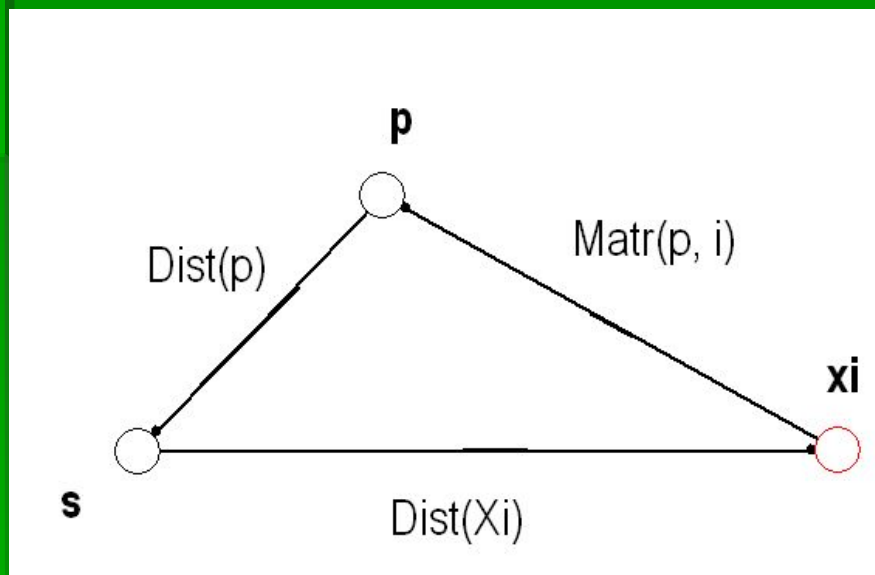
- Программа “ProGraph” была специально создана для создания графов в графической оболочке. Поддерживает возможность добавления алгоритмов на графах.

# Алгоритм Дейкстры

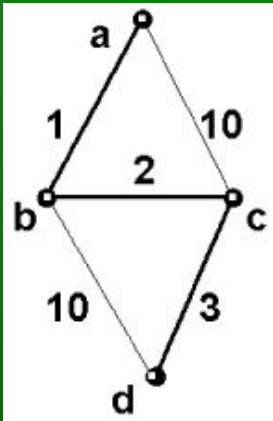
- Мы рассмотрим один из основных алгоритмов поиска кратчайших путей в графе – алгоритм Дейкстры, применимый для графов с неотрицательными весами.

# Описание алгоритма

- Основная идея основана на простой формуле:  
( $\text{Dist}(x)$  – расстояние до вершины  $x$  из исходной)
- $\text{Dist}(X_i) = \text{Минимум}(\text{Dist}(X_i), \text{Dist}(p) + \text{Matr}(p, i))$



# Описание алгоритма



	a	b	c	d
a	0	1	10	-
b	1	0	2	10
c	10	2	0	3
d	-	10	3	0

- Допустим, нам надо найти кратчайший путь из вершины A в вершину D.
- Перебираем все возможные расстояния до вершин, находим из них минимальное и выводим кратчайший путь.

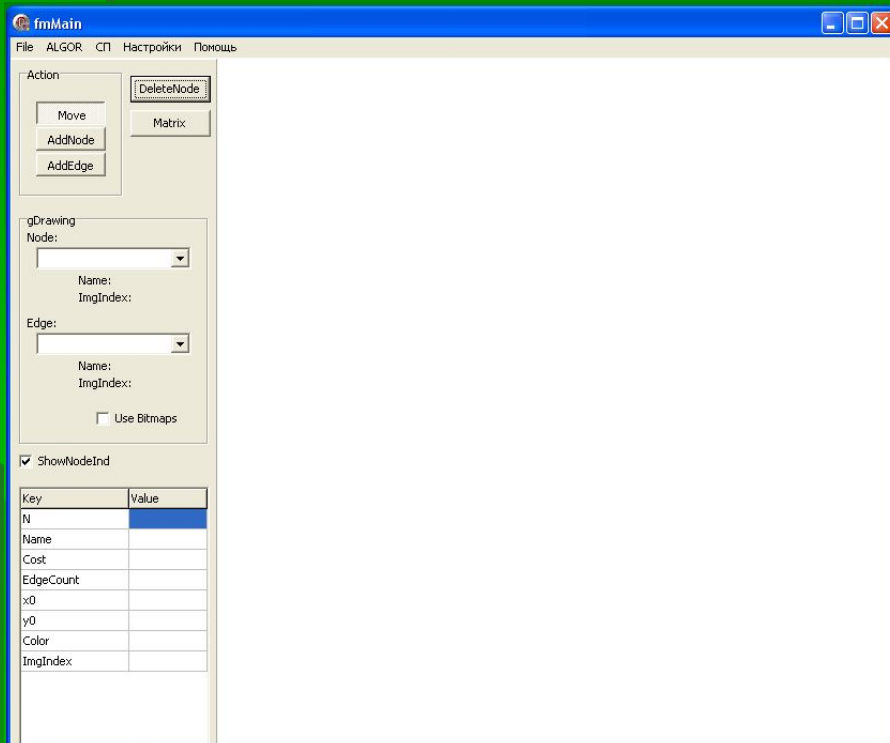
# Описание работы программы

Работа делится на две части:

- 1.Создание графа в Редакторе.
- 2.Применение алгоритма Дейкстры к получившемуся графу и просмотр результата.

# Создание графа в Редакторе

## 1) Запустите программу “ProGraph.exe”



Вы увидите это окно.

В данном окне вы должны ввести параметры:

Количество вершин графа ('AddNode')

Ребра и их вес ('AddNode', 'Matrix' – веса ребер)

Имена вершин

(ПКМ на вершине, поле 'NodeName')

Здесь вы можете дополнительно выбрать графическое изображение вершин.

# Создание графа в Редакторе

1) У вас должно получиться примерно так:

Key	Value
N	5
Name	Мой компьютер
Cost	0
EdgeCount	2
x0	569
y0	146
Color	clWhite
ImgIndex	2

Мы видим пример сети, оформленной в виде графа. Расстояние между вершинами показаны на линиях. В оформлении вершин используется пиктограмма компьютера.

Для сохранения полученного графа выбираем из меню File -> Save as и сохраняем под любым именем.

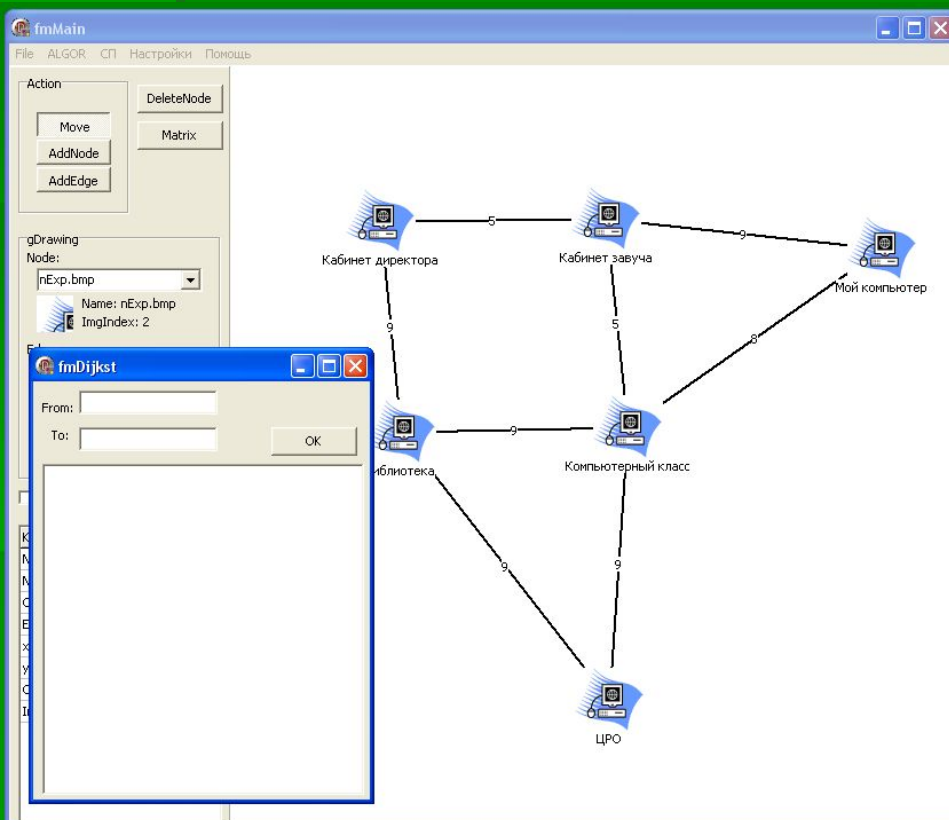
Полученную заготовку будем использовать для второй части.



# Применение алгоритма Дейкстры

Для вызова программы загружаем (File -> Load) ранее сохранённый файл. Открываем из главного меню 'ALGOR -> algor\_Dijkst'.

Появится новое окно, в котором необходимо задать начальный и конечный номер вершины.



(Чтобы переключить показ 'Имена вершин/Индексы' необходимо поставить флажок в поле 'ShowNodInd')

Заполнить поля 'From' и 'To' и запустить алгоритм на выполнение ('OK')

# Просмотр результата

Вы увидите результат работы:

В окне задания параметров появится строка с длиной кратчайшего пути и сам путь.

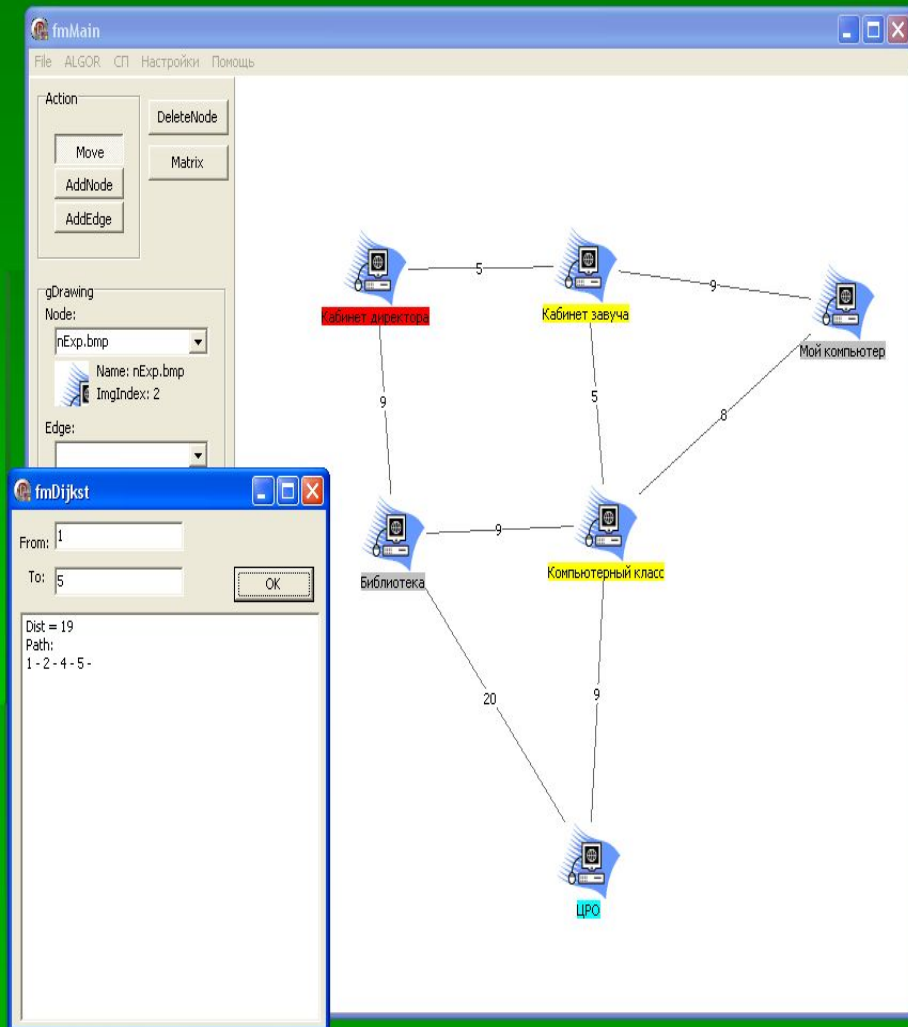
В окне редактора отобразится пройденный путь и вершины окрасятся в следующие цвета:

**Красный** – начальная вершина.

**Синий** – конечная вершина.

**Желтый** – вершины искомого пути.

Серый – вершины, посещенные при работе алгоритма, но не включённые в конечный путь.



# Достоинства программы

С помощью этой программы вы можете создать любой граф с помощью удобного редактора графов: схема метро, карта городов, компьютерные сети, карту лабиринта и многое другое.

Представить его в графическом виде, добавляя названия вершин, пиктограммы, расстояния.

Определить кратчайший путь между двумя заданными вершинами и увидеть результат работы алгоритма в графическом и текстовом виде.

Программа была создана на языке “Delphi” с использованием объектно-ориентированного программирования.

Данная программа может быть использована для подготовки к ЕГЭ по информатике.

## Список использованной литературы

**Кирюхин В.М., Лапунов А.В., Окулов С.М.** Задачи по информатике: Международные олимпиады 1989 – 1996: Для факультативов по информатике в старших классах. – М.: АБФ, 1996

**Боглаев Ю.П.** Вычислительная математика и программирование. – М.: Высшая школа, 1990

**Кушниренко А.Г., Лебедев Г.В.** Программирование для математиков. – М.: Наука, 1988.

**Майерс Г.** Искусство тестирования программ. – М.: Финансы и статистика, 1982.

**Никольская И.Л.** Математическая логика. – М.: Высшая школа, 1981.