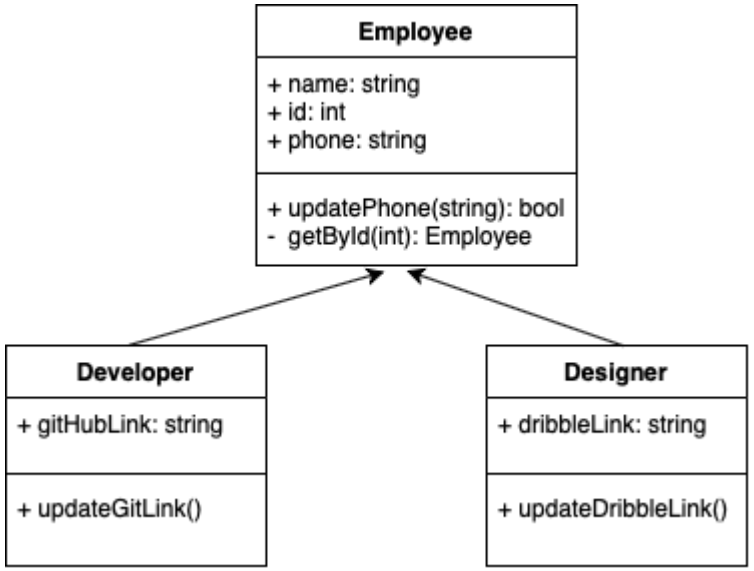
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

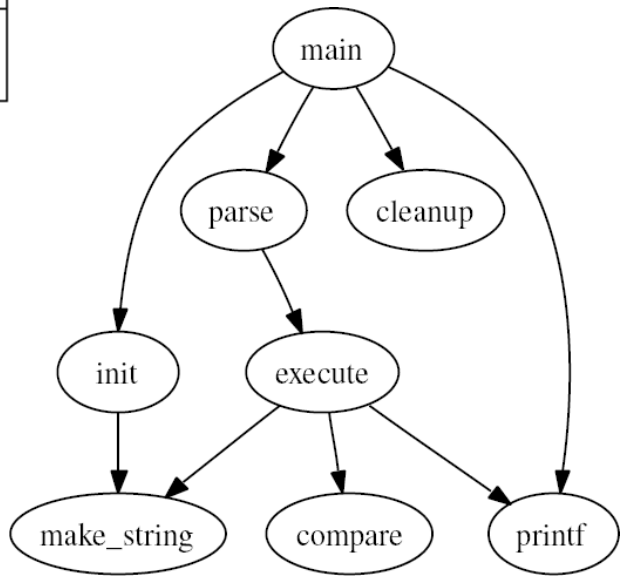
Компьютерные технологии в научных исследованиях

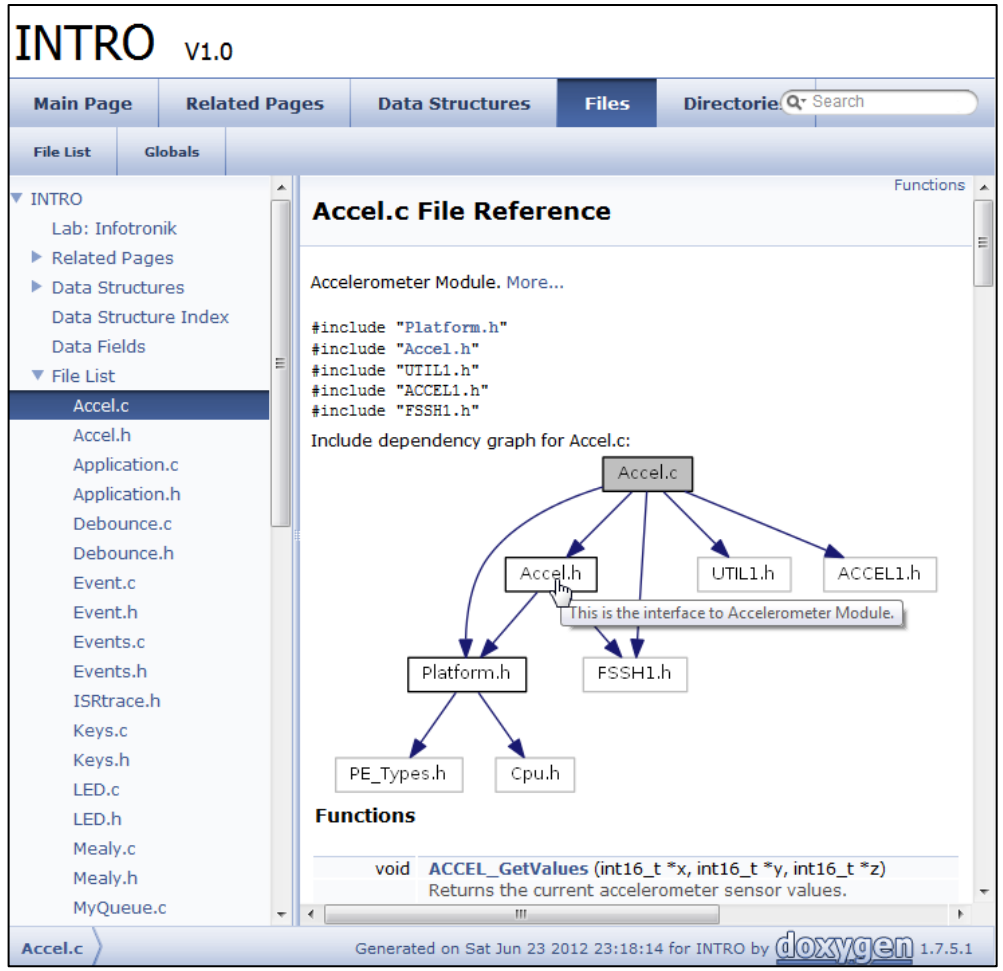
Семинар №5

Работа с пакетом graphviz

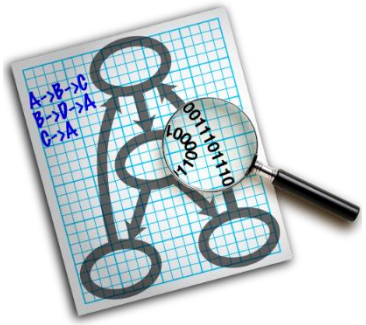
Задача визуализации графов



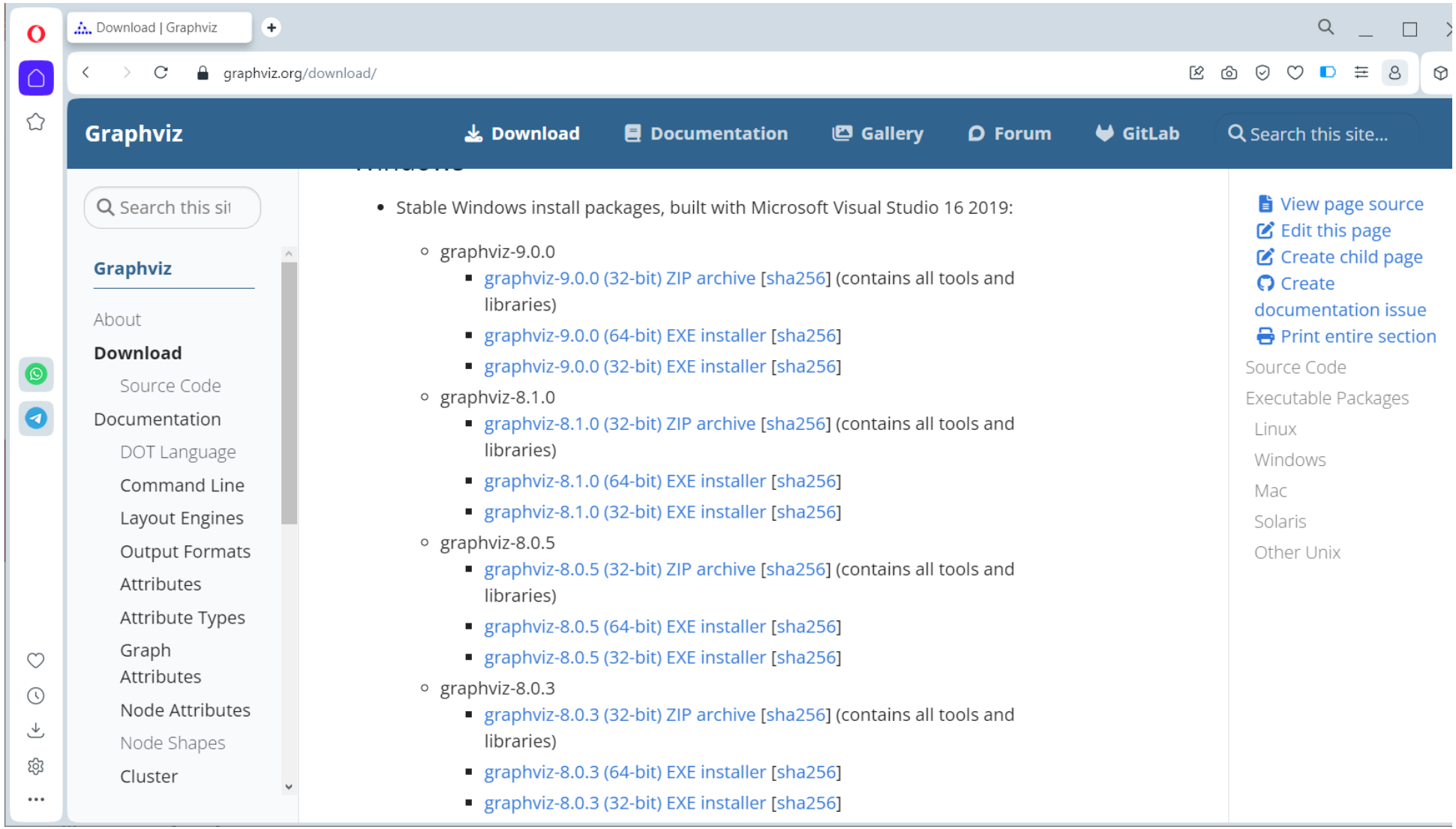




Программа Graphviz



URL: https://graphviz.org



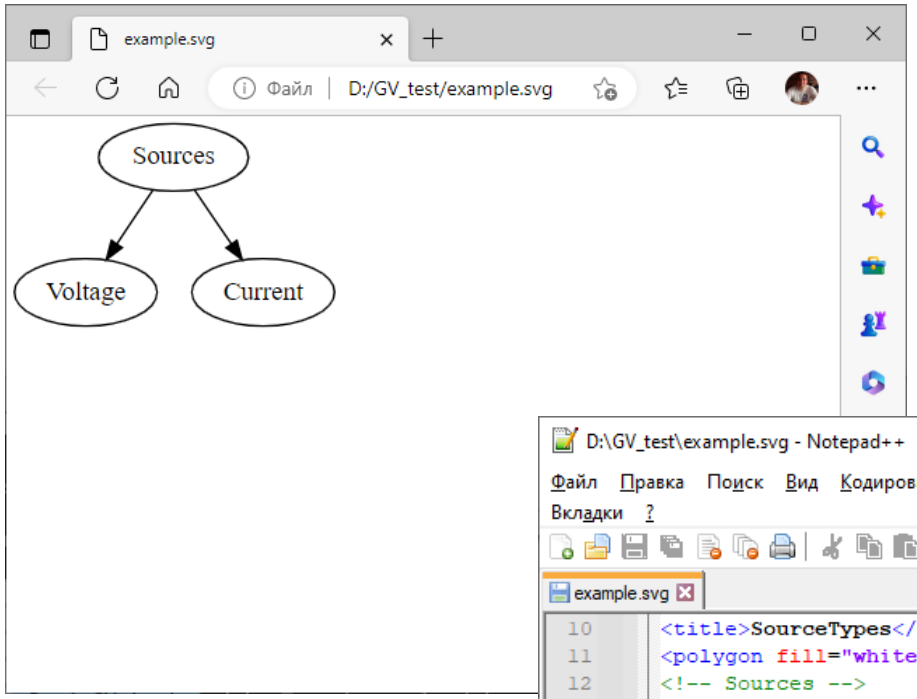
Выходные форматы GraphViz

Форматы изображений:

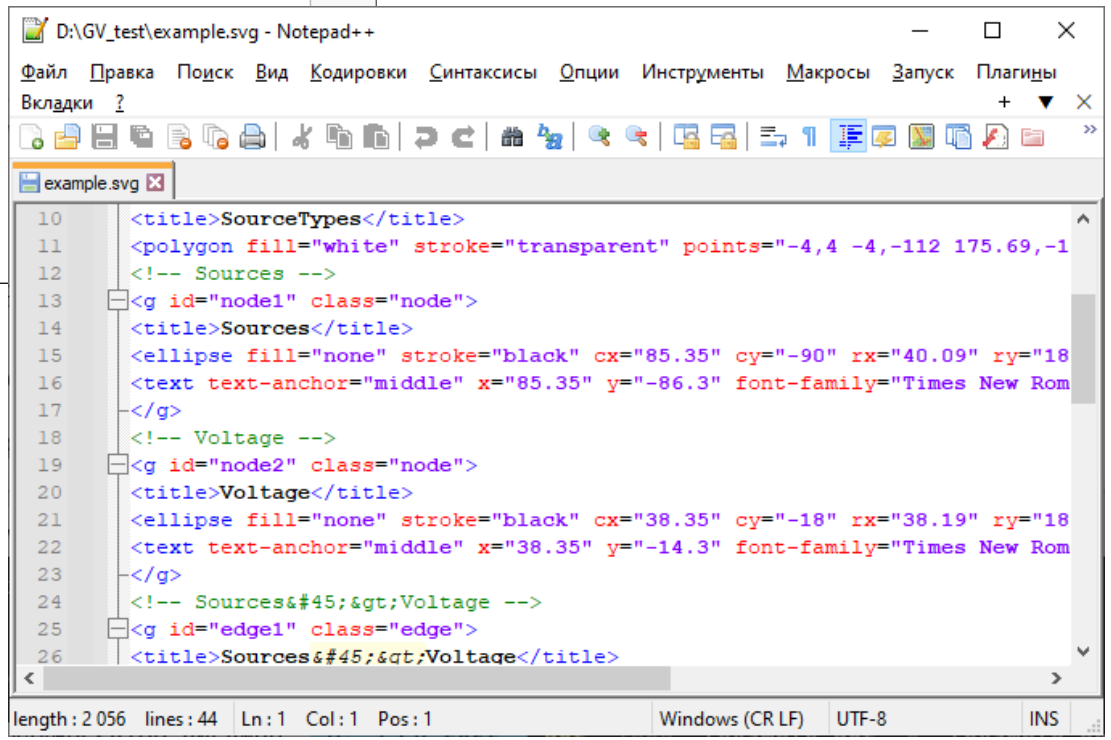
- bmp
- gif
- ico
- jpeg
- png
- psd
- tga
- tiff
- webp

Текстовые форматы изображений:

- fig
- pic
- ps
- svg



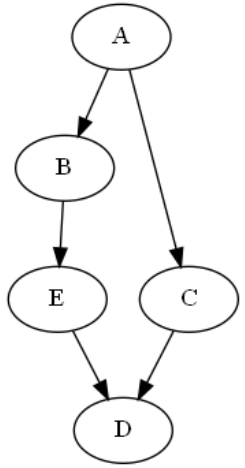

```
dot
-Tpng
<имя файла с графом>
-o
<имя файла с результатом>
```



Синтаксис формата dot

граф	::=	[strict] (graph|digraph) [имя_графа] '{' список_объявлений '}'
список_объявлений	::=	[объявление [';'] список_объявлений]
объявление	::=	подграф | узел | ребро | атрибут | имя '=' имя
подграф	::=	[subgraph [имя_подграфа]] '{' список_объявлений '}'
узел	::=	имя_узла [список_атрибутов]
ребро	::=	(имя_узла | имя подграфа) данные_ребра [список_атрибутов]
данные_ребра	::=	тип_связи (имя_узла | имя подграфа) [данные ребра]
тип_связи	::=	(-- | ->)

```
digraph {
  A -> B;
  A -> C -> D;
  B -> E -> D;
}
```

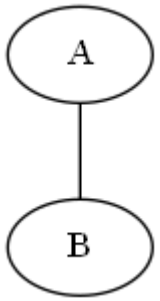


Типы графов: направленный и ненаправленный (1)

graph

Создаёт неориентированный граф:

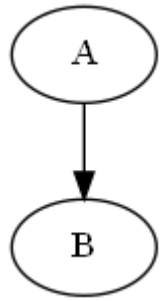
```
graph {
    A -- B
}
```



digraph

Создаёт ориентированный граф

```
digraph {
    A -> B
}
```



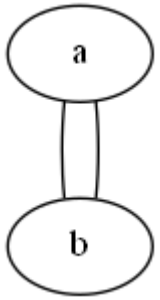
Описание графов

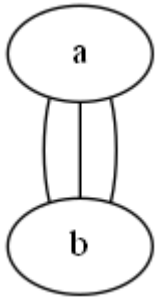
```
graph {
  a -- b
  a -- b
}
```

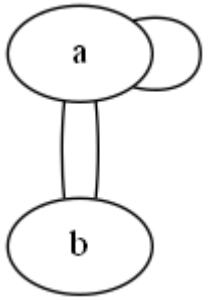
```
graph {
  a -- b
  b -- a
}
```

```
graph {
  a -- b
  a -- b
  b -- a
}
```

```
graph {
  a -- b
  a -- a
  b -- a
}
```

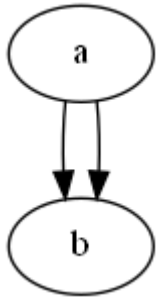




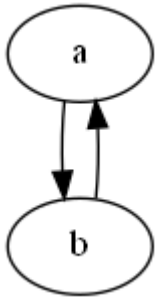


Описание орграфов

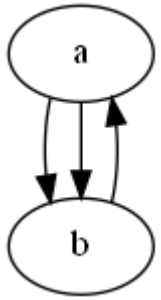
```
digraph {
  a -> b
  a -> b
}
```



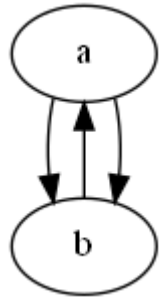

```
digraph {
  a -> b
  b -> a
}
```



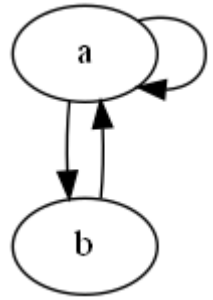

```
digraph {
  a -> b
  a -> b
  b -> a
}
```




```
digraph {
  a -> b
  b -> a
  a -> b
}
```



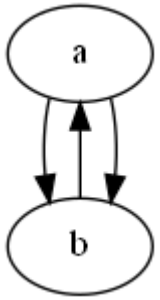

```
digraph {
  a -> b
  b -> a
  a -> a
}
```



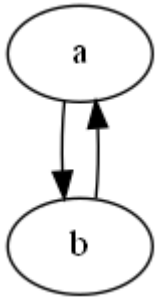
Ограничение числа рёбер: ключевое слово strict

Пример для ориентированных графов

```
digraph {
  a -> b
  b -> a
  a -> b
}
```

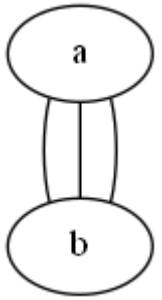



```
strict digraph {
  a -> b
  b -> a
  a -> b
}
```

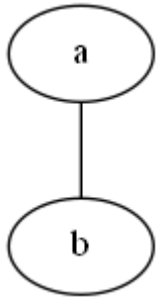


Пример для неориентированных графов

```
graph {
  a -- b
  a -- b
  b -- a
}
```

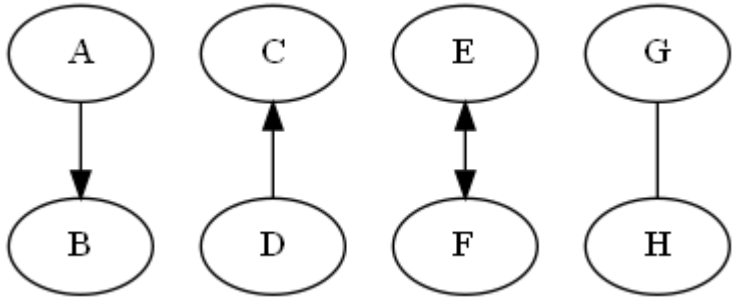



```
strict graph {
  a -- b
  a -- b
  b -- a
}
```

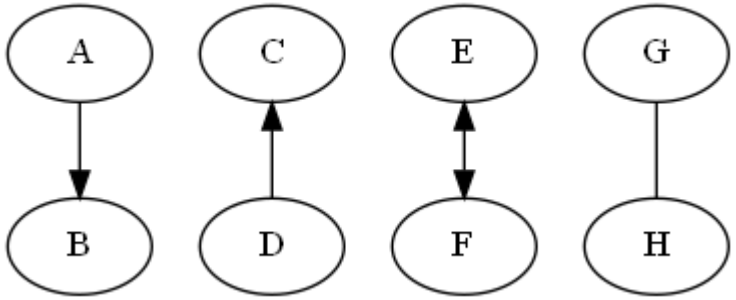


Атрибуты. Явное задание ориентации ребра (3)

```
digraph {
  a -> b [dir=forward]
  c -> d [dir=back]
  e -> f [dir=both]
  g -> h [dir=none]
}
```




```
graph {
  a -- b [dir=forward]
  c -- d [dir=back]
  e -- f [dir=both]
  g -- h [dir=none]
}
```



Возможные значения атрибута dir (применим только для рёбер):

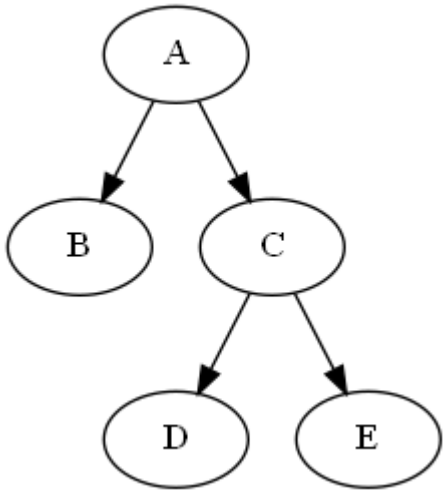
`forward` - значение по умолчанию для орграфов
`back`
`both`
`none` - значение по умолчанию для неориентированных графов

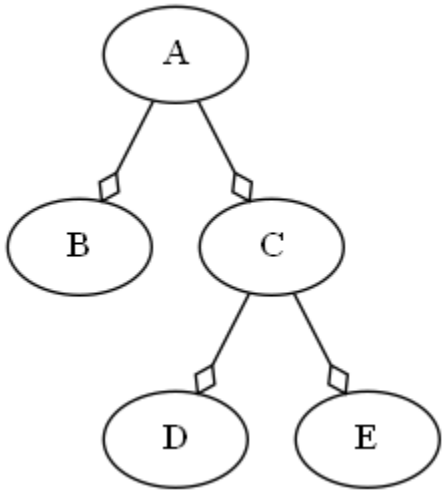
Описание для группы рёбер

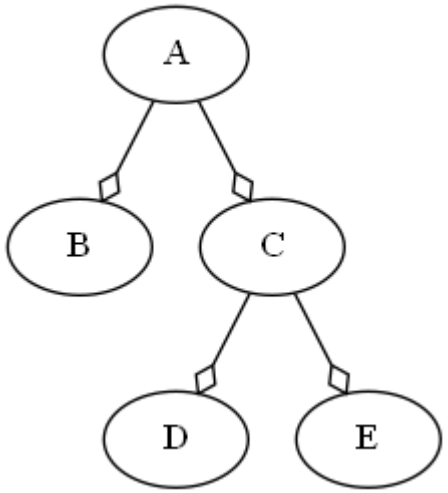
```
digraph {
  A -> B
  A -> C
  C -> D
  C -> E
}
```

```
digraph {
  A -> B [arrowhead=odiamond]
  A -> C [arrowhead=odiamond]
  C -> D [arrowhead=odiamond]
  C -> E [arrowhead=odiamond]
}
```

```
digraph {
  edge   [arrowhead=odiamond]
  A -> B
  A -> C
  C -> D
  C -> E
}
```

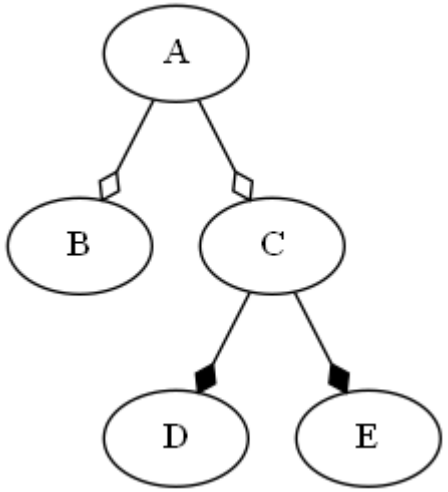




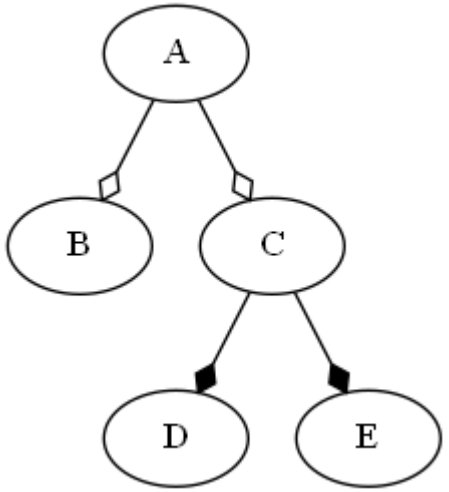


Описание рёбер (2)

```
digraph {
  edge   [arrowhead=odiamond]
  A -> B
  A -> C
  C -> D [arrowhead=diamond]
  C -> E [arrowhead=diamond]
}
```

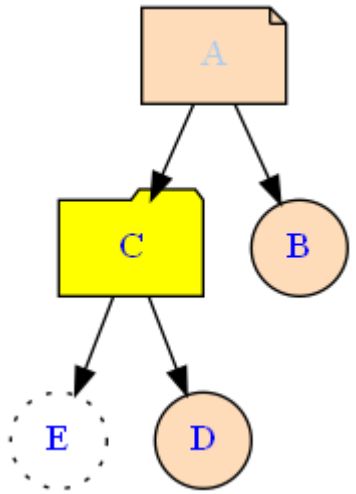



```
digraph {
  edge   [arrowhead=odiamond]
  A -> B
  A -> C
  edge   [arrowhead=diamond]
  C -> D
  C -> E
}
```



Задание вида узлов (2)

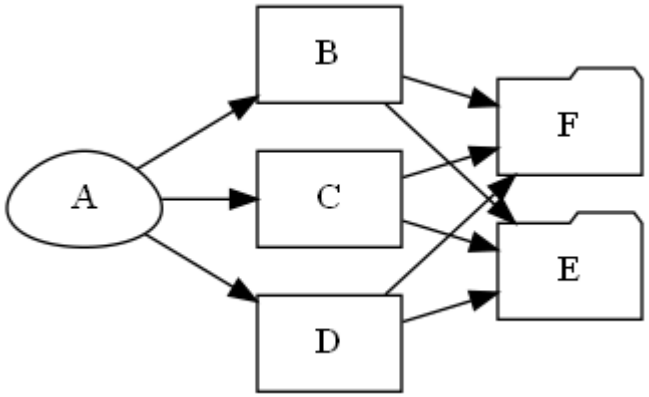
```
digraph {
  node [fontcolor=blue shape=circle style=filled fillcolor="#fedcba"]

  A [fontcolor="#abcdef" shape=note]
  C [shape=folder fillcolor="#ffff00"]
  E [style=dotted]

  A -> B
  A -> C
  C -> D
  C -> E
}
```



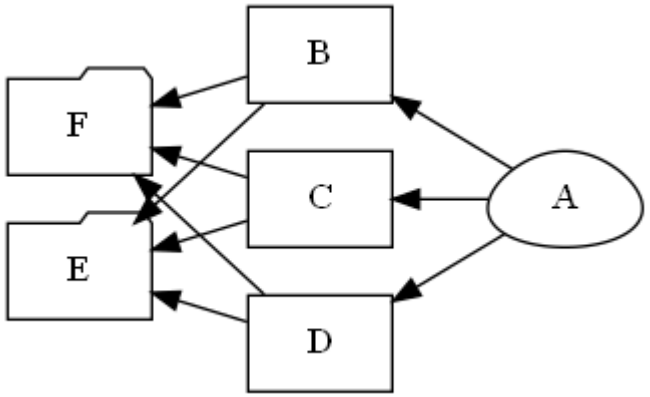
Изменение ориентации графа

```
digraph A {

  rankdir=LR;

  node [shape=egg]
  A
  node [shape=box]
  B; C; D
  node [shape=folder]
  F; E

  A -> {B, C, D} -> {F, E}

}
```

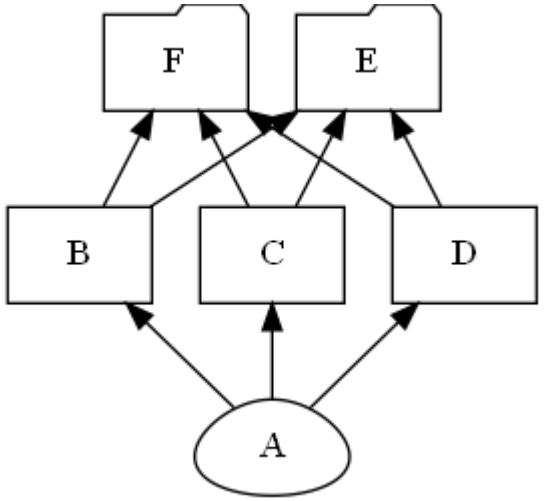
rankdir=LR;



rankdir=RL;

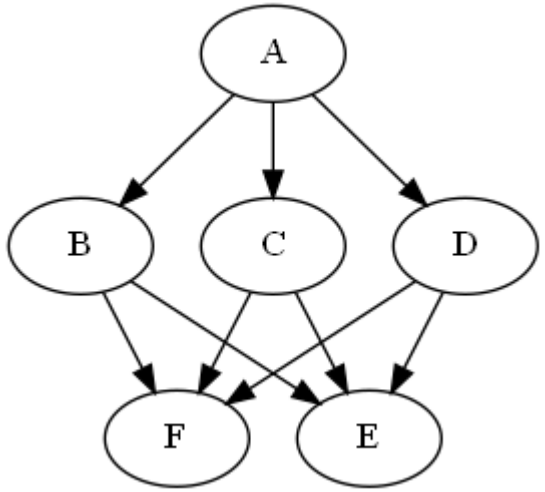


rankdir=BT;

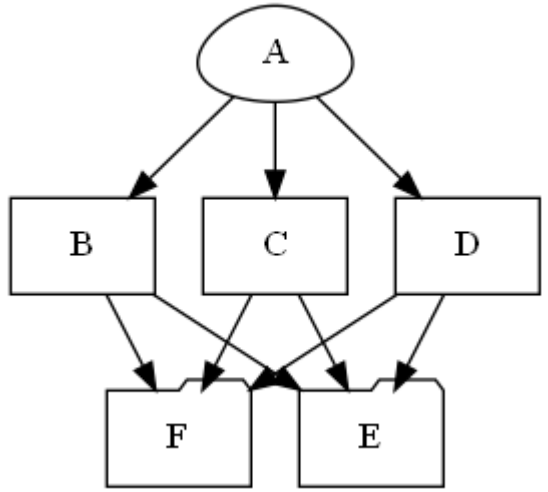


Группирование узлов (2)

```
digraph {

  A -> {B, C, D} -> F

}
```

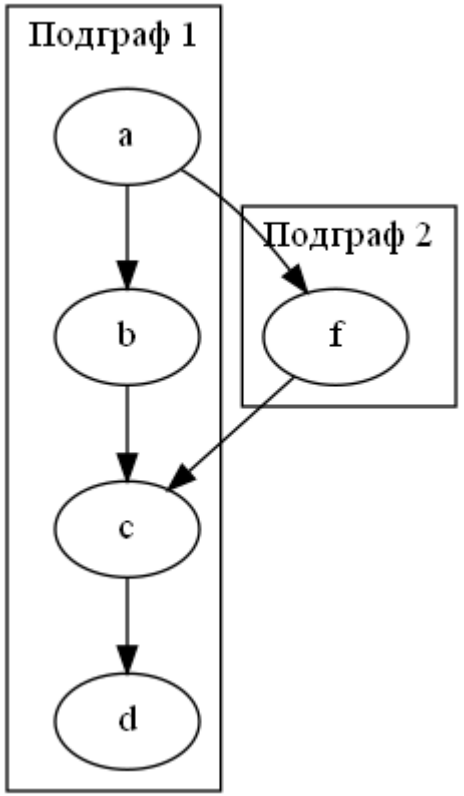



```
digraph {

  node [shape=egg]
  A
  node [shape=box]
  B; C; D
  node [shape=folder]
  F; E

  A -> {B, C, D} -> {F, E}

}
```



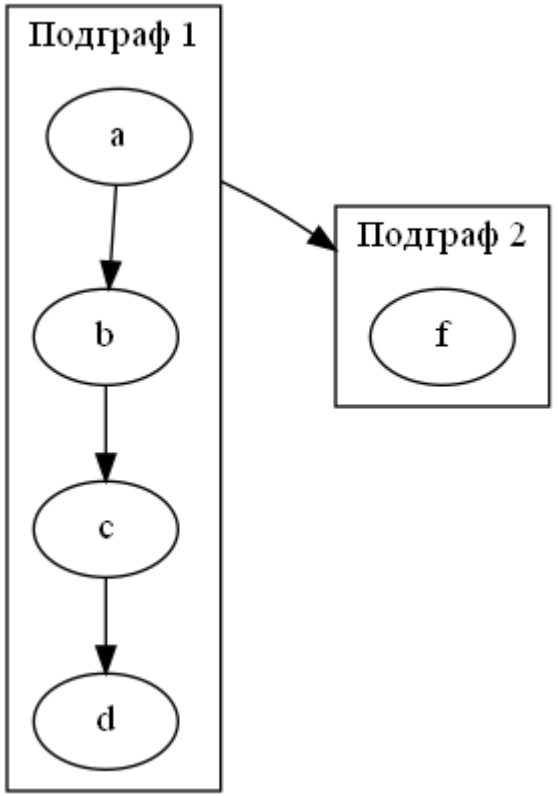
Группирование узлов подграфами (1)

```
digraph {
  subgraph cluster_0 {
   label="Подграф 1";
   a -> b;
   b -> c;
   c -> d;
  }

  subgraph cluster_1 {
   label="Подграф 2";
   a -> f;
   f -> c;
  }
}
```



Группирование узлов подграфами (2)

```
digraph {
  graph [compound=true nodesep=1 ranksep=0.5]
  subgraph cluster_0 {
    label="Подграф 1";
    a -> b;
    b -> c;
    c -> d;
  }

  subgraph cluster_1 {
    label="Подграф 2";
    f;
  }
  a -> f [ltail=cluster_0 lhead=cluster_1]
}
```



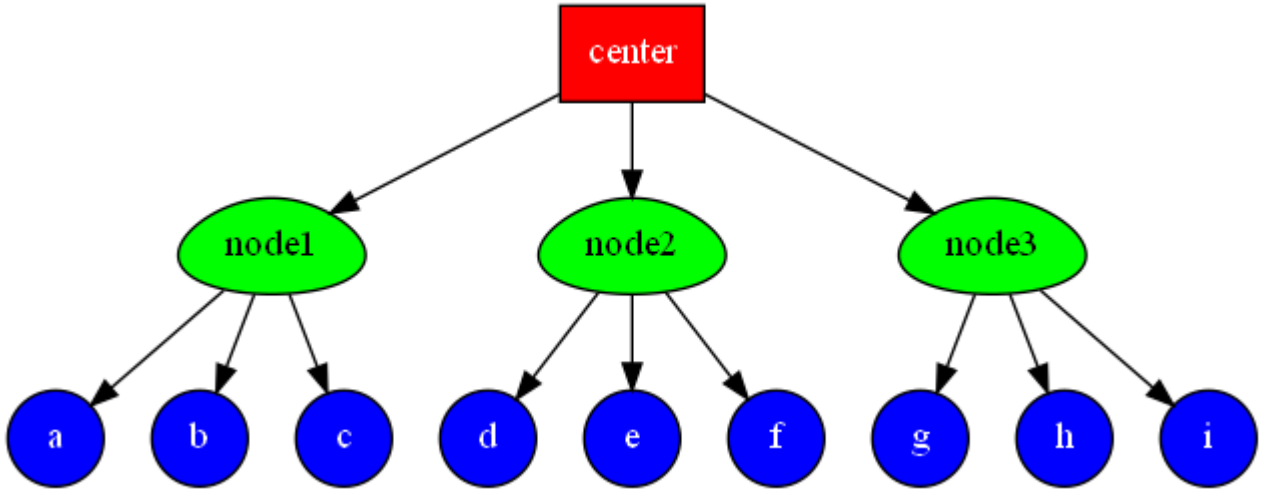
`compound=true` - разрешить связывать подграфы

`nodesep=1` - длина ребра между подграфами

`ranksep=0.5` - длина ребра внутри подграфа

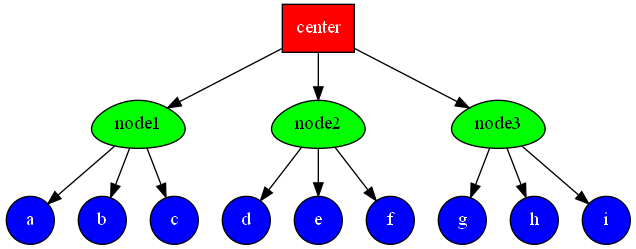
Алгоритмы визуализации графа (1)

```
digraph {
  node [style=filled fillcolor=red shape=box fontcolor=white]
  center
  node [style=filled fillcolor=green shape=egg fontcolor=black]
  node1; node2; node3
  node [style=filled fillcolor=blue shape=circle fontcolor=white]

  center -> node1
  center -> node2
  center -> node3

  node1 -> a
  node1 -> b
  node1 -> c

  node2 -> d
  node2 -> e
  node2 -> f

  node3 -> g
  node3 -> h
  node3 -> i
}
```

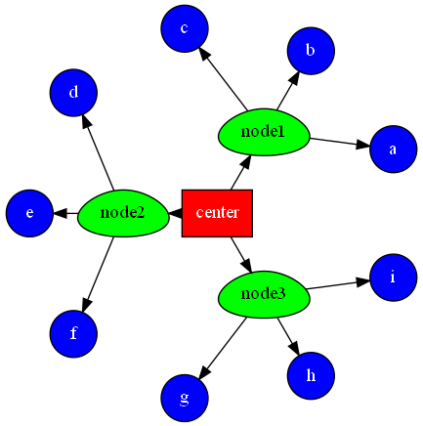


Алгоритмы визуализации графа (2)

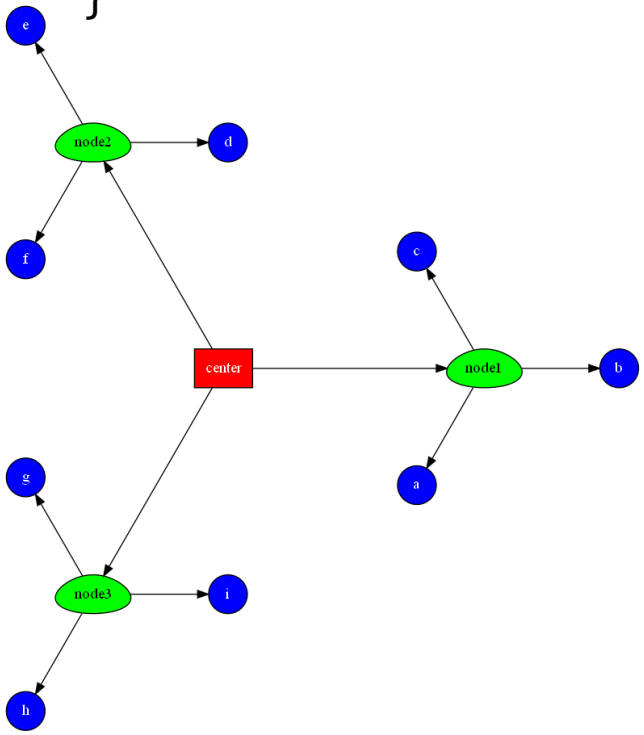
```
digraph {
  graph [layout=dot]
  ...
}
```



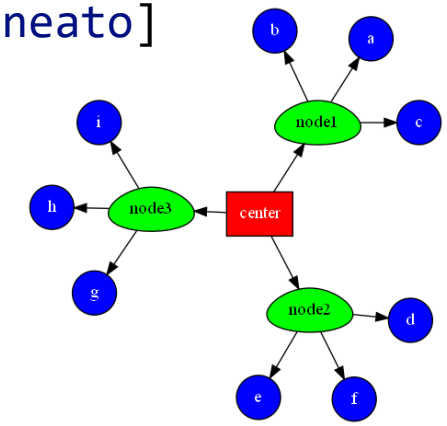

```
digraph {
  graph [layout=twopi]
  ...
}
```




```
digraph {
  graph [layout=circo]
  ...
}
```

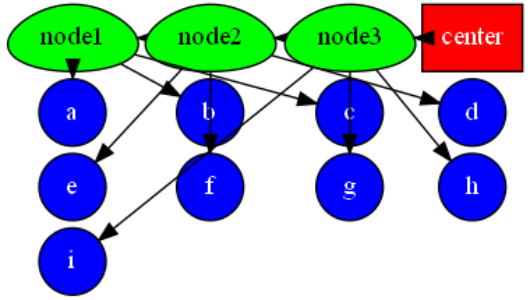



```
digraph {
  graph [layout=neato]
  ...
}
```

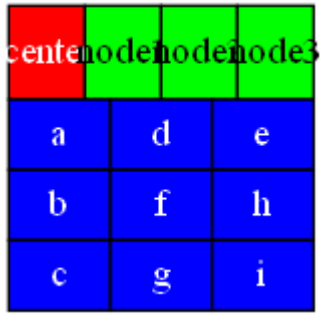


Алгоритмы визуализации графа (3)

```
digraph {
  graph [layout=osage]
  ...
}
```




```
digraph {
  graph [layout=patchwork]
  ...
}
```




```
digraph {
  graph [layout=fdp]
  ...
}
```