Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

# *ТРАДИЦИОННЫЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ПОСОБИЕ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ*

*Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-40 03 01 «Искусственный интеллект»*



Минск БГУИР 2016

УДК 004.8(076)
ББК 32.813я73
Т65

Авторы:
В. В. Голенков, Н. А. Гулякина, И. Т. Давыденко, Д. В. Шункевич

Рецензенты:
кафедра интеллектуальных систем Белорусского государственного университета (протокол №12 от 12.05.2015);

ведущий научный сотрудник государственного научного учреждения «Объединенный институт проблем информатики Национальной академии наук Беларуси», кандидат физико-математических наук, доцент Ю. В. Поттосин



**Традиционные** и интеллектуальные информационные Т65 технологии. Пособие для практических занятий : пособие / В. В. Голенков [и др.]. – Минск : БГУИР, 2016. – 64 с. : ил.
ISBN 978-985-543-210-5.

Пособие включает пять практических занятий, в которых изучаются основы теории графов, принципы обработки графовых структур в традиционных компьютерах, основы формализации знаний, представления графовых структур в интеллектуальных системах, базовые принципы обработки графовых структур в интеллектуальных системах. Занятия содержат теоретические сведения, варианты индивидуальных заданий, примеры решения задач и выполнения алгоримов.

**УДК 004.8(076)**
**ББК 32.813я73**

**ISBN 978-985-543-210-5**

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

При изучении дисциплины «Традиционные и интеллектуальные информационные технологии» первокурсники знакомятся с широко используемыми пакетами прикладных программ, современными системами контроля версий, базовыми принципами функционирования интеллектуальных систем, а также приобретают базовые навыки формализации знаний и программирования на специализированных языках обработки знаний.

Целью практических занятий является закрепление теоретического курса, приобретение навыков решения теоретико-графовых задач, активизация самостоятельной работы студентов.

Пособие состоит из пяти практических занятий, включенных в программу дисциплины на второй семестр, по следующим темам:

1. Основы теории графов.
2. Принципы обработки графовых структур в традиционных компьютерах.
3. Основы формализации знаний.
4. Представление графовых структур в интеллектуальных системах.
5. Базовые принципы обработки графовых структур в интеллектуальных системах.

# 1 ОСНОВЫ ТЕОРИИ ГРАФОВ

*Содержание занятия*: понятие графовой структуры; типология графовых структур, основные определения.

## 1.1 Теоретические сведения

Теорию графов начали разрабатывать для решения некоторых задач о геометрических конфигурациях, состоящих из точек и линий. В этих задачах несущественно, соединены ли точки конфигурации отрезками прямых или криволинейными дугами, какова длина линий и другие геометрические характеристики конфигурации. Важно лишь то, что каждая линия соединяет какие-либо две из заданных точек. Таким образом, можно дать определение графа как совокупности двух множеств V (точек) и E (линий), между элементами которых определено отношение инцидентности, причем каждый элемент $e \in E$ инцидентен ровно двум элементам $v, u \in V$. Элементы множества V называются вершинами графа G, элементы множества E – его ребрами.

Дадим более формальное определение понятия графа:

***Граф*** – алгебраическая система, модель, сигнатура которой содержит только бинарные отношения. Обычно рассматриваются графы с одним бинарным отношением в сигнатуре. Если такое единственное бинарное отношение является симметричным или его связки являются неориентированными множествами, то такой граф называется неориентированным, в других случаях – ориентированным.

***Подграфом*** графа *A* называется граф, носитель и все сигнатурные отношения которого являются соответственно подмножествами носителя и сигнатурных отношений графа *A*. Элементы носителя графа называют ***вершинами*** графа, а элементы сигнатурных отношений – ***ребрами*** в симметричных отношениях и ***дугами*** – в несимметричных.

***Смежными*** вершинами называют вершины, соединенные одним ребром либо дугой. Смежными ребрами называют ребра, имеющие общую вершину.

***Смежностным*** неориентированному графу *A* называется граф, множество вершин которого взаимно однозначно соответствует множеству ребер графа *A* таким образом, что если два ребра смежны в графе *A*, то смежны и соответствующие вершины в смежностном графе и наоборот.

Граф называется ***взвешенным*** графом, если его каждому ребру или дуге сопоставлено число.

***Полный*** граф – граф, в котором любые две вершины соединены ребром.

***Множество компонент связности*** графа – множество его связных подграфов, каждая пара которых не образует связный граф. ***Связный граф*** – граф, у которого любая пара его вершин соединена простой цепью. ***Простая цепь*** – цепь, у которой все вершины различны. ***Цепь*** – маршрут, у которого все ребра различны.

***Маршрут*** – чередующаяся последовательность вершин и ребер, начинающаяся с вершины A (начало) и заканчивающаяся вершиной B (конец). Каждые два соседних элемента этой последовательности инцидентны друг другу. Длина маршрута – количество ребер в нем. Расстояние между двумя вершинами связного графа – это длина простой цепи, соединяющей их. Эксцентриситет вершины связного графа – расстояние до самой удаленной от нее вершины.

***Диаметр*** связного графа – наибольший из его эксцентриситетов. ***Радиус*** связного графа – наименьший из его эксцентриситетов. ***Центром*** связного графа называется вершина, эксцентриситет которой равен радиусу графа. Ребро графа – ***мост***, если его удаление приводит к возрастанию числа компонент связности в нем. ***Цикл*** – цепь, которая начинается и заканчивается одной вершиной. ***Простой цикл*** – цикл, являющийся простой цепью.

***Эйлеров граф*** – граф, для которого существует цикл, содержащий все его ребра или дуги. Связный неориентированный граф, для которого отсутствуют циклы, называется ***деревом***. Граф называется ***двудольным*** тогда и только тогда, когда все его вершины могут быть разбиты на два множества, причем любая пара вершин одного множества не является смежной в этом графе. Двудольный неориентированный граф называется ***дольнорегулярным*** тогда и только тогда, когда для каждой доли существует число, для любой вершины этой доли совпадающее с числом всех ребер графа, инцидентных этой вершине.

### 1.2 Варианты индивидуальных заданий

1. Построить неориентированный связный граф с указанным числом вершин, каждая из которых инцидентна указанному одинаковому для всех вершин числу ребер (если такой граф существует).
2. Построить ориентированный связный граф с указанным числом вершин, такой, что из каждой вершины выходит и в каждую вершину входит указанное одинаковое для всех вершин количество дуг (количество входящих = количество выходящих = N).
3. Достроить исходный неориентированный граф до графа с указанным числом вершин, каждая из которых инцидентна указанному одинаковому для всех вершин числу ребер (если такой граф существует).

4. Достроить исходный неориентированный граф до связного дольнорегулярного двудольного графа с указанным числом вершин, для каждой доли которого указаны числа, задающие вершине число инцидентных ребер.
5. Для исходного неориентированного графа выделить компоненты связности, для каждой компоненты связности найти величину диаметра и радиуса, определить вершины, являющиеся ее центрами.
6. В исходном неориентированном графе найти все ребра, являющиеся мостами этого графа.
7. В исходном графе найти все вершины, являющиеся точками сочленения этого графа.
8. Определить, является ли исходный неориентированный граф деревом. Если да, то найти центр (или два центра) этого дерева и вывести поуровневое (относительно центра) описание этого дерева.
9. Определить, является ли исходный ориентированный граф ориентированным деревом. Если да, то зафиксировать его начальную вершину и вывести поуровневое описание этого дерева.
10. Определить, является ли исходный неориентированный граф эйлеровым графом, и если да, то построить для него один из эйлеровых циклов.
11. Определить, является ли исходный ориентированный граф эйлеровым графом, и если да, то построить для него один из ориентированных эйлеровых циклов.
12. В исходном неориентированном графе выделить все максимальные полные подграфы.
13. Разработать алгоритм нахождения кратчайшего расстояния между двумя вершинами (взвешенного) ориентированного графа.
14. Найти все минимальные простые циклы в заданном неориентированном графе.
15. Раскрасить граф минимальным числом цветов.
16. Дан неориентированный граф, реализовать процедуру, которая строит смежностный граф для заданного.

# 2 ПРИНЦИПЫ ОБРАБОТКИ ГРАФОВЫХ СТРУКТУР В ТРАДИЦИОННЫХ КОМПЬЮТЕРАХ

*Содержание занятия*: представление графовых структур в памяти традиционных компьютеров.

## 2.1 Теоретические сведения

Рассмотрим основные подходы к представлению графовых структур в памяти традиционных компьютеров:

- матрица инцидентности;
- список ребер (список инцидентности);
- матрица смежности;
- список смежности.

**Матрица инцидентности и список ребер**. Задать граф – значит описать множества его вершин и ребер, а также отношение инцидентности. Когда граф G – конечный, для описания его вершин и ребер достаточно их занумеровать. Пусть $v_1$, $v_2$, ..., $v_n$ – вершины графа G; $e_1$, $e_2$, ..., $e_т$ – его ребра. Отношение инцидентности можно определить матрицей $\|\varepsilon_{ij}\|$, имеющей *m* строк и *n* столбцов. Столбцы соответствуют вершинам графа, строки – ребрам. Если ребро $e_i$ инцидентно вершине $v_j$, то $\varepsilon_{ij} = 1$, в противном случае $\varepsilon_{ij} = 0$. Это так называемая *матрица инцидентности* неориентированного графа G, которая является одним из способов его определения. Для графа на рисунке 1 она дана в таблице 1.

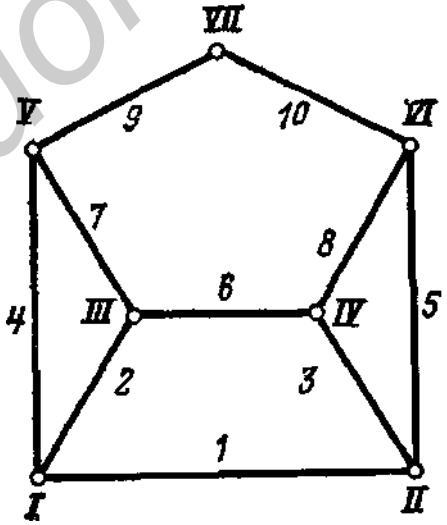


Рисунок 1 – Пример неориентированного графа G

Таблица 1 – Матрица инцидентности графа G

| Ребра | Вершины | | | | | | |
|---|---|---|---|---|---|---|---|
| | I | II | III | IV | V | VI | VII |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

В матрице инцидентности $\|\varepsilon_{ij}\|$ ориентированного графа G, если вершина $v_j$ – начало дуги $a_i$ то $\varepsilon_{ij} = -1$; если $v_j$ – конец $a_i$, то $\varepsilon_{ij} = 1$; если $a_i$ – петля, а $v_j$ – инцидентная ей вершина, то $\varepsilon_{ij} = \alpha$, где $\alpha$ – любое число, отличное от 1, 0 и –1; в остальных случаях $\varepsilon_{ij} = 0$ (пример – таблица 2 для графа на рисунке 2).

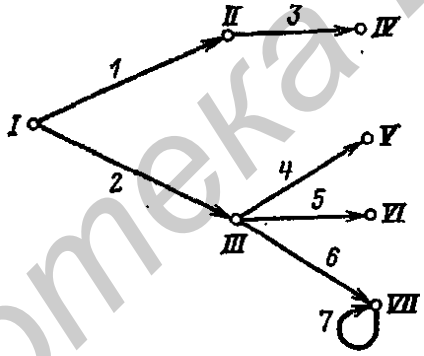


Рисунок 2 – Пример ориентированного графа G

Таблица 2 – Матрица инцидентности графа G

| Ребра | Вершины | | | | | | |
|---|---|---|---|---|---|---|---|
| | I | II | III | IV | V | VI | VII |
| 1 | –1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | –1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | –1 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | –1 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | –1 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | –1 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |

В каждой строке матрицы инцидентности для неориентированного или ориентированного графа только два элемента отличны от 0 (или один, если ребро является петлей). Поэтому такой способ задания графа оказывается недостаточно экономным. Отношение инцидентности можно еще задать ***списком ребер*** графа. Каждая строка этого списка соответствует ребру, в ней записаны номера вершин, инцидентных ему. Для неориентированного графа порядок этих вершин в строке произволен, для ориентированного – первым стоит номер или наименование начала ребра, а вторым – конца ребра. В таблицах 3 и 4 приводятся списки ребер для графов, изображенных на рисунках 1 и 2 соответственно.

По списку ребер графа легко построить его матрицу инцидентности. Действительно, каждая строка этого списка соответствует строке матрицы с тем же номером. Для неориентированного графа в строке списка указаны номера элементов строки матрицы инцидентности, равные 1, и для ориентированного графа в этой строке первым стоит номер элемента строки матрицы, равного 1, а вторым – номер элемента, равного +1.

Таблица 3 – Список ребер неориентированного графа G

| Ребра | Вершины |
|---|---|
| 1 | I, II |
| 2 | I, III |
| 3 | II, IV |
| 4 | I, V |
| 5 | II, VI |
| 6 | III, IV |
| 7 | III, V |
| 8 | IV, VI |
| 9 | V, VII |
| 10 | VI, VII |

Таблица 4 – Список ребер ориентированного графа G

| Ребра | Вершины |
|---|---|
| 1 | I, II |
| 2 | I, III |
| 3 | II, IV |
| 4 | III, V |
| 5 | III, VI |
| 6 | III, VII |
| 7 | VII, VII |

**Матрица смежности графа.** Это квадратная матрица $\|\delta_{ij}\|$, столбцам и строкам которой соответствуют вершины графа. Для неориентированного графа $\delta_{ij}$ равно количеству ребер, инцидентных i-й и j-й вершинам, для ориентированного графа этот элемент матрицы смежности равен количеству ребер с началом в i-й вершине и концом в j-й. Таким образом, матрица смежности неориентированного графа симметрична ($\delta_{ij}=\delta_{ji}$), а ориентированного – необязательно. Если она все же симметрична, то для каждого ребра ориентированного графа имеется ребро, соединяющее те же вершины, но идущее в противоположном направлении. Ориентированный граф с симметричной матрицей смежности канонически соответствует неориентированному графу, имеющему ту же матрицу смежности.

Матрицы смежности рассмотренных ранее графов (см. рисунки 1, 2) приводятся в таблицах 5 и 6.

Таблица 5 – Матрица смежности неориентированного графа G

| Вершины | I | II | III | IV | V | VI | VII |
|---|---|---|---|---|---|---|---|
| I | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| II | 1 | 0 | 0 | 1 | 0 | I | 0 |
| III | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| IV | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| V | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| VI | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| VII | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

Таблица 6 – Матрица смежности ориентированного графа G

| Вершины | I | II | III | IV | V | VI | VII |
|---|---|---|---|---|---|---|---|
| I | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| II | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| III | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| IV | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VI | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VII | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Матрица смежности также определяет соответствующий неориентированный или ориентированный граф. Число его вершин равно размерности матрицы n; i–й и j–й вершинам графа инцидентны $\delta_{ij}$ ребер. Для неориентированного графа $\delta_{ij}=\delta_{ji}$ и все его ребра определяются верхним

правым треугольником матрицы, расположенным над диагональю, включая последнюю. Количество их равно сумме $\delta_{ij}$ по этому треугольнику.

Ребра ориентированного графа определяются всеми элементами $\delta_{ij}$ матрицы смежности. В обоих случаях по матрице смежности легко строится, например, список ребер, определяющий граф. Элементу матрицы смежности, расположенному в i-й строке и j-м столбце, соответствуют $\delta_{ij}$ строк списка ребер (при $\delta_{ij}=0$ – ни одной строки), в каждой из которых записаны номера i, j. Для неориентированного графа эти строки соответствуют только элементам описанного ранее верхнего правого треугольника матрицы смежности, т. е. элементам $\delta_{ij}$ с $j \geq i$, а для ориентированного графа нужно рассматривать все элементы $\delta_{ij}$.

**Список смежности.** Данный способ представления больше подходит для разреженных графов, т. е. графов, у которых количество ребер гораздо меньше, чем количество вершин в квадрате ($|E| << |V|^2$).

В данном представлении используется некоторый массив A, содержащий |V| списков. В каждом списке A[v] содержатся все вершины u, так что между v и u есть ребро. Память, требуемая для представления, равна O(|E| + |V|), что является лучшим показателем, чем матрица смежности для разреженных графов.

Главный недостаток этого способа представления в том, что нет быстрого способа проверить, существует ли ребро (u, v).

### 2.2 Варианты индивидуальных заданий

Необходимо программно реализовать решение одной из предложенных теоретико-графовых задач, используя один из рассмотренных вариантов представления графовых структур.

Список теоретико-графовых задач:

I. Определить вид графа:

1) дерево;
2) ациклический граф;
3) полуэйлеров/эйлеров граф;
4) гамильтонов граф;
5) связный граф;
6) сильно-связный граф;
7) двусвязный граф;
8) двудольный неориентированный граф;
9) регулярный, реберно-регулярный неориентированный граф;
10) симметричный, антисимметричный, частично симметричный орграф;

11) транзитивный, антитранзитивный, частично транзитивный орграф;
12) рефлексивный, антирефлексивный, частично рефлексивный орграф;
13) функциональный, контрафункциональный орграф;
14) односторонне связный орграф;
15) кактус;
16) турнир, транзитивный турнир;
17) граф паппа;
18) планарный граф;
19) транзитируемый граф.

II. Определить числовую характеристику графа:

1) радиус;
2) диаметр;
3) средний диаметр;
4) полустепени захода/исхода и средние полустепени всех вершин в орграфе;
5) минимальную/среднюю/максимальную степень ребра в неориентированном графе;
6) число вершинной связности;
7) число реберной связности;
8) среднее и максимальное расстояние между центральными вершинами неориентированного графа;
9) число хорд неориентированного графа;
10) минимальное и среднее расстояние между периферийными вершинами неориентированного графа;
11) окружение орграфа;
12) обхват орграфа;
13) число компонент связности неориентированного графа;
14) число хадвигера для неориентированного графа;
15) определить толщину неориентированного графа;
16) индекс компонент относительно простой цепи в неориентированном графе;

III. Операции над графами. Найти:

1) декартово произведение двух неориентированных графов;
2) декартову сумму двух неориентированных графов;
3) прямое (тензорное) произведение двух неориентированных графов;
4) сильное произведение двух неориентированных графов;
5) композицию двух неориентированных графов;
6) модульное произведение двух неориентированных графов;

7) большое модульное произведение двух неориентированных графов;
8) объединение множества неориентированных графов;
9) пересечение множества неориентированных графов;
10) дополнение и фактор-дополнение неориентированного графа;
11) граф инциденций неориентированного графа;
12) реберный граф для неориентированного графа;
13) граф смежностей для неориентированного графа;
14) тотальный граф для неориентированного графа;
15) граф замыкания неориентированного графа;
16) граф конденсации для орграфа;
17) граф каркасов для неориентированного графа.

IV. Поиск в графе. Найти:

1) эксцентриситет каждой вершины в неориентированном графе;
2) эйлеров цикл в графе;
3) гамильтонов цикл;
4) компоненты связности в неориентированном графе;
5) сильные компоненты связности в орграфе;
6) вершины с указанной степенью;
7) минимальный остов в неориентированном графе;
8) доли неориентированного графа;
9) тупики/антитупики в ориентированном графе;
10) максимальный простой разрез;
11) минимальный простой разрез;
12) множество ребер, удаление которых приводит к увеличению числа компонент связности ориентированного графа;
13) точки сочленения неориентированного графа;
14) мосты в неориентированном графе;
15) множество вершин, удаление которых приводит к увеличению числа компонент связности ориентированного графа;
16) циклы указанной длины;
17) максимальный путь между заданными вершинами;
18) критические пути во взвешенном неориентированном графе;
19) множество вершин, удаление которых приводит к увеличению числа компонент связности неориентированного графа;
20) простые цепи указанной длины;
21) вершины с указанной полустепенью;
22) все доминирующие вершины;
23) центры графа;

24) все периферийные вершины;
25) множество ребер, удаление которых приводит к увеличению числа компонент связности неориентированного графа;
26) звезды с заданным числом листьев;
27) дерево кратчайших путей;
28) n-фактор для указанного графа;
29) подграфы в неориентированном графе, изоморфные графу-образцу;
30) множество различных суграфов неориентированного графа;
31) сформировать множество различных подграфов неориентированного графа;

V. Приведение графа к указанному виду. Найти:

1) минимальное множество вершин неориентированного графа, удаление которых позволяет сделать его деревом;
2) минимальное множество ребер неориентированного графа, удаление которых позволяет сделать его деревом;
3) минимальное множество вершин неориентированного графа, удаление которых позволяет сделать его планарным;
4) минимальное множество ребер графа, удаление которых позволяет сделать его планарным.

# 3 ОСНОВЫ ФОРМАЛИЗАЦИИ ЗНАНИЙ

*Содержание занятия:* представление простейших видов знаний в памяти интеллектуальных систем.

## 3.1 Теоретические сведения

В рамках данного занятия необходимо формализовать на языке семантических сетей фактографическое высказывание согласно варианту индивидуального задания. Более подробную информацию о принципах формализации на языке SC можно получить на лекциях и в литературе.

***Пример*** *(рисунок 3)*

Дана равнобедренная трапеция, основания которой a, b равны 9 и 21 см соответственно, высота h равна 8 см. Вокруг трапеции описана окружность радиусом x см.

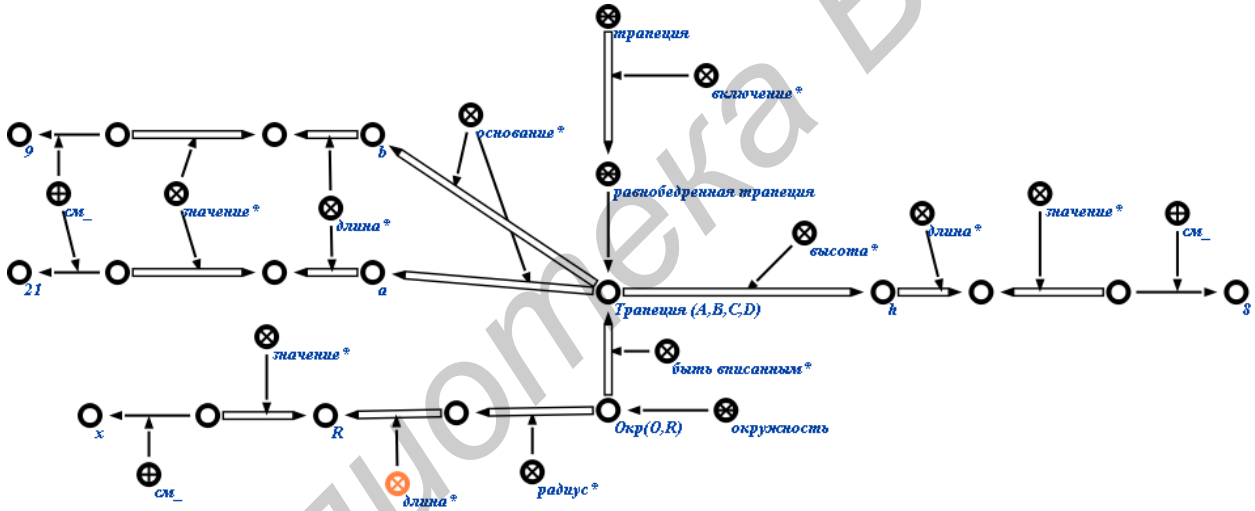


Рисунок 3 – Пример фактографического высказывания

## 3.2 Варианты индивидуальных заданий

1. Медиана AM треугольника ABC равна половине стороны BC. Угол между AM и высотой AH равен 40°.
2. На одной стороне угла с вершиной O взята точка A, а на другой – точки B и C , причем точка B лежит между O и C . Проведена окружность с центром O1, вписанная в треугольник OAB , и окружность с центром O2, касающаяся стороны AC и продолжений сторон OA и OC треугольника AOC.
3. Диагонали выпуклого четырехугольника ABCD пересекаются в точке E, AB = AD, CA – биссектриса угла C, угол BAD = 140°, угол BEA = 110°.

4. Дана окружность с центром O. На продолжении хорды AB за точку B отложен отрезок BC, равный радиусу. Через точки C и O проведена секущая CD (D – точка пересечения с окружностью, лежащая вне отрезка CO).
5. В треугольнике ABC известны углы: $A = 45^o$, $B = 15^o$. На продолжении стороны AC за точку C взята точка M, причем CM = 2AC.
6. В треугольнике ABC угол C – тупой, биссектриса BE угла B делит сторону AC на отрезки AE = 3, EC = 2. Известно, что точка K, лежащая на продолжении стороны BC за вершину C, является центром окружности, проходящей через точки C, E и точку пересечения биссектрисы угла B с биссектрисой угла ACK.
7. Треугольники ABC и ADC имеют общую сторону AC, стороны AD и BC пересекаются в точке M. Углы B и D равны по $40^o$. Расстояние между вершинами D и B равно стороне AB, угол AMC = $70^o$.
8. Биссектрисы треугольника ABC пересекаются в точке O. Через точку O проходят две прямые, которые параллельны прямым AB и AC и пересекаются с BC в точках D и E. Периметр треугольника OED равен отрезку BC.
9. В треугольнике ABC известно, что AB = BC, AC = 10. Из точки D, являющейся серединой AB, проведен перпендикуляр DE к стороне AB до пересечения со стороной BC в точке E. Периметр треугольника ABC равен 40.
10. В равнобедренном треугольнике с основанием AC проведена биссектриса угла C, которая пересекает боковую сторону AB в точке D. Точка E лежит на основании AC так, что DE перепендикулярна DC.

# 4 ПРЕДСТАВЛЕНИЕ ГРАФОВЫХ СТРУКТУР В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

*Содержание занятия:* представление графовых структур и отношений на них в виде семантических сетей.

## 4.1 Теоретические сведения

В рамках данного занятия необходимо выделить абсолютные и относительные понятия для своей теоретико-графовой задачи, вариант которой был выбран в занятии 2. Список понятий должен включать не только непосредственно необходимые понятия для решения вашей задачи, но и те, на основе которых они определены. Таким образом, выделяется целая иерархия понятий, для каждого из них разрабатывается способ представления его экземпляров в SC-коде.

Для данного занятия необходимо подготовить отчет в электронном варианте о проделанной работе, который должен содержать следующее:

1) перечень выделенных понятий со следующей информацией:
   a) название понятия;
   b) абсолютное или относительное;
   c) определение на естественном языке;
   d) пример представления экземпляра понятия в SC-коде на SCg;
2) пять примеров на SCg исходных и выходных данных для вашей программы.

Для рисования SCg-текстов необходимо использовать редактор KBE (Knowledge Base Editor). Установочный файл и zip-архив собранной версии под Windows находятся на кафедральном информационном сервере, где также лежит пример отчета по данному этапу работы.

### 4.1.1 Формализация понятий «неориентированный» и «ориентированный» граф

Неориентированный граф G (рисунок 4):

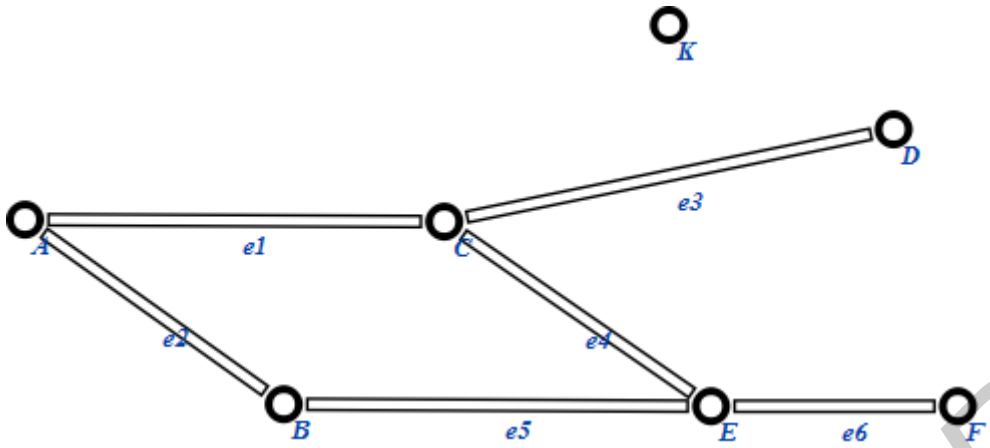


Рисунок 4 – Неориентированный граф G (представление не на SCg)

Граф G, расписанный классическим способом на языке теории множеств, будет выглядеть так:

$G = (Vg, Eg)$;

$Vg = \{A, B, C, E, D, F, K\}$;

$Eg = \{\{A,B\}, \{A,C\}, \{C,E\}, \{C,D\}, \{B,E\}, \{E, F\}\}$. (1)

### 4.1.2 Перевод записи в SC-код

Для представления неориентированного графа в SC-код вводится абсолютное понятие *неориентированный граф* (в дальнейшем будем использовать такое форматирование для идентификации sc-элементов в тексте). Теперь можно преобразовать запись на языке множеств, задающую граф G, в SCg-конструкцию, которая изображена на рисунке 5.

Приведенный выше способ задания графа на языке классической теории множеств является распространенным, но с использованием SC-кода можно найти другую форму, так как есть возможность задавать ролевые отношения. Поэтому вводятся два относительных понятия (ролевых отношения): *вершина'* и *ребро'*. Тогда можно сформулировать, что неориентированный граф задается множеством объектов, в которых объект с ролью *вершина'* является вершиной графа, а объект с ролью *ребро'* – ребром графа. На языке теории множеств, расширенном возможностью задавать атрибуты (роли) у элементов кортежа, граф G будет задаваться выражением (2).

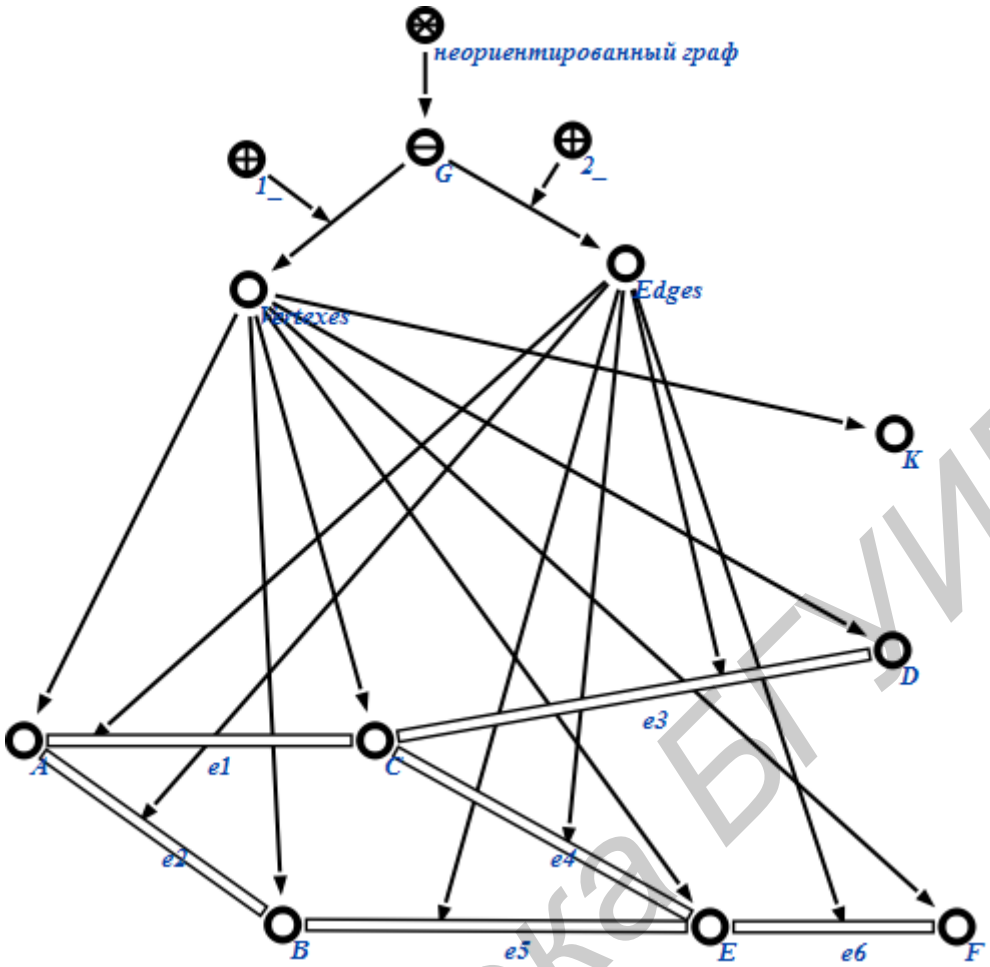


Рисунок 5 – Классический способ задания неориентированного графа G на SCg

$$G = \langle \text{вершина}' : A, \text{вершина}' : B, \text{вершина}' : C, \text{вершина}' : E, \text{вершина}' : D, \text{вершина}' : F, \text{вершина}' : K, \text{ребро}' : \{A,B\}, \text{ребро}' : \{A,C\}, \text{ребро}' : \{C,E\}, \text{ребро}' : \{C,D\}, \text{ребро}' : \{B,E\}, \text{ребро}' : \{E, F\}\rangle. \quad (2)$$

Перевод выражения (2) в SCg показан на рисунке 6.

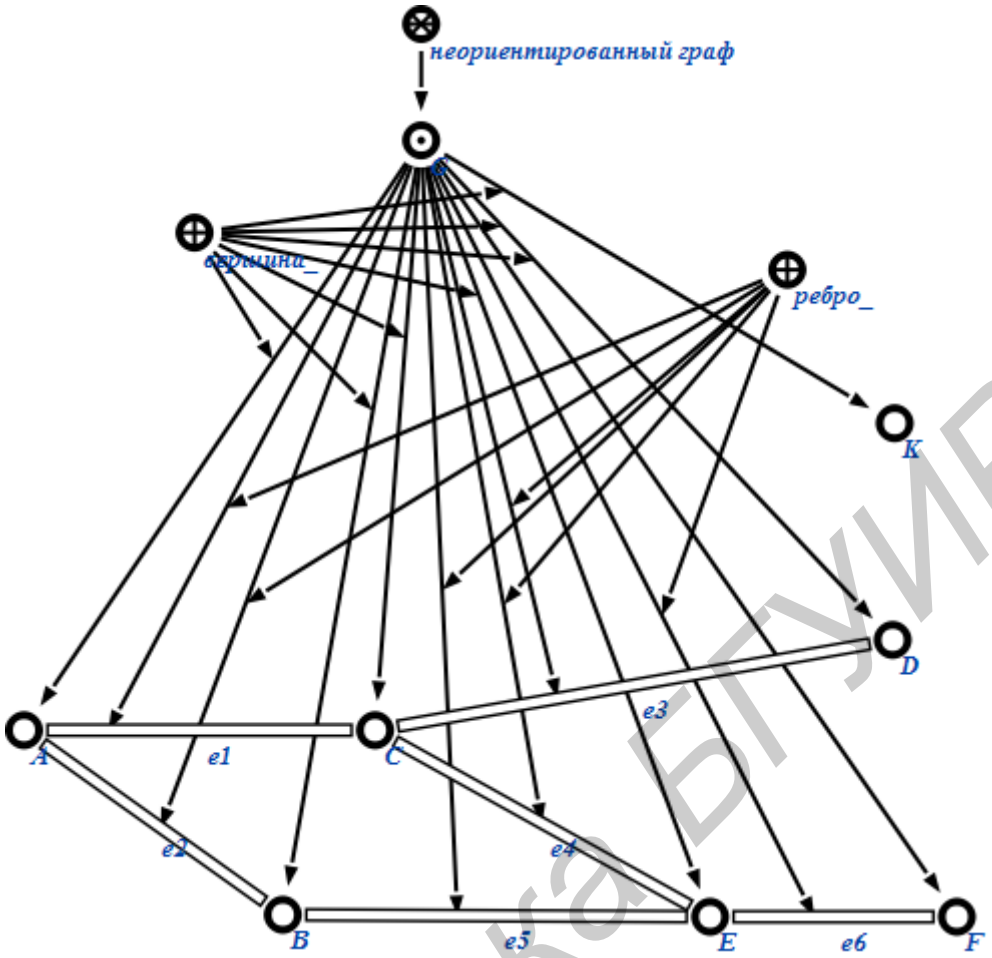


Рисунок 6 – Основной способ задания графа G на SCg

Тип SCg-узла используется для обозначения sc-структуры, т. е. множества объектов, которые могут выступать как целое. Объекты, которые входят в sc-структуру, при совместном рассмотрении порождают некоторое новое качество. Рассмотрим это через сравнение множества *неориентированный граф* с множеством G, которое является конкретным неориентированным графом.

Множество *неориентированный граф* является sc-понятием (абсолютным), т. е. sc-множеством, все элементы которого обладают некоторым заданным свойством. В случае sc-понятия *неориентированный граф* его элементы должны обозначать неориентированные графы.

Предположим, в множестве *неориентированный граф* есть 5 элементов (неориентированных графов). Добавим в это множество еще 5 неориентированных графов. При добавлении элементов мощность множества изменилась, но качественно множество неориентированного граф осталось таким же. Этот факт лаконично можно выразить следующим образом: количество элементов множества, которое является sc-понятием, никогда не переходит в качество.

Множество G – это sc-множество, составленное из sc-элементов, которые вместе обладают некоторой целостностью, имеющей важные свойства (они образуют неориентированный граф только в том случае, если рассмотрены как единое целое). Если добавить в множество G новый элемент (вершину или ребро), то получится уже другой граф. Таким образом, можно заключить, что количество элементов для множества G переходит в качество. Множества, для которых выполняется это свойство, являются sc-структурами.

Стоит отметить, что связки также являются структурами. Однако неориентированный граф G, не только обозначает сам факт существования связи между объектами, но и включает связки между своими собственными элементами. Такие структуры, как граф G, не являются связками, назовем их одноуровневыми реляционными структурами. Способ представления графов в виде одноуровневых реляционных структур используется в проекте базы знаний по теории графов технологии OSTIS (OSTIS GT), поэтому ниже будет рассмотрен только этот способ, и в рамках практического занятия должен использоваться именно этот метод.

Представим ориентированный граф на SCg. Для этого преобразуем неориентированный граф G в ориентированный граф $G_d$ (рисунок 7).

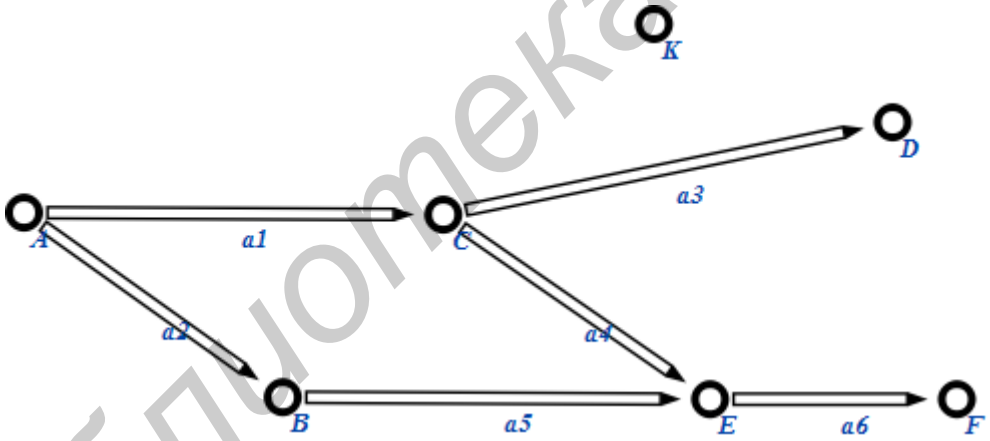


Рисунок 7 – Ориентированный граф $G_d$ (представление не на SCg)

Для представления $G_d$ на SCg вводится абсолютное понятие *ориентированный граф* и относительное понятие (ролевое отношение) *дуга'*. С использованием этих понятий граф $G_d$ на SCg можно представить так, как показано на рисунке 8.

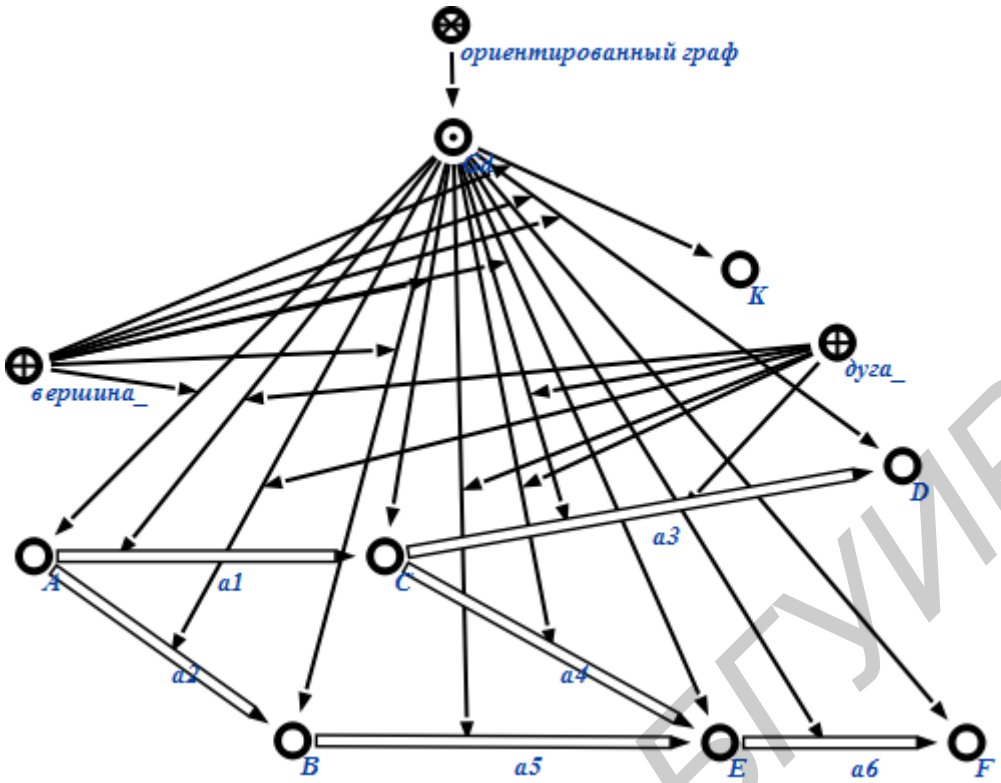


Рисунок 8 – Ориентированный граф $G_d$ на SCg

Если сравнить запись графа G (см. рисунок 6) на SCg и графа $G_d$ (см. рисунок 8), то видим, что разница только в использовании:

1) узла *ориентированный граф* вместо *неориентированный граф*;
2) узла *дуга'* вместо *ребро'*;
3) ориентированных дуг общего вида вместо неориентированных.

Можно заметить, что графы на SCg с использованием ролевых отношений *вершина'* и *ребро'* получаются очень громоздкими, поэтому в дальнейших объяснениях для представления неориентированных и ориентированных графов будет использоваться сокращенная запись. С использованием сокращенной записи граф G, заданный на рисунке 6, мы будем задавать так, как показано на рисунке 9. Данная запись представлена исключительно для облегчения восприятия человеком, для машины является важным наличие опущенных атрибутов.

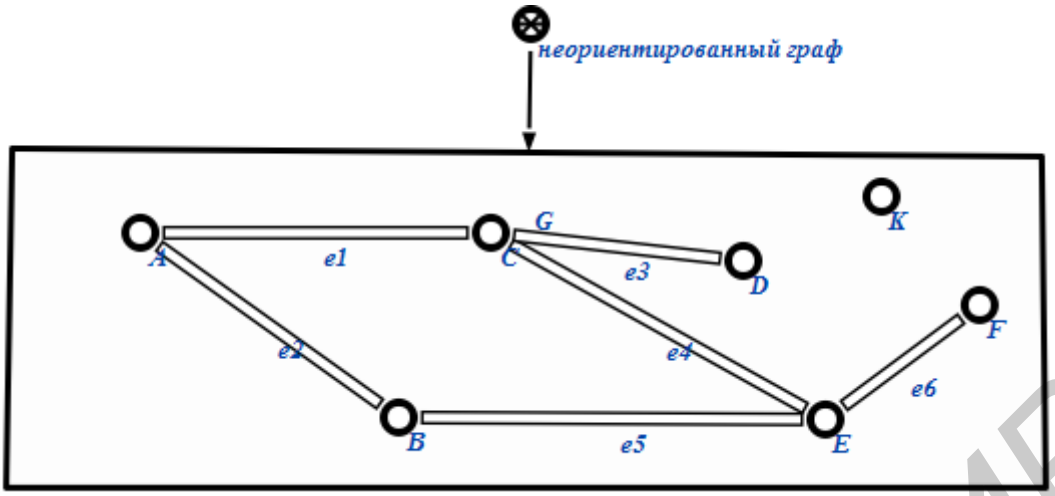


Рисунок 9 – Сокращенная форма задания графа G на SCg

Таким образом, мы рассмотрели способ формализации с использованием семантических сетей неориентированного и ориентированного графов. Дальше будет рассмотрен способ формализации минимального пути – второго понятия, необходимого для решения задачи.

### 4.1.3 Формализация понятия «минимальный путь»

Рассмотрим более подробно понятие «маршрут».

Маршрутом в графе G называется чередующаяся последовательность вершин и ребер $v_0$, $x_1$, $v_1$, ..., $v_{n-1}$, $x_n$, $v_n$. Эта последовательность начинается и кончается вершиной, и каждое ребро последовательности инцидентно двум вершинам, одна из которых непосредственно предшествует ему, а другая непосредственно следует за ним. Указанный маршрут соединяет вершины $v_0$ и $v_n$, и его можно обозначить $v_0$, $v_1$, ..., $v_n$ (наличие ребер подразумевается). Маршрут называется цепью, если все его ребра различны, и простой цепью (путем), если все вершины (а следовательно, и ребра) различны.

Таким образом, путь – это маршрут, поэтому будет рассматриваться именно это более общее понятие.

Очевидно, маршрут – это относительное понятие, так как конкретный маршрут существует в связи с конкретным графом. Поэтому для представления маршрутов вводится бинарное ориентированное отношение *маршрут**. Первым компонентом связки этого отношения будет знак графа, а вторым – знак структуры маршрута. Пример связки приведен на рисунке 10.

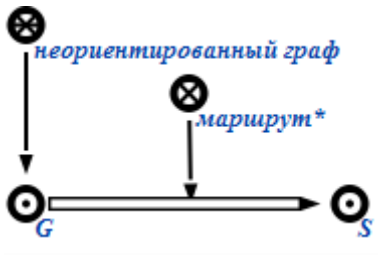


Рисунок 10 – Пример связки отношения *маршрут**

Теперь нам необходимо выяснить, из чего же состоит структура S на рисунке 10. Для этого рассмотрим в графе G (см. рисунок 9) маршрут R между вершинами A и F, который задается последовательностью A, $e_2$, B, $e_5$, E, $e_6$, F (это один из минимальных путей между A и F). Если маршрут состоит из вершин и ребер, то его можно представить как подграф графа, на котором задается маршрут.

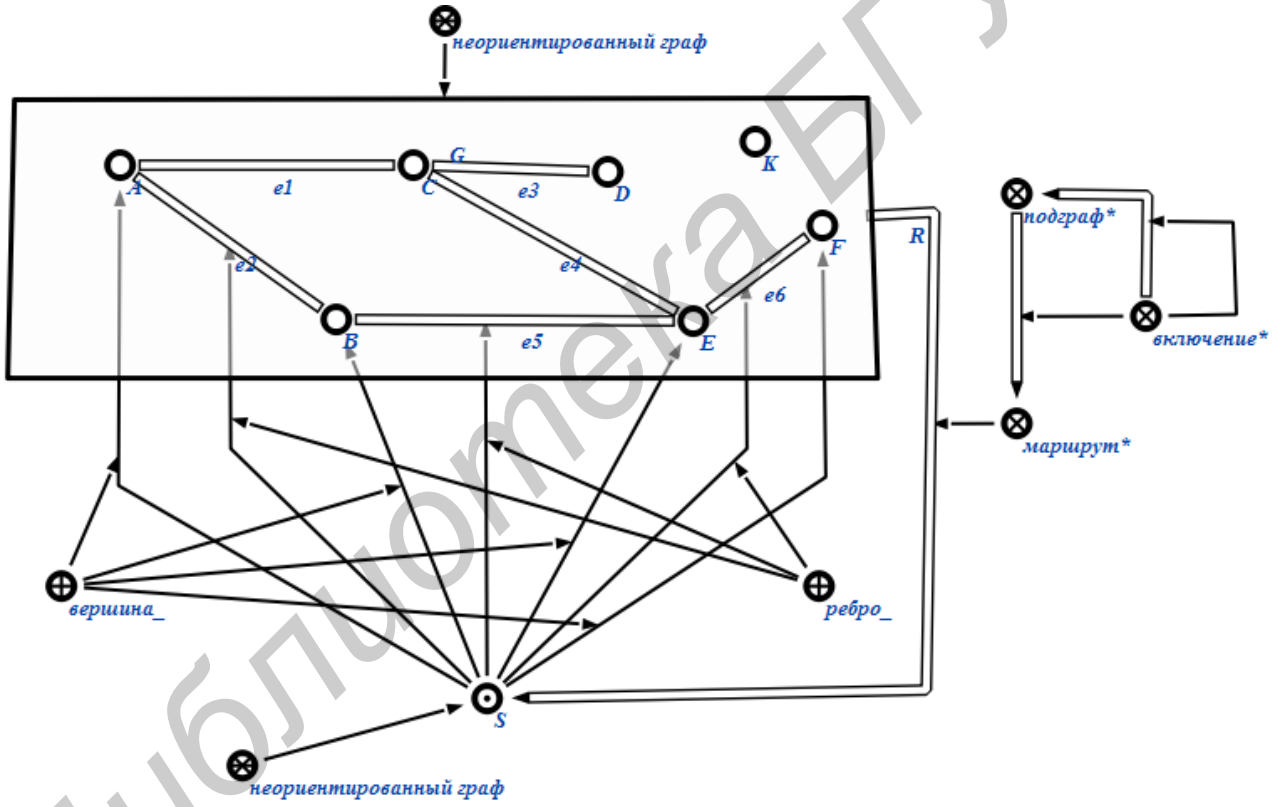


Рисунок 11 – Представление маршрута как подграфа графа G

На рисунке 11 появилось бинарное ориентированное отношение *подграф*,* суть которого есть связывание одного графа с другим, который является подграфом первого. Также из рисунка 11 можно заметить, что отношение *включение** содержит отношение *подграф*,* т. е. *подграф** – аналог *включения**, но только для графов. А отношение *маршрут** является подмножеством отношения *подграф**. Таким образом, граф структуры S является подграфом исходного графа G.

Теперь можно попробовать нарисовать SCg-конструкцию, которая задаст маршрут A, $e_2$, B, $e_5$, E, $e_4$, C, $e_1$, A, $e_2$, B.

На рисунке 11 представлен не путь, потому что вершины и ребра повторяются, но это допустимый маршрут. На рисунке 12 повторяющиеся вершины и ребра имеют более светлый оттенок. Поэтому переходим к поиску способа задать структуру второго компонента связки отношения *маршрут** (структура S на рисунке 10).

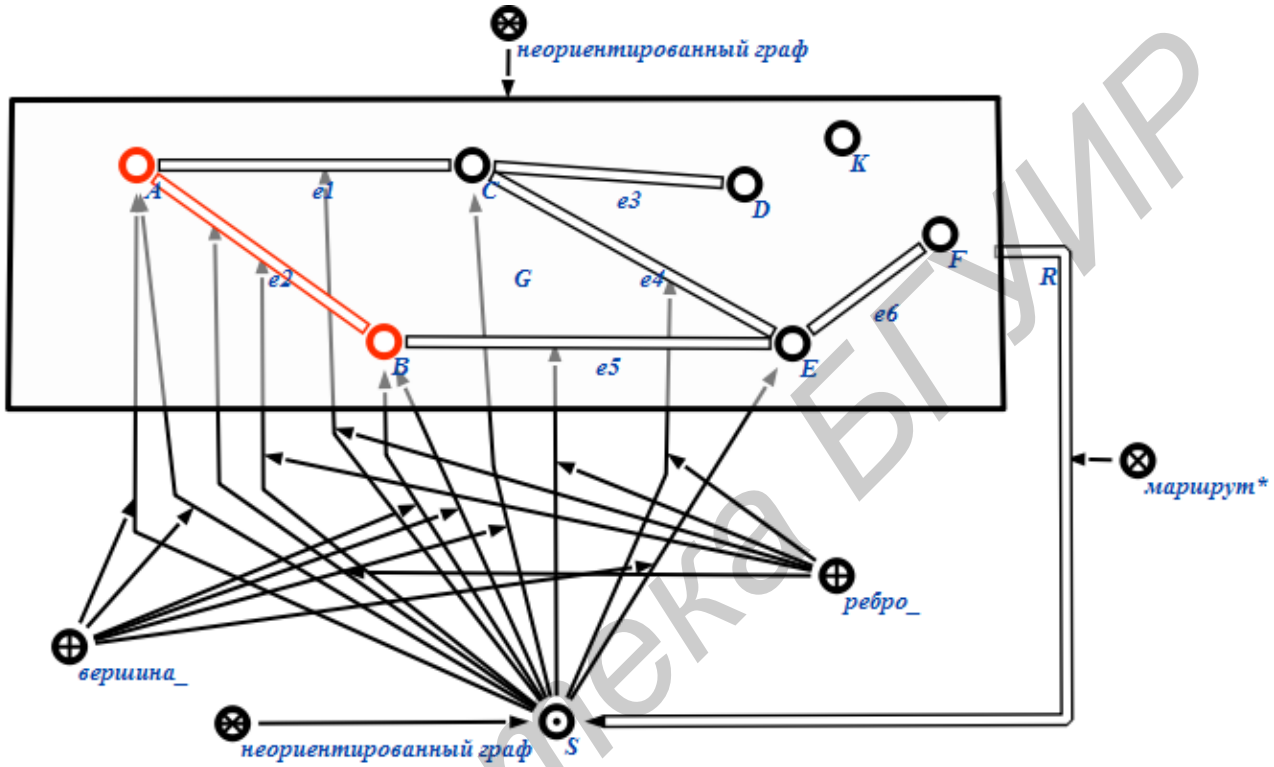


Рисунок 12 – Проблема представления маршрута как подграфа графа G

Вернемся к рассмотрению маршрута R в графе G. Тогда в основу способа представления положили, что маршрут состоит из вершин и ребер и он есть подграф графа, на котором задается. Может быть, слабость этого подхода была в том, что не рассматривался маршрут как последовательность вершин и ребер, т. е. не был задан порядок. Зададим последовательность элементов маршрута, например, при помощи ориентированного графа. Вершинами такого графа будут элементы последовательности, а дуги будут задавать отношение предыдущий/следующий. Первый элемент последовательности специально не указан, потому что первым элементом является вершина, в которую нет входящих дуг. Последним элементом последовательности будет вершина, из которой нет исходящих дуг. Этот способ представления маршрута R показан на рисунке 13 (элементы маршрута обозначены темными линиями, связки следующий/предыдущий – светлыми).

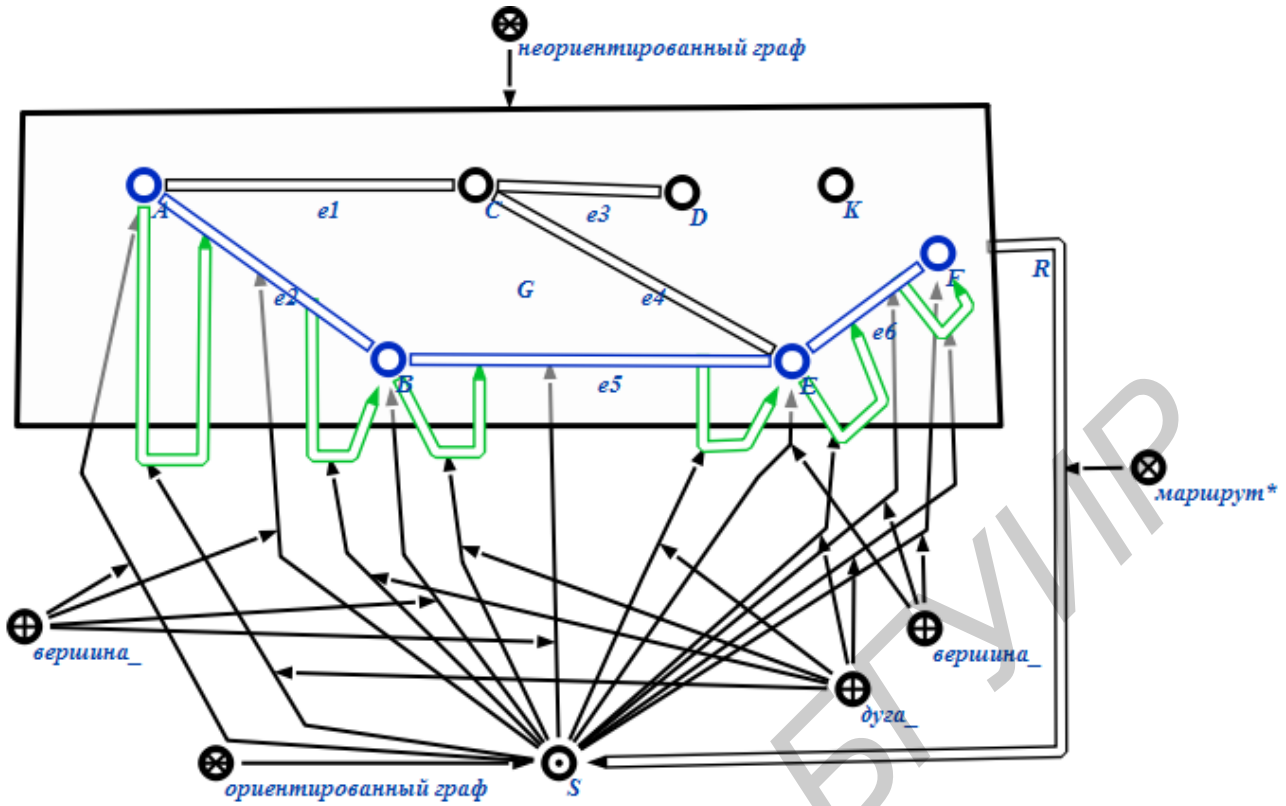


Рисунок 13 – Представление маршрута через ориентированный граф последовательности

Но, даже используя новый способ представления, нельзя задать маршрут, который не является цепью или простой цепью (путем). Новый способ представления все равно ограничен.

В определении маршрута сказано, что маршрут состоит из вершин и ребер. Но при более строгом рассмотрении оказывается, что это не совсем так. Мы нашли два ограниченных способа для представления маршрута, пользуясь буквально этим определением, а задачу еще не решили. Возможно, маршрут состоит не из последовательности вершин и ребер, а из последовательности посещений вершин и ребер. Рассмотрим именно такую точку зрения. Тогда можно преобразовать ориентированный граф последовательности S из рисунка 13 в ориентированный граф посещений S на рисунке 14. Вершина графа S на рисунке 14 обозначает посещение вершины графа G, а дуга графа S – посещение ребра графа G.

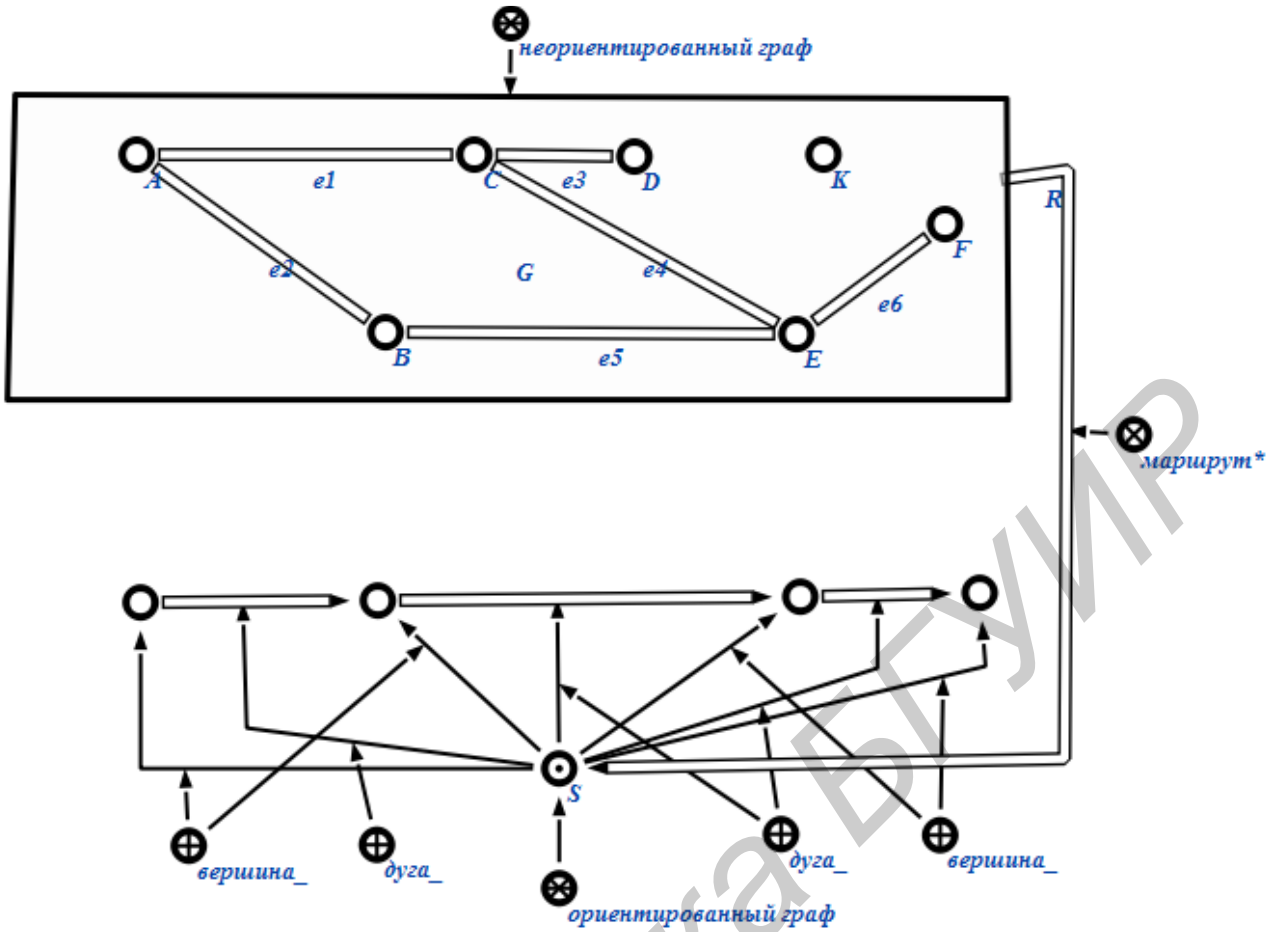


Рисунок 14 – Представление структуры S маршрута R как ориентированного графа посещений

На рисунке 14 не хватает только связей между элементами посещения из графа S и посещенными элементами из графа G. Для того чтобы можно было закончить конструкцию рисунка 13, необходимо дополнить соответствие между двумя множествами.

Введем относительное понятие (тернарное отношение) *соответствие**, связка которого включает следующие три компонента (пример связки на рисунке 15):

1) множество, которое является областью определения соответствия (множество X);
2) множество, которое является областью значений соответствия (множество Y);
3) бинарное ориентированное отношение, которое устанавливает соответствие между элементом из области определения и элементом из области значения (отношение *Cr**).

Рассмотрим маршрут R как всюду определенное, функциональное соответствие между ориентированным графом (структурой маршрута) S и неориентированным исходным графом G. Тогда на основе рисунков 14 и 15 построим окончательный вариант представления маршрута R (рисунок 16). Рассмотрим отличия рисунка 16 от рисунка 14:

1. Для наглядности упрощен ориентированный граф S (скрыты узлы *вершина'* и *дуга'*).
2. Показано, что отношение *маршрут** является подмножеством отношения *соответствие**.
3. В связке R первым компонентом является граф S, а вторым – граф G. На рисунке 14 было наоборот. Чтобы понять, почему связка оказалась перевернутой, рассмотрим рисунок 15. Видно, что первым компонентом связки отношения *соответствие** должно быть множество, которое является областью определения, а вторым – множество, которое является областью значения. Именно поэтому произошла такая перестановка.
4. Появилось отношение (третий компонент связки R), которое задает соответствие между посещением и посещенным элементом. Обратите внимание на направление бинарных ориентированных пар этого отношения.

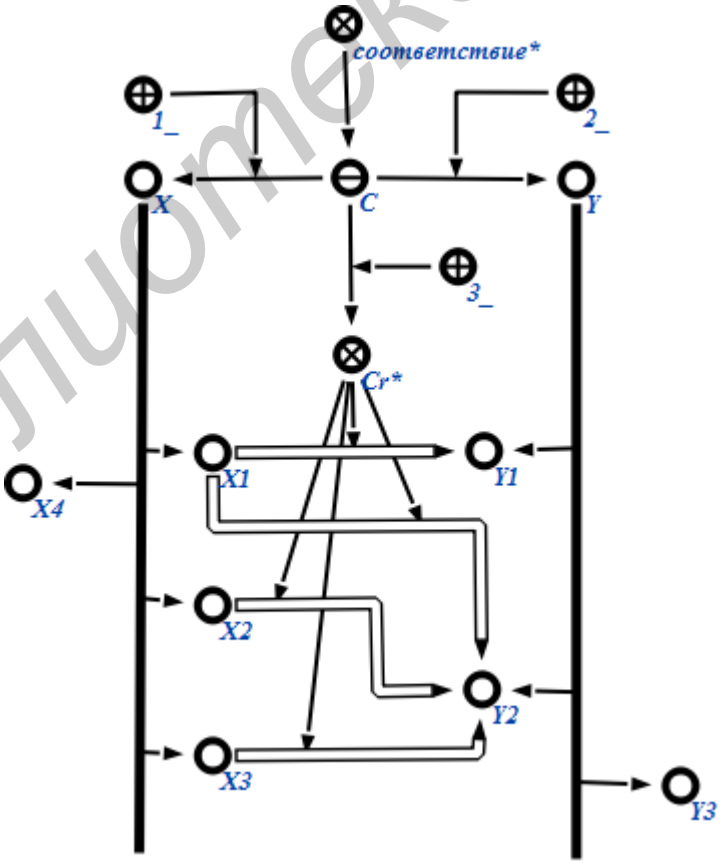


Рисунок 15 – Пример связки C отношения *соответствие** между множествами X и Y

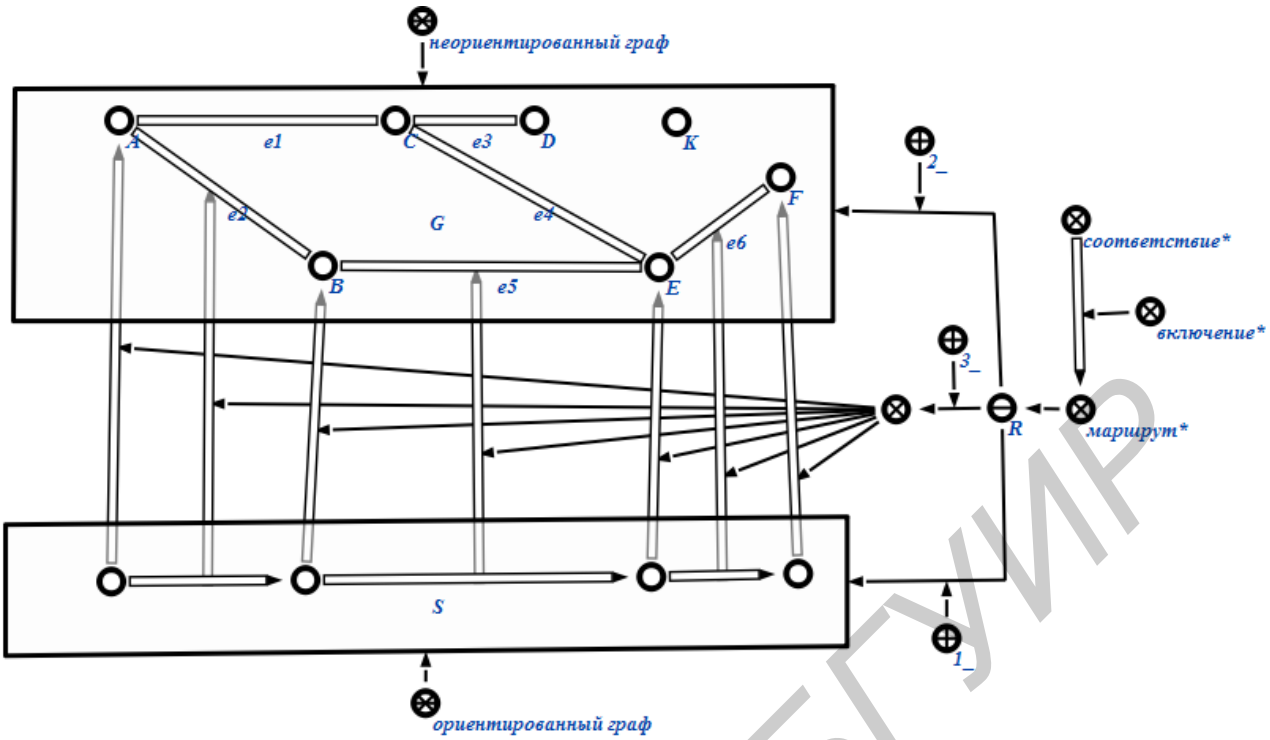


Рисунок 16 – Окончательный вариант представления маршрута R, как соответствия между графом S и графом G

Попробуем представить при помощи данного способа (рисунок 16) маршрут, в котором есть повторяющиеся вершина и/или ребра. Введем относительные понятия (бинарные ориентированные отношения) *цепь** и *путь**, которые связаны с отношением *маршрут** так, как показано на рисунке 17.

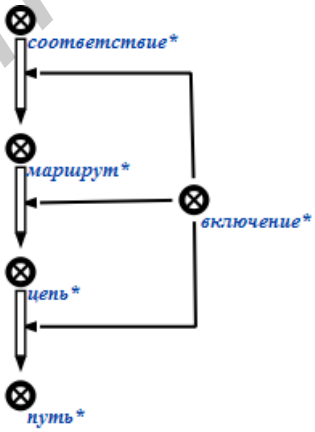


Рисунок 17 – Связь отношений *цепь** и *путь** с отношением *маршрут**

Тогда один из минимальных путей R (A, $e_2$, B, $e_5$, E, $e_6$, F) в графе G между вершинами A и F будет представляться в SC-коде так, как показано на рисунке 18.

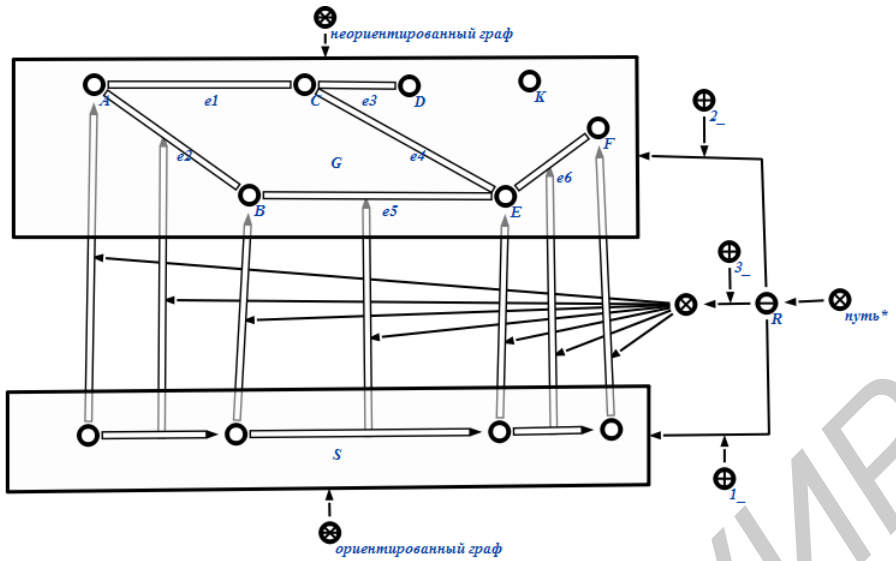


Рисунок 18 – Один из минимальных путей R между вершинами A и F графа G

Теперь перейдем к тому, как можно обобщить представление различных видов графов в SC-коде.

### 4.1.4 Обобщение различных видов графов с использованием понятия графовой структуры

В пункте 4.1.3 было дано представление наиболее общего понятия по отношению к понятию *путь**, а именно – маршрута. Таким образом, с определением кодирования понятия *маршрут** было определено и кодирование понятия *путь**. А в пункте 4.1.2 при рассмотрении неориентированных и ориентированных графов было дано только то, что необходимо для решения задачи, без характеристики более общих случаев. Теперь необходимо посмотреть на задачу представления графов в SC-коде более широко. Для этого стоит обратиться к базе знаний по теории графов из проекта OSTIS GT. Неполная иерархия различных типов графов из этой базы знаний изображена на рисунке 19. Узлы ориентированного и неориентированного графов составляют только «низ» иерархии, так как определены на основе более общих понятий. А начнем рассматривать иерархию с «верха», а именно, с понятия *графовая структура*.

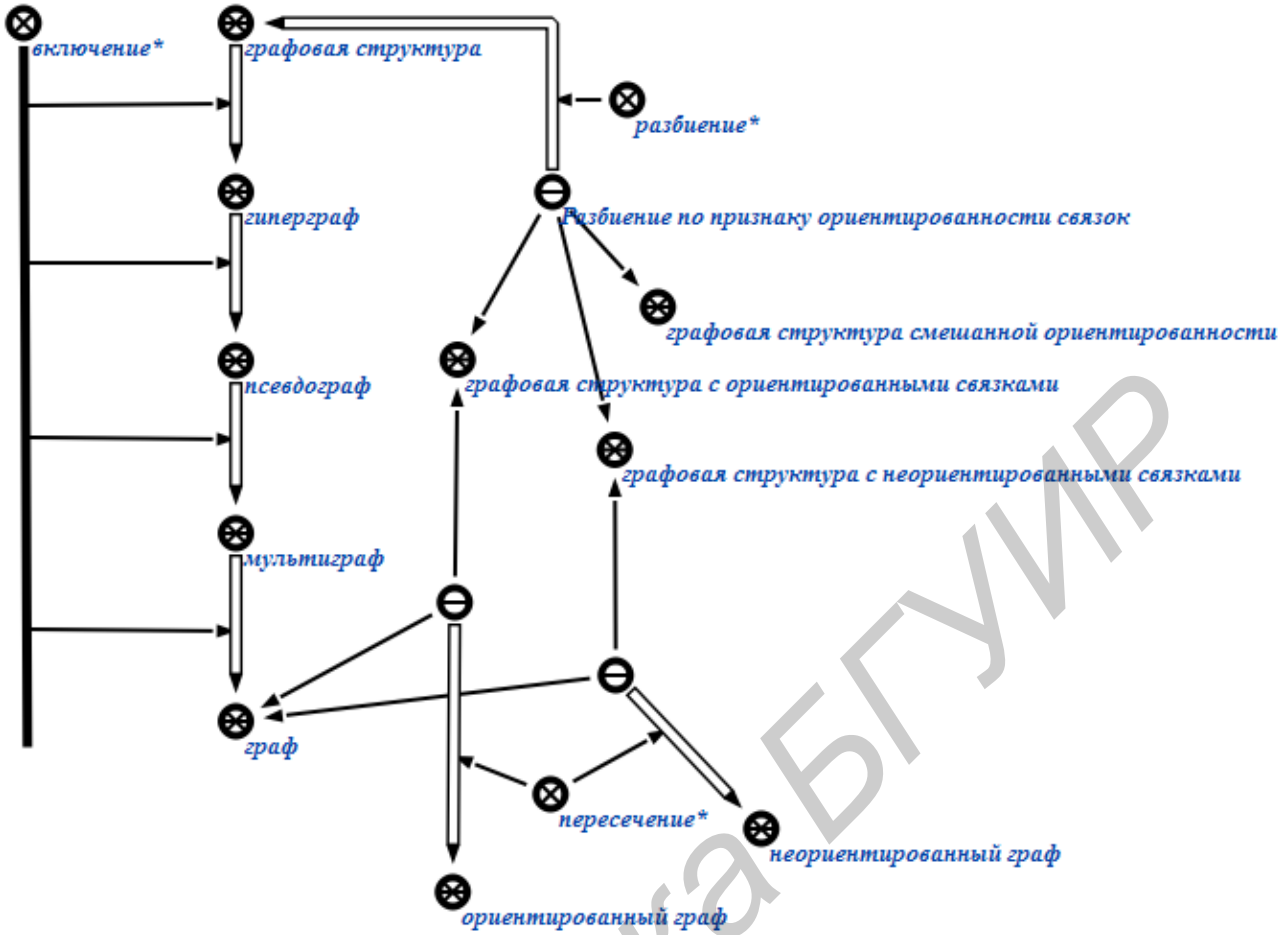


Рисунок 19 – Иерархия различных типов графовых структур (неполная)

Экземпляр понятия *графовая структура* – это такая одноуровневая реляционная структура, которая содержит объекты с ролью *вершина'*, а связки между этими объектами – с ролью *связка'*. На элементы с ролью *связка' графовой структуры* не накладывается ограничений, как на элементы с ролью *ребро' неориентированного графа*, а именно:

- они могут быть любой арности, а не только бинарными;
- они могут быть как ориентированными, так и неориентированными;
- их компонентами могут быть не только вершины, но и другие связки (т. е. разрешены связки, «выходящие» из связок, и связки, входящие в другие связки).

На рисунке 20 приведен пример графовой структуры $G_s$, в которой четыре вершины и три связки. Графовая структура вполне может включать элементы с ролью *ребро'*. Понятие *связка'* более общее, чем *ребро'*. Чтобы лучше увидеть связь между этими ролевыми отношениями, рассмотрите рисунок 21, на котором изображена иерархия ролевых отношений элементов графовой структуры из базы знаний по теории графов. Теперь перейдем к краткому описанию абсолютных понятий *гиперграф*, *псевдограф*, *мультиграф* и *граф*.

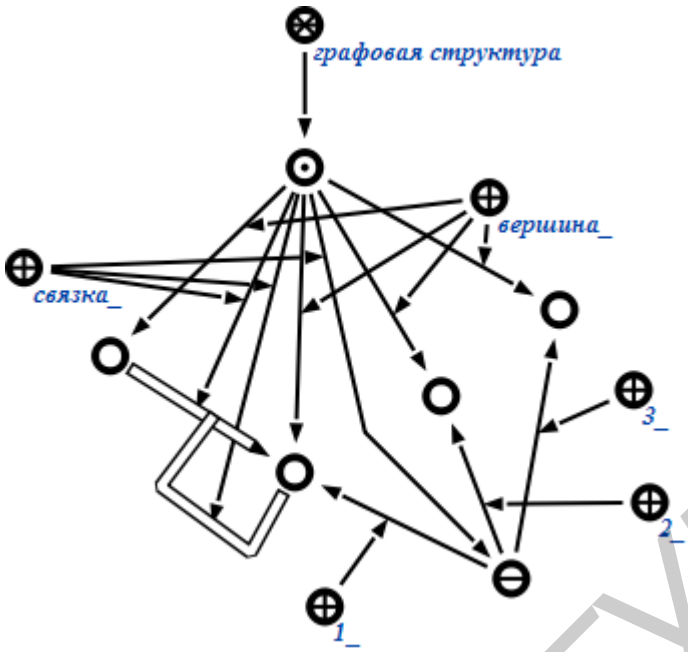


Рисунок 20 – Пример графовой структуры $G_s$

Экземпляр понятия *гиперграф* (см. рисунок 19) – это такая графовая структура, в которой элемент с ролью *связка'* может иметь в качестве своих компонент только элемент с ролью *вершина'* этой графовой структуры. На арность связок никакого ограничения не накладывается. Для указания связки введены ролевые отношения *гиперсвязка'*, *гипердуга'*, *гипперребро'* (рисунок 21). Обратите внимание на то, что на рисунке 21 *гипердуга'* определена как пересечение понятий *гиперсвязка_* и ориентированная связка. Аналогично дела обстоят с понятием *гипперребро'*.

Экземпляр понятия *псевдограф* (см. рисунок 19) – это такой гиперграф, в котором гиперсвязка может иметь только две компоненты. Для указания связки введены ролевые отношения *бинарная связка', петля', дуга', ребро'* (см. рисунок 21). Петлей называется бинарная связка, у которой оба компонента одинаковы.

Экземпляр понятия *мультиграф* (см. рисунок 19) – это такой псевдограф, в котором не может быть петель.

Экземпляр понятия *граф* (см. рисунок 19) – это такой мультиграф, в котором не может быть кратных связок, т. е. связок, у которых первая и вторая компоненты совпадают.

А теперь самостоятельно на основе рисунка 19 рассмотрите то, как определены уже знакомые вам понятия *ориентированный граф* и *неориентированный граф.*

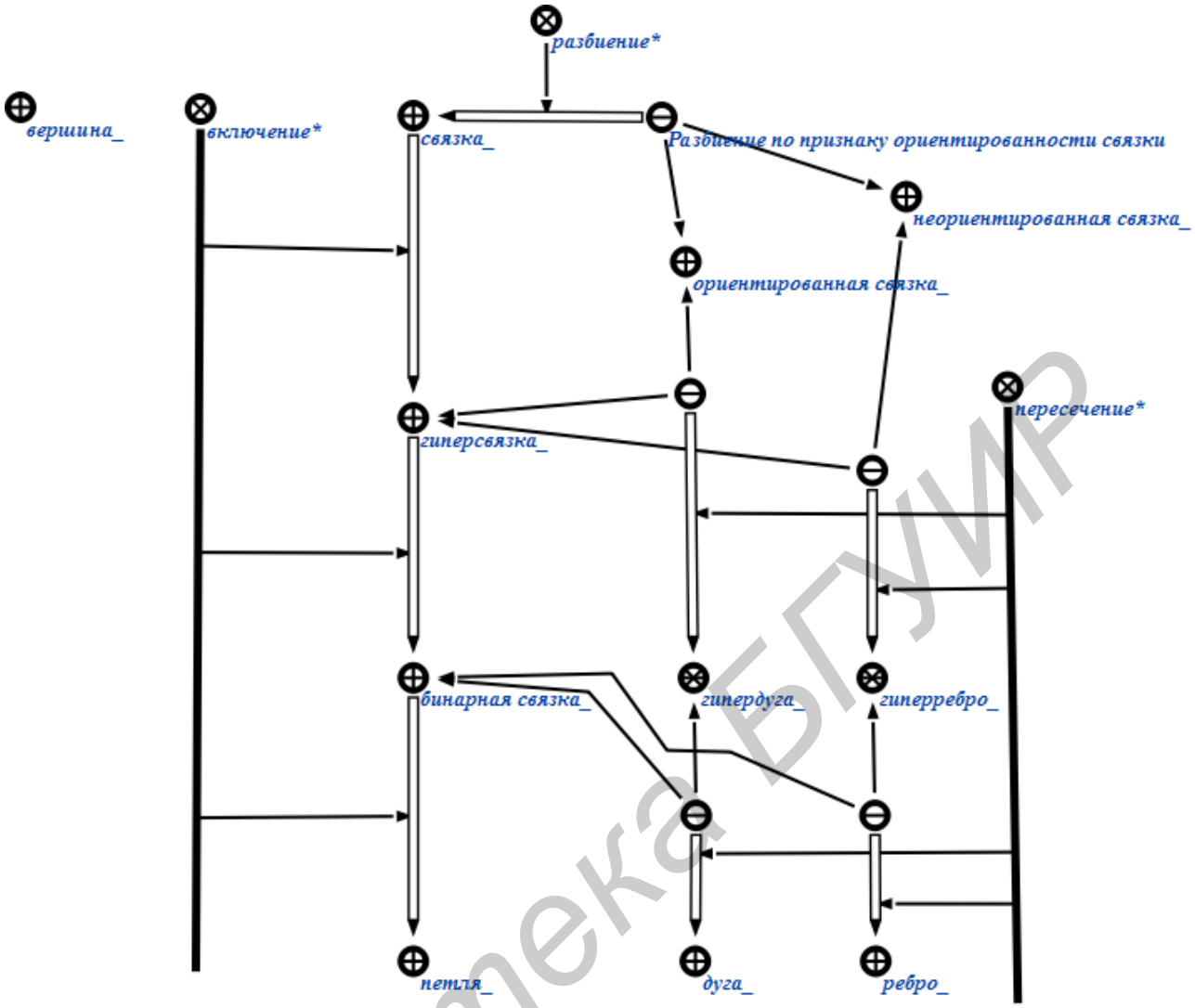


Рисунок 21 – Иерархия ролей элементов графовой структуры (неполная)

### 4.1.5 Пример выполнения задания

*Список понятий*:

1. Графовая структура (абсолютное понятие) – это такая одноуровневая реляционная структура, объекты которой могут играть роль либо вершины, либо связки (рисунок 22):

– вершина (относительное понятие, ролевое отношение);

– связка (относительное понятие, ролевое отношение).

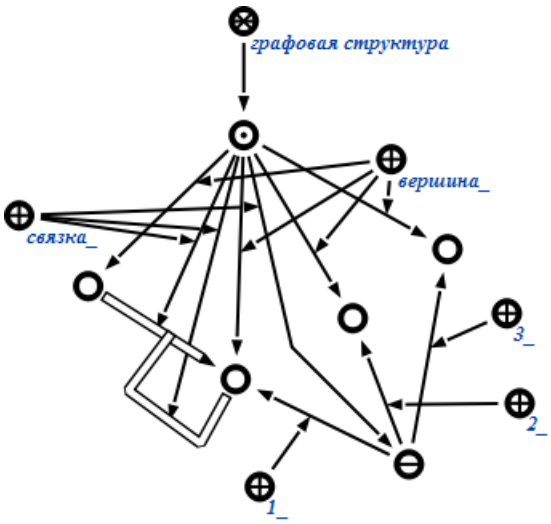


Рисунок 22 – Графовая структура

2. Графовая структура с ориентированными связками (абсолютное понятие) (рисунок 23):
   – ориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается ориентированным множеством.

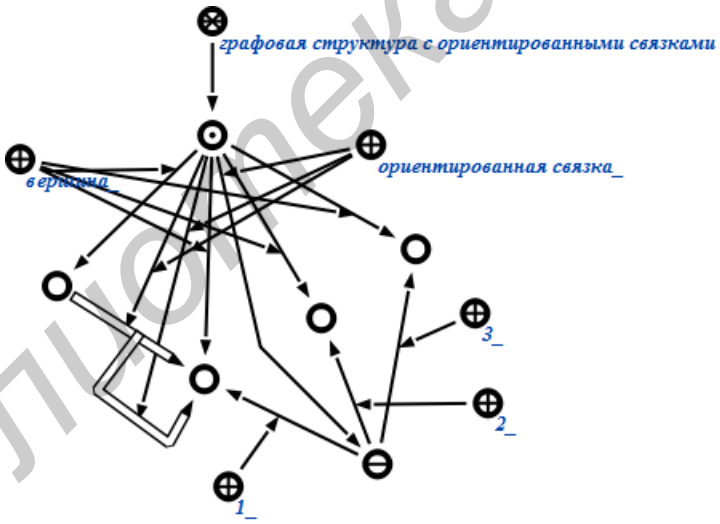


Рисунок 23 – Графовая структура с ориентированными связками

3. Графовая структура с неориентированными связками (абсолютное понятие) (рисунок 24):
   – неориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается неориентированным множеством.

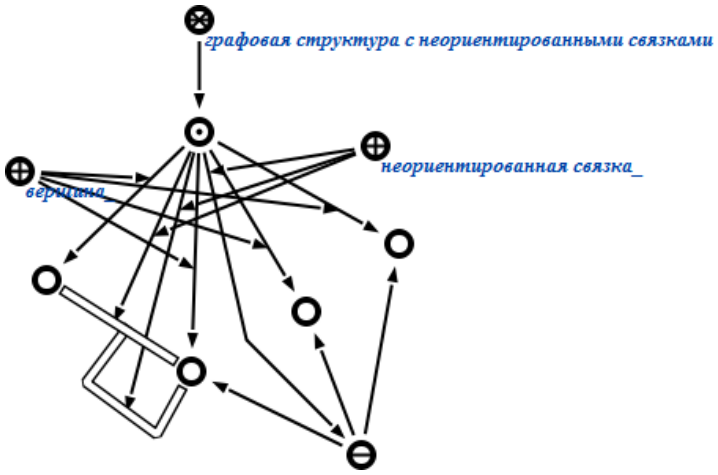


Рисунок 24 – Графовая структура с неориентированными связками

4. Гиперграф (абсолютное понятие) – это такая графовая структура, в которой связки могут связывать только вершины (рисунок 25):
   – гиперсвязка (относительное понятие, ролевое отношение);
   – гипердуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;
   – гиперребро (относительное понятие, ролевое отношение) – неориентированная гиперсвязка.

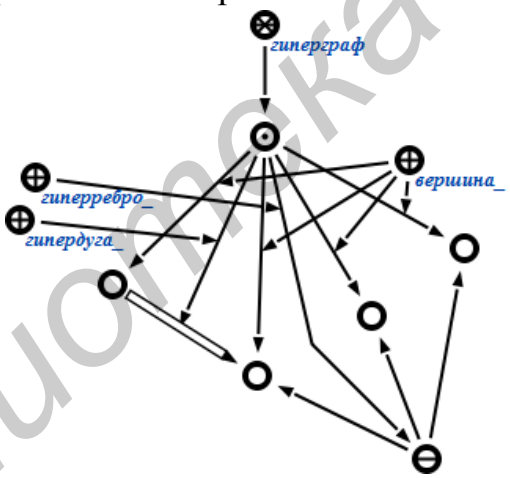


Рисунок 25 – Гиперграф

5. Псевдограф (абсолютное понятие) – это такой гиперграф, в котором все связки должны быть бинарными (рисунок 26):
   – бинарная связка (относительное понятие, ролевое отношение) – гиперсвязка арности 2;
   – ребро (относительное понятие, ролевое отношение) – неориентированная гиперсвязка;
   – дуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;
   – петля (относительное понятие, ролевое отношение) – бинарная связка, у которой первый и второй компоненты совпадают.

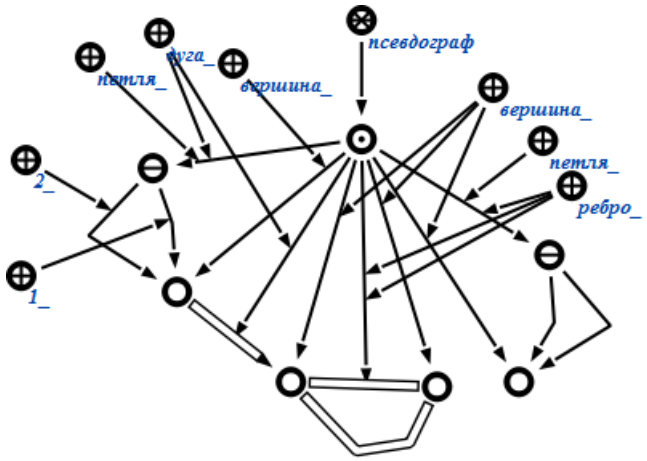


Рисунок 26 – Псевдограф

6. Мультиграф (абсолютное понятие) – это такой псевдограф, в котором не может быть петель (рисунок 27).

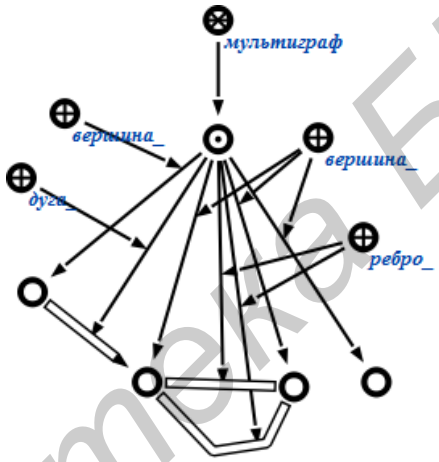


Рисунок 27 – Мультиграф

7. Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратных связок, т. е. связок, у которых первый и второй компоненты совпадают (рисунок 28).

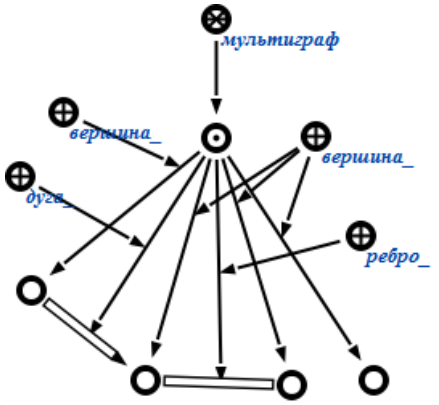


Рисунок 28 – Граф (абсолютное понятие)

8. Неориентированный граф (абсолютное понятие) – это такой граф, в котором все связки являются ребрами (рисунок 29).

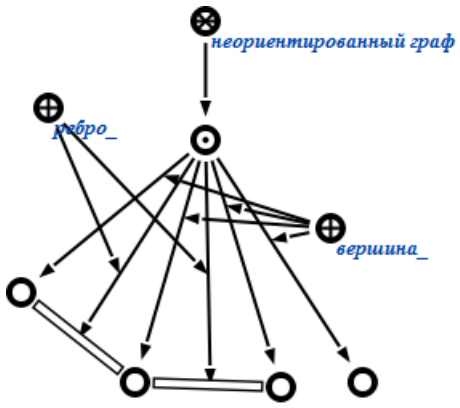


Рисунок 29 – Неориентированный граф (абсолютное понятие)

9. Ориентированный граф (абсолютное понятие) – это такой граф, в котором все связки являются дугами (рисунок 30).

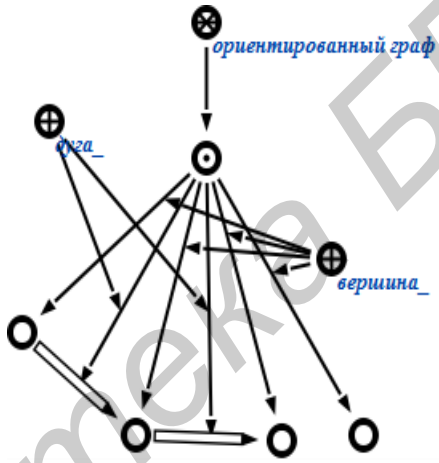


Рисунок 30 – Ориентированный граф (абсолютное понятие)

10. Маршрут (относительное понятие, бинарное ориентированное отношение) – это чередующаяся последовательность вершин и гиперсвязок в гиперграфе, которая начинается и заканчивается вершиной, и каждая гиперсвязка последовательности инцидентна двум вершинам, одна из которых непосредственно предшествует ей, а другая непосредственно следует за ней. В примере на рисунке 31 показан маршрут A, CON1, C, CON2, D, CON3, B, CON1, A в гиперграфе. Графы в примере приведены в сокращенной форме, CON1 – это тернарная неориентированная связка (гиперсвязка), CON2 и CON3 – бинарные связки (гиперсвязки).

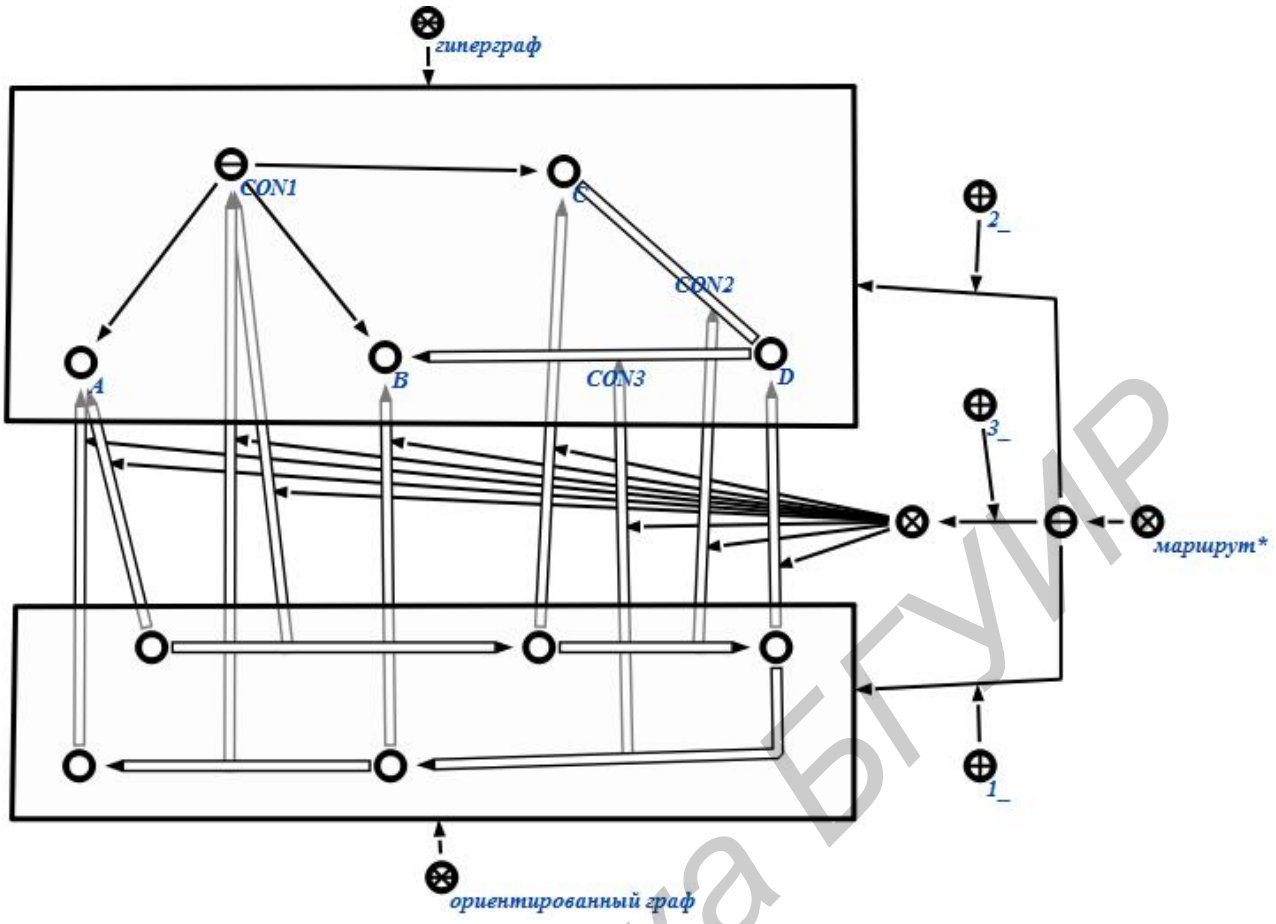


Рисунок 31 – Маршрут (относительное понятие)

11. Цепь (относительное понятие, бинарное ориентированное отношение) – это маршрут, все гиперсвязки которого различны. В примере на рисунке 32 показана цепь A, CON1, C, CON2, D, CON3, B, CON1, A в гиперграфе.

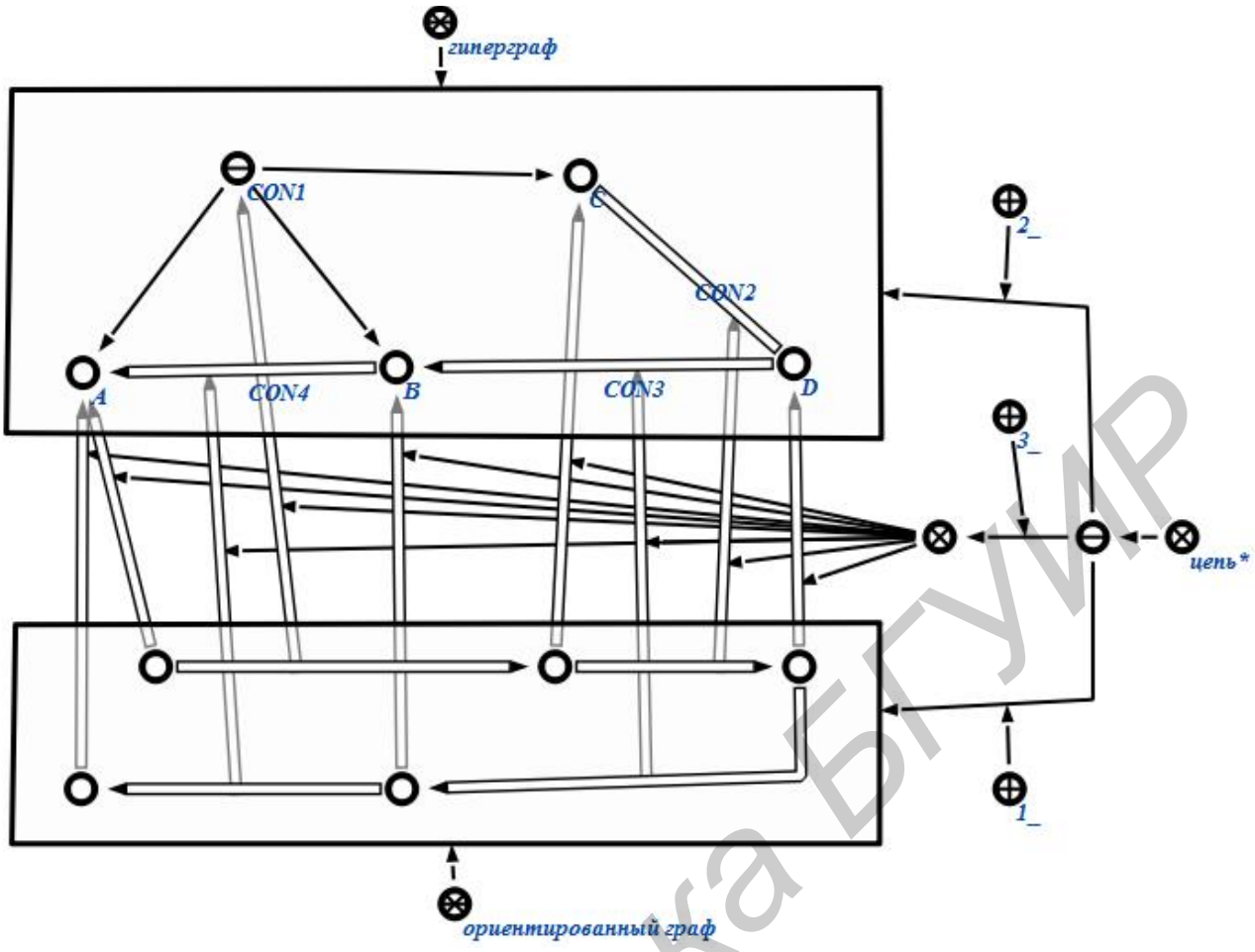


Рисунок 32 – Цепь (относительное понятие)

12. Простая цепь, путь (относительное понятие, бинарное ориентированное отношение) – это цепь, в которой все вершины различны. В примере на рисунке 33 показан путь A, CON1, C, CON2, D, CON3, B в гиперграфе.

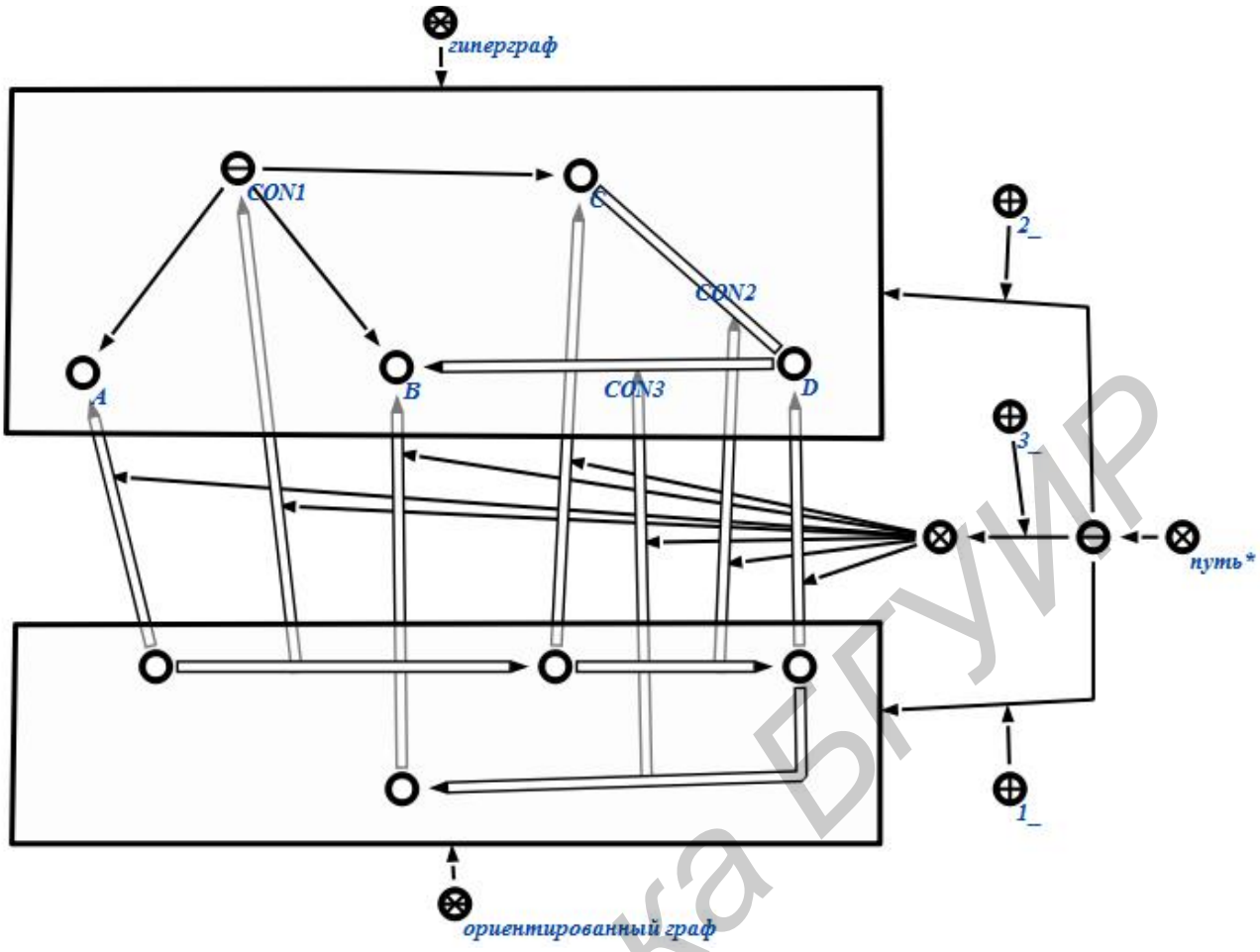


Рисунок 33 – Простая цепь (относительное понятие)

*Тестовые примеры*

Во всех тестах графы будут приведены в сокращенной форме со скрытыми ролями элементов графа.

**Тест 1** (рисунок 34)

Вход: необходимо найти минимальный путь между вершинами A и C.

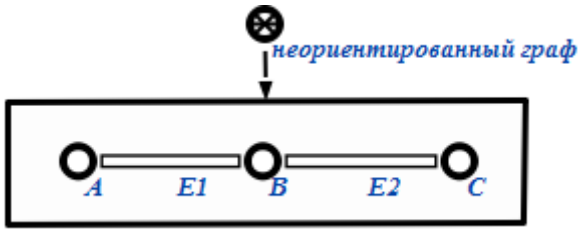


Рисунок 34 – Неориентированный граф к тесту 1

Выход: будет найден единственный минимальный путь A, E1, B, E2, C, (рисунок 35).

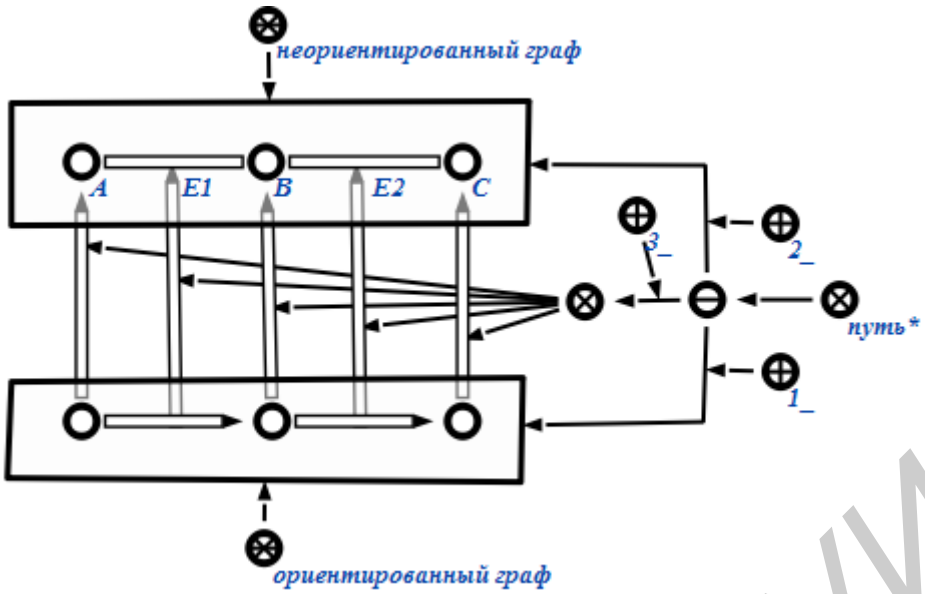


Рисунок 35 – Минимальный путь

**Тест 2** (рисунок 36)

Вход: необходимо найти минимальный путь между вершинами A и F.

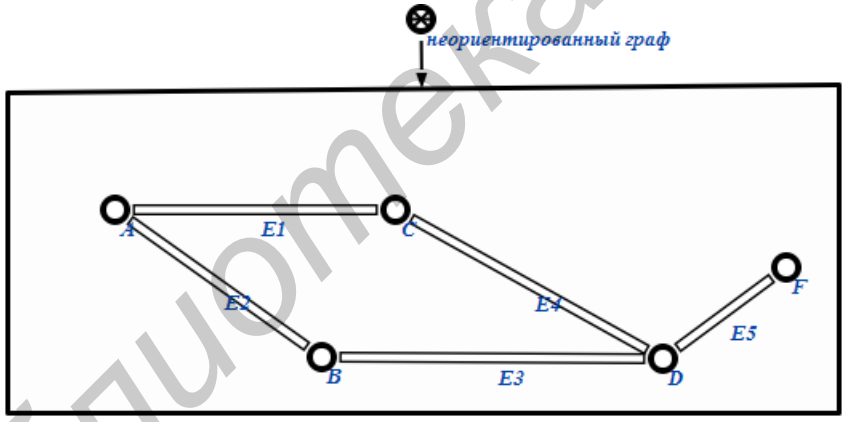


Рисунок 36 – Неориентированный граф к тесту 2

Выход: будет найден один из двух минимальных путей A, E2, B, E3, D, E5, F (рисунок 37).

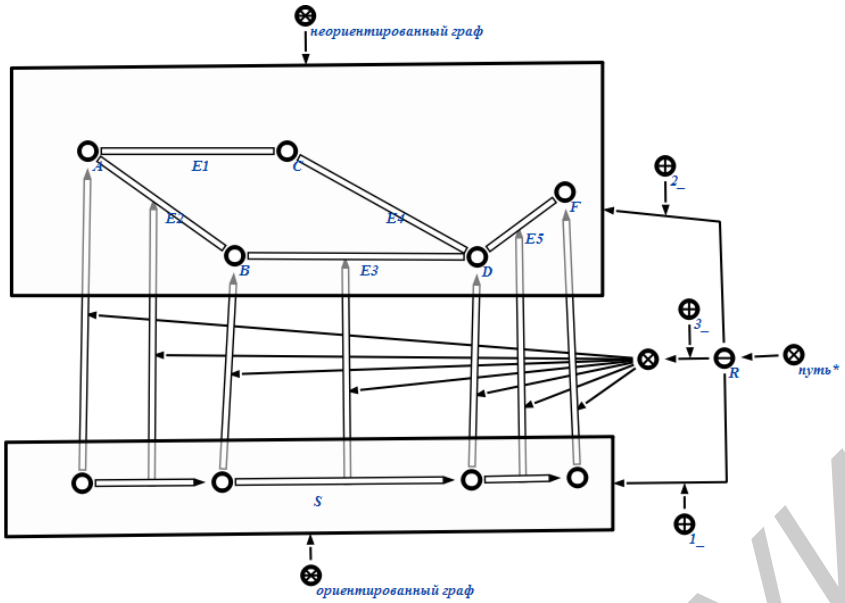


Рисунок 37 – Минимальные пути к тесту 2

**Тест 3** (рисунок 38)

Вход: необходимо найти минимальный путь между вершинами A и K.

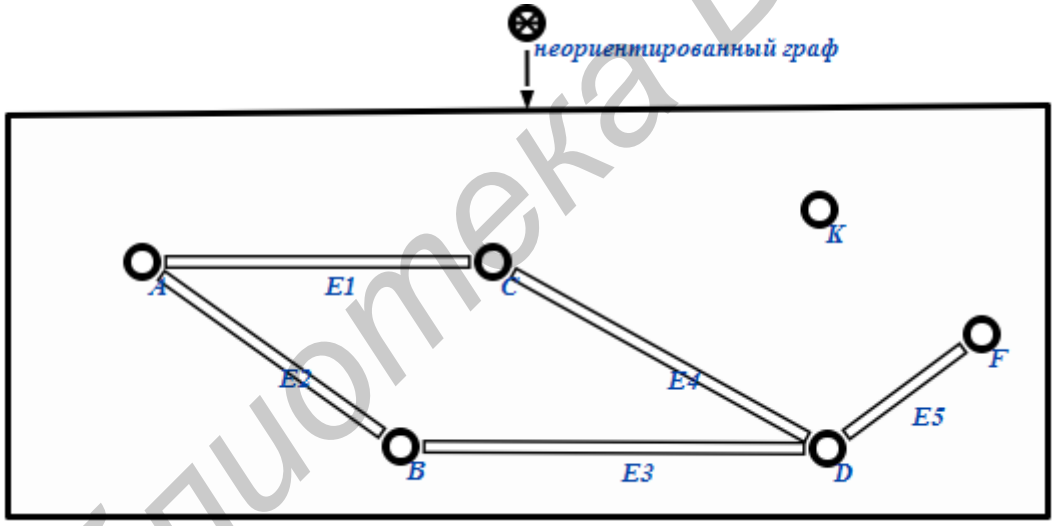


Рисунок 38 – Неориентированный граф к тесту 3

Выход: Минимальный путь между вершинами A и K не существует. Программа должна вернуть ошибку вызывающему контексту.

**Тест 4** (рисунок 39)

Вход: необходимо найти минимальный путь между вершинами V5 и V11.

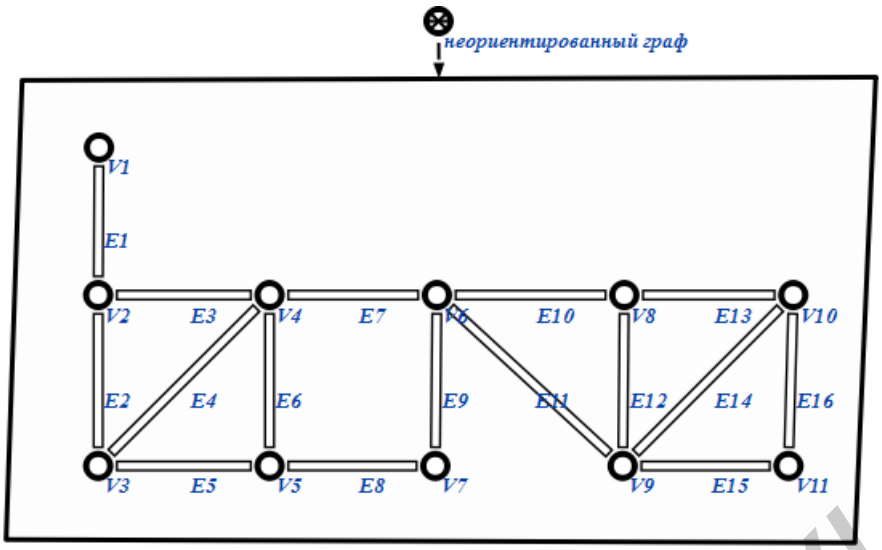


Рисунок 39 – Неориентированный граф к тесту 4

Выход: будет найден один из двух минимальных путей V5, E8, V7, E9, V6, E11, V9, E15, V11 (рисунок 40).

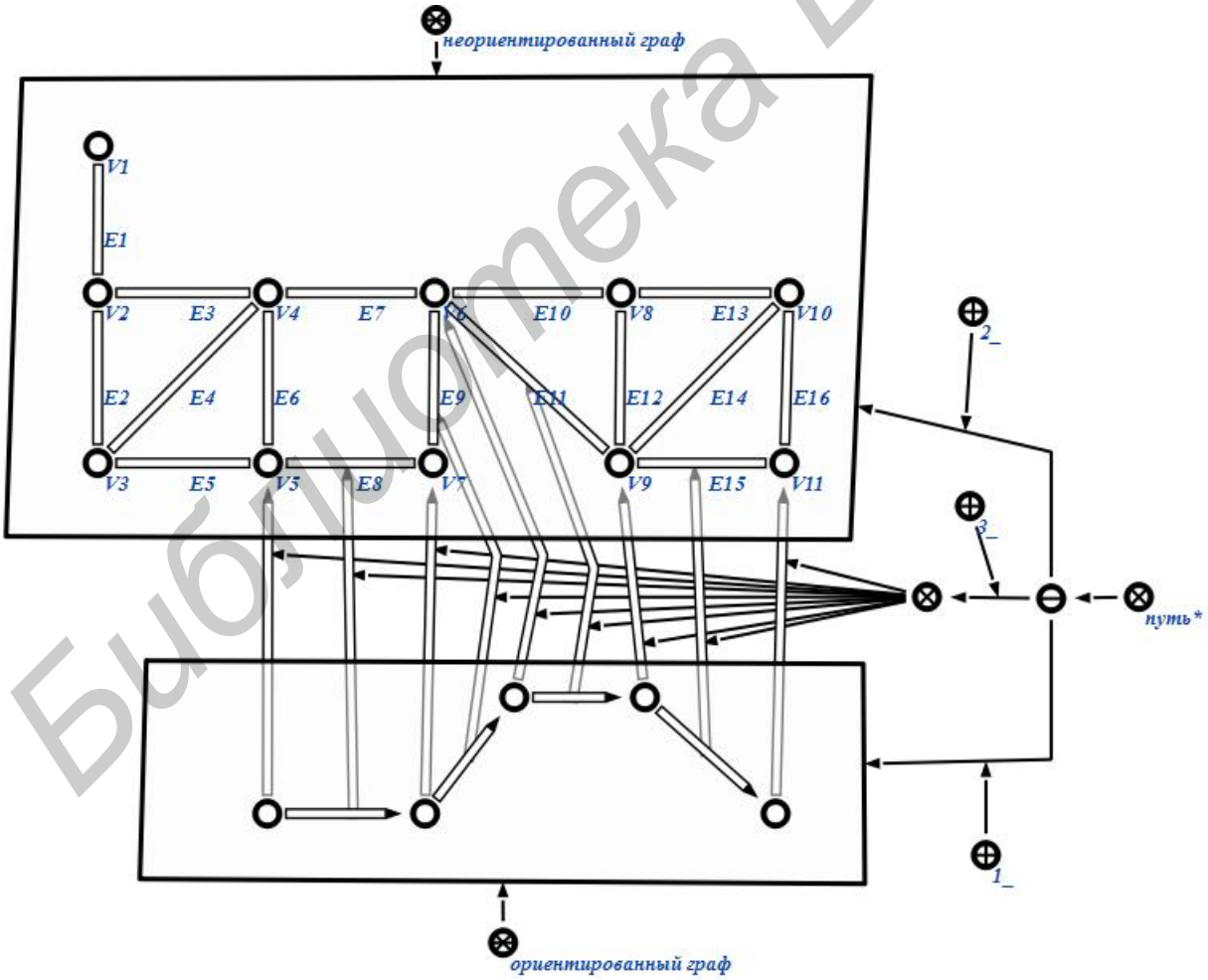


Рисунок 40 – Минимальные пути к тесту 4

**Тест 5** (рисунок 41)

Вход: необходимо найти минимальный путь между вершинами V1 и V9.

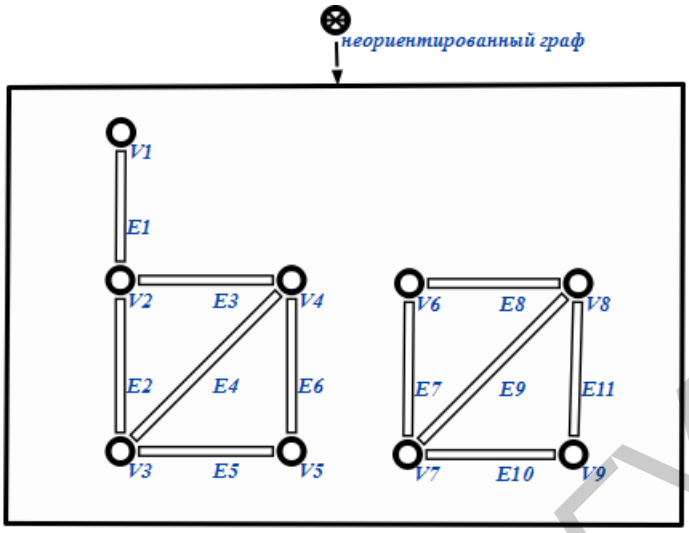


Рисунок 41 – Неориентированный граф к тесту 5

Выход: минимальный путь между вершинами V1 и V9 не существует. Программа должна вернуть ошибку вызывающему контексту.

### 4.2. Варианты индивидуальных заданий

Варианты индивидуальных заданий к занятию 4 аналогичны перечисленным в занятии 2 (подраздел 2.2).

# 5 БАЗОВЫЕ ПРИНЦИПЫ ОБРАБОТКИ ГРАФОВЫХ СТРУКТУР В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

*Содержание занятия*: разработка алгоритма решения предложенной теоретико-графовой задачи на основе формального представления графовой структуры.

## 5.1 Теоретические сведения

В рамках данного занятия необходимо продемонстрировать пошаговое выполнение алгоритма в sc-памяти. Сам алгоритм и структуры данных для него уже исследовались на предыдущих занятиях, а сейчас продемонстрируем графодинамику выполнения алгоритма. Это значит, что вся информация, необходимая для работы вашего алгоритма, должна храниться в sc-памяти и там же и обрабатываться. Недостаток данного требования – невозможность использования привычной матрицы смежности/инцидентности.

### 5.1.1 Волновой алгоритм поиска одного из минимальных путей в неориентированном графе

*Описание алгоритма*

В рамках алгоритма волной называется множество вершин, каждая из которых является в обрабатываемом графе смежной хотя бы одной вершине из предыдущей волны. Волна, для которой нет предыдущей волны, называется начальной и состоит из вершины, от которой начинается поиск минимального пути. Волна, включающая конечную вершину пути, называется конечной. Таким образом, алгоритм можно задать следующим перечнем шагов:

1. Добавить все вершины графа, кроме начальной вершины пути, во множество непроверенных вершин.
2. Создать новую волну и добавить в нее начальную вершину пути. Начальная волна – это новая волна. Новой волной будем называть последнюю созданную волну.
3. Сформировать следующую волну для новой волны. В нее попадет та вершина, которая является смежной вершине из новой волны и присутствует во множестве непроверенных вершин. Если вершина попала в формируемую волну, то ее надо исключить из множества непроверенных вершин. Созданную волну необходимо установить как следующую для новой волны, и после этого созданную волну считать новой волной.

4. Если новая волна пуста, то между вершинами не существует пути. Завершить алгоритм.
5. Если в текущей волне есть конечная вершина, то перейти к шагу 6, иначе – к шагу 3.
6. Сформировать один из минимальных путей, проходя в обратном порядке по списку волн. Завершить алгоритм.

*Пример выполнения алгоритма в sc-памяти*

Перед демонстрацией выполнения алгоритма в sc-памяти необходимо установить некоторые соглашения по формирование SCg-рисунков.

При записи графов будет использоваться сокращенная форма: атрибуты для вершин и связок не приводятся (используется только для наглядности).

В качестве программных переменных, применяемых в ходе работы алгоритма, будут использоваться sc-переменные. Для указания значения sc-переменной используется отношение *значение**. На рисунке 42 задано значение для sc-переменной *_beg_vertex*.

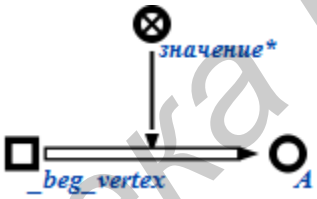


Рисунок 42 – Пример указания значения sc-переменной *_beg_vertex*

Однако будем сокращать способ, показанный на рисунке 42, опуская знак отношения *значение**. Таким образом, будет использована форма, показанная на рисунке 43. Важно, что полученная sc-конструкция не является корректной в семантическом смысле, но в данном разделе использование именно такой формы позволит сделать рисунки более понятными.

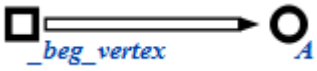


Рисунок 43 – Пример сокращенного указания значения sc-переменной*_beg_vertex*

Последнее, о чем стоит упомянуть, это использование цвета с целью явного выделения изменений, произошедших с sc-конструкцией:

- фиолетовым – sc-переменные, значение которых изменилось;
- зеленым – созданные sc-элементы;
- синим – измененные sc-элементы (например, sc-множества, в которые был включен или из которых был исключен элемент).

*Демонстрация алгоритма*

Выполнение алгоритма продемонстрируем через состояния sc-памяти на каждом элементарном этапе решения задачи. Для всех элементарных этапов, для которых это возможно, в конце их краткого описания в скобках будет указан соответствующий номер шага алгоритма, рассмотренного выше.

***Задание входного графа, начальной и конечной вершины пути для работы алгоритма*** *(рисунок 44)*

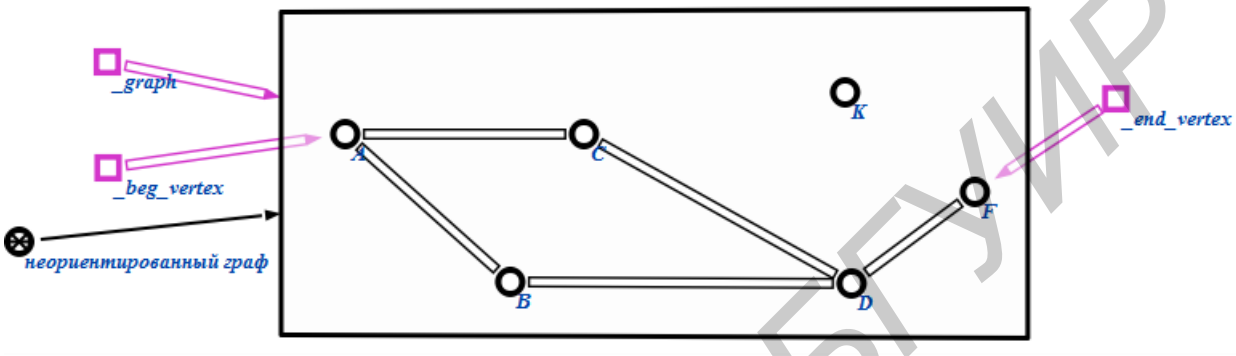


Рисунок 44 – Задание входного графа, начальной и конечной вершины пути для работы алгоритма

Переменные изменятся следующим образом:

- *_graph* получит в качестве значения sc-узел неориентированного графа;
- *_beg_vertex* получит в качестве значения вершину A, которая будет начальной для поиска минимального пути;
- *_end_vertex* получит в качестве значение вершину F, которая будет конечной для поиска минимального пути. Таким образом, из состояния sc-памяти на этом шаге вам должно быть ясно, что будет производиться поиск минимального пути между вершинами A и F.

***Шаг 1. Создание множества непроверенных вершин*** *(рисунок 45)*

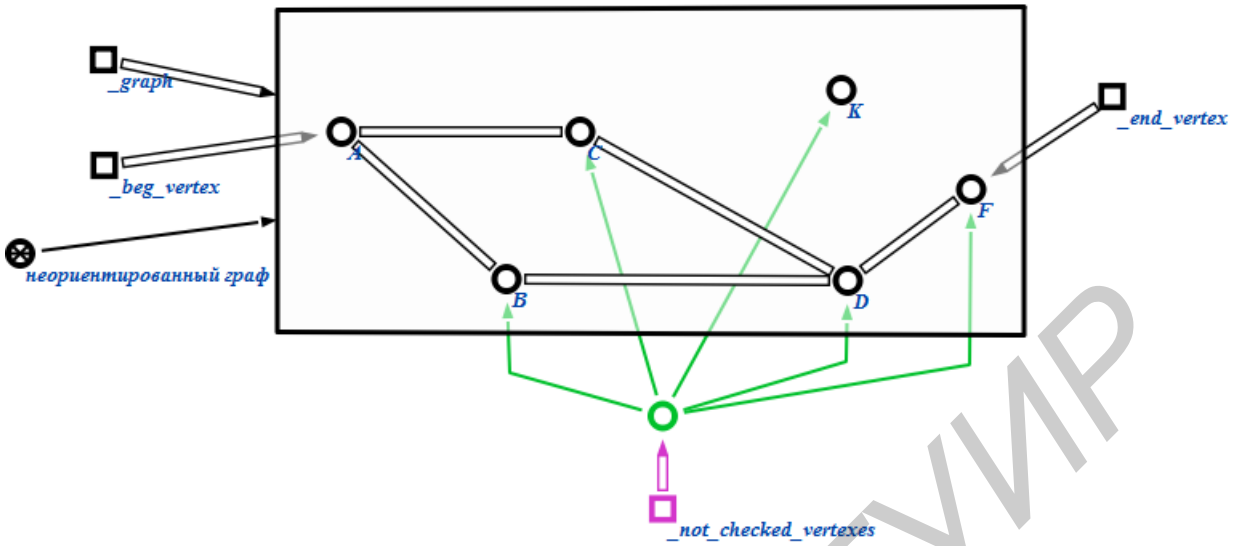


Рисунок 45 – Создание множества непроверенных вершин

Переменная *_not_checked_vertexes* получит в качестве значения множество непроверенных вершин обрабатываемого графа (в это множество не включена начальная вершина пути A).

***Шаги 2, 3. Создание волны, включающей начальную вершину A*** *(рисунок 46)*

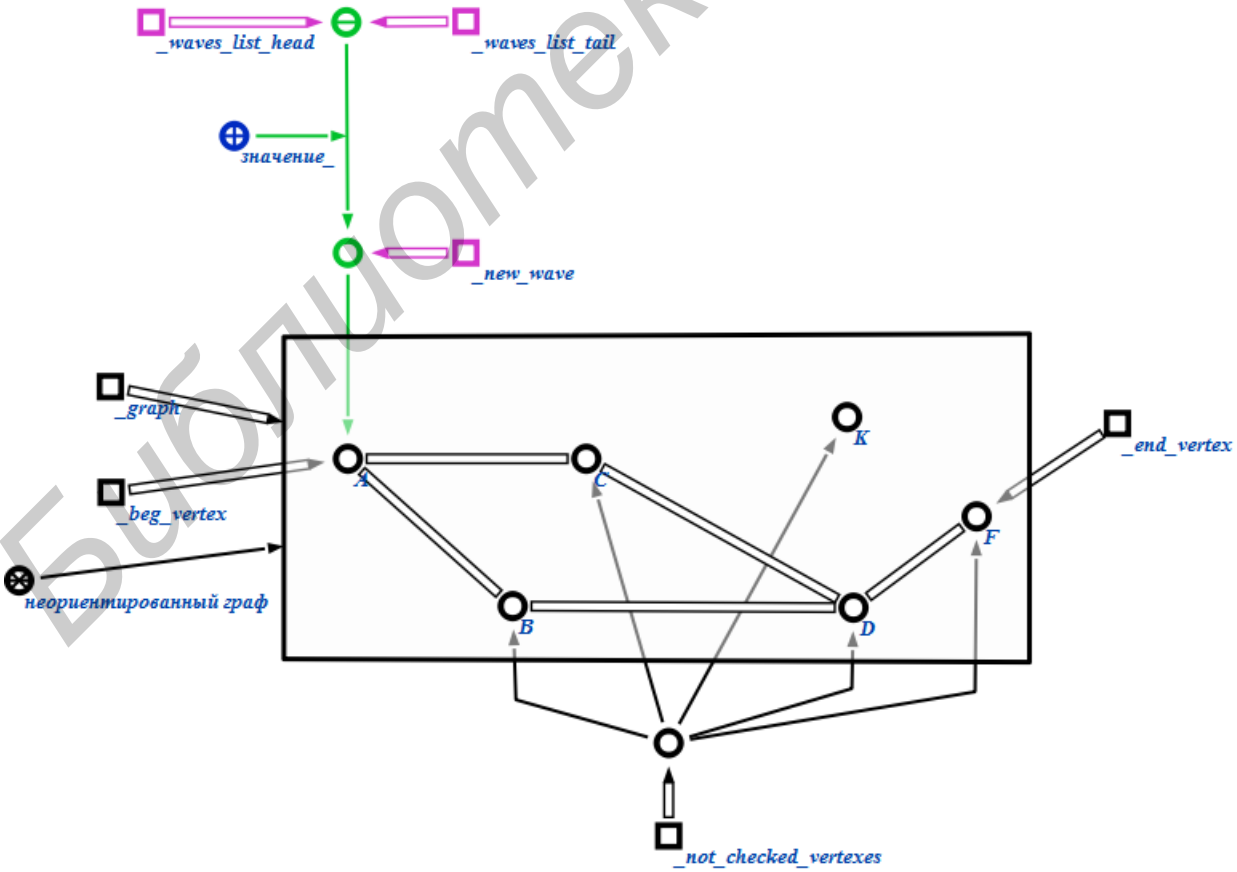


Рисунок 46 – Создание волны, включающей начальную вершину A

На этом этапе программа создает первую волну из списка волн. Первая волна содержит только начальную вершину пути A. Переменная *_new_wave* получает в качестве значения созданную волну, и в будущем будет всегда указывать на вновь созданную волну.

Переменная *_waves_list_head* указывает на начальный элемент списка волн, а переменная *_waves_list_tail* сейчас и в последующих шагах – на концевой элемент списка волн.

***Шаг 4. Создание волны, включающей вершины B и C*** *(рисунок 47)*

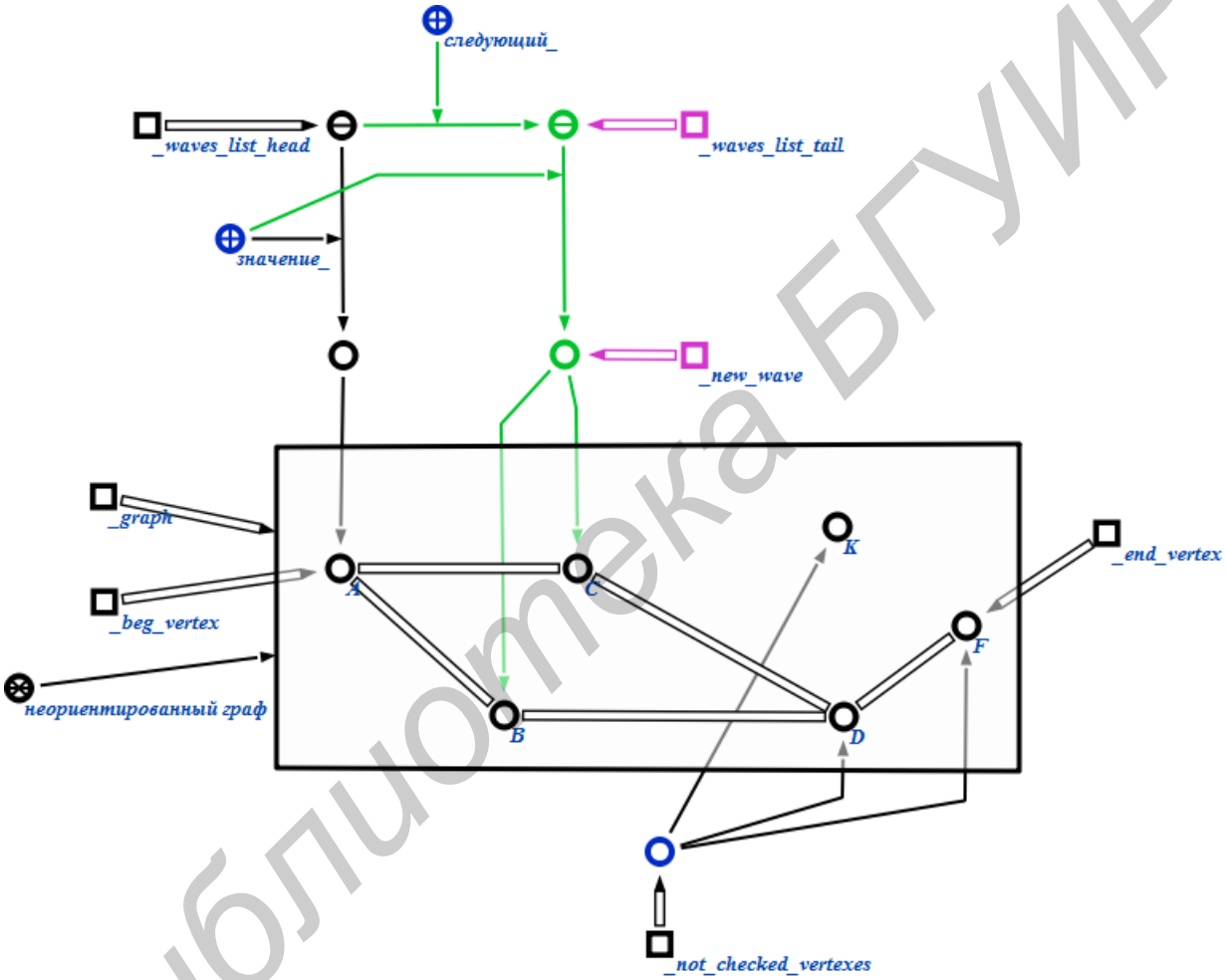


Рисунок 47 – Создание волны, включающей вершины B и C

Для вершин из предыдущей волны являются смежными и входящими во множество проверенных вершин только вершины B и C. Из них формируется новая волна. Эти вершины исключаются из множества непроверенных вершин (см. значение переменной *_not_checked_vertexes*).

Переменная *_waves_list_tail* получает в качестве значения созданный элемент списка, а переменная *_new_wave* – созданную волну.

***Шаг 5. Создание волны, включающей вершину D*** *(рисунок 48)*

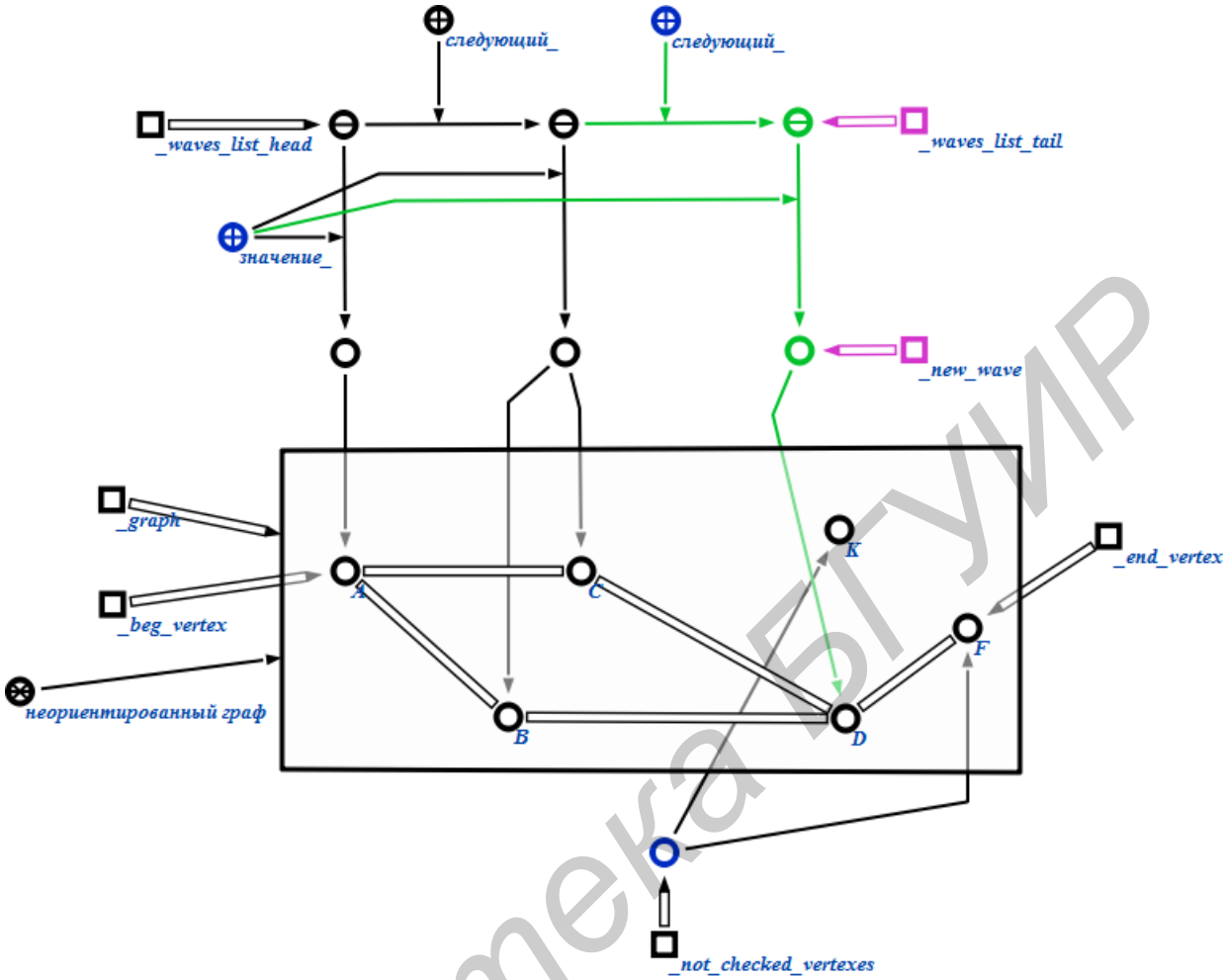


Рисунок 48 – Создание волны, включающей вершину D

Для вершин из предыдущей волны является смежной и входящей во множество проверенных вершин только вершина D. Из нее формируется новая волна. Эта вершина исключается из множества непроверенных вершин (см. значение переменной *_not_checked_vertexes*).

Переменная *_waves_list_tail* получает в качестве значения созданный элемент списка, а переменная *_new_wave* – созданную волну.

***Шаги 6, 7, 8. Создание волны, включающей вершину F*** *(рисунок 49)*

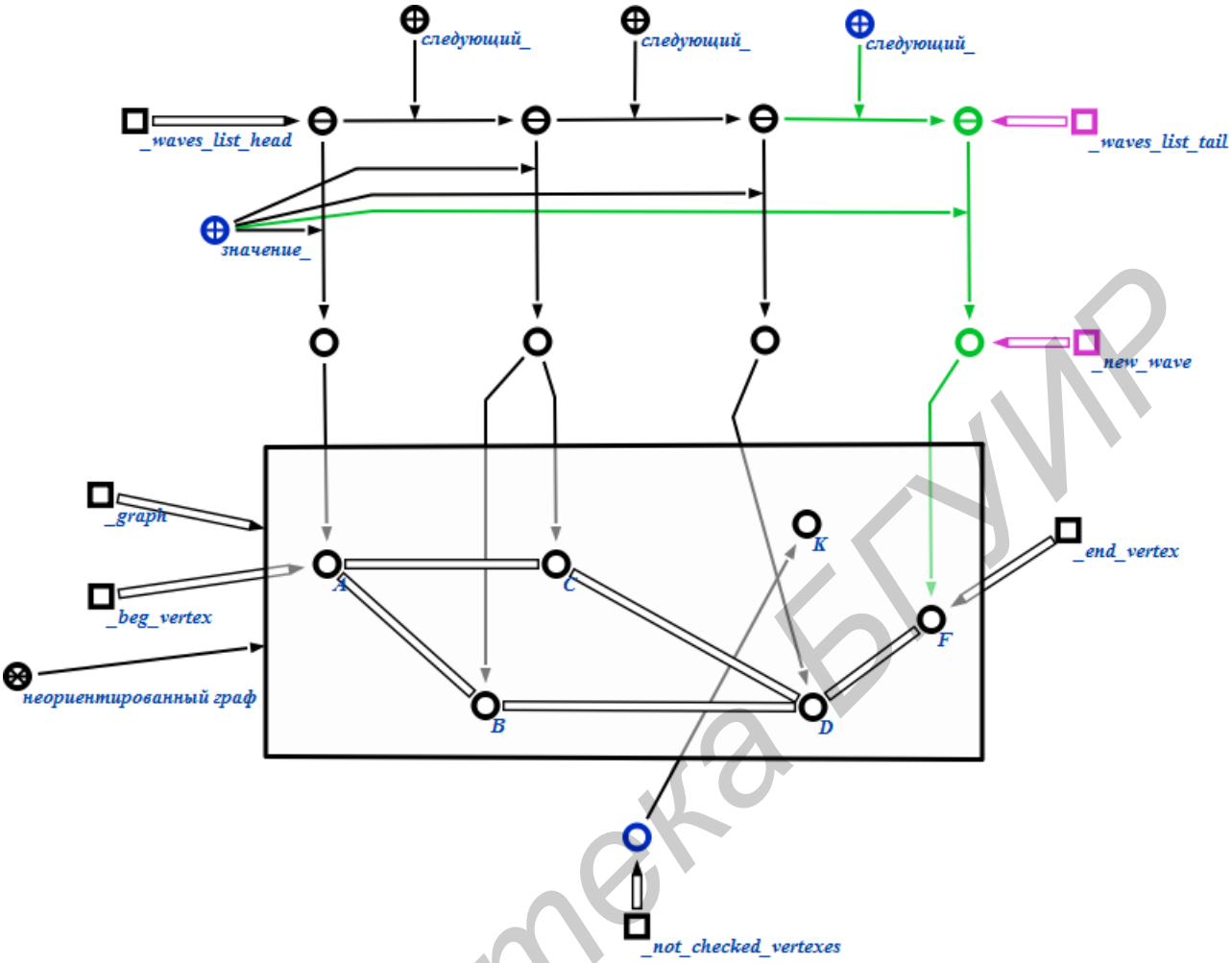


Рисунок 49 – Создание волны, включающей вершину F

Для вершины из предыдущей волны является смежной и входящей во множество проверенных вершин только вершина F. Из нее формируем новую волну. Эта вершина исключается из множества непроверенных вершин (см. значение переменной *_not_checked_vertexes*).

Переменная *_waves_list_tail* получает в качестве значения созданный элемент списка, а переменная *_new_wave* – созданную волну.

Так как эта волна содержит конечную вершину пути F, то на следующих этапах перейдем к генерации одного из найденных минимальных путей.

***Шаг 9. Удаление множества непроверенных вершин*** *(рисунок 50)*

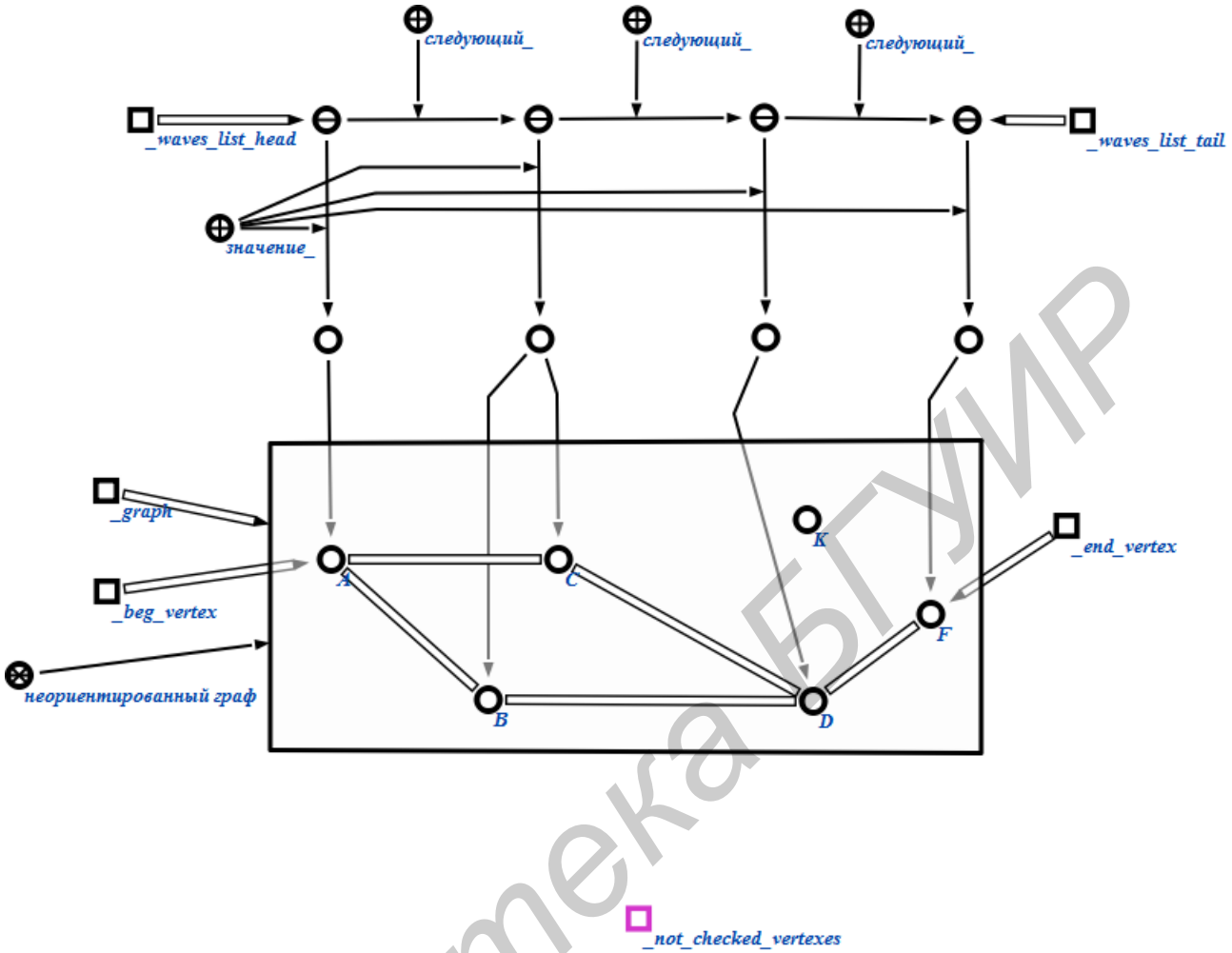


Рисунок 50 – Удаление множества непроверенных вершин

Удаляем значение переменной *_not_checked_vertexes*, так как оно уже не нужно.

***Шаг 10. Создание связки отношения путь**** *(рисунок 51)*

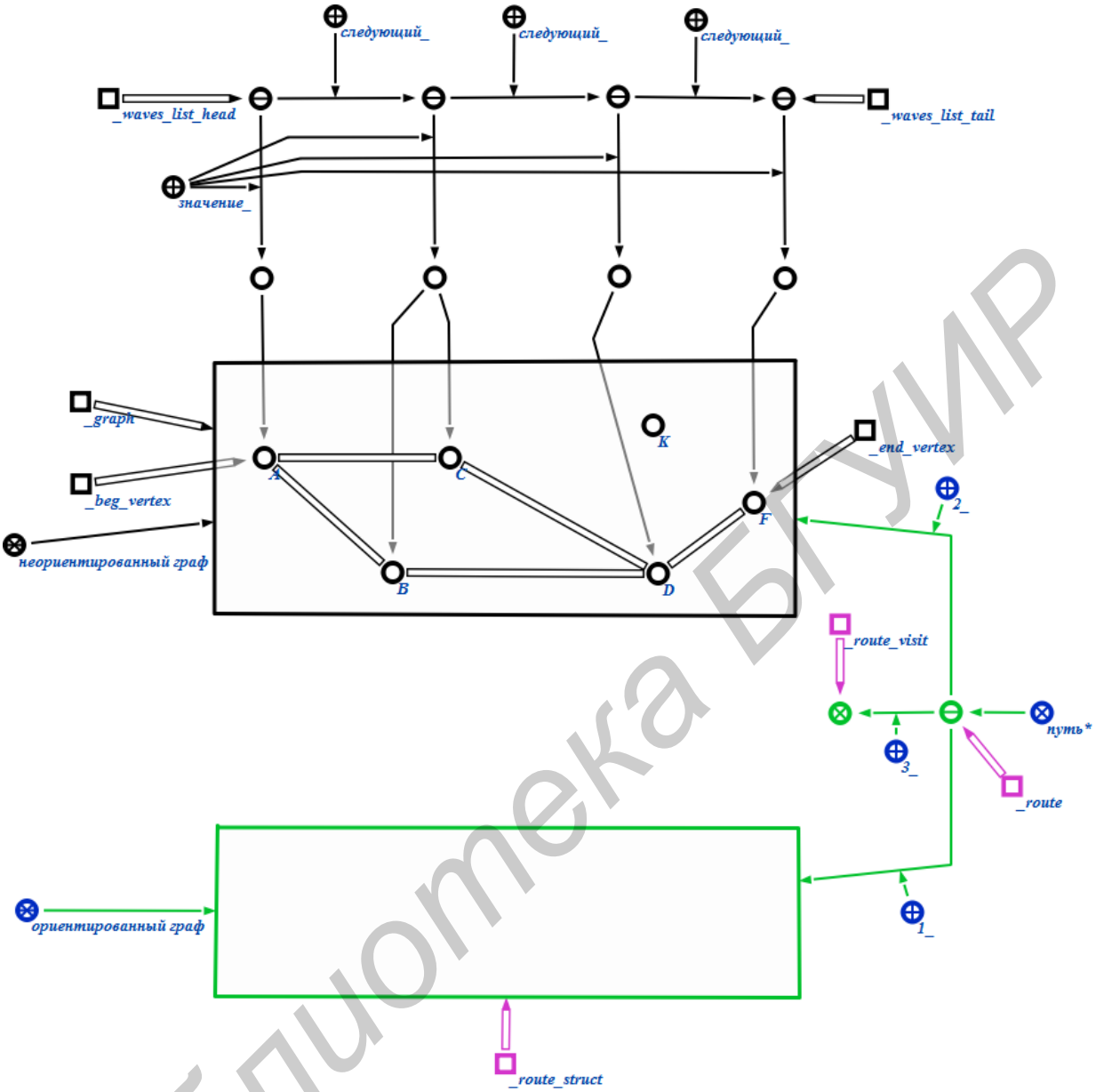


Рисунок 51 – Создание связки отношения *путь**

Начинаем генерацию одного из найденных минимальных путей.

Создадим связку отношения *путь** и установим ее в качестве значения переменной *_route*. Переменная *_route_struct* получит в качестве значения ориентированный граф структуры пути, а переменная *_route_visit* – отношения посещения.

***Шаг 11. Добавление в структуру пути посещения конечной вершины генерируемого пути*** *(рисунок 52)*

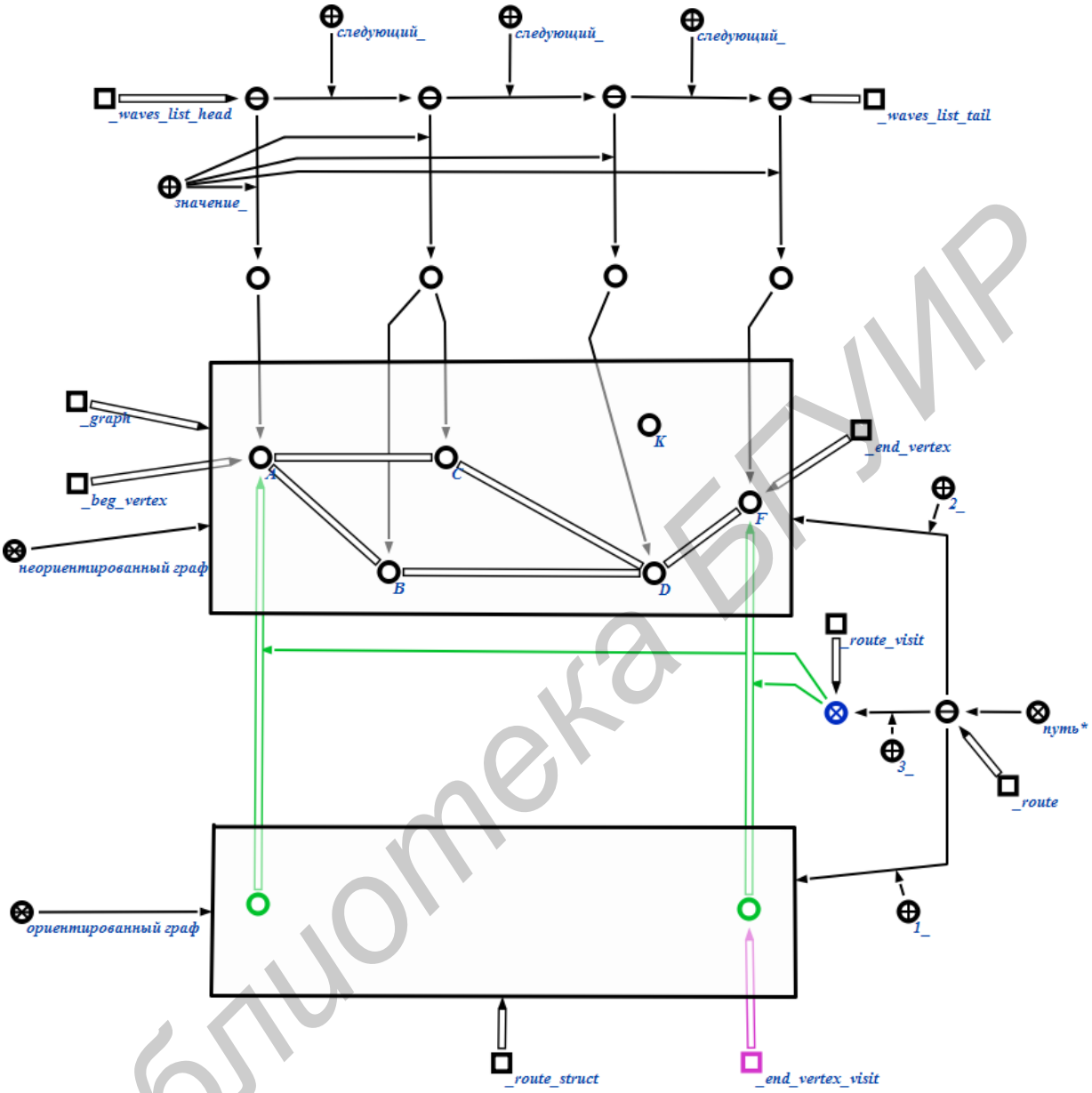


Рисунок 52 – Добавление в структуру пути посещения конечной вершины генерируемого пути

Добавим в структуру генерируемого пути посещение конечной вершины F. Созданное посещение получит в качестве значения переменная *_end_vertex_visit*.

***Шаг 12. Установка переменных для 1-й итерации цикла генерации структуры пути*** *(рисунок 53)*

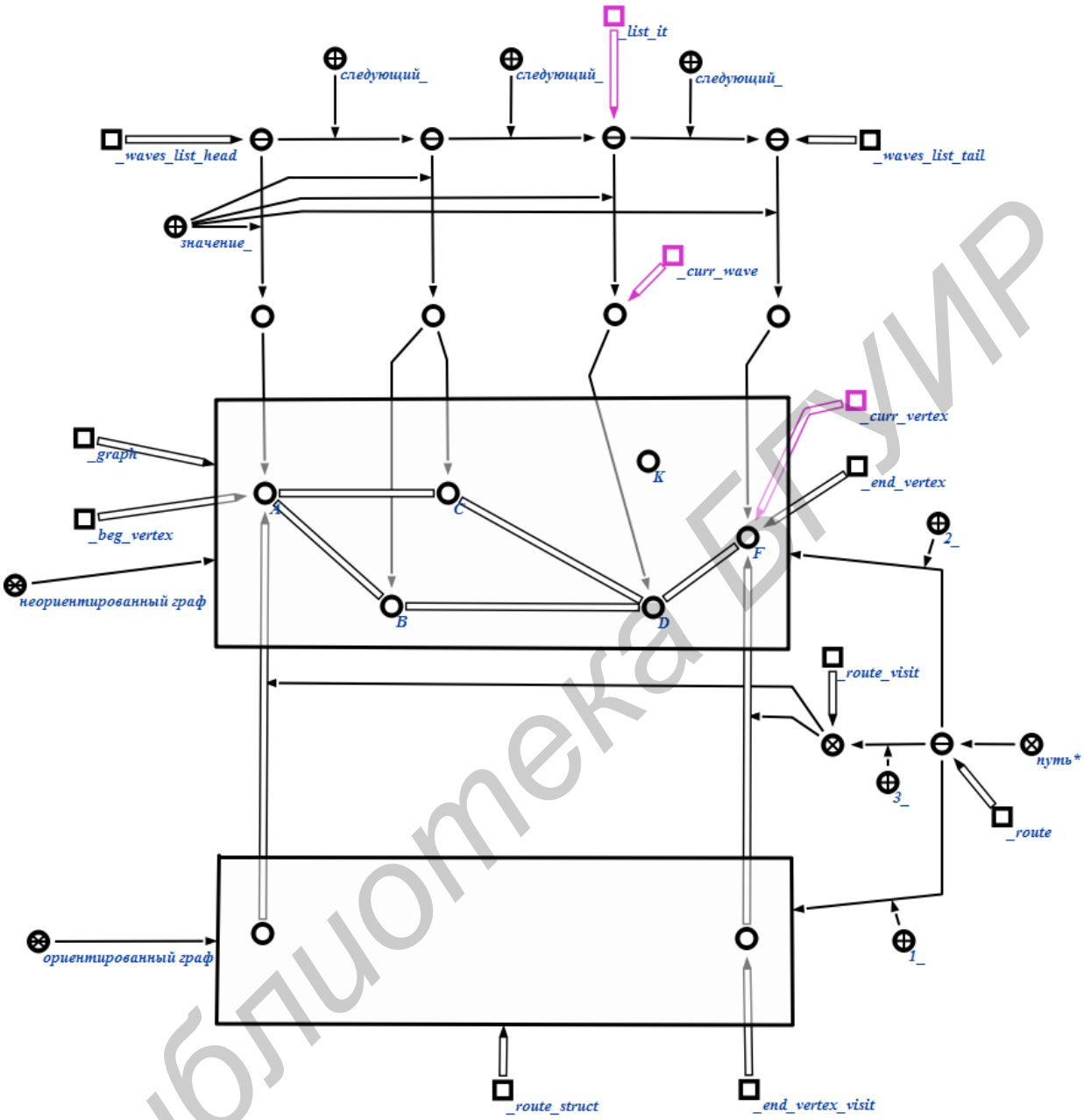


Рисунок 53 – Установка переменных для 1-й итерации цикла генерации структуры пути

Структура пути строится начиная с конечной вершины пути F.

На этом этапе переменная *_curr_vertex* получит в качестве значения текущую обрабатываемую вершину, переменная *_list_it* – текущий обрабатываемый элемент списка волн, переменная *_curr_wave* – текущую обрабатываемую волну.

***Шаг 13. Создание посещения вершины на 1-й итерации цикла генерации структуры пути*** *(рисунок 54)*

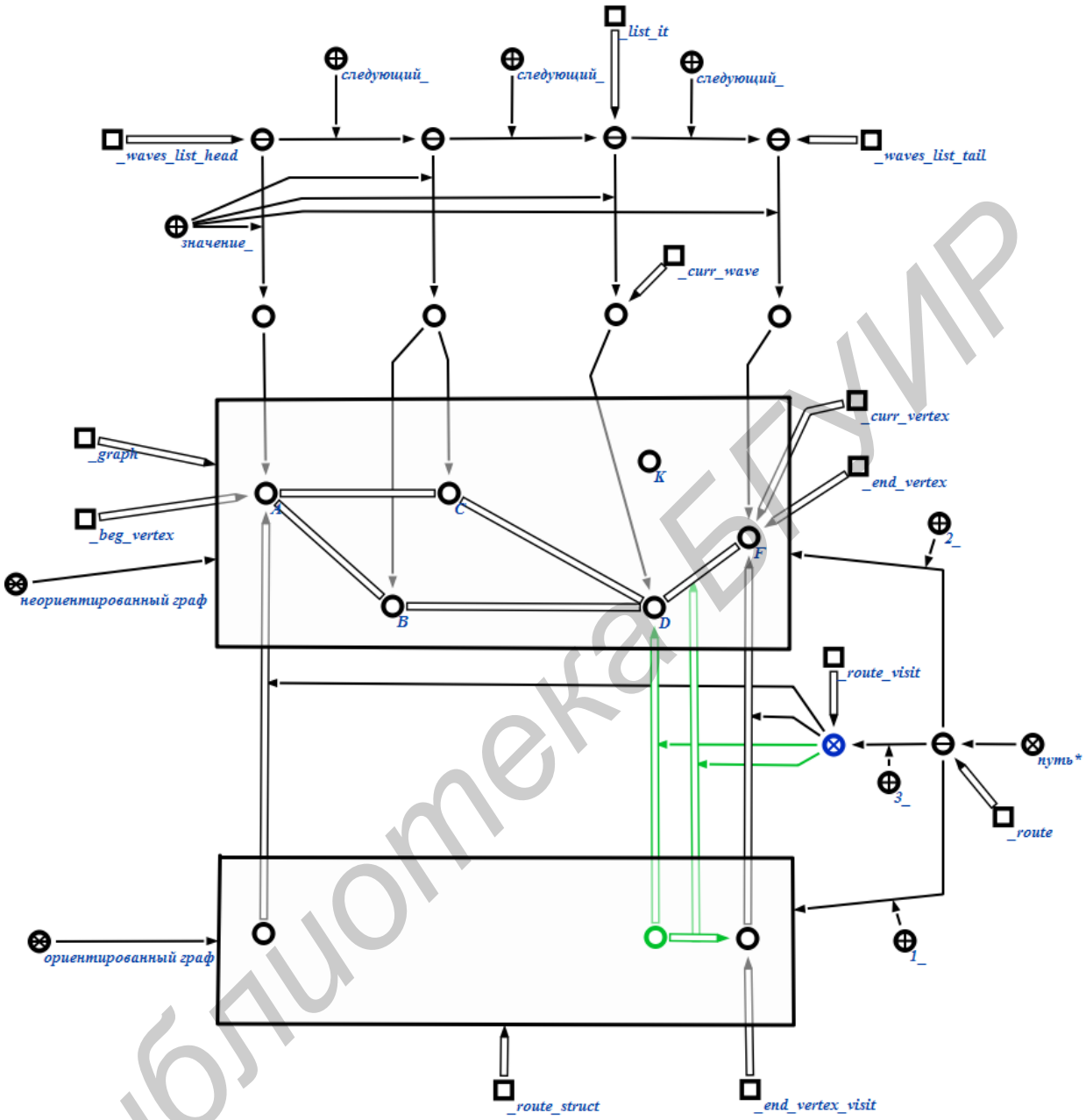


Рисунок 54 – Создание посещения вершины на 1-й итерации цикла генерации структуры пути

Создадим посещение для одной из вершин из волны *_curr_wave*, которая является смежной вершине из переменной *_curr_vertex*. Для связывающего их ребра тоже создадим посещение.

***Шаг 14. Переход ко 2-й итерации цикла генерации структуры пути*** *(рисунок 55)*

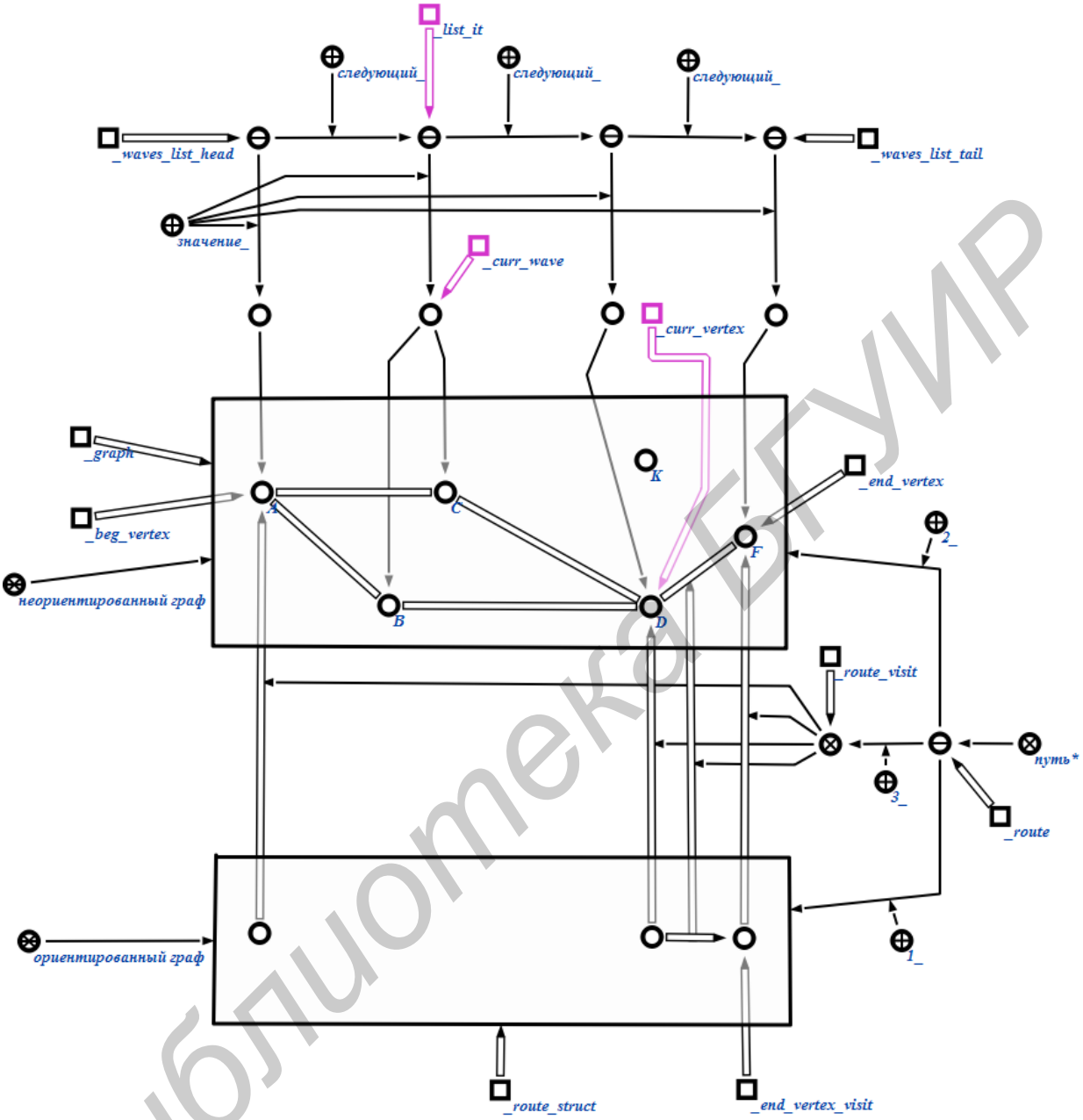


Рисунок 55 – Переход ко 2-й итерации цикла генерации структуры пути

На этом этапе переменная *_curr_vertex* получит в качестве значения вершину, для которой было создано посещение на предыдущей итерации, переменная *_list_it* – предшествующий элемент списка волн для значения переменной *_curr_wave* на предыдущем этапе, переменная *_curr_wave* – обрабатываемую волну для элемента списка из установленного значения *_list_it*.

***Шаг 15. Создание посещения вершины на 2-й итерации цикла генерации структуры пути*** *(рисунок 56)*

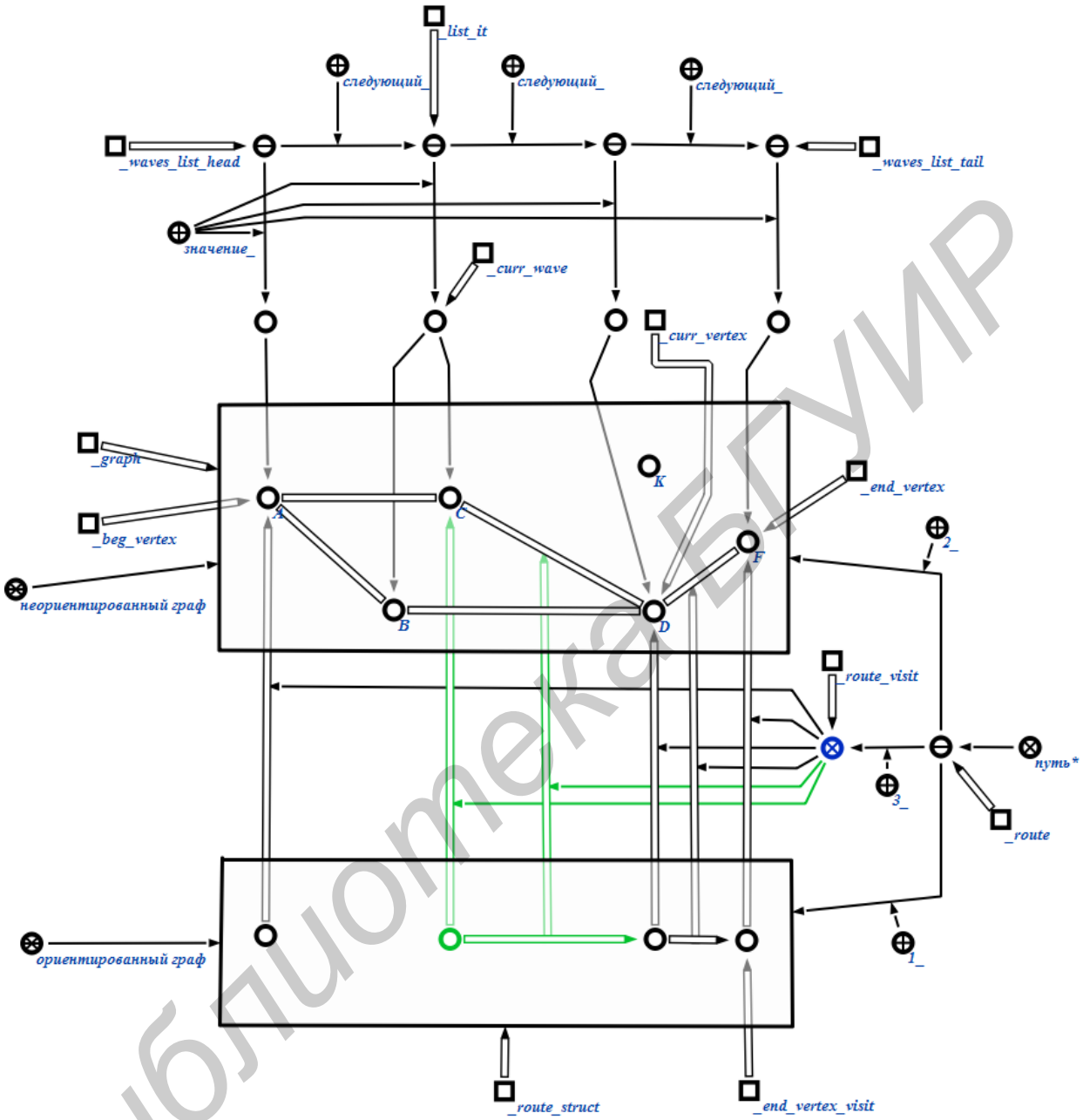


Рисунок 56 – Создание посещения вершины на 2-й итерации цикла генерации структуры пути

Создадим посещение для одной из вершин из волны *_curr_wave*, которая является смежной вершине из переменной *_curr_vertex*. На эту роль была выбрана вершина C. Для ребра, связывающего две вершины, тоже создадим посещение.

***Шаг 16. Переход к 3-й итерации цикла генерации структуры пути*** *(рисунок 57)*

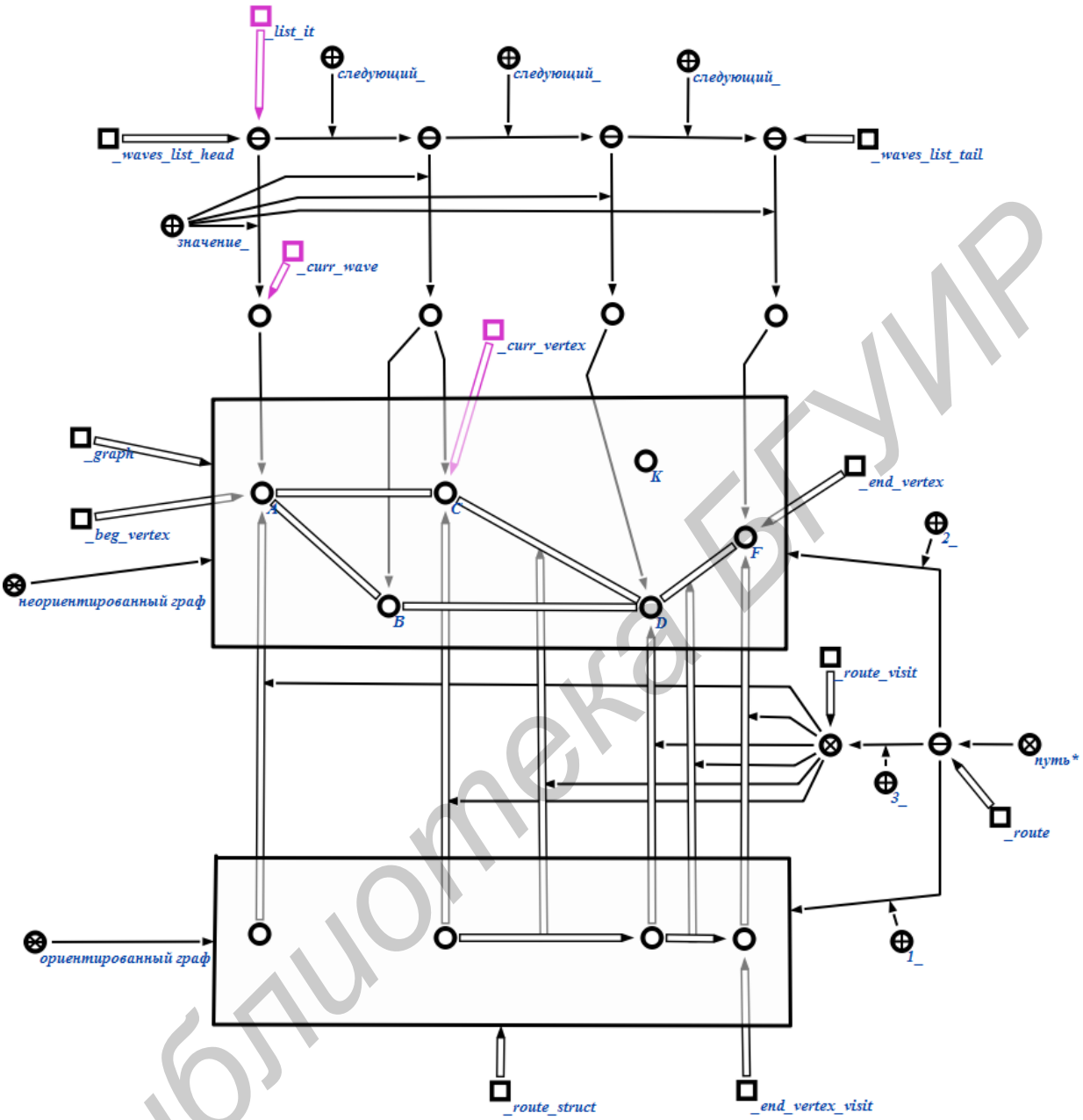


Рисунок 57 – Переход к 3-й итерации цикла генерации структуры пути

На этом этапе переменная *_curr_vertex* получит в качестве значения вершину, для которой было создано посещение на предыдущей итерации, переменная *_list_it* – предшествующий элемент списка волн для значения переменной *_curr_wave* на предыдущем этапе, переменная *_curr_wave* – обрабатываемую волну для элемента списка из установленного значения *_list_it*.

***Шаг 17. Создание посещения вершины на 3-й итерации цикла генерации структуры пути*** *(рисунок 58)*

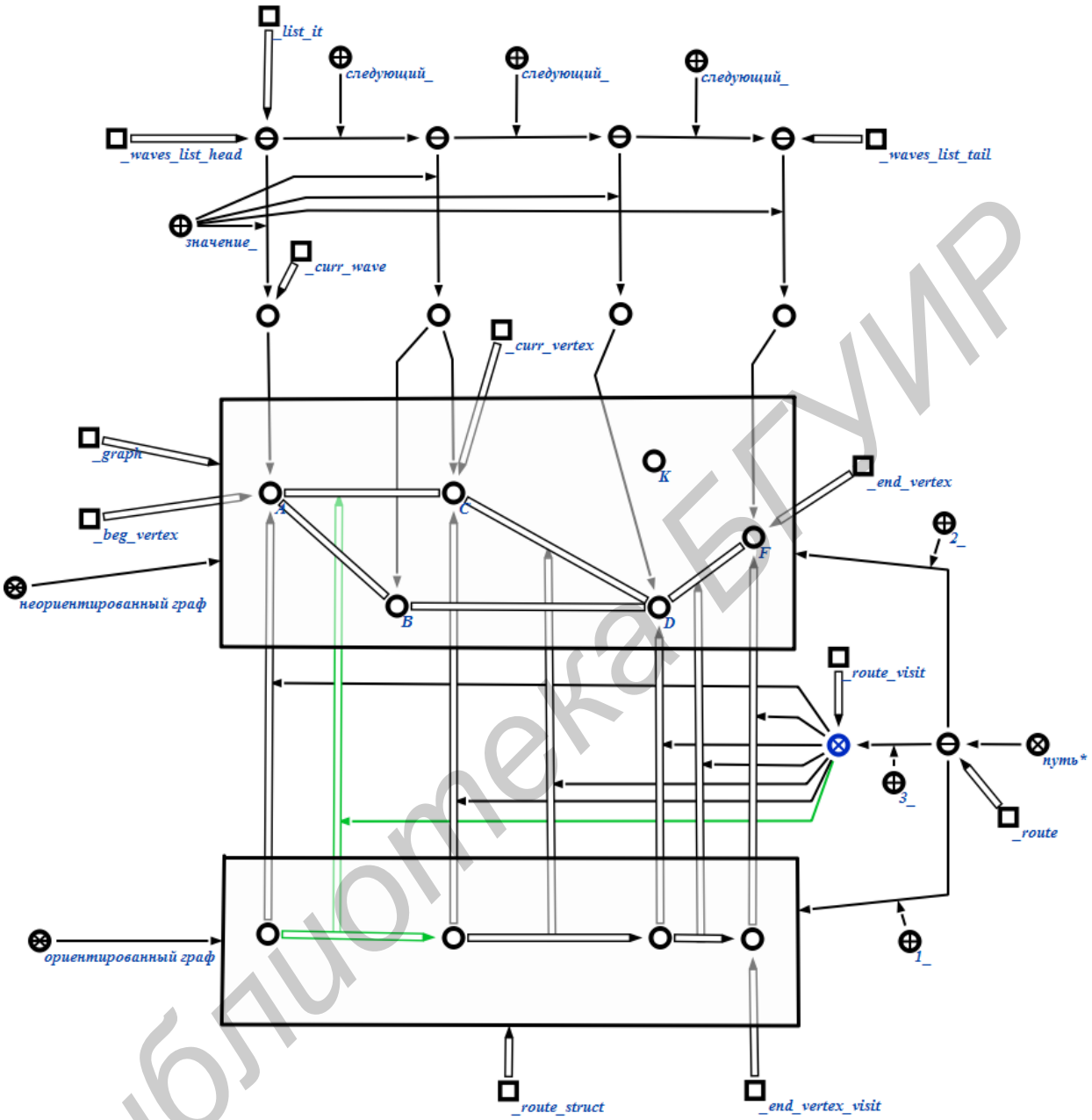


Рисунок 58 – Создание посещения вершины на 3-й итерации цикла генерации структуры пути

Так как для вершины A уже создано посещение, то делать это повторно алгоритм не будет. А вот посещение ребра между вершинами A и C необходимо создать.

Структура пути создана, поэтому завершаем цикл и переходим к очистке sc-памяти от уже ненужных sc-конструкций.

***Шаг 18. Удаление списка волн*** *(рисунок 59)*

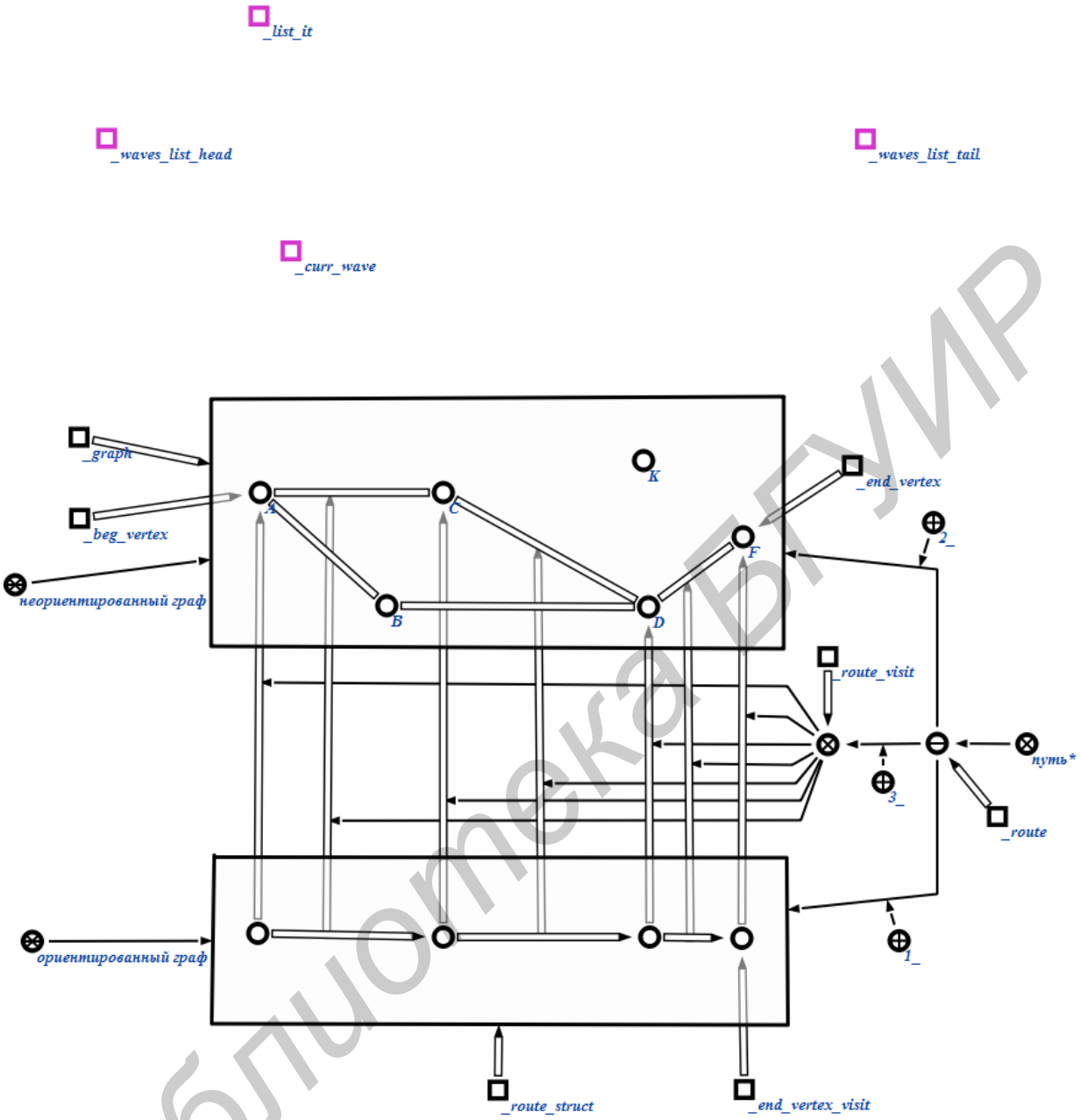


Рисунок 59 – Удаление списка волн

Удалим список волн. Переменные *_list_it*, *_waves_list_head*, *_curr_wave*, *_waves_list_tail* окажутся без значений.

***Шаг 19. Результат работы алгоритма*** *(рисунок 60)*

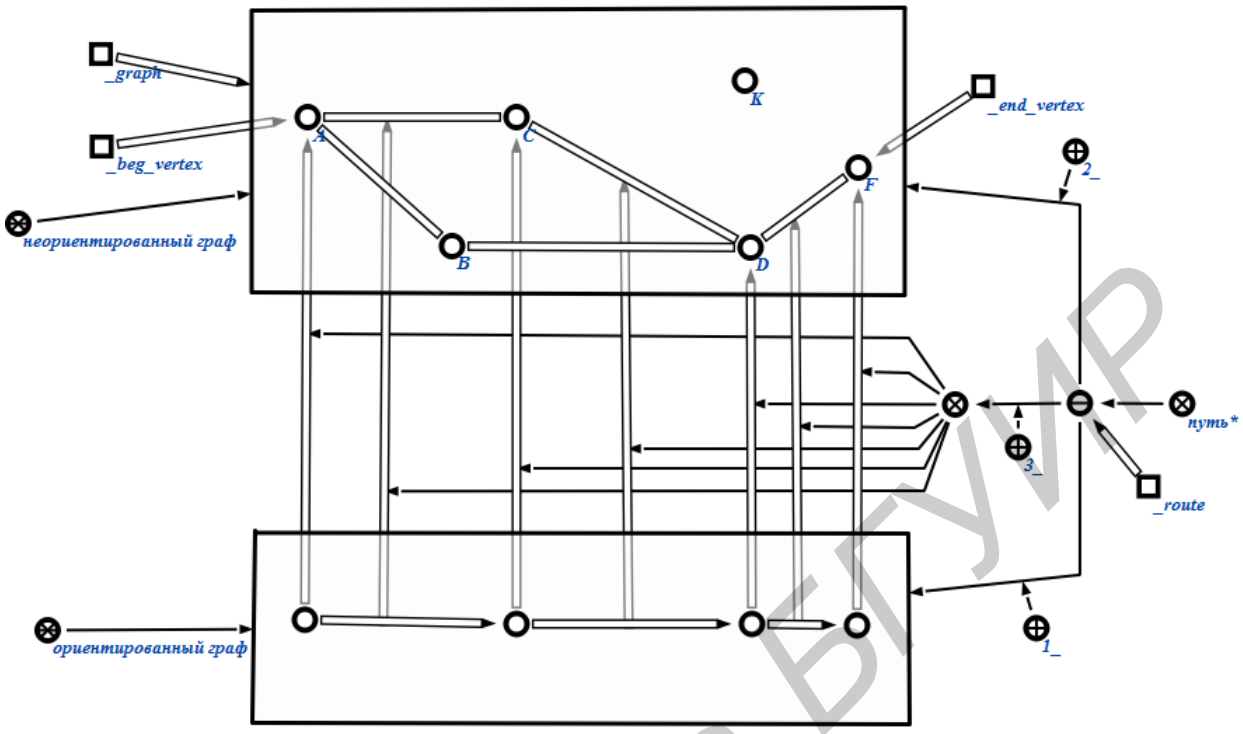


Рисунок 60 – Результаты работы алгоритма

На данном этапе продемонстрирован результат работы алгоритма, значение переменной *_route* будет возвращено в вызывающий контекст.

### 5.2 Варианты индивидуальных заданий

Варианты индивидуальных заданий к занятию 5 аналогичны перечисленным в занятии 2 (подраздел 2.2).

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Кузнецов, О. П. Дискретная математика для инженера / О. П. Кузнецов, Г. М. Адельсон-Вельский. – М. : Энергоиздат, 1988.

2 Представление и обработка знаний в графодинамических ассоциативных машинах / В. В. Голенков [и др.]; под ред. В. В. Голенкова. – Минск : БГУИР, 2001.

3 Семантическая технология проектирования интеллектуальных решателей задач на основе агентно-ориентированного подхода / В. В. Голенков, Д. В. Шункевич, И. Т. Давыденко // Программные системы и вычислительные методы. – 2013. – №1.

4 Базы знаний интеллектуальных систем : учебник / Т. А. Гаврилова [и др.]. – СПб. : Питер, 2001.

5 Шункевич, Д. В. Принципы построения машин обработки знаний интеллектуальных систем на основе семантических сетей / Д. В. Шункевич // Электроника-инфо. – 2014. – №3.

6 Давыденко, И. Т. Семантическая модель и средства компонентного проектирования баз знаний на основе унифицированных семантических сетей: Электроника-инфо. – 2013. – №11.

7 Давыденко, И. Т. Технология компонентного проектирования баз знаний на основе унифицированных семантических сетей / И. Т. Давыденко // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS–2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февраля 2013 г. – Минск : БГУИР, 2013.

8 Шункевич, Д. В. Модели и средства компонентного проектирования машин обработки знаний на основе семантических сетей / Д. В. Шункевич // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS–2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февраля 2013 г. – Минск : БГУИР, 2013.

Св. план 2015, поз. 27

*Учебное издание*

**Голенков** Владимир Васильевич
**Гулякина** Наталья Анатольевна
**Давыденко** Ирина Тимофеевна
**Шункевич** Даниил Вячеславович

# *ТРАДИЦИОННЫЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ПОСОБИЕ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ*

ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 15.06.2016. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,95. Уч.-изд. л. 4,0. Тираж 100 экз. Заказ. 151.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6