Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

К защите допустить:
Заведующий кафедрой
_____ В. В. Голенков

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по дисциплине «Проектирование программ в интеллектуальных системах»:

Алгоритм вычисления окружения орграфа

БГУИР КРЗ 1–40 03 01 01 32 ПЗ

Студент: Е. А. Македон

Руководитель: Д. В. Шункевич

Минск 2016

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

В курсовой работе используются следующие условные обозначения:
БЗ — база знаний;
ИСС — интеллектуальная справочная система;
SC — Semantic Code;
SCg —Semantic Code Graphical;
SCn — Semantic Code Natural;
SCp — Semantic Code Programm;

ВВЕДЕНИЕ

Целью выполнения курсовой работы по дисциплине "Проектирование программ в интеллектуальных системах"является закрепление навыков:

– разработки программных проектов и фрагментов проектов с применением современных методик и инструментальных средств;

– применения методов из области теории графов для решения прикладных задач в различных предметных областях;

– использования инструментальных средств разработки интеллектуальных систем

– создания программ, ориентированных на обработку различных видов знаний, хранимых в памяти интеллектуальных систем.

Задачей курсовой работы является разработка алгоритма вычисления окружения орграфа и реализация данного алгоритма на графовом языке программирования SCP.

1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Язык SCP

Язык SCP – это графовый язык процедурного программирования, предназначенный для эффективной обработки однородных семантических сетей с теоретико-множественной интерпретацией, закодированных с помощью SC-кода. Язык SCP является языком параллельного асинхронного программирования.

Языком представления данных для текстов языка SCP (scp-программ) является SC-код и, соответственно, любые варианты его представления. Язык SCP сам построен на основе SC-кода, вследствие чего scp-программы сами по себе могут входить в состав данных scp-программ, в т.ч. по отношению к самим себе. Таким образом, язык SCP предоставляет возможность построения реконфигурируемых программ. Однако для обеспечения возможности реконфигурирования программы непосредственно в процессе ее интерпретации необходимо на уровне интерпретатора языка SCP (scp-интерпретатора) обеспечить уникальность каждой исполняемой копии исходной программы.

Язык SCP рассматривается как ассемблер для графодинамического компьютера, ориентированного на хранение и обработку семантических сетей.

1.2 Понятие графа и ориентированного графа

Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратных связок, т.е. связок у которых первый и второй компоненты совпадают. Представление графа показано на рисунке 1.1.

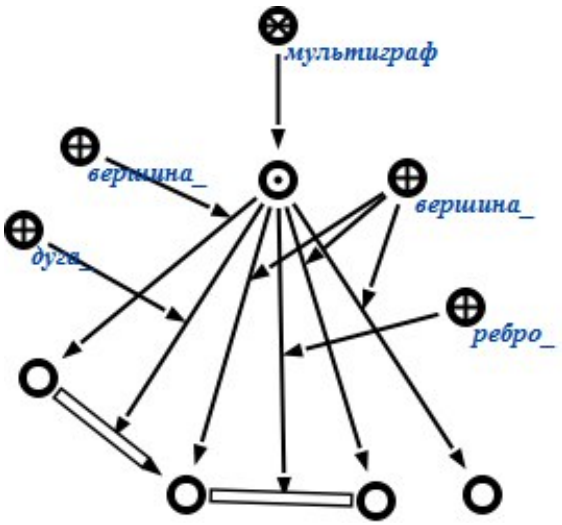

Рисунок *1.1* – Граф.

Ориентированный граф (абсолютное понятие) - это такой граф, в котором все связки являются дугами. Представление ориентированного графа показано на рисунке 1.2.

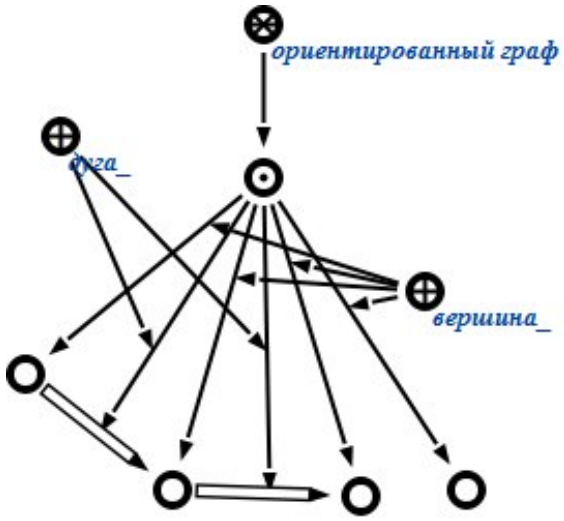

Рисунок *1.2* – Ориентированный граф.

Окружение орграфа – это длина самого длинного простого цикла в ориентированном графе.

Цикл - это цепь, в которой первая и последняя вершины совпадают. При этом длиной цикла называется число составляющих его рёбер. Цикл называется простым, если ребра в нём не повторяются.

2 ТЕСТОВЫЕ ПРИМЕРЫ

2.1 Тест 1

Вход:

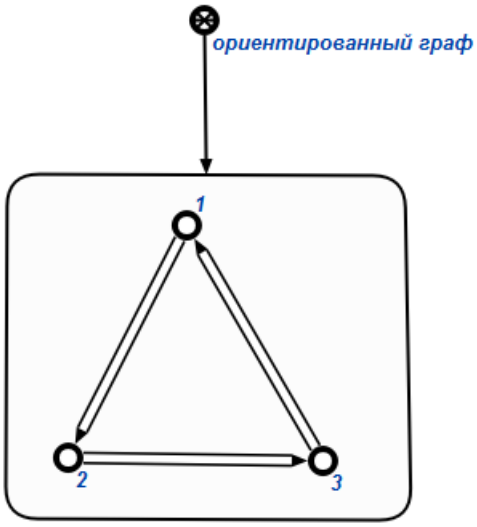

Рисунок *2.1* – Ориентированный граф на входе. Тест 1.

Выход:

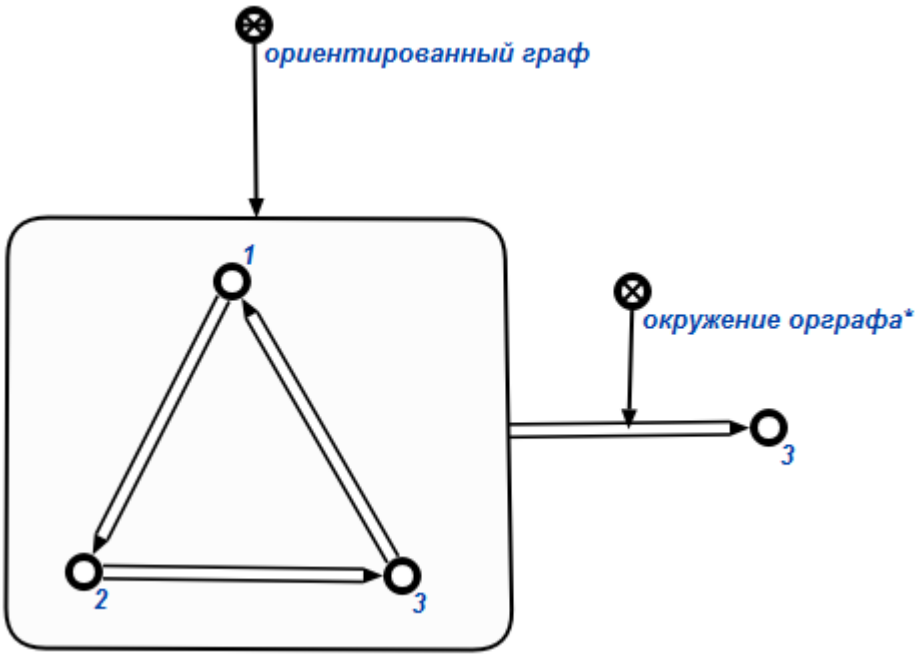

Рисунок *2.2* – Ориентированный граф на выходе. Тест 1.

2.2 Тест 2

Вход:

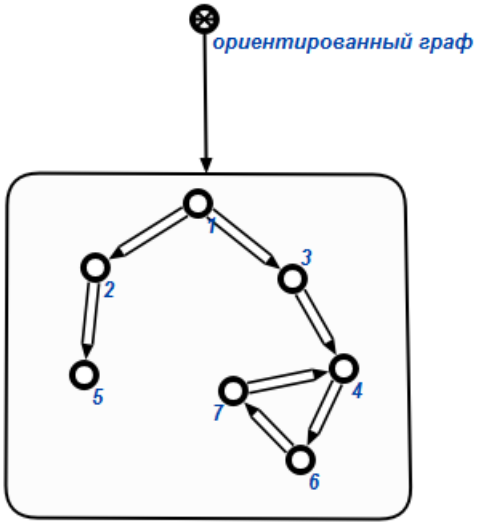

Рисунок *2.3* – Ориентированный граф на входе. Тест 2.

Выход:

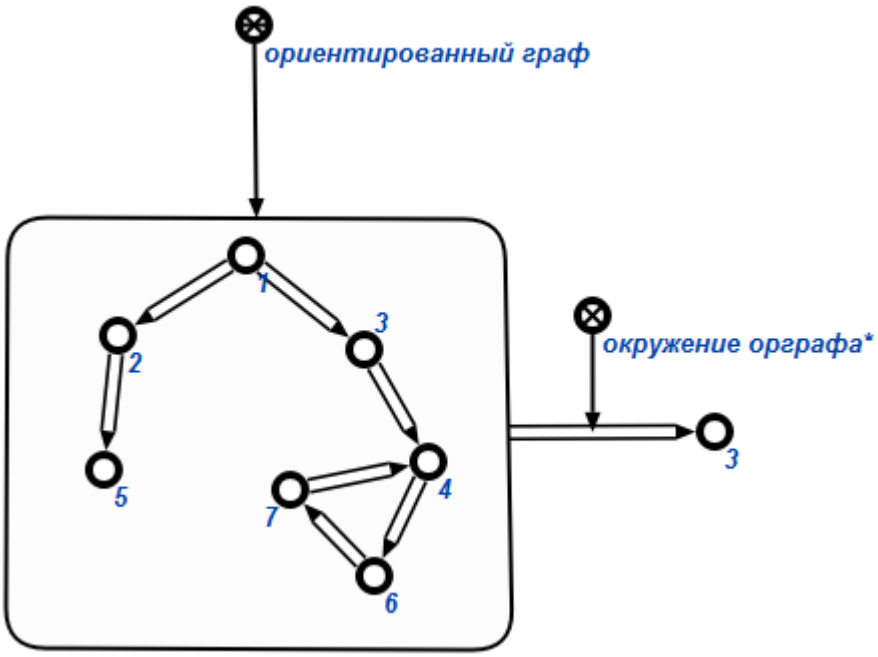

Рисунок *2.4* – Ориентированный граф на выходе. Тест 2.

2.3 Тест 3

Вход:

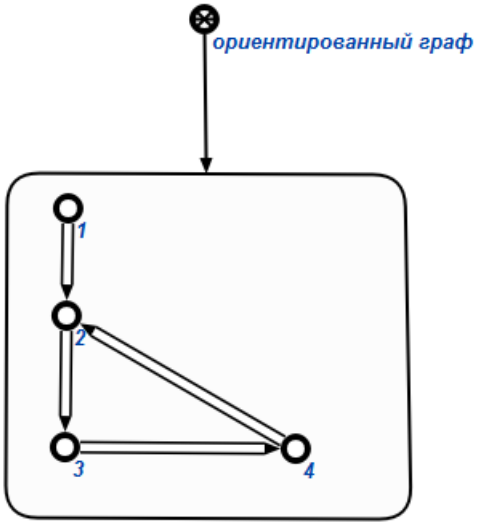

Рисунок *2.5* – Ориентированный граф на входе. Тест 3.

Выход:

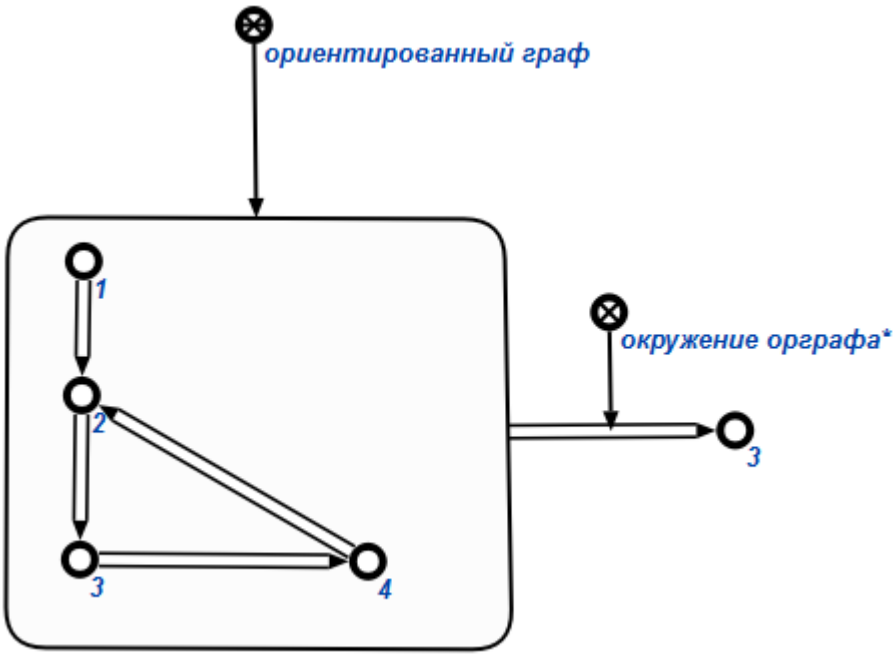

Рисунок *2.6* – Ориентированный граф на выходе. Тест 3.

2.4 Тест 4

Вход:

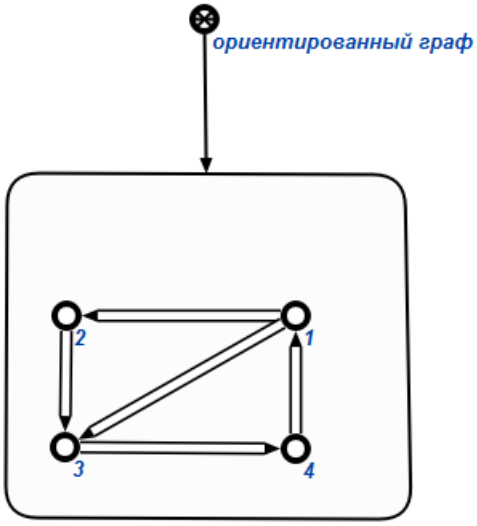

Рисунок *2.7* – Ориентированный граф на входе. Тест 4.

Выход:

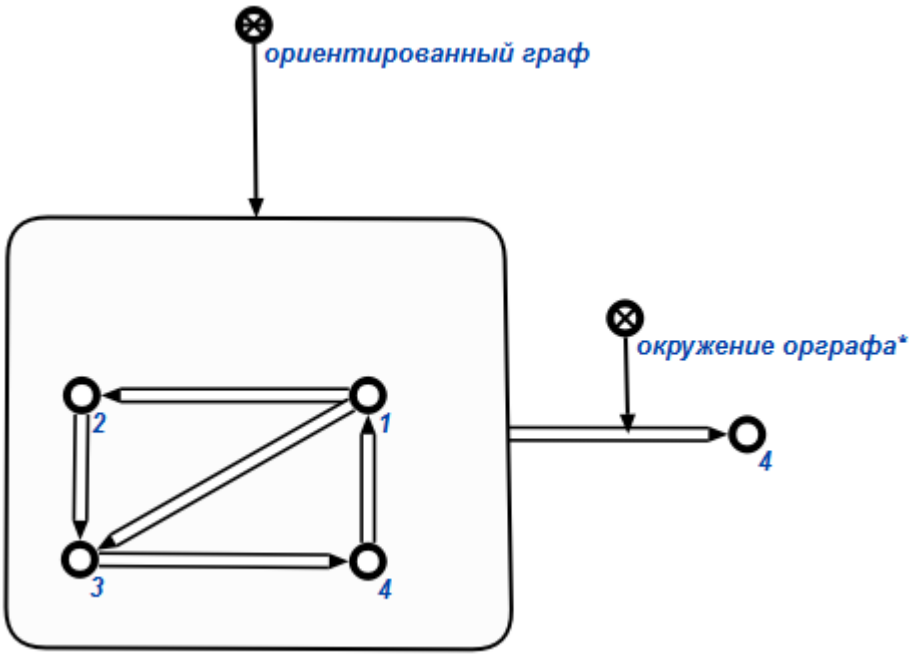

Рисунок *2.8* – Ориентированный граф на выходе. Тест 4.

2.5 Тест 5

Вход:

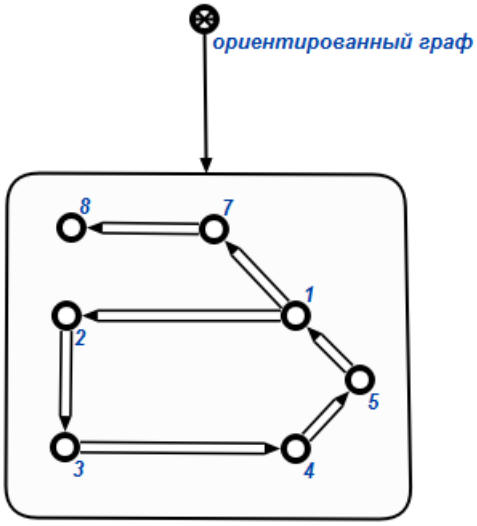

Рисунок *2.9* – Ориентированный граф на входе. Тест 5.

Выход:

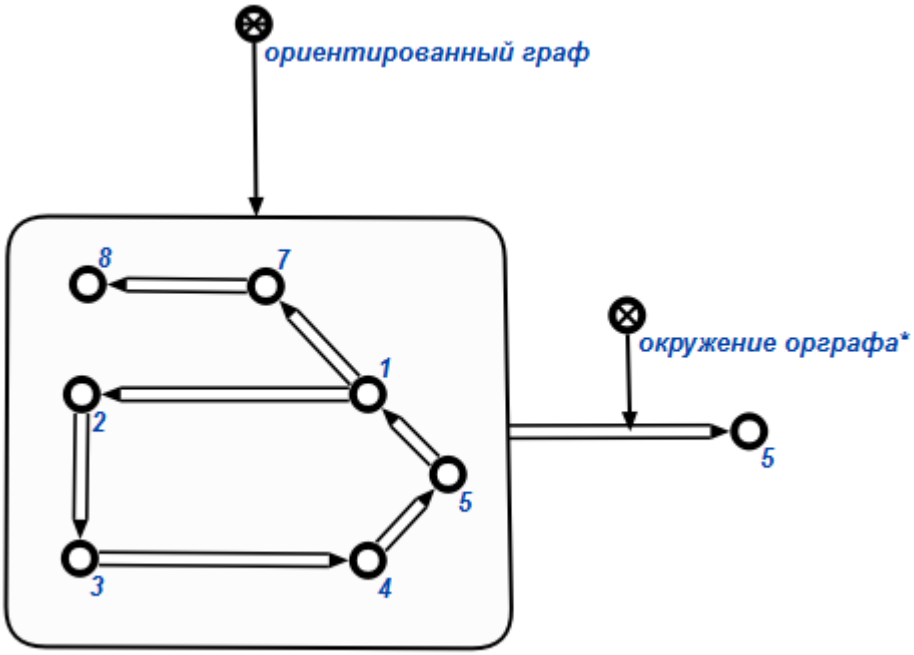

Рисунок *2.10* – Ориентированный граф на выходе. Тест 5.

3 АЛГОРИТМ ВЫЧИСЛЕНИЯ ОКРУЖЕНИЯ ОРГРАФА

3.1 Описание алгоритма

1) Создание переменной _graph и добавление ориентированного графа в переменную _graph.
2) Создание переменных _printed_vertexes (для хранения пройденных вершин), _visit (для хранения посещенных вершин), _max_cycle (для хранения длины максимального цикла), _kol (для подсчета количества элементов), _rrel_parent (для хранения дуги из текущей вершины на родителя).
3) Присвоение переменной _kol значение 0, а переменной _max_cycle значение -1.
4) Создание переменных _cycle_st (для хранения вершины начала цикла), _cycle_end (для хранения вершины конца цикла).
5) Берем вершину графа, не принадлежащую _printed_vertexes.
6) Если такая вершина не найдена, переходим к пункту (29).
7) Если такая вершина была найдена, переходим к пункту (8).
8) Добавляем вершину в переменную _cycle_st.
9) Добавляем вершину в переменную _visit.
10) Устанавливаем дугу на предыдущую вершину и добавляем ее в переменную _rrel_parent.
11) Находим вершину, смежную с текущей.
12) Если вершина не найдена, переходим к пункту (27).
13) Если вершина принадлежит _visit, переходим к пункту (15).
14) Если вершина не принадлежит _visit, переходим к пункту (9).
15) Возвращаемся к предыдущей вершине.
16) Добавляем вершину в переменную _cycle_end.
17) Увеличиваем значение переменной _kol на 1.
18) Переходим по дуге, принадлежащей переменной _rrel_parent к следующей вершине.
19) Удаляем старую вершину из переменной _cycle_end и добавляем новую вершину в переменную _cycle_end.
20) Увеличиваем значение переменной _kol на 1.
21) Если вершина не принадлежит _cycle_end и _cycle_st, переходим к пункту (18).
22) Если вершина принадлежит _cycle_end и _cycle_st, переходим к пункту (23).
23) Сравнение _max_cycle и _kol.
24) Если _max_cycle меньше _kol, изменяем значение _max_cycle на

значение переменной _kol и переходим к пункту (26).
25) Иначе переходим к пункту (26).
26) Изменяем значение переменной _kol на 0.
27) Удаляем все элементы из переменных _rrel_parent, _cycle_st, _cycle_-end, _visit.
28) Исходную вершину добавляем в переменную _printed_vertexes и переходим к пункту (5).
29) Конечный ответ содержится в переменной _max_cycle.
30) Завершение алгоритма.

3.2 Переменные программы

1) _graf – переменная, принимающая значение sc-узла графа.
2) _printed_vertexes – переменная, принимающая значение множества всех пройденных вершин.
3) _visit – переменная, принимающая значение множества всех посещенных вершин.
4) _max_cycle - переменная для хранения значения длины максимального цикла.
5) _kol – переменная, принимающая значение кол-ва пройденных вершин.
6) _rrel_parent - переменная, для хранения дуги из текущей вершины на родителя.
7) _cycle_st - переменная, принимающая вершину начала цикла.
8) _cycle_end - переменная, принимающая вершину конца цикла.

3.3 Пример программы

1.

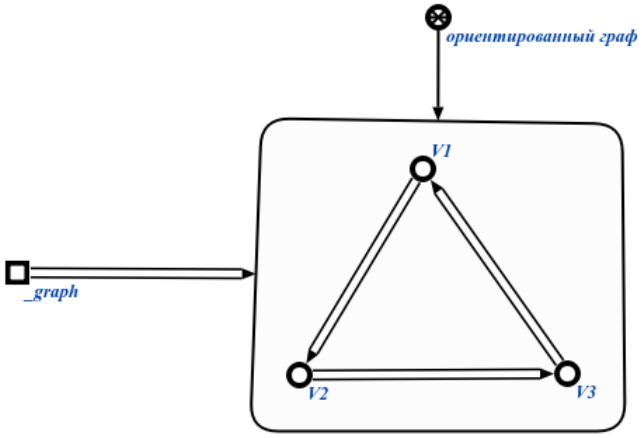

Рисунок *3.1* – Ориентированный граф на входе в программу

Запись в переменную _graph входного орграфа.

2.

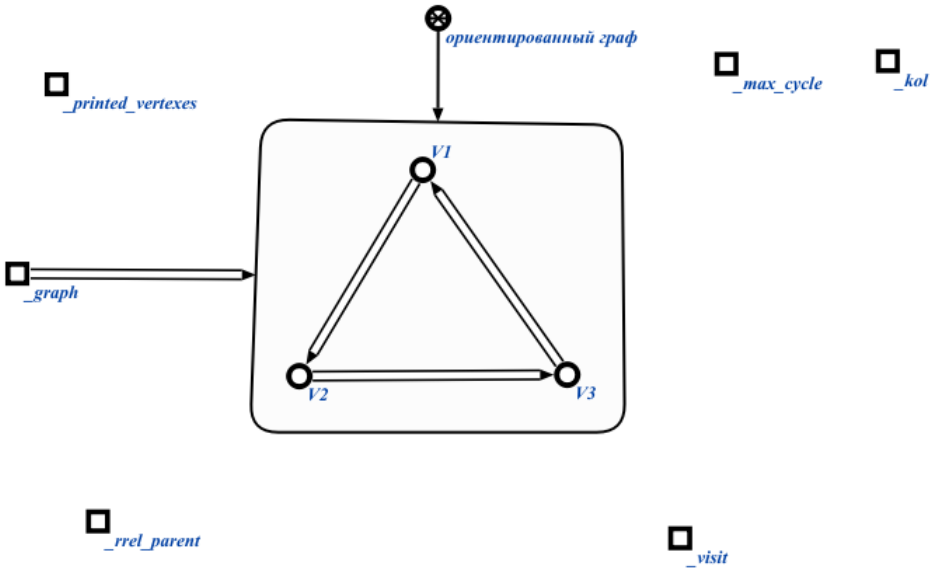

Рисунок *3.2* – Создание переменных

Создание переменных _printed_vertexes, _visit, _max_cycle, _kol, _rrel_parent.

3.

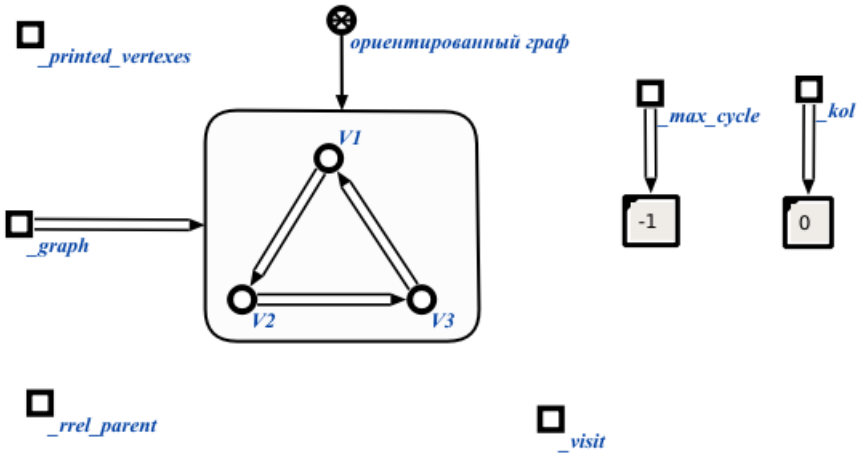

Рисунок *3.3* – Присвоение значений

Присвоение переменной _kol значение 0, а переменной _max_cycle значение -1.

4.

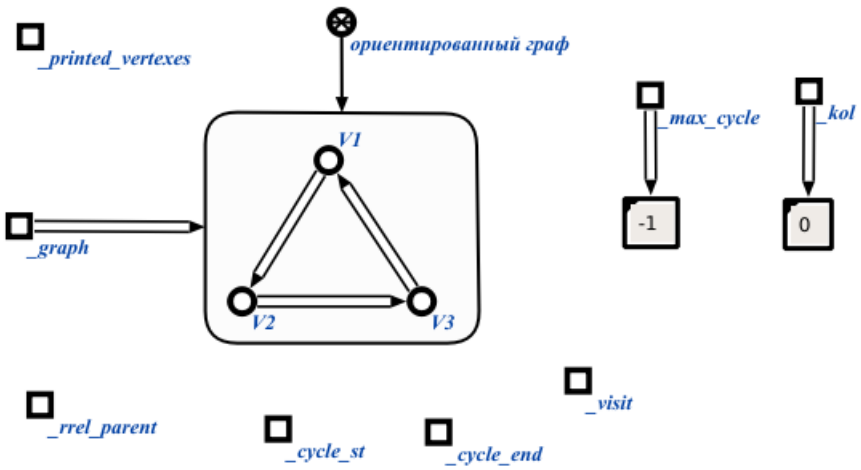

Рисунок *3.4* – Создание переменных

Создание переменных _cycle_st, _cycle_end.

5.

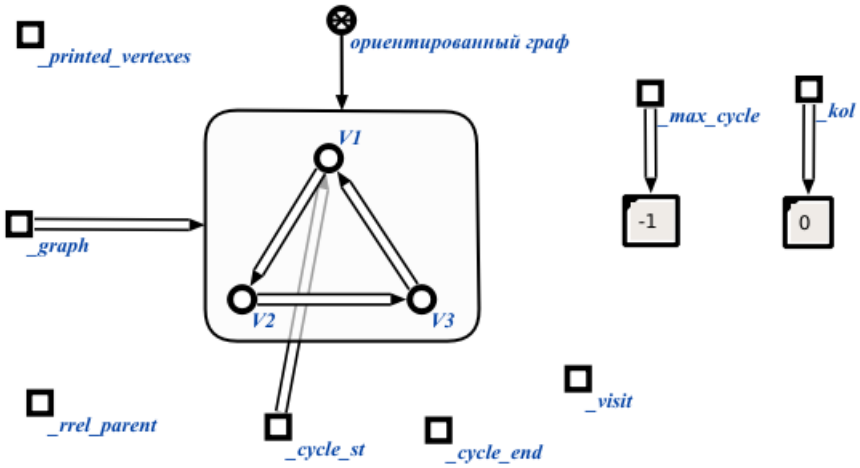

Рисунок *3.5* – Обход графа (шаг 1)

Добавление вершины V1 в переменную _cycle_st.

6.

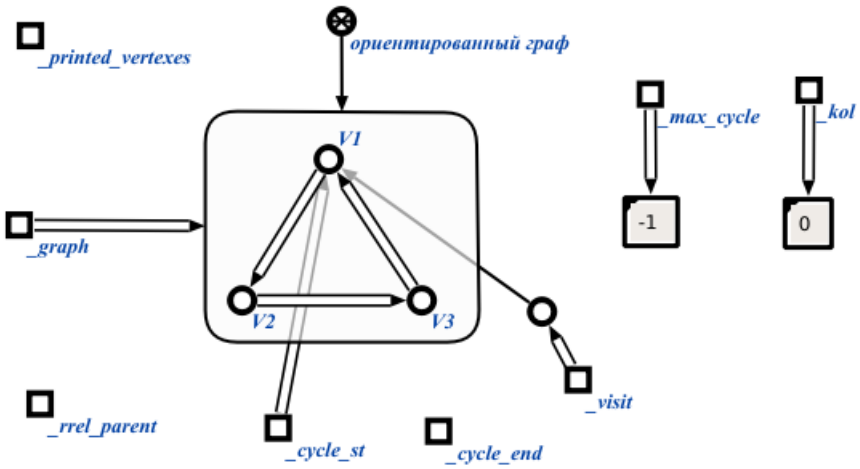

Рисунок *3.6* – Обход графа (шаг 2)

Добавление вершины V1 в переменную _visit.

7.

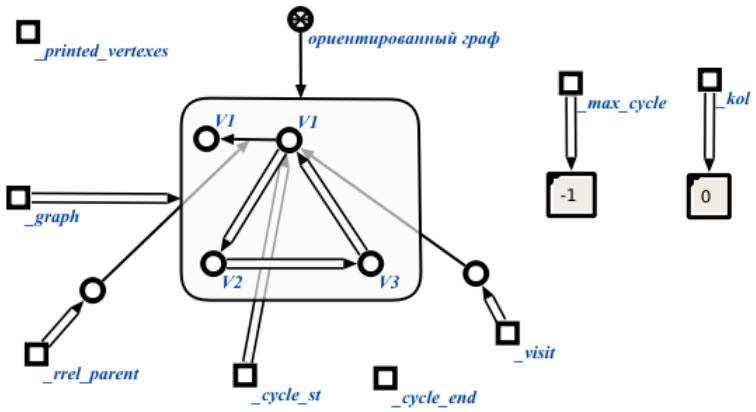

Рисунок *3.7* – Обход графа (шаг 3)

Создание дуги из вершины V1 в вершину V1 и добавление этой дуги в переменную _rrel_parent.

8.

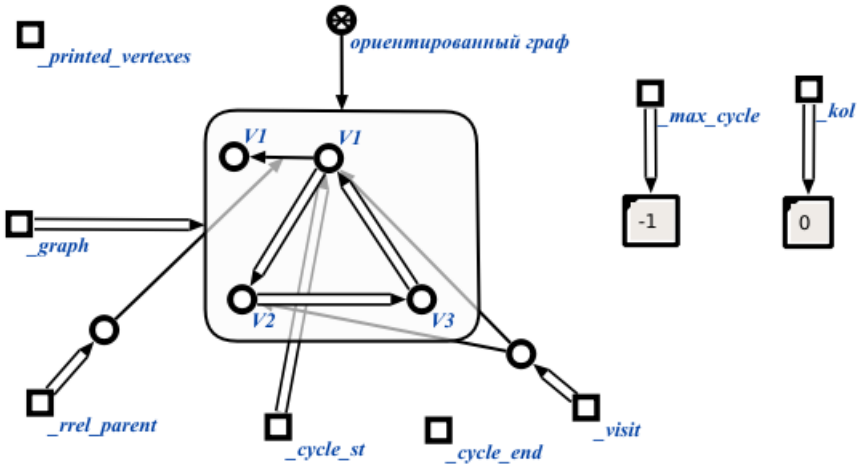

Рисунок *3.8* – Обход графа (шаг 4)

Добавление вершины V2 в переменную _visit.

9.

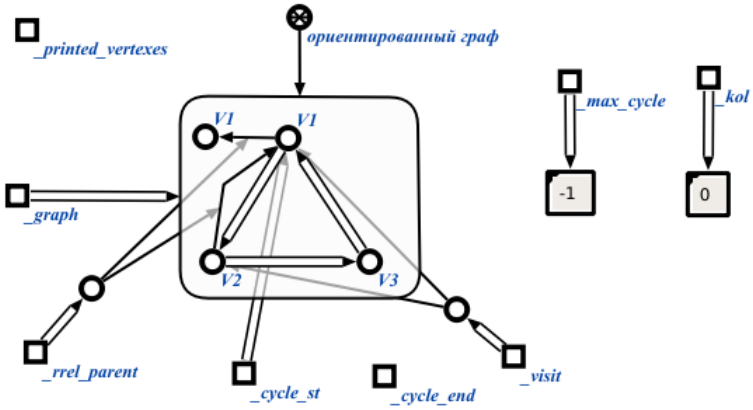

Рисунок *3.9* – Обход графа (шаг 5)

Создание дуги из вершины V2 в вершину V1 и добавление этой дуги в переменную _rrel_parent.

10.

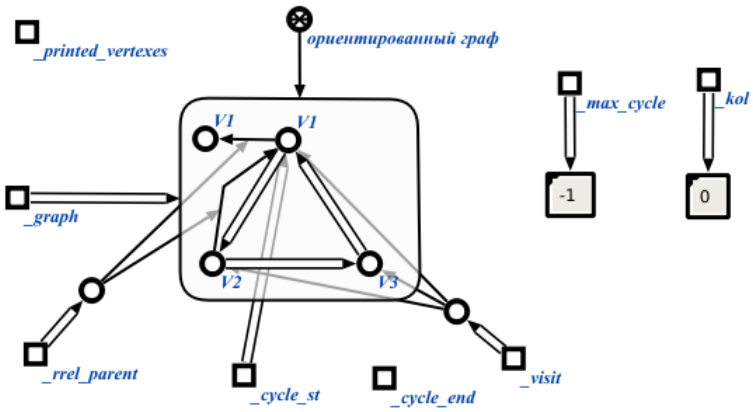

Рисунок *3.10* – Обход графа (шаг 6)

Добавление вершины V3 в переменную _visit.

11.

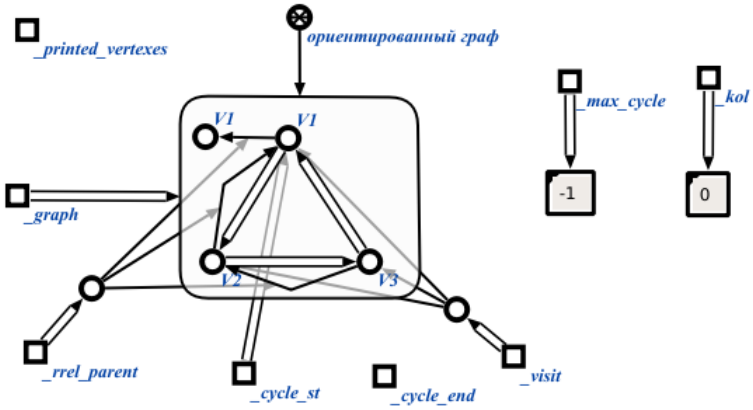

Рисунок *3.11* – Обход графа (шаг 7)

Создание дуги из вершины V3 в вершину V2 и добавление этой дуги в переменную _rrel_parent.

12.

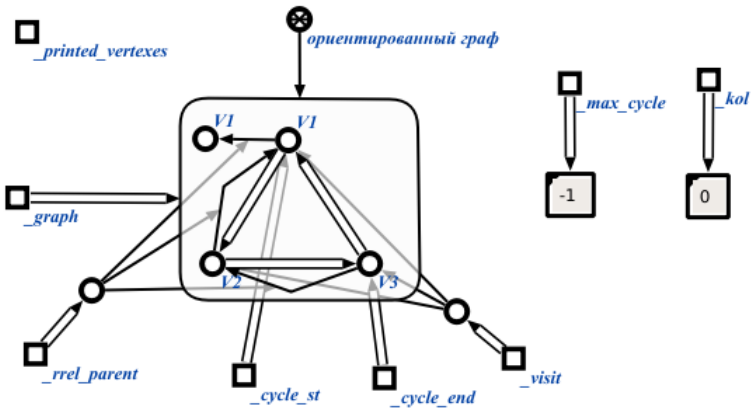

Рисунок *3.12* – Обход графа (шаг 8)

Добавление вершины V3 в переменную _cycle_end.

13.

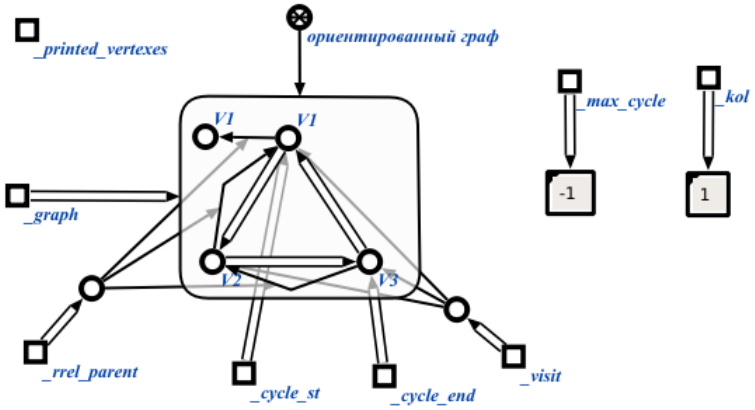

Рисунок *3.13* – Обход графа (шаг 9)

Увеличение значения переменной _kol на 1.

14.

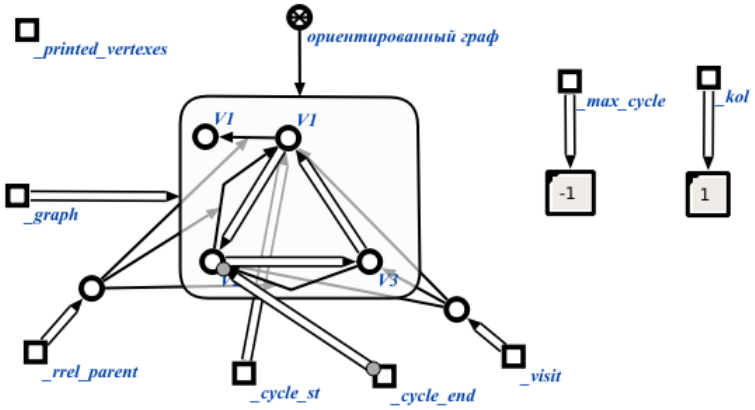

Рисунок *3.14* – Обход графа (шаг 10)

Переходим по дуге, принадлежащей переменной _rrel_parent из вершины V3 в вершину V2, удаляем вершину V3 из переменной _cycle_end и добавляем вершину V2 в переменную _cycle_end.

15.

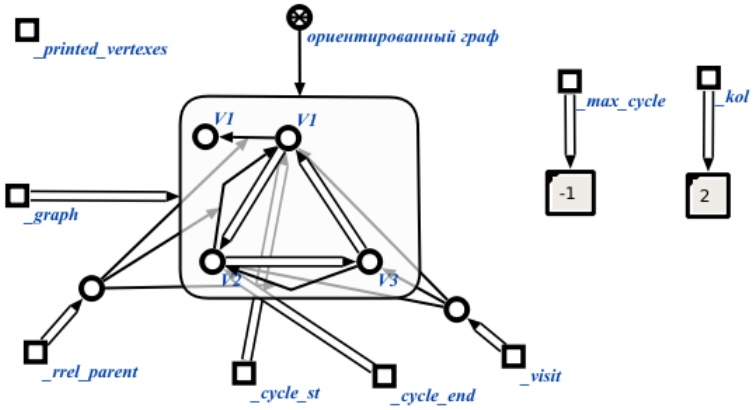

Рисунок *3.15* – Обход графа (шаг 11)

Увеличение значения переменной _kol на 1.

16.

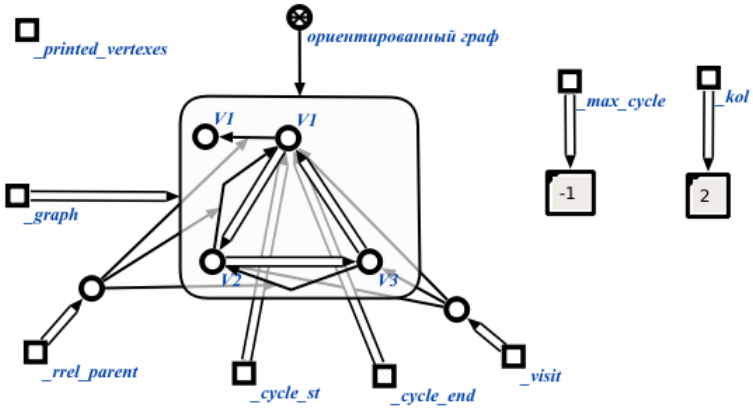

Рисунок *3.16* – Обход графа (шаг 12)

Переходим по дуге, принадлежащей переменной _rrel_parent из вершины V2 в вершину V1, удаляем вершину V2 из переменной _cycle_end и добавляем вершину V1 в переменную _cycle_end.

17.

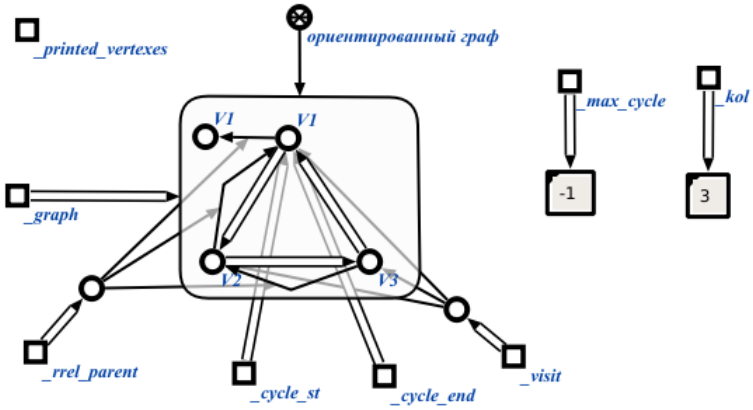

Рисунок *3.17* – Обход графа (шаг 13)

Увеличение значения переменной _kol на 1.

18.

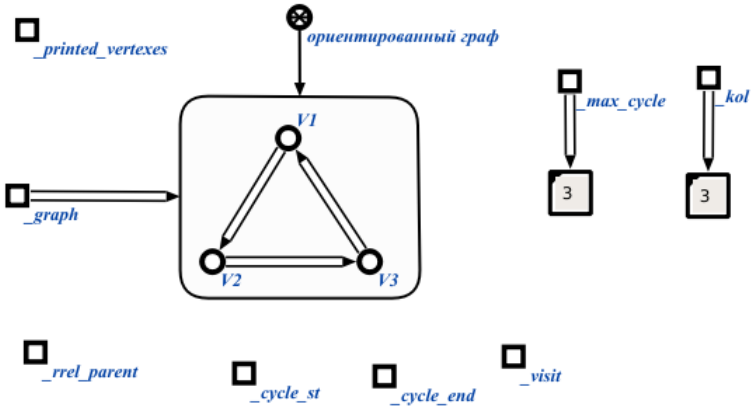

Рисунок *3.18* – Обход графа (шаг 14)

Сравнение переменных _max_cycle и _kol. Т.к. _max_cycle меньше _kol значение переменной _max_cycle меняется на значение переменной _kol.

19.

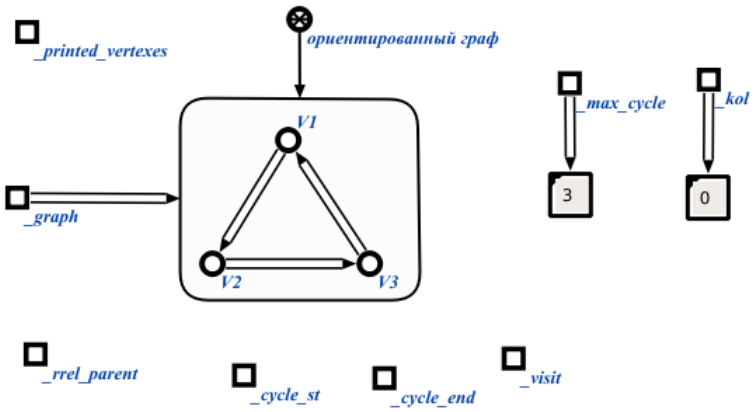

Рисунок *3.19* – Обход графа (шаг 15)

Обнуление переменной _kol.

20.

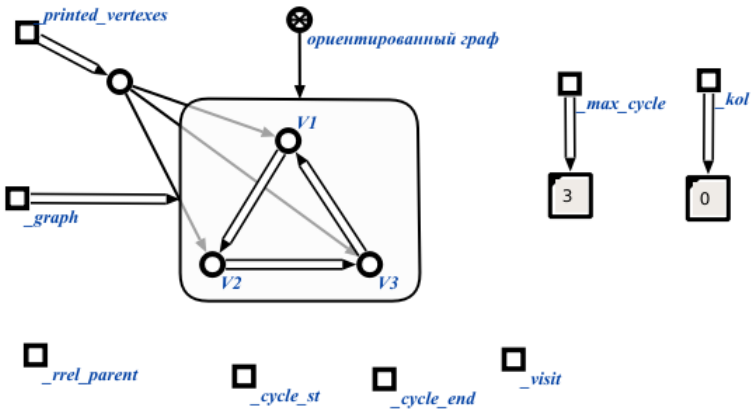

Рисунок *3.20* – Граф, после обхода из всех вершин.

Аналогично будет произведен обход графа из двух других вершин. На рисунке 3.20 представлен граф, после обхода.

21.

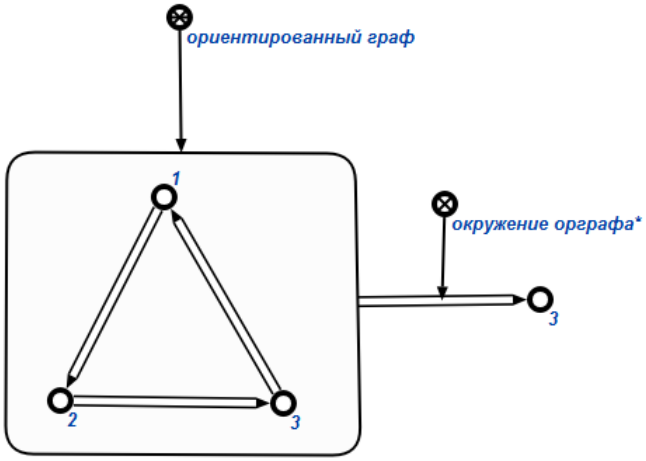

Рисунок *3.21* – Результат программы.

На рисунке 3.21 представлен результат работы программы. Окружением данного графа является значение переменной _max_cycle, равное 3.

4 ЛИЧНЫЙ ВКЛАД В РАЗВИТИЕ ПРОЕКТА СРЕДА УПРАВЛЕНИЯ ПРОЕКТИРОВАНИЕМ OSTIS-СИСТЕМ

4.1 Цели работы

Целью работы было пополнение БЗ проекта, а также поиск и устранение багов и недоточетов в БЗ.

4.2 Выполненные задачи

В этом семестре для развития проекта мною были выполнены следующие задачи:

1. Формализация абсолютного понятия "академическая успеваемость"представлена на рисунке 4.1.

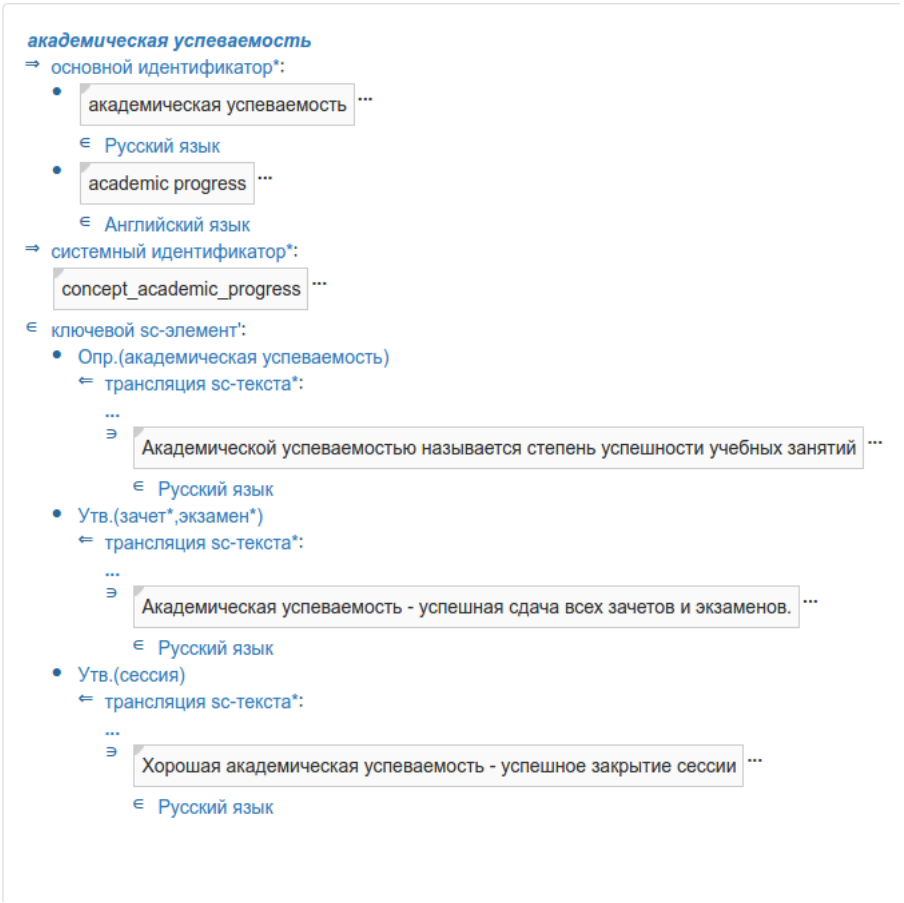

Рисунок *4.1* – Представление в БЗ понятия "академическая успеваемость".

2. Формализация относительного понятия "экзамен*"представлена на рисунке 4.2.

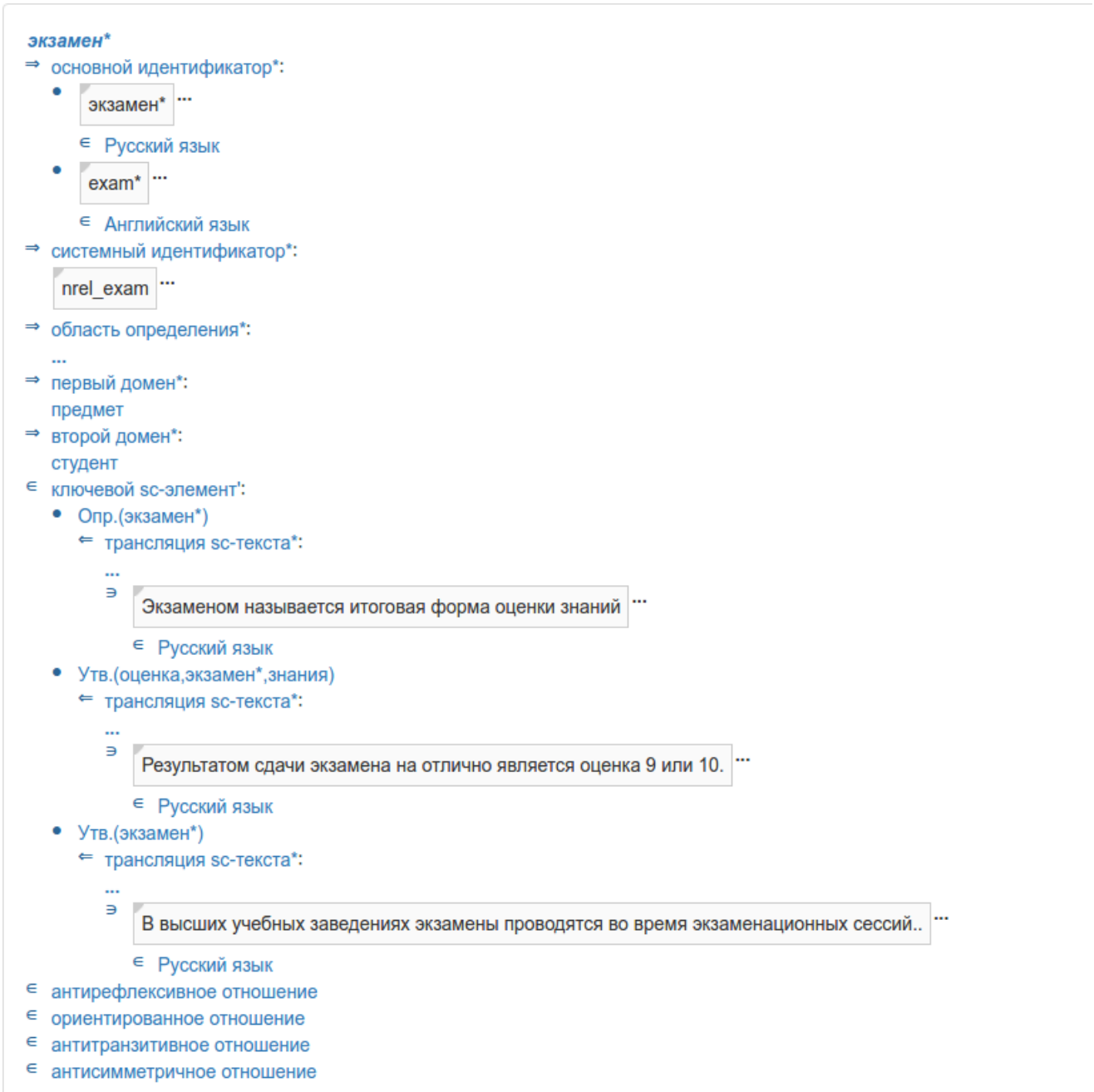

Рисунок *4.2* – Представление в БЗ понятия "экзамен*".

3. Формализация абсолютного понятия "студент"представлена на рисунке 4.3.

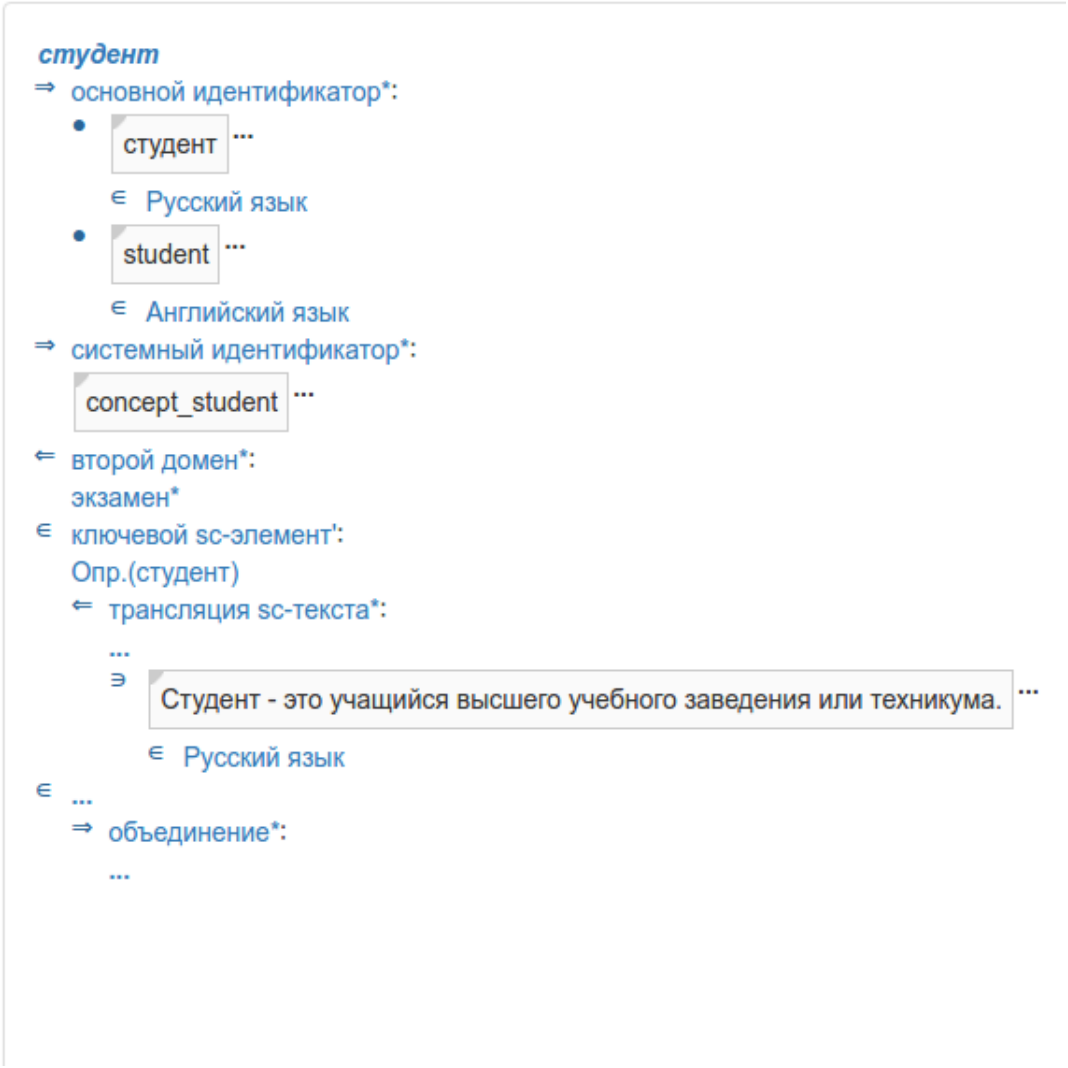

Рисунок *4.3* – Представление в БЗ понятия "студент".

4. Формализация абсолютного понятия "предмет"представлена на рисунке 4.4.

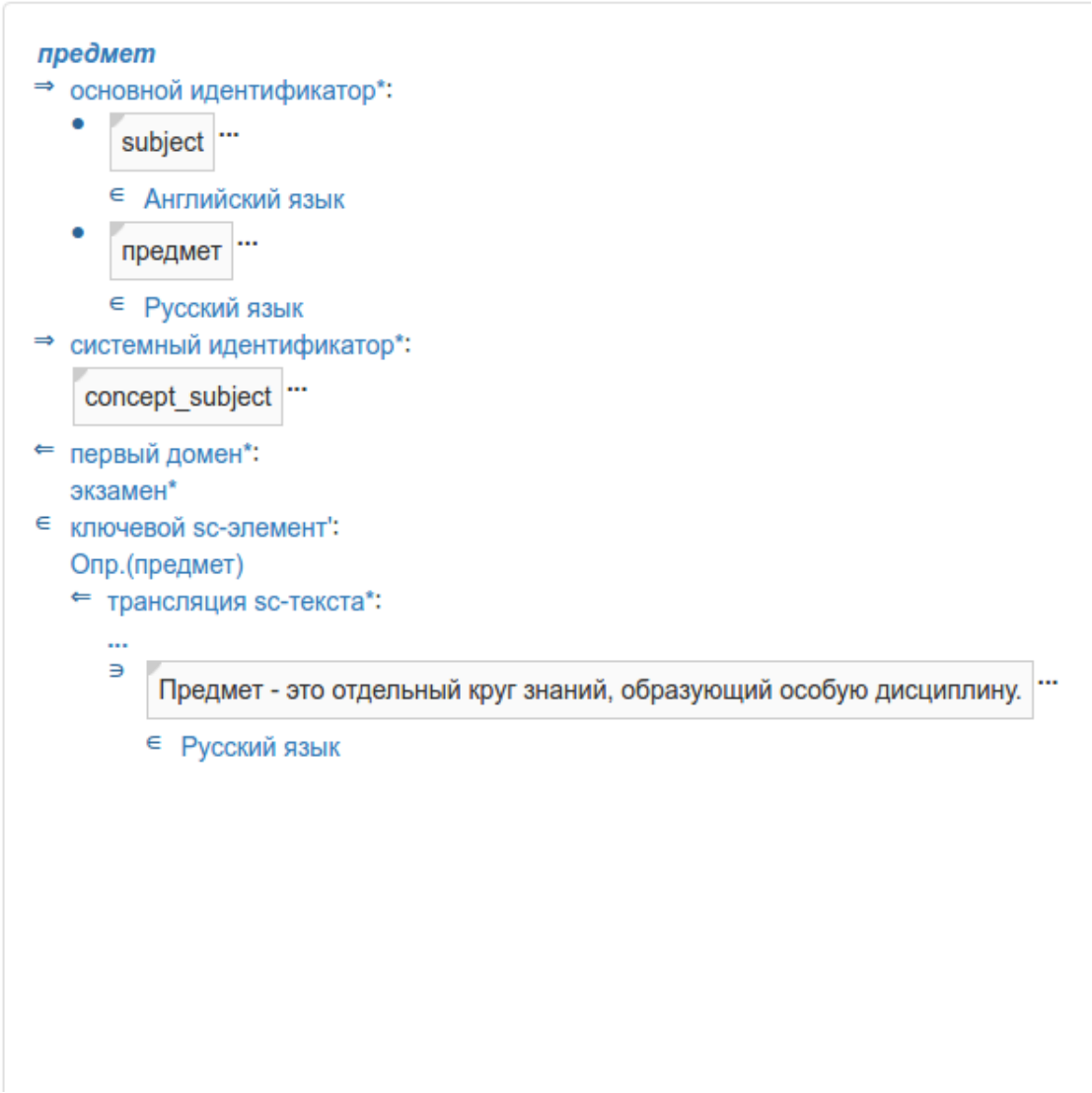

Рисунок *4.4* – Представление в БЗ понятия "предмет".

5. Формализация абсолютного понятия "внедрение"представлена на рисунке 4.5 и рисунке 4.6.

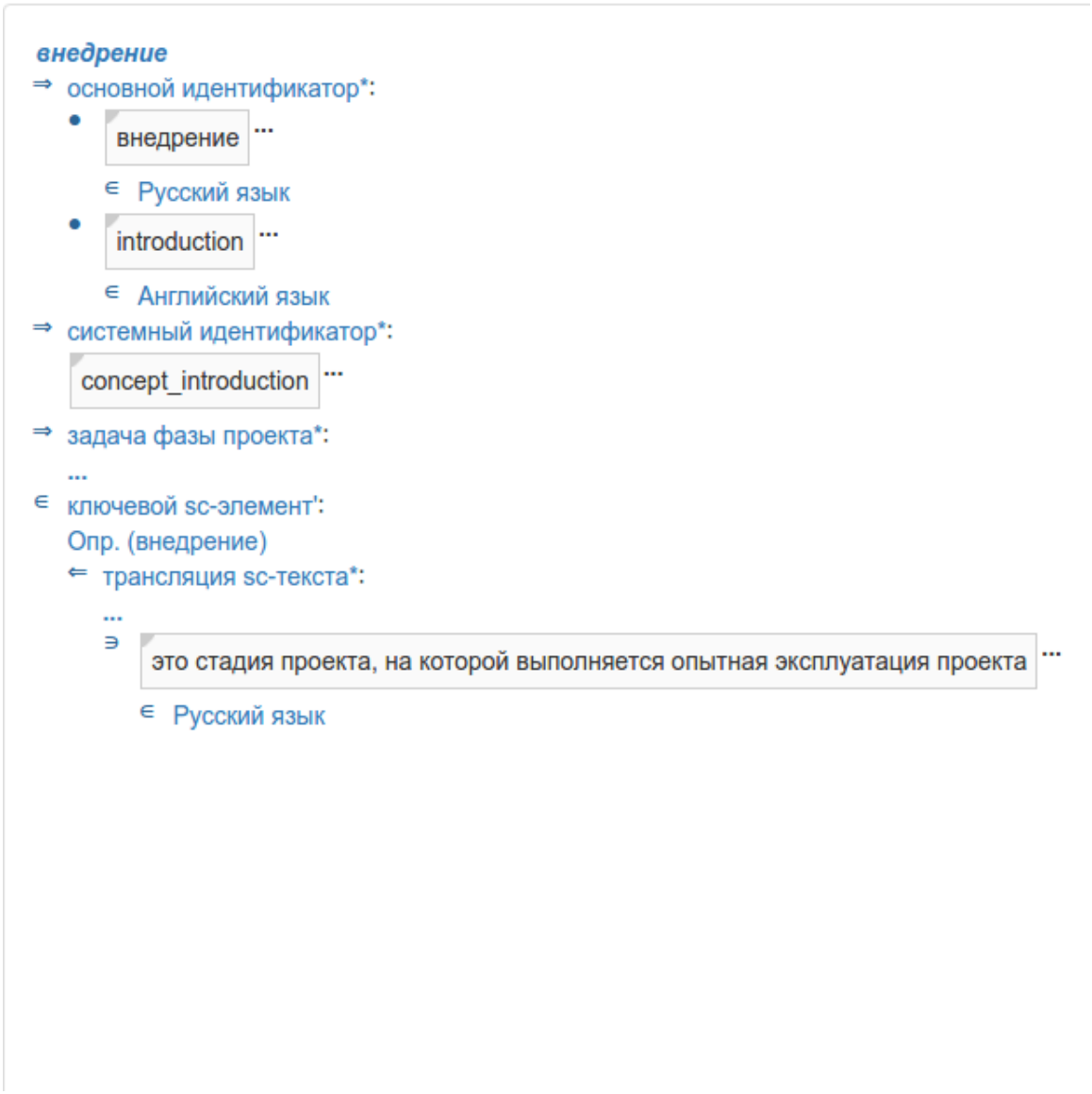

Рисунок *4.5* – Представление в БЗ понятия "внедрение".

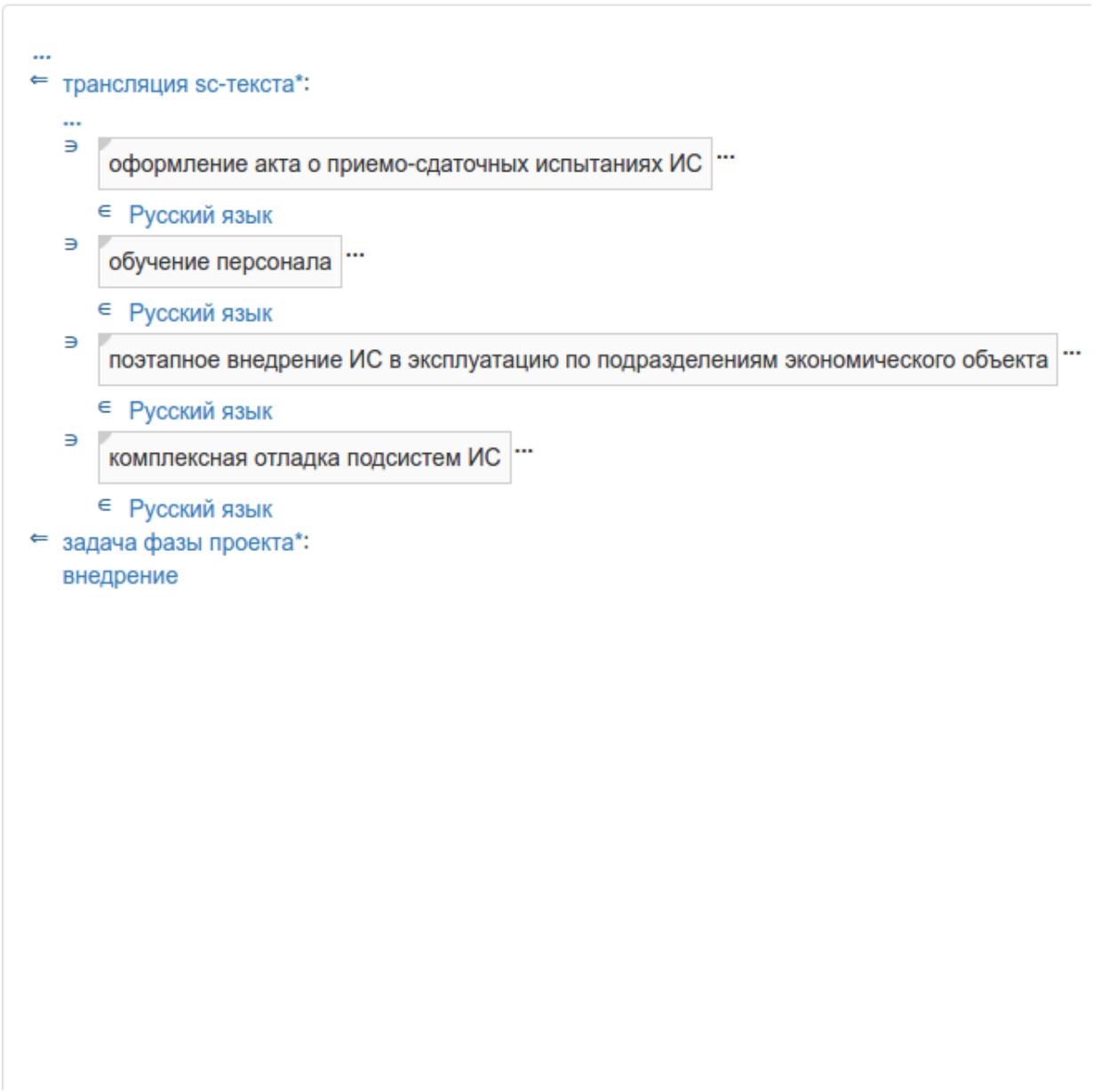

Рисунок *4.6* – Представление в БЗ понятия "внедрение".

6. Формализация абсолютного понятия "задача проекта"представлена на рисунке 4.7.

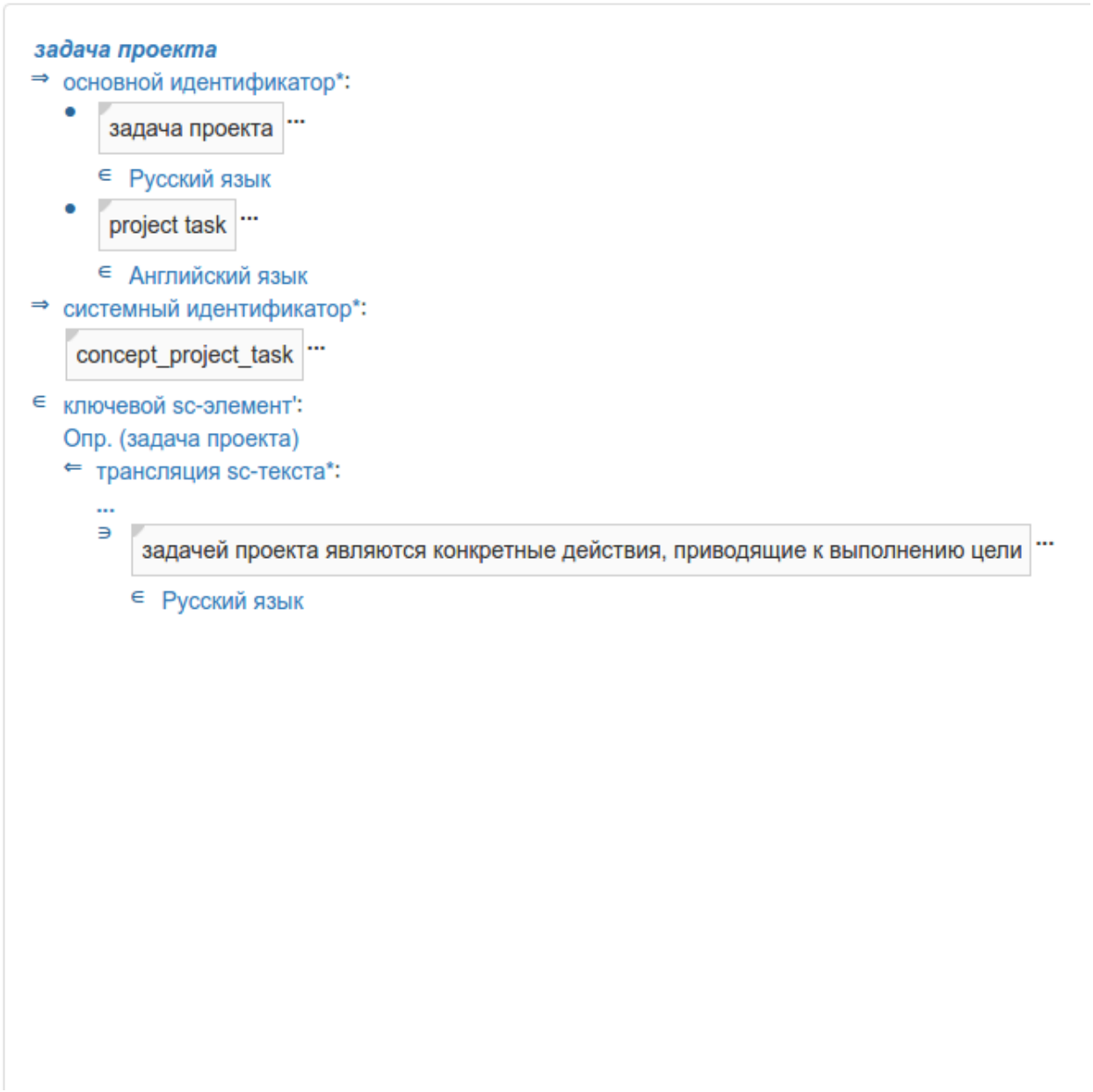

Рисунок *4.7* – Представление в БЗ понятия "задача проекта".

7. Формализация относительного понятия "задача фазы проекта*"представлена на рисунке 4.8.

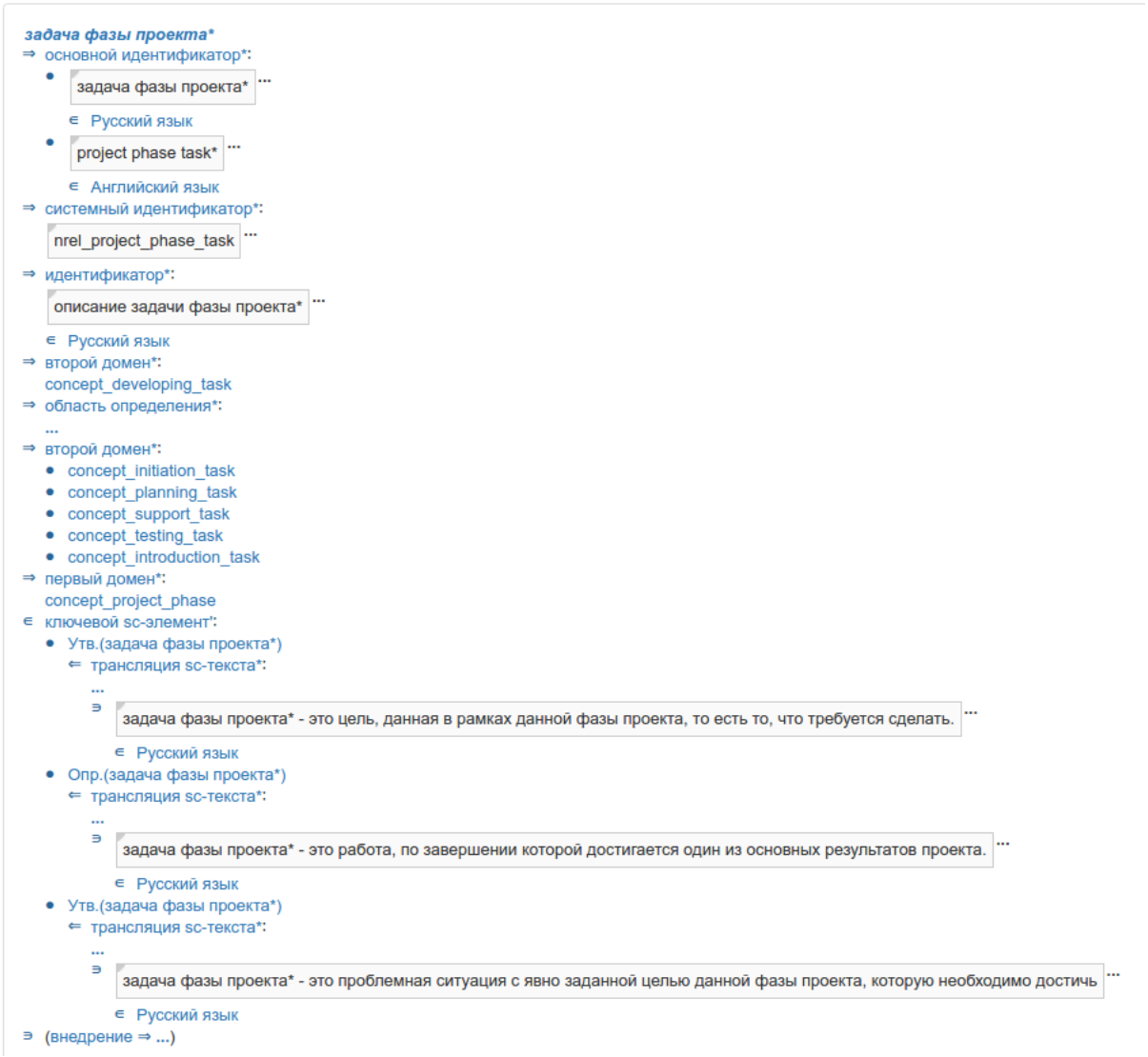

Рисунок *4.8* – Представление в БЗ понятия "задача фазы проекта*".

8. Формализация абсолютного понятия "студенческий проект"представлена на рисунке 4.9.

Рисунок *4.9* – Представление в БЗ понятия "студенческий проект".

9. Формализация относительного понятия "субпроект*"представлена на рисунке 4.10.

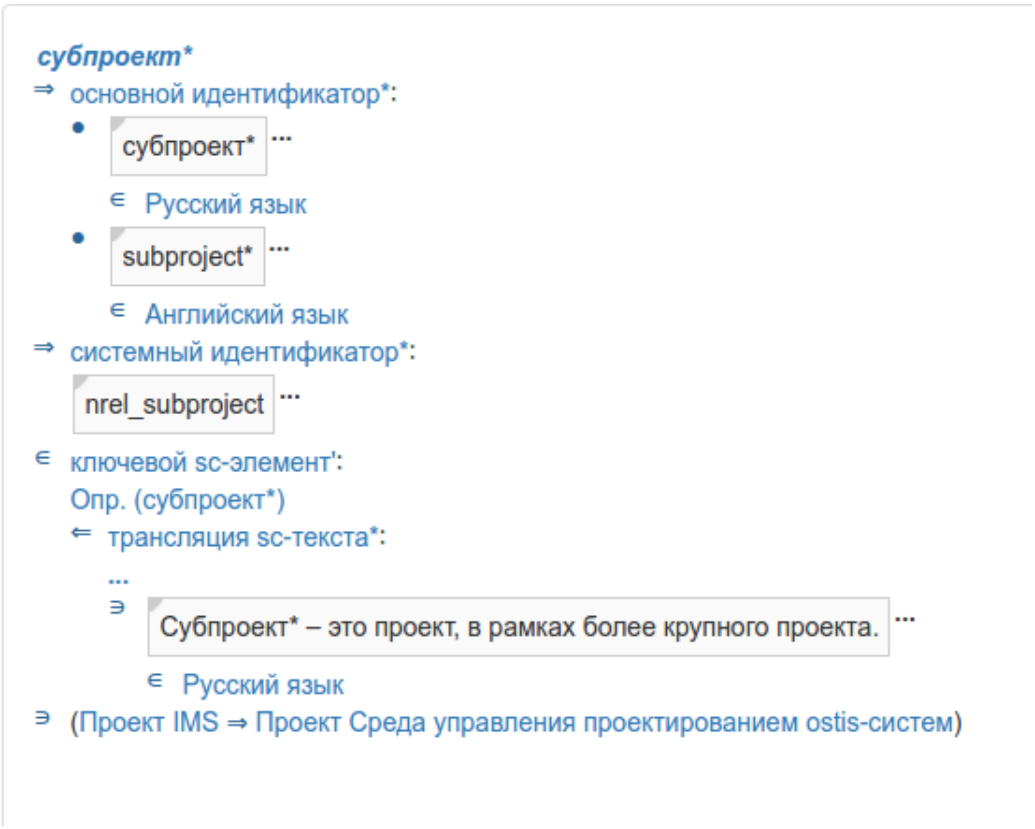

Рисунок *4.10* – Представление в БЗ понятия "субпроект*".

10. Исправление бага на стартовой странице проекта.

Представление в БЗ до исправления представлен0 на рисунке 4.11.

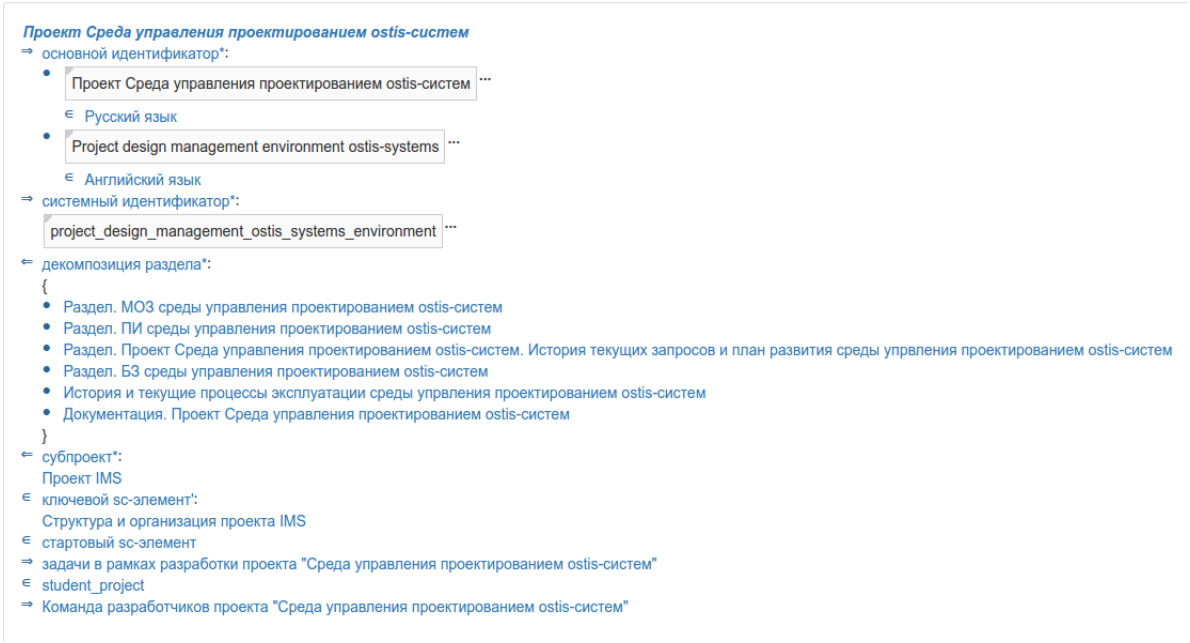

Рисунок *4.11* – Стартовая страница до исправления.

Представление в БЗ до исправления представлен0 на рисунке 4.12.

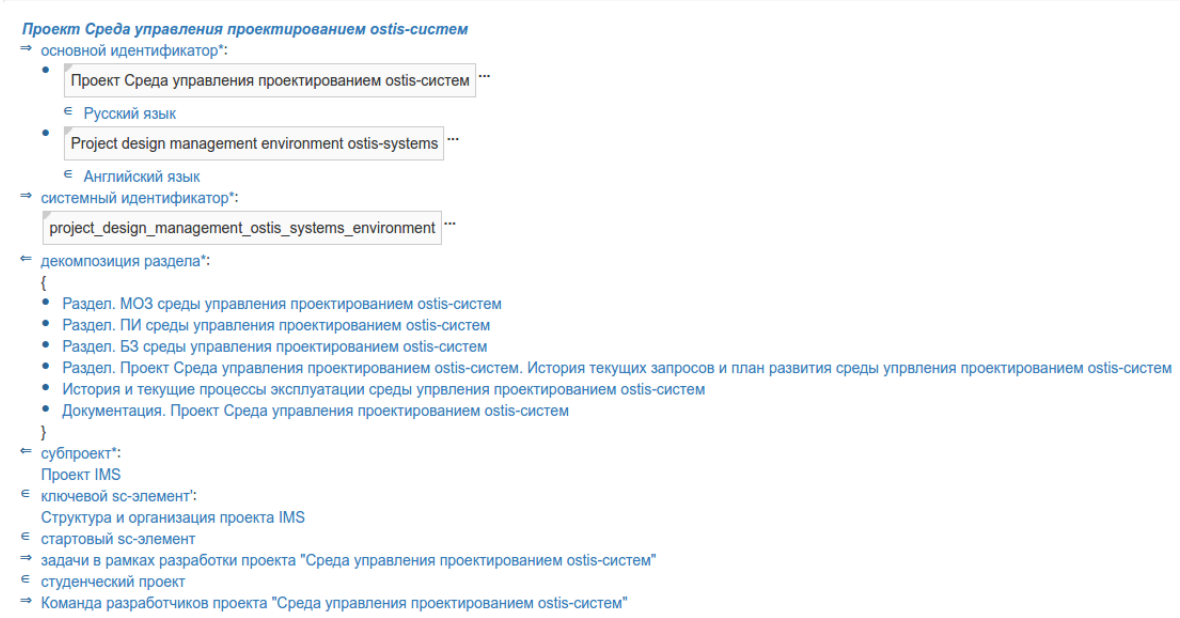

Рисунок *4.12* – Стартовая страница после исправления.

ЗАКЛЮЧЕНИЕ

В этом семестре мною был разработан алгоритм вычисления окружения орграфа, а также внесен определенный вклад в развитие проекта Среда управления проектированием ostis-систем, а именно мною было формализовано 6 абсолютных понятий, 3 относительных понятия, а также найден и исправлен баг на стартовой странице проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Голенков, В.В. Представление и обработка знаний в графодинамических ассоциативных машинах / В.В. Голенков; Ed. by И.Т. Давыденко и др.]; под ред. В.В. Голенкова Д.В. Шункевич. — Минск, 2001.

[2] Голенков, В.В. Семантическая технология проектирования интеллектуальных решателей задач на основе агентно-ориентированного подхода / В.В. Голенков; Ed. by И.Т.Давыденко; Д.В.Шункевич. — Программные системы и вычислительные методы, 2013. – No1.

[3] Шункевич, Д.В. Модели и средства компонентного проектирования машин обработки знаний на основе семантических сетей. / Д.В. Шункевич; БГУИР. — 2013.

[4] Сайт системы OSTIS [Электронный ресурс].-Режим доступа:. `http://ims.ostis.net/`.

[5] Сайт проекта Среда управления проектированием ostis-систем [Электронный ресурс].-Режим доступа:. `http://85.143.221.50:8082/`.