Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

РАСЧЕТНАЯ РАБОТА

по дисциплине «Представление и обработка информации в интеллектуальных системах»

на тему

Найти кол-во компонент связности в неориентированном графе

Выполнил: П. И. Кадиков

Студент группы
321702

Проверила: Н. В. Малиновская

Минск 2024

1 Введение

Цель: Получить навыки формализации и обработки информации с использованием семантических сетей.
Задача: Найти кол-во компонент связности в неориентированном графе.

2 Список понятий

1. ***Неориентированный граф***(абсолютное понятие)-граф, в котором все ребра являются звеньями, то есть порядок двух концов ребра графа не существенен.
   (a) Вершина (относительное понятие, ролевое отношение);
   (b) Связка (относительное понятие, ролевое отношение).

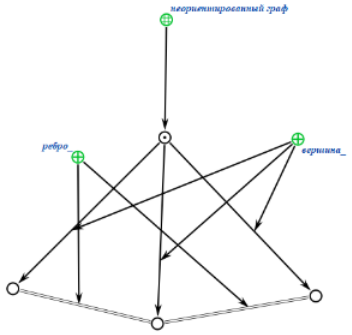

Рис. 1: Абсолютное понятие неориентированного графа.
На рисунке 1 представлено абсолютное понятие неориентированного графа.

2. ***Компонента связности неориентированного графа***(абсолютное понятие) - подмножество вершин, достижимых из какой-то заданной вершины.
   (a) Вершина (относительное понятие, ролевое отношение);
   (b) Ребро (относительное понятие, ролевое отношение).

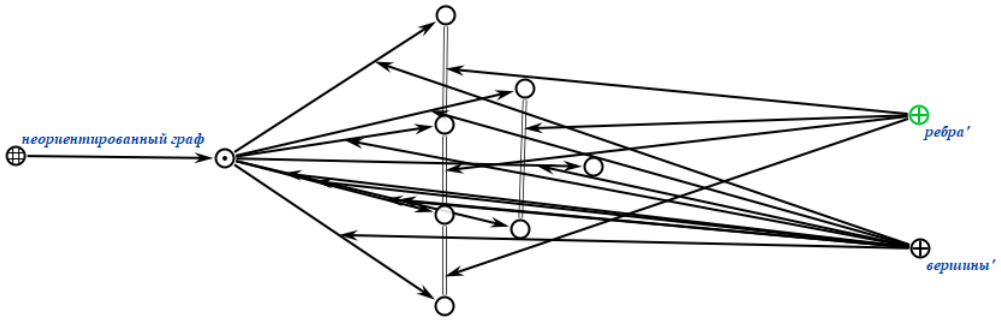

Рис. 2: Компоненты связности.
На рисунке 2 представлены компоненты связности неориентированного графа.

3 Тестовые примеры

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

3.1 Тест 1

Вход: Необходимо найти количество компонент связности неориентированного графа.

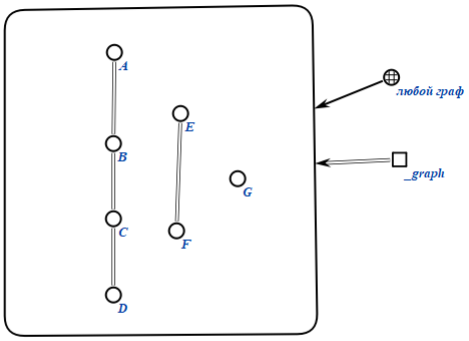

Рис. 3: Вход теста 1.
На рисунке 3 представлен вход теста 1.

Выход: Результатом станет 3, так как граф имеет только 3 компоненты связности.

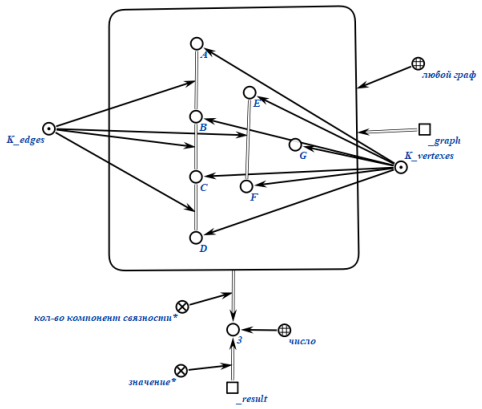

Рис. 4: Выход теста 1.
На рисунке 4 представлен выход теста 1.

3.2 Тест 2

Вход: Необходимо найти количество компонент связности неориентированного графа.

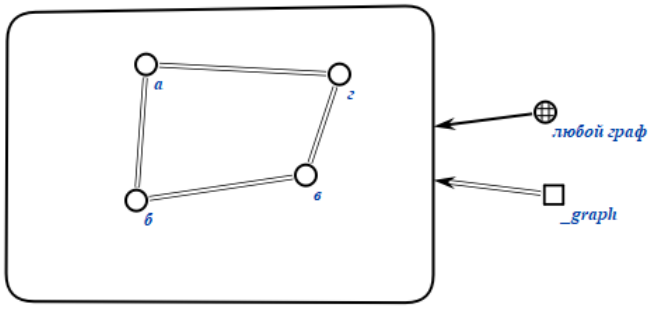

Рис. 5: Вход теста 2.
На риснуке 5 представлен вход теста 2.

Выход: Результатом станет 1, так как граф имеет только 1-у компоненту связности.

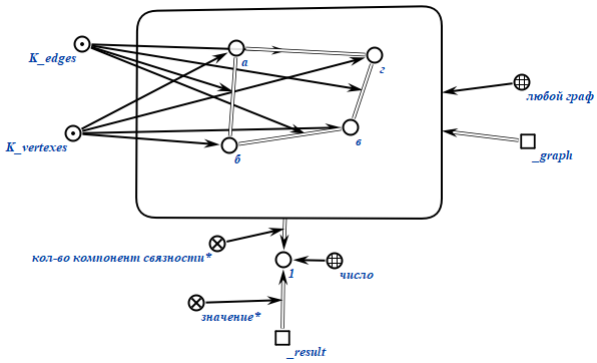

Рис. 6: Выход теста 2.
На рисунке 6 представлен выход теста 2.

3.3 Тест 3

Вход: Необходимо найти количество компонент связности неориентированного графа.

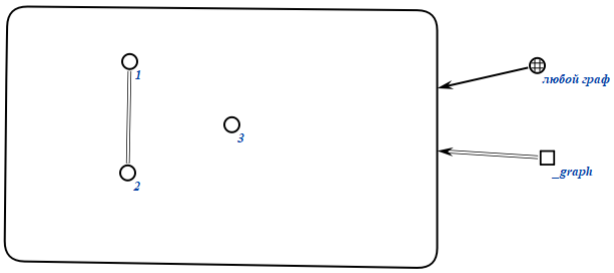

Рис. 7: Вход теста 3.
На риснуке 7 представлен вход теста 3.

Выход: Результатом станет 2, так как граф имеет только 2 компоненты связности.

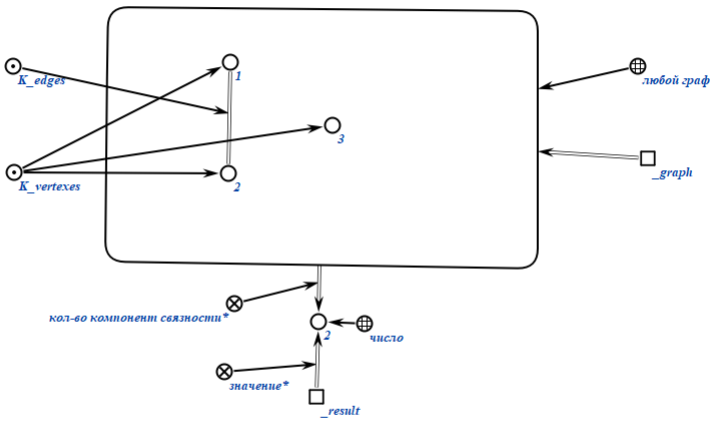

Рис. 8: Выход теста 3.
На риснуке 8 представлен выход теста 3.

3.4 Тест 4

Вход: Необходимо найти количество компонент связности неориентированного графа.

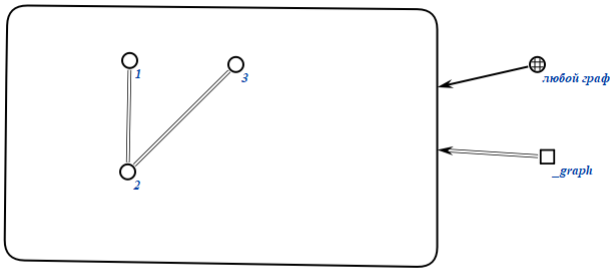

Рис. 9: Вход теста 4.
На рисунке 9 представлен вход теста 4.

Выход: Результатом станет 1, так как граф имеет только 1 компоненту связности.

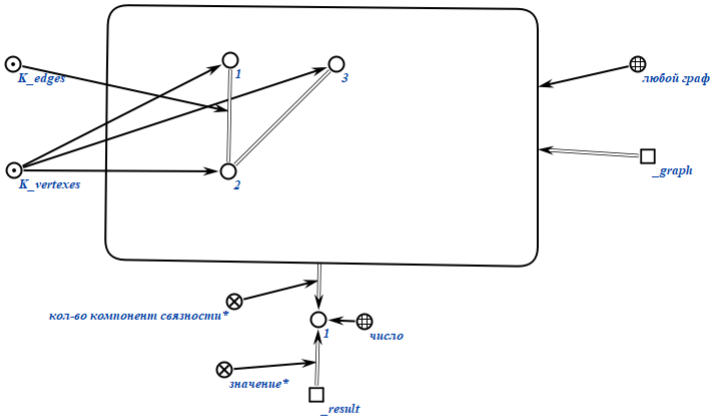

Рис. 10: Выход теста 4.
На рисунке 10 представлен выход тесат 4.

3.5 Тест 5

Вход: Необходимо найти количество компонент связности неориентированного графа.

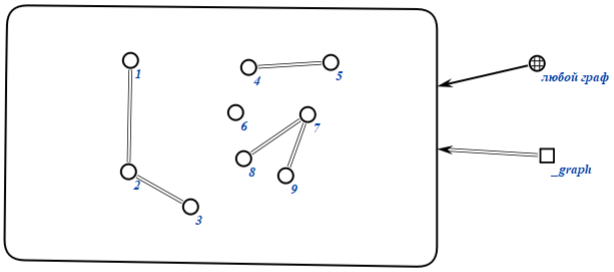

Рис. 11: Вход теста 5.
На рисунке 11 представлен вход теста 5.

Выход: Результатом станет 4, так как граф имеет только 4 компоненты связности.

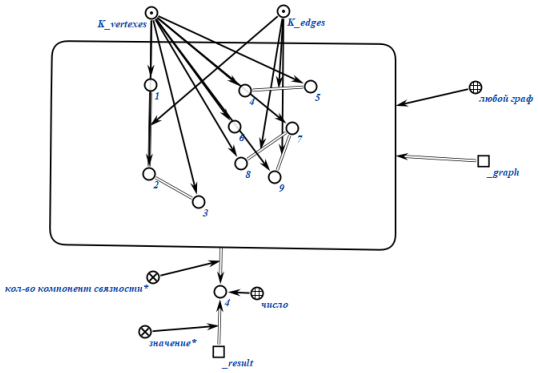

Рис. 12: Выход теста 5.
На рисунке 12 представлен выход теста 5.

4 Пример работы алгоритма в семантической памяти

(a) Входной граф.
_graph получит в качестве значения sc-узел неориентированного графа:

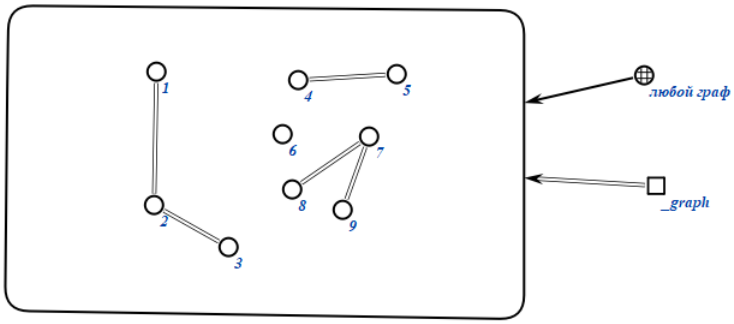

Рис. 13: Входной граф.
На рисунке 13 представлен входной граф.

(b) Добавляем все вершины графа во множество непосещенных вершин.
Переменная _not_checked_vertexes получит в качестве значения множество непроверенных вершин обрабатываемого графа.

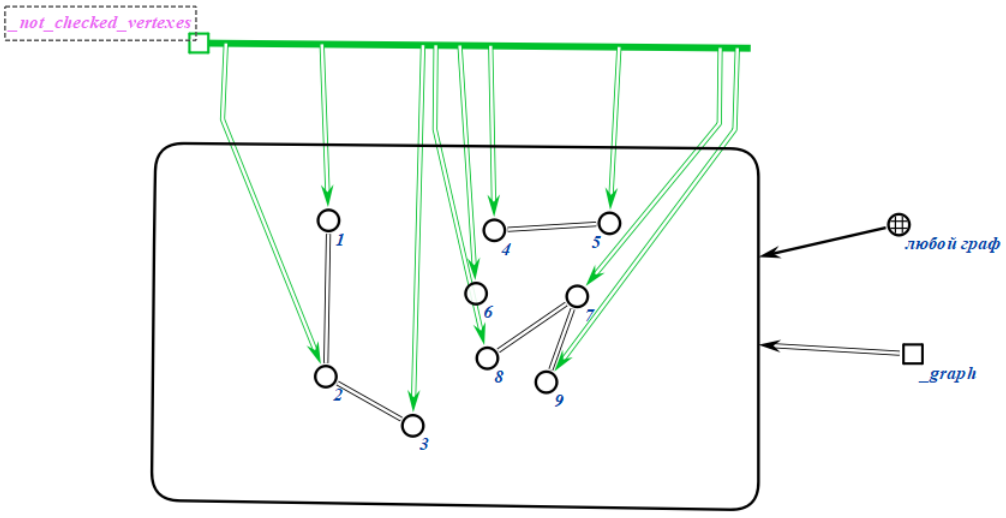

Рис. 14: Непосещённые вершины.
На рисунке 14 представлена переменная "Непосещённые вершины".

(c) Добавляем все вершины во множество конечных вершин, которые имеют либо 1 ребро, либо вообще не имеют ребер

Переменная _finite_vertexes получит в качестве значения множество конечных вершин обрабатываемого графа.

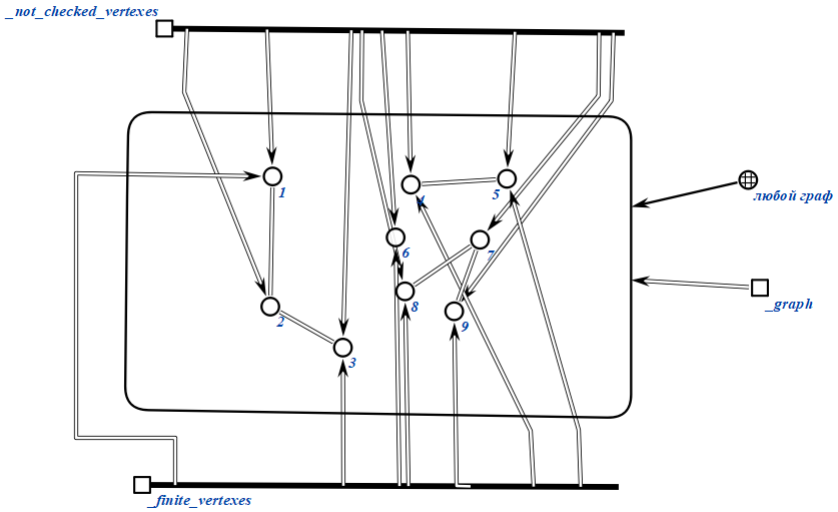

Рис. 15: Конечные вершины.
На рисунке 15 представлена переменная "Конечная вершины".

(d) Создаём счетчик кол-ва компонент связности неориентированного графа.

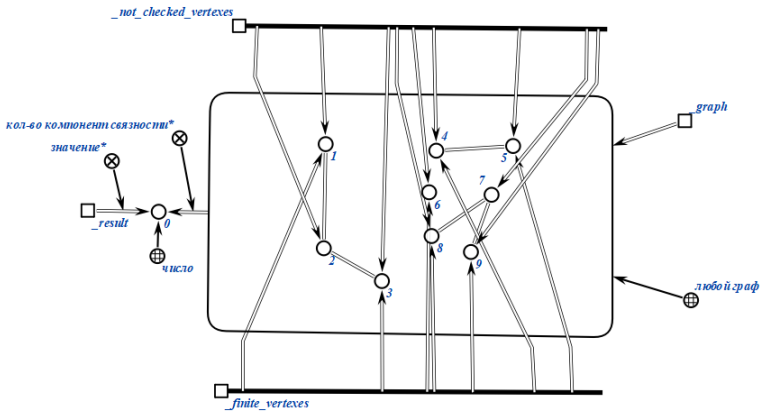

Рис. 16: Счётчик комопнент связности.
На рисунке 16 мы вводим счётчик компонент связности.

(e) Создаем переменную _checked_vertexes, которая будет хранить в себе посещенные вершины.

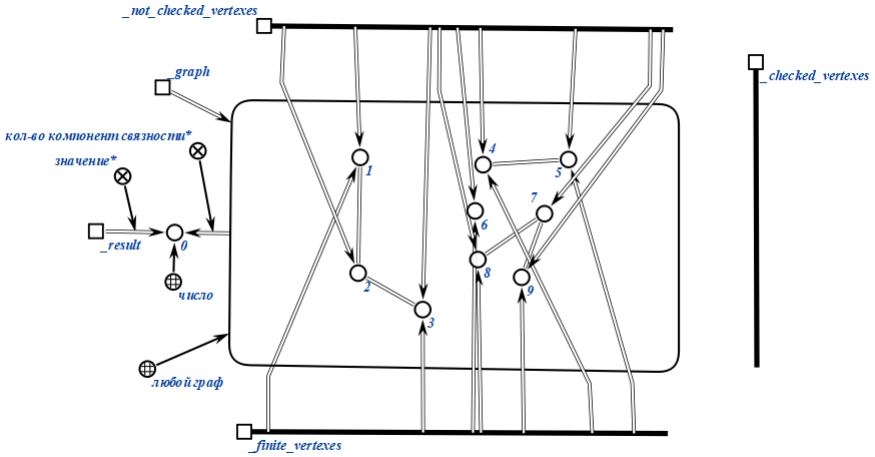

Рис. 17: Посещенные вершины.
На рисунке 17 представленна переменная "Посещённые вершины".

(f) Начинаем поиск кол-ва компонент связностей с вершины под индексом 1, одноременно занося ее во мн-во посещенных вершин и удаляя из множества непосещенных вершин.

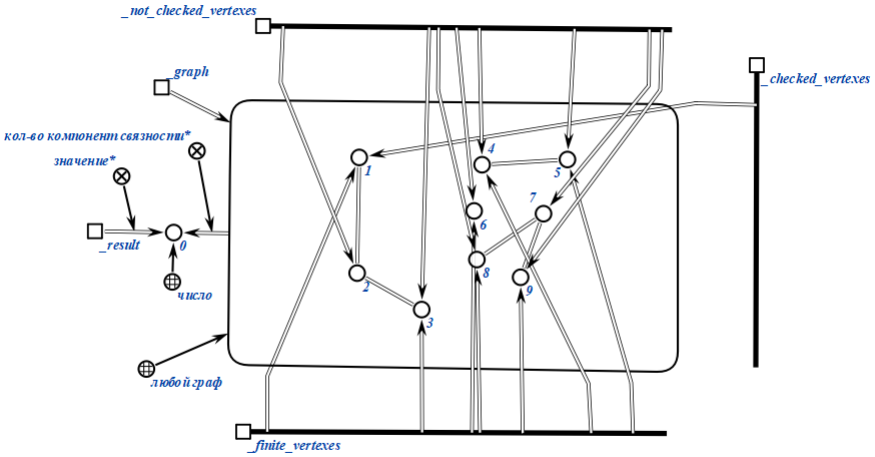

Рис. 18: Начало поиска из вершины с индеком 1.
На рисунке 18 представлено начало поиска из вершины с индексом 1.

(g) Идём из виршины с индексом 1 к вершине с индексом 2. Заносим вершину с индексом 2 во множество посещенных вершин и удаляем ее же из мн-ва непосещенных вершин.

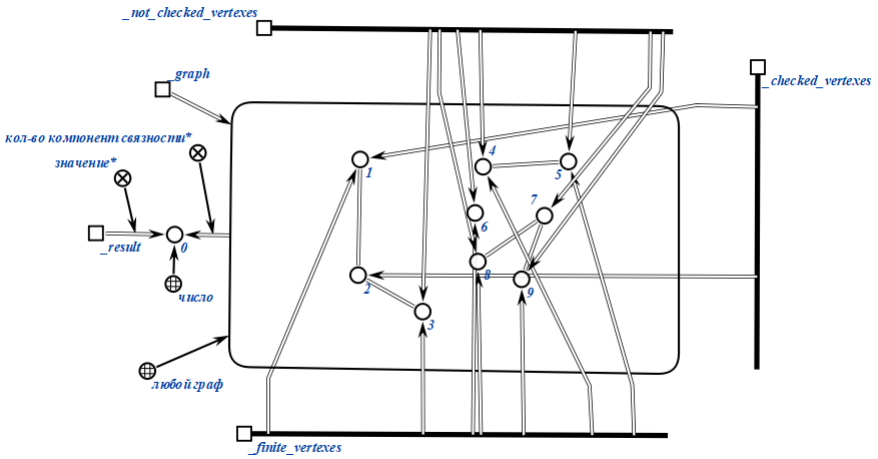

Рис. 19: Идём в вершину с индексом 2.
На рисунке 19 продемонстрирован поиск из вершины с индексом 1 в 2.

(h) Затем программа идет из вершины с индексом 2 в вершину с индексом 3, так как между ними имеется ребро и вершина с индексом 3 еще не внесена во множество посещенных вершин. Соответственно удаляем эту вершину из мн-ва непосещенных вершин и вносим во мн-во посещенных вершин.

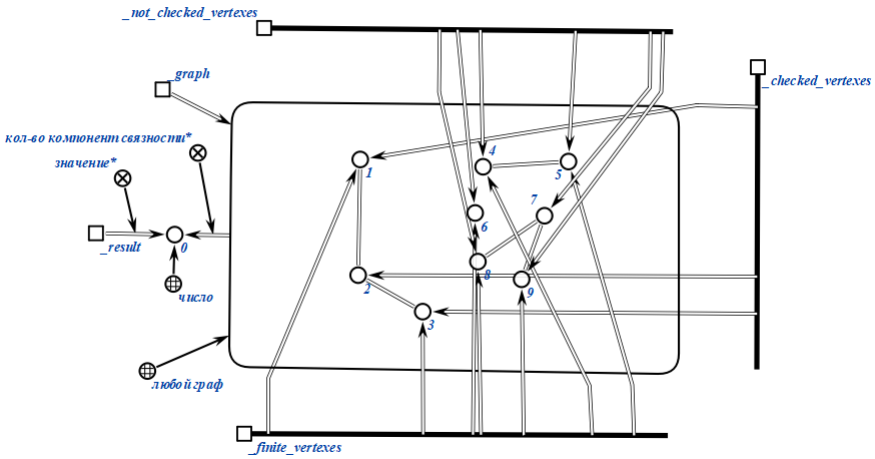

Рис. 20: Идём в веришну с индексом 3.
На рисунке 20 продемонстрирован поиск из вершины с индексом 2 в 3.

(i) Больше у вершины с индексом 3 нету других вершин, с которыми она была бы связана ребром и которые бы входили во множество непосещенных вершин. Такая же ситуация и с вершиной с индексом 1. Также программа видит, что вершины с индексом 1 и 3 входят во множество конечных вершин(_finite_vertexes). Соответственно программа понимает, что счетчик кол-ва компонент связности нужно увеличить на 1(что она собственно и делает).

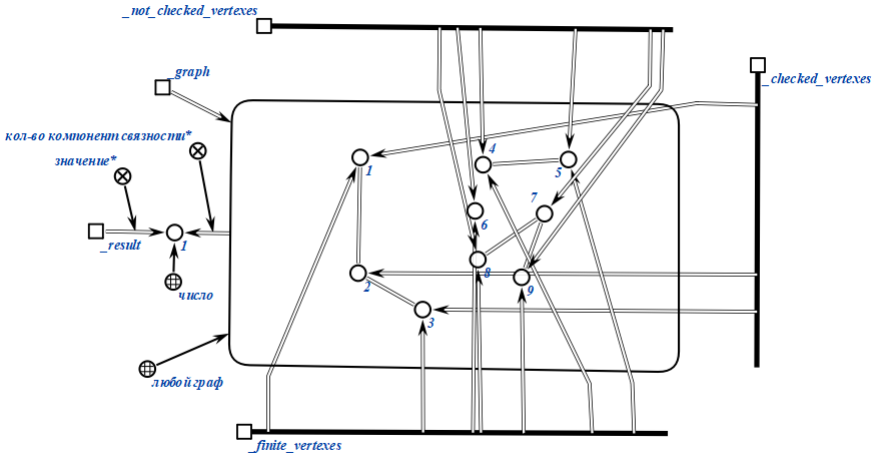

Рис. 21: Увеличиваем счётчик кол-ва компонент связности на 1 значение.
На рисунке 21 представлено увеличение счётчика на одно значение.

(j) Затем программа начинает выполнение это-го же алгоритма, но начинает с другой непосещенной и крайней(конечной) вершины вершины(например, с индексом 4), удаляет ее из мн-ва непосещенных вершин и вносит во мн-во посещенных вершин.

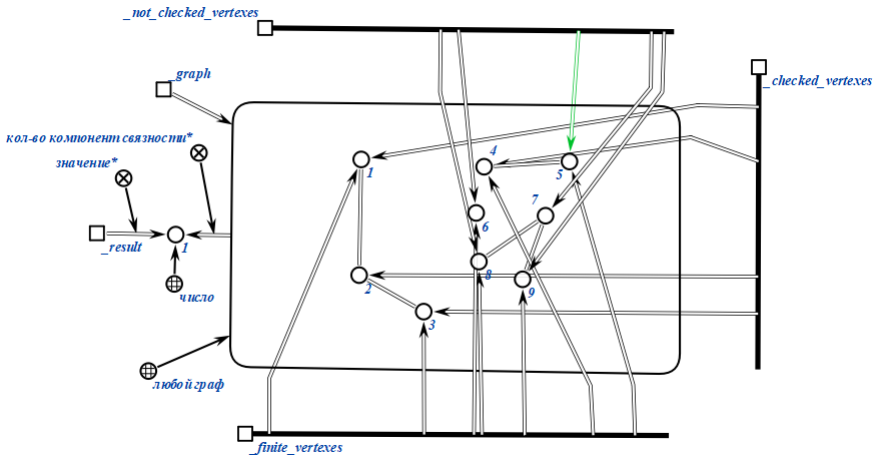

Рис. 22: Продолжение этого же алгоритма с другой компонентой связности.
На рисунке 22 представлено продолжение алгоритма с другой компонентой связности.

(k) Программа идёт далее к веришне с индексом 5, удаляет ее из мн-ва непосещенных вершин и вносит во мн-во посещенных вершин.

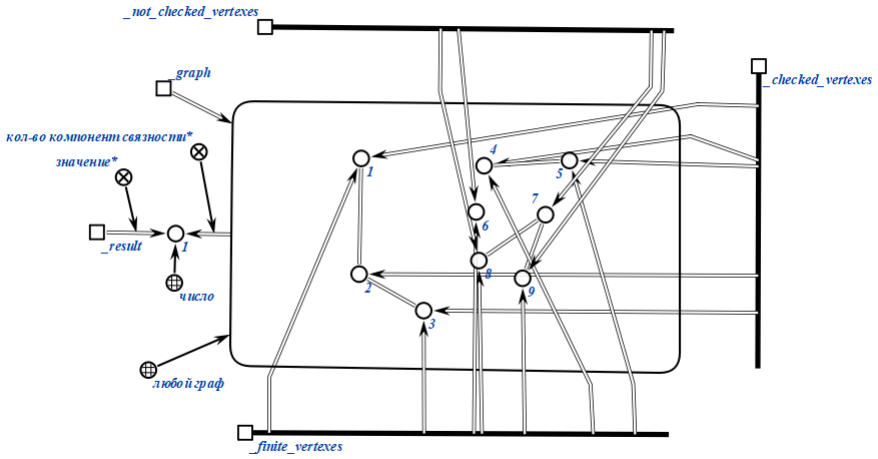

Рис. 23: Идём к вершине с индексом 5.

На рисунке 23 представлено продолжение алгоритма из вершины с индексом 4 в 5.

(l) Больше у вершины с индексом 5 нету других вершин, с которыми она была бы связана ребром и которые бы входили во множество непосещенных вершин. Такая же ситуация и с вершиной с индексом 4. Также программа видит, что вершины с индексом 4 и 5 входят во множество конечных вершин(_finite_vertexes). Соответственно программа понимает, что счетчик кол-ва компонент связности нужно увеличить на 1(что она собственно и делает).

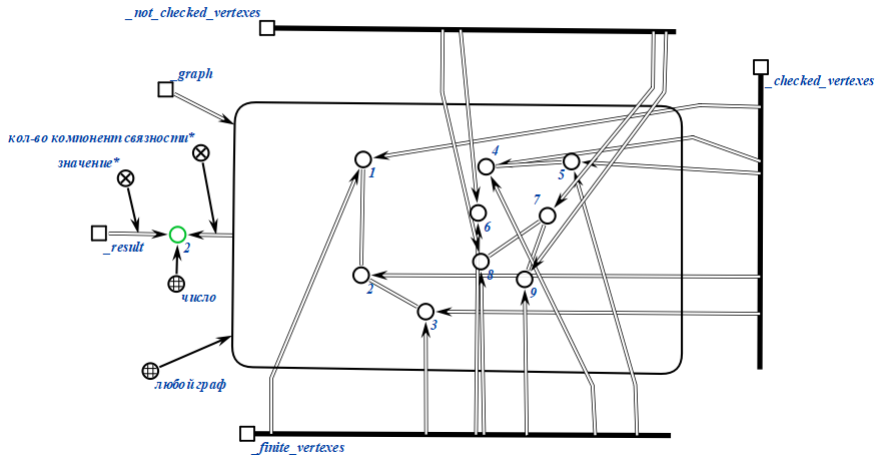

Рис. 24: Увеличение счётчика компонент связности на одно значение.

На рисунке 24 представлено увеличение значения счётчика компонента связности на одно значение.

(m) Переходим к вершине с индексом 6(т.к.ее индекс минимален среди непосещенных оставшихся вершин и она находиться во множествах таких, как конечняя(крайняя) вершина и непосещенные вершины).Удаляем ее из мн-ва непосещенных вершин и добавляем во мн-во посещенных вершин. У вершины с индексом 6 нету других вершин, с которыми она была бы связана ребром и которые бы входили во множество непосещенных вершин. Также программа видит, что вершина с индексом 6 входит во множество конечных вершин(_finite_vertexes). Соответственно программа понимает, что счетчик кол-ва компонент связности нужно увеличить на 1(что она собственно и делает).

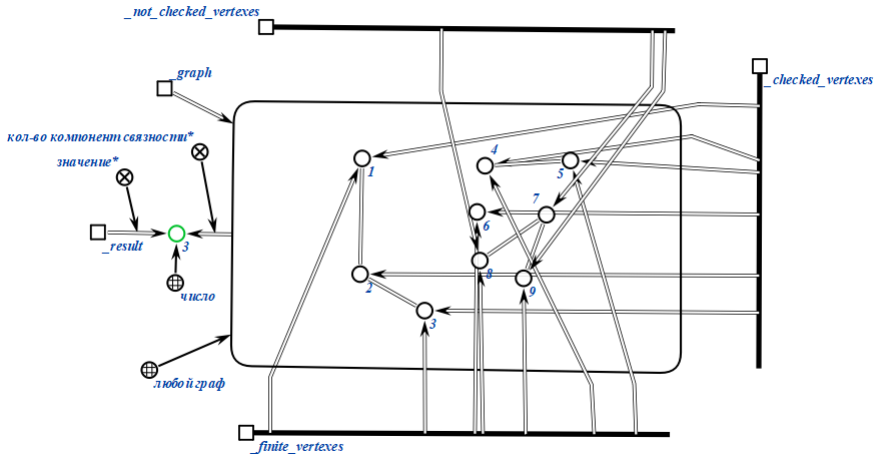

Рис. 25: Переход к вершине с индексом 6 и увеличение счетчика на одно значение.
На рисунке 25 предсатвлен переход использование алгоритма к другой компоненте связности.

(n) Переходим к вершине с индексом 8(т.к.ее индекс минимален среди непосещенных оставшихся вершин входящих во множества, как конечняя(крайняя) вершина и непосещенные вершины). Удаляем ее из мн-ва непосещенных вершин и добавляем во мн-во посещенных вершин.

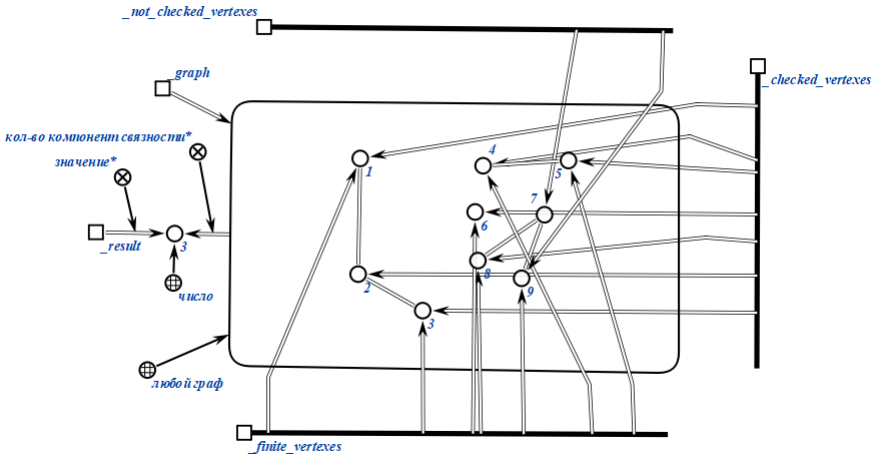

Рис. 26: Переход к вершине с индексом 8.
На рисунке 26 представлен переход использования алгоритма к вершине с индексом 8.

(o) Далее программа видит, что вершина с индексом 8 связана с вершиной с индексом 7 ребром, которая входит во мн-во непосещенных веришн, следовательно она идёт к вершине 7, удаляем ее(7) из мн-ва непосещенных вершин и добавляем во мн-во посещенных вершин.

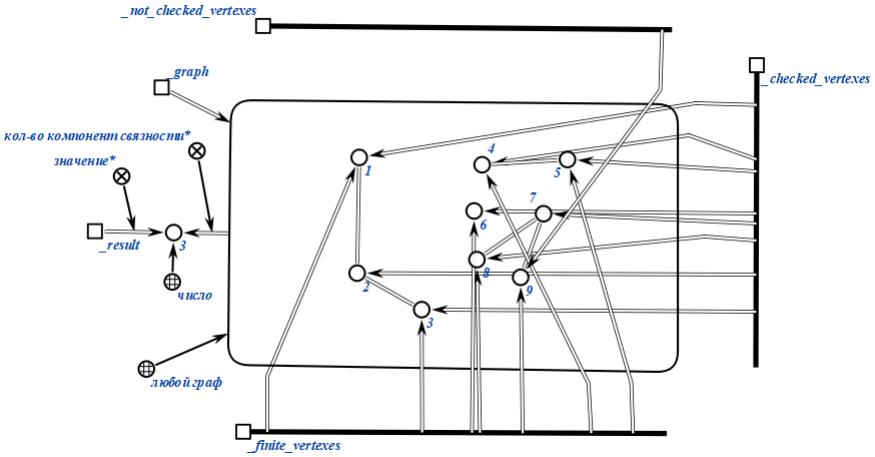

Рис. 27: Переход к вершине с индексом 7.
На рисунке 27 представлен переход использования алгоритма к другой компоненте связности.

(p) Далее программа видит, что вершина с индексом 7 связана с вершиной с индексом 9 ребром, которая входит во мн-во непосещенных веришн, следовательно она идёт к вершине 9, удаляем ее(9) из мн-ва непосещенных вершин и добавляем во мн-во посещенных вершин.

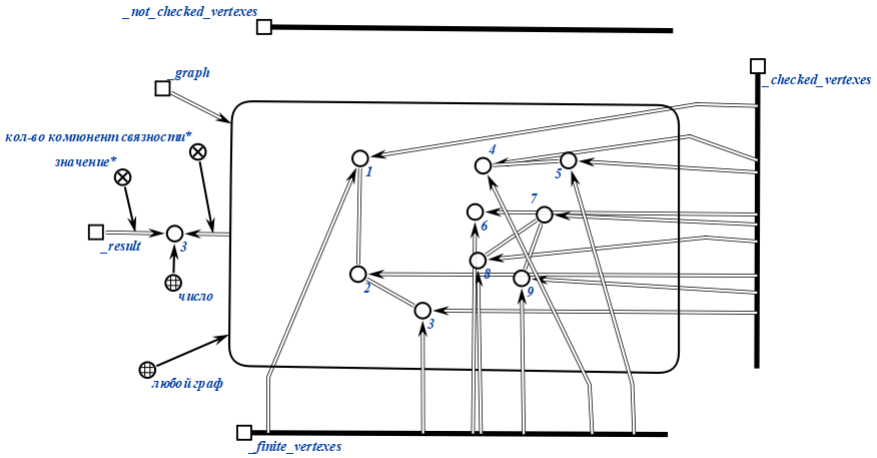

Рис. 28: Переход к вершине с индексом 9.
На рисунке 28 представлен переход из вершины с индексом 7 в 9.

(q) Больше у вершины с индексом 9 нету связей с вершинами, которые бы входили во множество непосещенных вершин. Так же программа видит, что вершины с индексом 8 и 9 входят в мн-во крайних вершин. Следовательно программа добавляет одно значение в счетчик компонент связности.

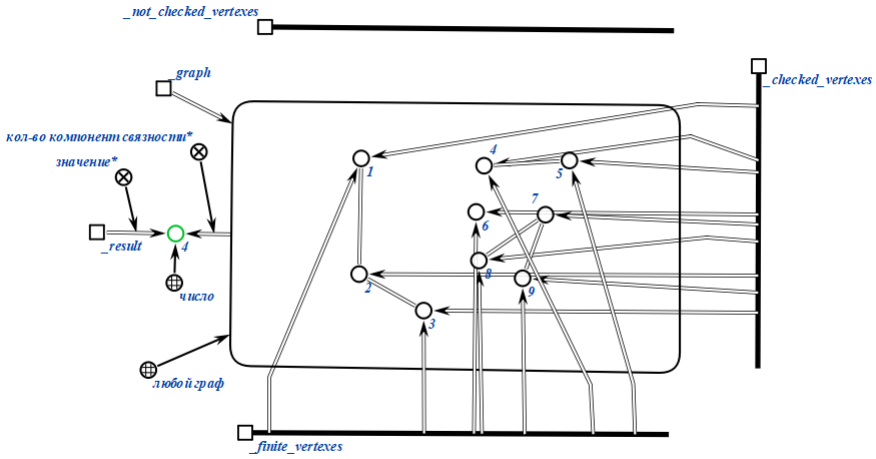

Рис. 29: Увеличение показания счетчика компонент связности на одно значение.

На рисунке 29 представлено увеличение показания счётчика компонента связности на одно значение.

(r) Программа после каждого дабваления в значение счетчик всегда смотрела, не пустое ли мн-во непосещенных вершин, так как оно до этого момента было не пустым, то программа выполняла свою работу. На данном этапе мн-во непосещенных вершин пусто, что гласит нам о том, что программа успешно выполнила свою работу с ответом 4(компоненты связности в неориентированном графе).

5 Заключение

Формализовал алгоритм поиска компонент связности в неориентированном графе. Наглядно рассмотрен алгоритм поиска в глубину. Также алгоритм реализован на языке программирования C++. Получен практический опыт в формализации алгоритмов в KBE. Получен практический опыт в разработке структуры алгоритма. Получен опыт структурного подхода проектирования системы (в данном случае алгоритма) для выполнения определённой задачи - поиск комопнент связности в неориентированно графе.

Список литературы

[1] Ф. Харарри. *Теория графов*. Эдиториал УРСС, Москва, 2018. 304 с.

[2] О. Оре. *Теория графов*. Наука, Москва, 1980. 336 с.

[3] О. П. Кузнецов, Г. М. Адельсон-Вельский. *Дискретная математика для инженера*. Энергоатомиздат, Москва, 1988. 480 с.

[4] Д. Кормен. *Алгоритмы. Построение и анализ*. Вильямс, Москва, 2015. 1328 с.

[5] M. Wooldridge. *An introduction to multiagent systems*. 2nd ed. J. Wiley, Chichester, 2009. 484 p.