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 «Традиционные интеллектуальные информационные технологии»

на тему   
«Нахождение окружения в ориентированном графе»

Выполнил ( Македон Е. А. )

студент группы

521702

Проверил ( Шункевич Д. В. )

Минск 2016

**Цель:** Получить навыки формализации и обработки информации с использованием семантических сетей

**Задача:** Нахождение окружения в ориентированном графе

1. **Список понятий**
2. Ориентированный граф (абсолютное понятие) - это такой граф, в котором все связки являются дугами:

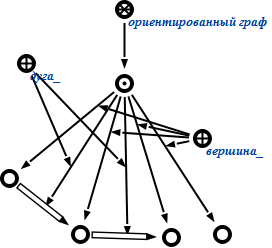


Рисунок 1.1 - Ориентированный граф.

1. Окружение графа – это длина самого длинного простого цикла в графе

**2. Тестовые примеры**

**2.1 Тест 1**

**Вход:**

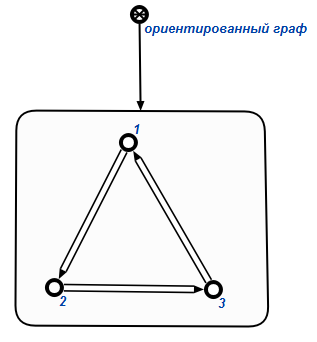
****

Рисунок 2.1 - Ориентированный граф на входе. Тест 1.

**Выход:**

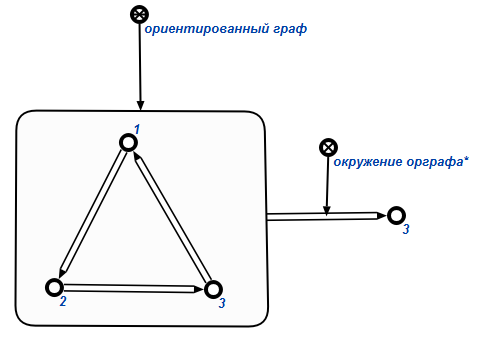
****

Рисунок 2.2 - Ориентированный граф на выходе. Тест 1.

**2.2 Тест 2**

**Вход:**

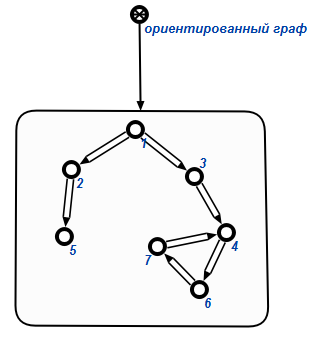
****

Рисунок 2.3 - Ориентированный граф на входе. Тест 2.

**Выход:**

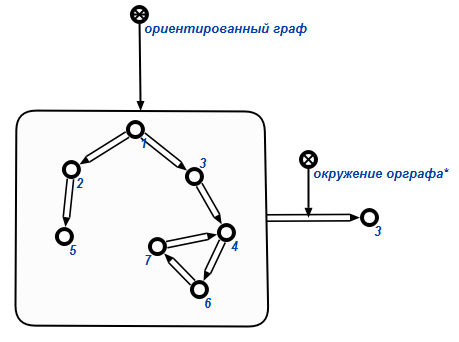


Рисунок 2.4 - Ориентированный граф на выходе. Тест 2.

**2.3 Тест 3**

**Вход:**

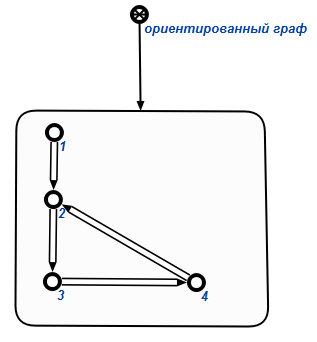
****

Рисунок 2.5 - Ориентированный граф на входе. Тест 3.

**Выход:**

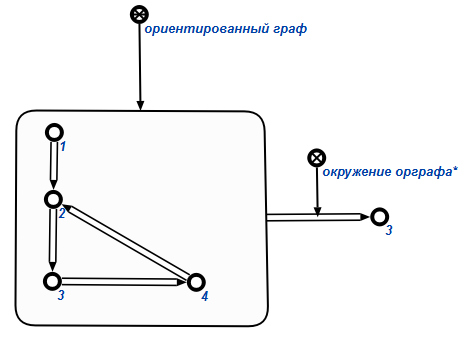


Рисунок 2.6 - Ориентированный граф на выходе. Тест 3.

**2.4 Тест 4**

**Вход:**

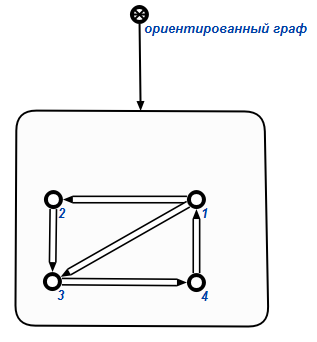
****

Рисунок 2.7 - Ориентированный граф на входе. Тест 4.

**Выход:**

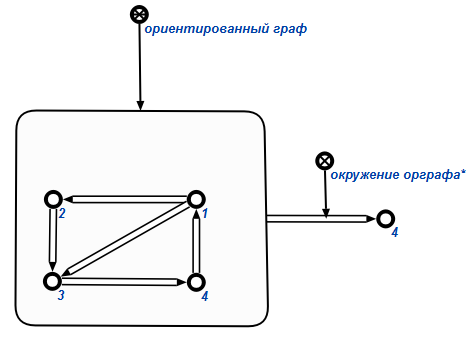


Рисунок 2.8 - Ориентированный граф на выходе. Тест 4.

**2.5 Тест 5**

**Вход:**

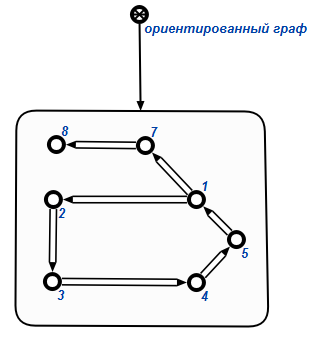
****

Рисунок 2.9 - Ориентированный граф на входе. Тест 5.

**Выход:**

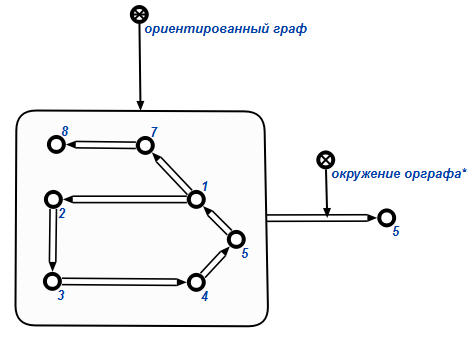
****

Рисунок 2.10 - Ориентированный граф на выходе. Тест 5.

1. **Алгоритм**

**3.1 Постановка задачи**

Задачей является нахождение окружения в ориентированном графе.

**3.2 Описание алгоритма**

* 1. Добавление всех вершин в множество graf.
  2. \_K=0 - переменная для подсчета элементов цикла; A – множество, которое хранит все найденные циклы графа.
  3. Создание множества used размерностью с количеством всех вершин и заполнение его значениями false - вершина не пройдена.
  4. Берем вершину из множества graf.
  5. В множестве used изменяем значение вершины на true - вершина пройдена и \_K увеличиваем на 1.
  6. Берем вершину, смежную с выбранной.
  7. Если вершина со значением false, переходим к пункту (5).
  8. Если вершина со значением true и является вершиной, с которой мы начинали поиск, записываем значение переменной \_K в множество A, \_K=0 и переходим к пункту (4), выбрав следующую вершину из множества graf.
  9. Если вершина со значением true и не является вершиной, с которой мы начинали поиск, \_K=0 и переходим к пункту (4), выбрав следующую вершину из множества graf и изменив все значения в множестве used на false.
  10. Если пройдены все вершины из множества graf, максимальный его элемент будет являться окружением орграфа.

**3.3 Переменные программы**

1. \_graf – переменная, принимающая значение sc-узла графа.
2. \_beg\_vertex – переменная, принимающая значения вершин графа.
3. \_end\_vertex – переменная, принимающая значение последней вершины графа.
4. \_not\_checked\_vertexes – переменная, принимающая значениемножества всех непройденных вершин.
5. \_kol – переменная, принимающая значение кол-ва пройденных вершин.

**3.4 Пример программы**



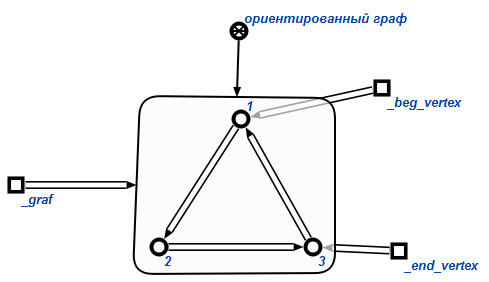


Рисунок 3.1 – Входные данные.  
  
Переменная \_graf принимает значение sc-узла ориентированного графа.  
Переменная \_beg\_vertex принимает значение первой вершины.  
Переменная \_end\_vertex принимает значение последней вершины.

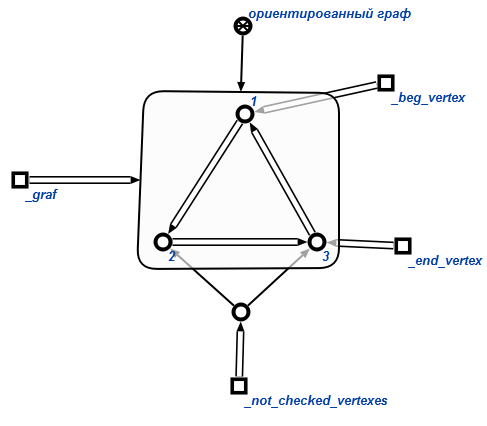
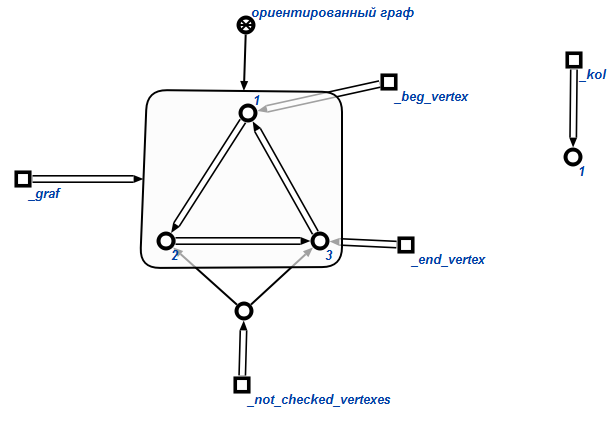
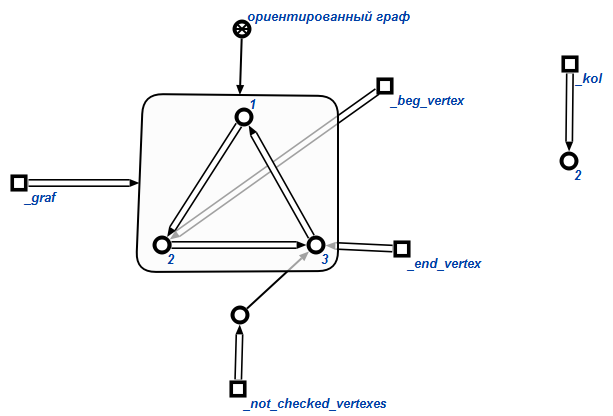
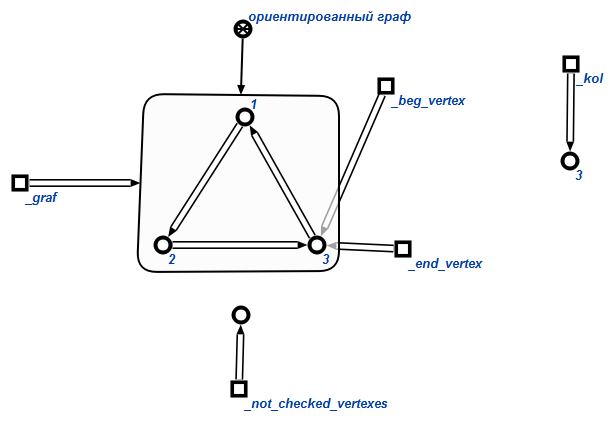
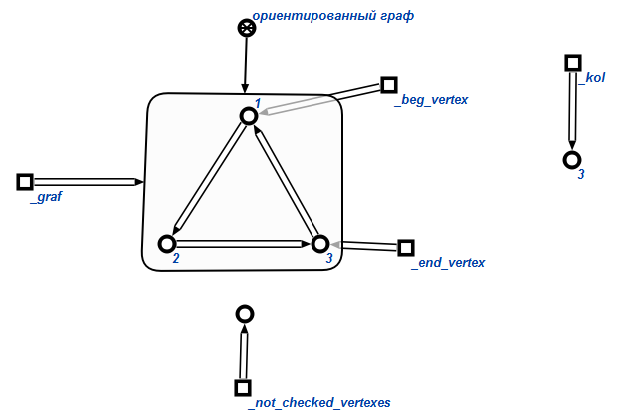
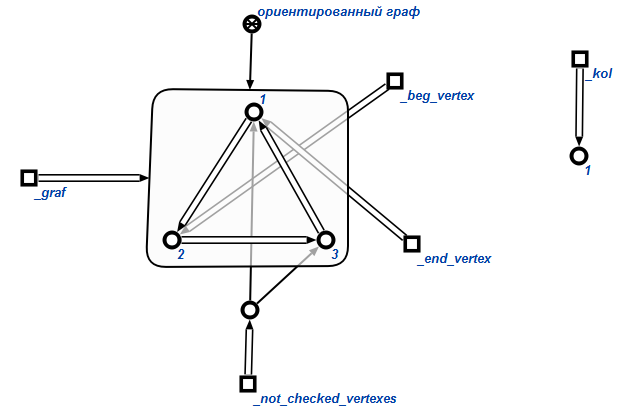
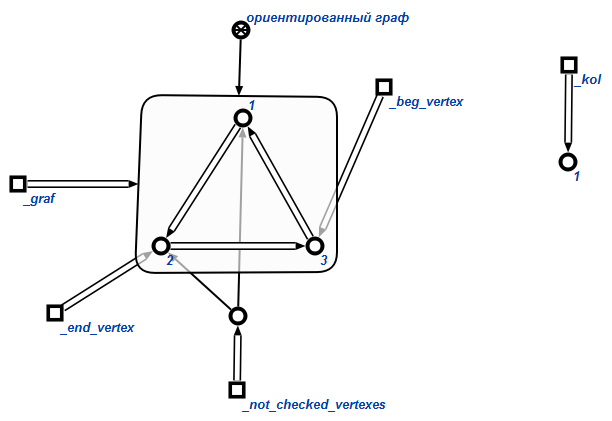
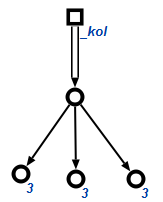
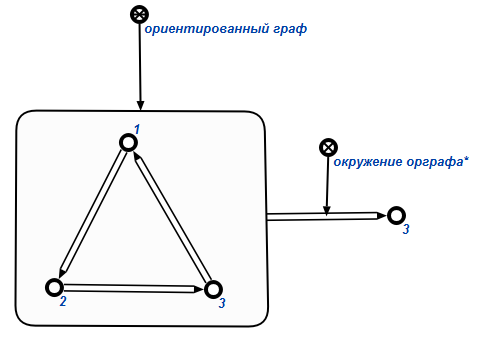


Рисунок 3.2 – Создание множества непройденных вершин.  
  
Переменная \_not\_checked\_vertexes принимает значение множества всех непройденных вершин.

1.   
   Рисунок 3.3 – Обход графа (шаг 1).  
     
   Начинается обход графа с первой вершины.  
   Переменная \_kol принимает значения 1.

1.   
   Рисунок 3.4 – Обход графа (шаг 2).  
     
   Переменная \_beg\_vertex принимает значение второй вершины.  
   Переменная \_kol принимает значение 2.
2.   
   Рисунок 3.5 – Обход графа (шаг 3).  
     
   Переменная \_beg\_vertex принимает значение третьей вершины.  
   Переменная \_not\_checked\_vertexes принимает значение пустого множества.  
   Переменная \_kol принимает значение 3.
3.   
   Рисунок 3.6 – Обход графа (шаг 4).  
   Переменная \_beg\_vertex принимает значение первой вершины.  
   Программа приходит в вершину, которая уже была пройдена и является вершиной, с которой мы начинали обход, значит все пройденные вершины составляют цикл.  
   Переменная \_kol принимает значение 3 - мощность найденного цикла.
4.   
   Рисунок 3.7 – Входные данные для обхода графа со второй вершины.  
     
   Аналогично производится обход графа, начиная со второй вершины.
5. Рисунок 3.8 – Входные данные для обхода графа с третьей вершины.  
     
   Аналогично производится обход графа, начиная со третьей вершины.
6. В итоге переменная \_kol принимает значение множества, которое хранит мощности всех найденных циклов.  
      
   Рисунок 3.9 – Переменная \_kol после всех обходов графа.  
     
   Окружением орграфа является максимальное из этих значений.
7. ****  
   Рисунок 3.10 – Выходные данные.  
     
   На рисунке 3.10 показан ответ – результат работы программы.