# ВВЕДЕНИЕ

Умение правильно составить логическое выражение для условного оператора является обязательным для программиста. Задачи на проверку принадлежности точки некоторой области на плоскости позволяют проверить умение правильно составлять логические выражения и использовать в них различные сочетания арифметических выражений и логических операторов, что является немаловажным для любого начинающего программиста. Подобные задачи являются заданием С1 в ЕГЭ по информатике, но количество этих задач ограничено. Возникает необходимость иметь большое количество подобных заданий. Решением этой проблемы может стать создание инструмента, позволяющего некоторым образом генерировать область на плоскости.

Целью разрабатываемого продукта является автоматизация проверки способностей учащихся составлять сложные логические выражения для задач о принадлежности точки заданной области.

Назначением конечного продукта является упрощение проверки студентов, автоматизированный сбор результатов тестирования для подсчета статистики.

# 1 ТЕХНИЧЕСКИЙ ПРОЕКТ

## **Описание предметной области**

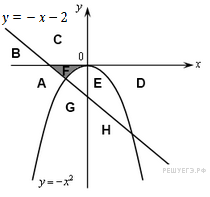
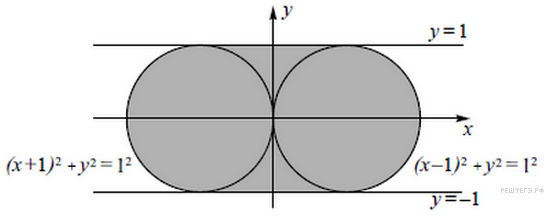
### **Основные теоретические сведения**

При изучении курса информатики, одним из важных разделов является умение правильно работать с логическими операциями, которые в свою очередь образуют логические выражения.

Умение правильно составлять подобные выражения является важным критерием для будущих программистов.

В экзаменационных задачах встречаются несколько разных видов: проверка на принадлежность точки заданным интервалам, проверка на принадлежность точки заданной области на плоскости и проверка на корректность ввода данных.

В разрабатываемом продукте будут реализованы задачи на принадлежность точки заданной области на плоскости. Примеры задач из части С ЕГЭ по информатике представлены на рисунке 1.1.



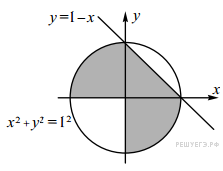
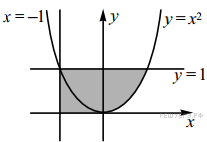


Рисунок 1.1 – Примеры задач из ЕГЭ по информатике

В ЕГЭ требуется составить программный код, который принимает данные (координаты точки) и проверяет ее на принадлежность указанной области. В данном программном продукте необходимо будет указать только логическое выражение, необходимое для проверки.

Логическое выражение в программировании — конструкция языка программирования, результатом вычисления которой является «истина» или «ложь».

В логическом выражении присутствуют операции сравнения, необходимые для выяснений значений логического выражения. Операции сравнения для логических выражений представлены в таблице 1.1.

Таблица 1.1 – Логические операции сравнения

|  |  |
| --- | --- |
| **Значение операции** | **Представление в программе** |
| Равно | == |
| Не равно | != |
| Больше | > |
| Меньше | < |
| Больше или равно | >= |
| Меньше или равно | <= |

Логические выражения, составленные из нескольких частей с операциями сравнения, называются сложными. Для сложных логических выражений характерны операции объединения. Логические операции объединения представлены в таблице 1.2.

Таблица 1.2 – Логические операции

|  |  |
| --- | --- |
| **Значение операции** | **Знак операции** |
| Логическое «И» | Выражение истинно, если оба операнда истинны. |
| Логическое «ИЛИ» | Выражение истинно, если хотя бы один операнд истинен. |
| Логическое «НЕ» (отрицание) | Меняет значение выражения с истинного на ложное, и наоборот. |

Операции в логическом выражении также подчиняются законам приоритета. Наивысший приоритет имеют выражения в скобках. Из операций, приведенных выше, наивысшим приоритетом обладает операция «НЕ», далее идет операция «И» и наименьший приоритет имеет операция «ИЛИ».

При написании логических выражений на языках высокого уровня компилятор данного языка проверяет строку на корректность ввода (не нарушены правила расстановки скобок, используются только те операции, которые заявлены как логические) и выполняют подстановку всех переменных и констант в булево выражение, что позволяет проверить правильность составления выражения на тестовом наборе точек. Также присутствует возможность переставлять местами операнды, меняя логические и арифметические знаки.

Сложность данного метода состоит в том, что проверять логическое выражение в рамках разрабатываемого продукта является затруднительным и ресурсоемким

Поэтому было принято решение, что проверка ответа будет производиться с помощью преобразования выражения в обратную польскую запись, что избавит от скобок, которые могут быть расставлены в местах, не требующих повышения приоритета.

Обратная польская запись - форма записи математических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная польская запись, обратная бесскобочная запись (ОБЗ), постфиксная нотация, бесскобочная символика Лукашевича, польская инверсная запись, ПОЛИЗ. Примером обратной польской записи является выражение , что в обычном представлении имеет вид .

Ввиду того, что обратная польская запись представлена строкой, то и проверка будет происходить посимвольно, поэтому необходимо разработать ряд ограничений, накладываемых на формат ввода ответа, позволяющих уберечь пользователя от некорректного ввода, что в свою очередь приведет к отрицательному результату проверки.

Ввиду сложности реализации аппарата проверки логических выражений было принято решение использовать две элементарные фигуры: окружность и прямую линию.

Уравнение окружности представлено в формуле 1.1.

|  |  |
| --- | --- |
| , | (1.1) |

где – координаты центра окружности; – радиус окружности; – переменные.

Уравнение окружности представлено в формуле 1.2.

|  |  |
| --- | --- |
| , | (1.2) |

где – угловой коэффициент; – свободный член; – переменные.

Задание выстраивается из данных фигур, и количество фигур, одновременно представленных в задании должно быть меньше 5.

Правильной записью для каждой представленной фигуры в задании должно быть уравнение, написанное в формулах 1.1 и 1.2. Любая другая запись будет являться ошибочной с точки зрения проверяющего аппарата.

Само задание должно состоять из двух частей: графическое представление заданной области (сочетание примитивов: окружность, ломаная, многоугольник и линия) и текстовое представление, описывающее необходимую область в Си – подобном стиле. Текстовое представление храниться в памяти компьютера и используется только для проверки ответа, введенного пользователем.

Все задания представлены шаблонами, хранящимися в заранее подготовленной базе шаблонов.

Пример задания, который должен генерироваться программным продуктом представлен на рисунке 1.2.

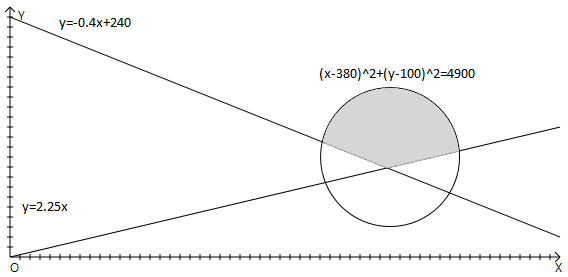


Рисунок 1.2 – Пример генерируемого задания

Для данной области эталонным решением будет являться следующее логическое выражение .

База шаблонов представляет собой XML-файл, разделенный тегами. Тег - элемент языка разметки гипертекста. Названия тегов и их назначение представлено в таблице 1.3.

Таблица 1.3 – Теги и их назначения

|  |  |
| --- | --- |
| **Название** | **Назначение** |
| <base>…</base> | Границы базы с шаблонами. |
| <template>…</template> | Границы шаблона. Имеет свойство «id», отвечающее за номер шаблона. |
| <primitive>…</primitive> | Количество и вид фигуры, используемой в шаблоне. |
| <circle>…</circle> | Окружность. Имеет свойство «id», необходимое для идентификации фигуры, и свойство «link», указывающее, с какой фигурой необходимо связать текущую. |
| <poluline>…</polyline> | Ломаная линия. Имеет свойство «id», необходимое для идентификации фигуры, свойство «link», указывающее, с какой фигурой необходимо связать текущую, и свойство «linecount», указывающее сколько линий будет. |

Продолжение таблицы 1.3.

|  |  |
| --- | --- |
| <polygon>…</ polygon > | Многоугольник. Имеет свойство «id», необходимое для идентификации фигуры, свойство «link», указывающее, с какой фигурой необходимо связать текущую, и свойство «linecount», указывающее сколько линий будет. |
| < area>…</area> | Обозначает границы эталонного решения для текущего шаблона с обозначением мест, куда необходимо вставить уравнения фигур, используемых в задании. |

Все вышеперечисленные теги являются необходимыми, и их искажение приведет к неисправности базы шаблонов.

## **Технология обработки информации**

### **Описание** **вариантов** **использования**

В ходе изучения предметной области выяснилось, что необходим один тип пользователя – пользователь, который может пройти тестирование на умение составлять логические выражения.

Обычный пользователь имеет возможность пройти тестирование (для этого необходимо запросить имя и фамилию), его результаты будут записаны в базу и выведены на экран.

### **Тестирование**

Тестирование проводится для того, чтобы оценить навыки студента в составлении логических выражений для оператора ветвления. Результаты тестирования оцениваются по 4-х балльной шкале (плохо, удовлетворительно, хорошо, отлично). Баллы зависят от процента правильных ответов тестируемого от общего количества вопросов.

Оценочная шкала описана в таблице 1.3.

Таблица 1.5 – Оценочная шкала по умолчанию

|  |  |
| --- | --- |
| **Оценка** | **Количество процентов** |
| Плохо | 0 – 59 |
| Удовлетворительно | 60 – 79 |
| Хорошо | 80 - 89 |
| Отлично | 90 - 100 |

Условия получения той или иной оценки неизменны и являются рекомендациями для преподавателя.

### **Алгоритм обработки базы с шаблонами**

Для представления в компьютере задания создается новый тип данных - Task. Он хранит в себе последовательность примитивов, выделяемую область и правильный ответ.

Заполнение полей экземпляра класса Task’a происходит по следующему алгоритму:

1. прочитать из тэга <template> значение поля “id” и записать в *Task*;
2. из элемента <primitives> прочитать в список примитивов все примитивы, используемые в шаблоне, и их свойства;
3. из элемента <area> прочитать область выделения и записать ее в *Task*;
4. конец.

### **Алгоритм подготовки базы заданий для тестирования**

1. Прочитать в *k* количество заданий в тесте из файла настроек;
2. объявить список заданий для тестирования;
3. открыть базу с шаблонами;
4. повторять от *i* = 0 до *k* с шагом 1
   1. прочитать очередной шаблон и записать его в список заданий;
5. конец.

### **Алгоритм вывода задания на экран**

1. Расставить элементы – примитивы на полотне;
2. в соответствии с заданием, выделить необходимую область на плоскости;
3. каждому элементу-примитиву добавить описание при наведении (указать координаты точек, размеры и прочие свойства, характерные для примитивов);
4. конец.

### **Алгоритм проверки ответа тестируемого**

1. Запросить у тестируемого ответ;
2. получить ответ от тестируемого в виде строки, удовлетворяющей требованиям C++;
3. перевести ответ пользователя в обратную польскую запись;
4. перевести эталонное решение в обратную польскую запись;
5. посимвольно сверить полученные строки из пунктов 3 и 4:
   1. если символы различны, то выдать отрицательный результат;
   2. если дошли до конца строки, а другая строка еще не кончилась, то вывести отрицательный результат;
   3. если обе строки закончились и не было найдено различающихся, то выдать положительный результат;
6. конец.

## **Входные и выходные данные**

Входные данные:

1. ФИО испытуемого (текст)
2. пароль администратора (текст, числа);
3. база с шаблонами заданий (XML файл, отвечающий требованиям таблицы 1.3);
4. файл настроек (XML файл, содержащий пароль и количество заданий в тесте)
5. ответы пользователя (строка, отвечающая требованиям корректности ввода – логическое выражение на Си подобном языке).

Выходные данные:

1. изображение на экране(картинка);
2. результаты теста (текст, числа).

## **Требования к техническому и программному обеспечению**

Разрабатываемая программная система должна соответствовать следующим минимальным требованиям, обусловленным заданием к проекту.

Рекомендуемая аппаратная конфигурация:

* Процессор с частотой 1,6 ГГц или выше
* 512 МБ ОЗУ
* 128 MБ свободного места на диске
* Жесткий диск
* Дисковод CD-ROM/DVD-ROM

Рекомендуемая операционная система:Microsoft Windows XP с пакетом обновлений 3(SP3) или старше.