Курсовая работа

Методические указания по выполнению

Курсовая работа должна выполняться после изучения всего теоретического материала и выполнения лабораторных работ. Курсовая работа состоит в написании программы в соответствии с заданием. В случае наличия ошибок в программе она возвращается на доработку.

**При выполнении работы над ошибками необходимо сохранять замечания преподавателя, а изменения в отчёт вносить другим цветом.**

Выполнение задания включает разработку программного средства, тестирование его на наборе данных и написание отчёта по работе.

Отчёт должен содержать:

1)      титульный лист;

2)      номер варианта и текст задания;

3)      описание алгоритма решения задачи с иллюстрацией его на своём примере;

4)      описание основных блоков программы;

5)      текст программы;

6)      результаты тестирования программы;

7)      распечатку файла результатов – 2–3 примера.

8) Программа должна быть написана в Delphi !

Высылаемый на проверку преподавателю архив должен содержать отчёт и все файлы программы – все модули, exe-модуль, файлы данных, файлы результатов.

**Внимание!!!**

**Необходимо предусмотреть обработку ошибок.           
Никакие входные данные не должны нарушать работу программы!! Программа не должна «зависать» или прекращать выполнение по неизвестной причине – обязательна выдача соответствующей диагностики.**

При выполнении задания следует использовать средства объектно-ориентированного программирования. Выбор конкретного средства разработки оставляется за студентом. При наличии определённых требований к программному обеспечению студенту следует уведомлять об этом преподавателя.

Рекомендуется при разработке программного средства использовать материалы лабораторных работ (в зависимости от темы задания).

Программа должна управляться посредством меню, в котором должны присутствовать следующие пункты: "Автор", "Тема" (с полной информацией о разработчике и теме задания), "Данные" (выбор способа задания исходных данных – чтение из файла или ввод с клавиатуры), "Расчёты", "Запись результатов в файл" – и другие, определяемые конкретным заданием. При вводе данных с клавиатуры необходимо использовать соответствующую форму, а также предусмотреть возможность вызова справки с примером формата данных. При чтении из файла – должна открываться своя папка. Все результаты расчётов должны отображаться на экране и выводиться в файл (по требованию пользователя). При введении автором каких-либо ограничений (размер алфавита и т.п.) они должны быть описаны в пояснительной записке и в соответствующем пункте меню.

Задание

Написать программу для автоматического построения регулярного выражения (РВ) по словесному описанию языка.

Вход программы: алфавит языка, обязательная начальная подцепочка, выбранный символ алфавита, его кратность (натуральное число), 2 числа – диапазон длин для генерации цепочек.

Выход: построенное регулярное выражение, результат генерации цепочек.

Подробно:

Язык задан своим алфавитом, обязательной начальной цепочкой и указанием кратности вхождений некоторого символа во все цепочки языка. В начальной цепочке не должно находиться символов, не содержащихся в алфавите. В крайнем случае она может быть и пустой. Кратность числа вхождений выбранного символа задаётся любым натуральным числом. Кратность, равная единице, равносильна отсутствию ограничений на вхождения данного символа, т.е. он может встречаться в цепочках любое количество раз. При любом значении кратности количество вхождений символа в цепочку может быть и нулевым.

Программа должна:

1.            по предложенному описанию регулярного языка строить РВ, генерирующее этот язык, в том виде, как было рассмотрено в теории, раздел 2.1.1;

***(2.1.1 Определение и свойства регулярных выражений***

Проблема генерирования бесконечных языков посредством конечных описаний решается различными способами. Одним из них является использование регулярных выражений для порождения бесконечных цепочек языка.

*Регулярное множество* и *регулярное выражение* для некоторого алфавита V определяется рекурсивно следующим образом:

* 0 – регулярное выражение, обозначает Æ регулярное множество;
* l – регулярное выражение, обозначает регулярное множество {l };
* " aÎ V a – регулярное выражение, обозначает регулярное множество {a};
* если p и q – произвольные регулярные выражения, обозначающие регулярные множества P и Q, то p+q, pq, p\* – регулярные выражения, обозначающие соответственно регулярные множества PÈ Q, PQ, P\*;
* ничто другое регулярным выражением и регулярным множеством не является.

Иными словами, *регулярные множества* – это цепочки символов над заданным алфавитом, построенные с использованием операций объединения, конкатенации и замыкания.

Все регулярные языки представляют собой регулярные множества.

Два регулярных выражения a и b *эквивалентны*: a  = b , если они обозначают одно и то же множество.

Каждое регулярное выражение обозначает одно и только одно регулярное множество, но для одного регулярного множества может существовать сколько угодно задающих его регулярных выражений.

При записи регулярных выражений используются круглые скобки, как для обычных арифметических выражений. При отсутствии скобок операции выполняются слева направо с учетом приоритета. Наивысшим приоритетом обладает операция итерации, затем конкатенации, потом – объединение множеств.

Например, язык, представляющий собой множество всех цепочек из нулей и единиц произвольной длины, может быть описан эквивалентными регулярными выражениями α1 и α2: α1 = (0+1)\*, α2 = (0\*1\*)\*. α1 = α2 .

Пусть a , b , g – регулярные выражения. Тогда свойства регулярных выражений можно записать в виде следующих формул:

|  |  |  |
| --- | --- | --- |
| 1. l + a a \*= l + a \*a = a \* 2. a + b = b + a 3. a + (b +g )= (a + b )+g 4. a (b +g )= a b +a g 5. (b +g )a =b a +g a |  a (b g )= (a b )g   a + a = a   a + a \*= a \*   l +a \*=a \*+l =a \*   0\*=l |  0a =a 0=0   0+a =a +0=a   l a =a l =a   (a \*)\*= a \* |

Все эти свойства доказываются с применением аппарата теории множеств, т.к. регулярные выражения – это обозначения для соответствующих множеств.)

2.            с помощью построенного РВ генерировать все цепочки языка в заданном пользователем диапазоне длин.

При генерации цепочек языка по РВ можно использовать любые структуры и алгоритмы, в том числе и прямой перебор.

Рассмотрим пример построения регулярного выражения.

Задан язык: алфавит {0,1,a,b}, обязательная начальная цепочка ‘01a’, количество вхождений символа ‘а’ кратно 2. Анализируем задание: язык будет состоять из цепочек с чётным количеством символов ‘а’, начинающихся с ‘01a’, например {01аa, 01аba, 01аab, 01аac, 01аaaa,…}. Для начала определим минимально возможную цепочку этого языка. Это ‘01aa’ – она удовлетворяет всем условиям и является при этом самой короткой. Это основа нашего РВ.

Удлинить эту цепочку можно, поместив после неё или между цепочкой ’01a’ и ‘a’ в ней любое количество любых символов алфавита, за исключением символа ‘a’ – для него нужно подсчитывать количество, поэтому его следует рассматривать особо. Заданную цепочку ’01a’ разрывать нельзя. Для генерации любого количества символов используется итерация, а выбор одного из нескольких символов записывается в виде суммы.

Итак, получим 01a(0+1+b)\*a(0+1+b)\*. Но по построенному таким образом выражению нельзя получить много символов ‘a’. Для увеличения количества ‘a’ можно добавить после построенного выражения (aa)\* для сохранения чётности. Но тогда нельзя будет между символами ‘a’ поместить другие символы алфавита. Для того чтобы сделать это возможным, заменим добавляемую конструкцию на ((0+1+b)\*a(0+1+b)\*a(0+1+b)\*)\*, сохранив чётность ‘a’ и добавив любое количество других символов. Итоговое регулярное выражение будет иметь вид:   
01a(0+1+b)\*a(0+1+b)\*((0+1+b)\*а(0+1+b)\*a(0+1+b))\*. Можно несколько упростить построенное выражение, убрав одну из скобок (0+1+b)\* – там, где она повторяется. Окончательный ответ: 01a(0+1+b)\*a(0+1+b)\*(а(0+1+b)\*a(0+1+b))\***.**