**ЗАДАНИЕ**

Написать программу для автоматического построения регулярной грамматики (леволинейной или праволинейной), эквивалентной заданному конечному автомату (КА).

Вход программы: КА в виде: алфавит языка, множество состояний, начальное состояние, множество заключительных состояний, функция переходов в табличном виде; тип грамматики (ЛЛ либо ПЛ), 2 числа – диапазон длин для генерации цепочек.

Выход: построенная грамматика (все 4 элемента), результат генерации цепочек.

Подробно:

Язык задан конечным автоматом. Автомат может быть полностью или неполностью определённым, детерминированным или недетерминированным, заключительных состояний может быть несколько.

Программа должна:

1.            по предложенному распознавателю регулярного языка в виде КА строить регулярную грамматику, генерирующую этот язык, в том виде, как она рассматривалась в теории, раздел 1.3.1;

2.            с помощью построенной грамматики генерировать все цепочки языка в заданном пользователем диапазоне длин.

Грамматика должна строиться и праволинейная, и леволинейная. Желаемый тип грамматики выбирается пользователем в меню. Все правила грамматики должны соответствовать выбранному типу! Отдельно следует указывать, какой нетерминальный символ является целевым. Если в грамматике используется пустое правило, то необходимо дать пояснение, каким именно символом обозначается пустая цепочка.

После построения грамматики пользователь может убедиться в её правильности путём генерации всех цепочек языка в том диапазоне длин, который он задаст. Генерацию каждой цепочки языка следует поэтапно отображать на экране в виде цепочки вывода (в соответствии с примерами раздела 1.4.1.). Генерация осуществляется в соответствии с лабораторной работой №1.

Рассмотрим пример построения регулярной грамматики.

|  |  |  |
| --- | --- | --- |
|  | ***вход*** | Задан ДКА M({S,A,B},{0,1,2},,S,{B}), функция переходов  представлена в табличном виде. Построить праволинейную грамматику. В правой части её правил вывода может быть только один нетерминал, и он должен располагаться справа от терминальных символов. |
| ***состояние*** | 0 | 1 | 2 |
| S | {S} | {A} | – |
| A | {B} | – | {B} |
| B | {A} | – | {A} |

Нетерминалами будут являться состояния ДКА: {S,A,B}, заключительному состоянию должно соответствовать пустое правило. Целевым символом будет нетерминал, соответствующий начальному состоянию исходного ДКА. Правила грамматики строятся по строкам таблицы переходов следующим образом: если Х(Y,t),X,YQ, t – входной символ, то соответствующее правило праволинейной грамматики будет иметь вид Y → tX.

Итак, грамматика имеет вид: G({0,1,2},{S,A,B},P,S), Р: S→0S|1А; A→0B|2B; B→0A|2A|.

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.

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.