Теория языков программирования и методы трансляции

Среда программирования – Рекомендуется использовать Delphi, C# или аналоги.

Внимание!   
При выполнении лабораторных работ необходимо предусматривать обработку любых возможных ошибок ввода. Программа не должна «зависать» или вести себя иным некорректным образом ни при каких начальных данных! При вводе неправильных начальных данных должно быть выведено сообщение об ошибке пользователя и предложено повторить ввод правильно (как вариант – можно программно запрещать ввод неправильных данных).

По каждой лабораторной работе необходимо выполнять отчёт, включающий в себя:

1. постановку задачи;
2. описание входных данных программы и её результатов;
3. описание основных переменных, а также основных блоков и подпрограмм;
4. алгоритм решения задачи;
5. текст программы;
6. результаты работы, если они есть (вариант – копия экрана), или результаты тестирования;
7. ответы на контрольные вопросы.

Преподавателю на проверку необходимо предоставлять отчёт, исходный код и откомпилированный модуль (exe-файл).

Лабораторная работа № 1 *Генерация цепочек языка*

Пусть язык задан контекстно-свободной грамматикой (теоретический материал разделов 1.1–1.4). Написать программу, которая по заданной грамматике будет генерировать ВСЕ цепочки языка в некотором диапазоне длин. Использовать только левосторонний или правосторонний вывод! Диапазон длин генерируемых цепочек должен задаваться пользователем при запуске программы.

Предусмотреть возможность выбора пользователю – использовать заданную в программе грамматику или вводить свою с клавиатуры.

На вход программы подаётся КС-грамматика (терминальный и нетерминальный алфавиты, целевой символ, правила вывода), задаётся диапазон длин цепочек, указывается тип вывода (левосторонний или правосторонний).

Рекомендуется для ввода исходных данных использовать соответствующую форму. При вводе правил грамматики не рекомендуется предоставлять пользователю излишнюю свободу действий, например, удобнее запретить ввод в левой части правил более чем одного нетерминального символа, чем проверять введённые правила на соответствие контекстно-свободному типу.

На выходе должен быть список построенных цепочек. Все цепочки в списке должны быть различны. При получении одинаковых цепочек (разными способами) их следует сохранять в списке выводимых цепочек только в случае выполнения дополнительного задания.

Проверить программу на примерах из лекционного курса и на заданиях из контрольных вопросов к теоретической части.

**Контрольные вопросы:**

1. Как поведёт себя программа, если при вводе правил грамматики сделать ошибку – ввести правила грамматики, не соответствующей КС-типу, или не использовать какие-то символы исходных алфавитов, или поместить в левой части правил терминальный символ?
2. Используется ли при генерации цепочек рекурсия? Если да, то как ограничена её глубина?
3. Какой язык порождается грамматикой G({0},{S,A},{S→AAA, A→AAA |0}, S)? Правильно ли Ваша программа работает на этом примере? Если нет – доработайте программу.
4. Разрешает ли Ваша программа использовать в исходной грамматике пустые правила? Если нет – доработайте программу.

### Лабораторная работа № 2 *Моделирование работы ДКА*

Пусть регулярный язык задаётся конечным автоматом – ДКА (теоретический материал разделов 1.5, 2.2). Написать программу, которая будет проверять по заданному автомату вводимую цепочку и делать вывод о том, принадлежит ли она рассматриваемому регулярному языку. В случае отрицательного ответа необходимо давать пояснение, по какой причине цепочка не принадлежит языку – например, «в цепочке присутствуют посторонние символы», «после прочтения цепочки автомат не пришёл в конечное состояние» и т.п. Исходный автомат вводить с клавиатуры в соответствии с определённым форматом. Ввод цепочек также производить с клавиатуры.

На вход программы подаётся ДКА (множество состояний, алфавит языка, начальное состояние, множество заключительных состояний, функция переходов в виде таблицы) и проверяемая цепочка символов (может вводиться многократно, т.е. возможно проверить любое количество цепочек). При этом в проверяемую цепочку могут входить и символы, не принадлежащие алфавиту языка; цепочка может быть и пустой.

Программа должна предоставлять пользователю возможность изменять начальное и конечные состояния с сохранением введённой функции переходов для заданного автомата.

Выход – отображение на экране процесса проверки цепочки в виде последовательной смены конфигураций и вывод результата – сообщения, принадлежит ли цепочка языку.

Проверить работу программы на примерах из лекций и задачах из контрольных вопросов./p>

**Контрольные вопросы:**

1. Как поведёт себя программа, если при вводе таблицы переходов ДКА сделать (случайно или преднамеренно) ошибку – например, ввести несуществующее состояние?
2. Все ли ячейки таблицы переходов исходного ДКА обязательно должны быть заполнены или можно использовать неполностью определённый ДКА?
3. В каком случае ДКА распознаёт пустую цепочку как цепочку языка?

### Лабораторная работа № 3 *Перевод с помощью МП-преобразователя*

Пусть дан преобразователь с магазинной памятью; написать программу, которая будет выполнять перевод цепочек с одного языка на другой с помощью заданного преобразователя (теоретический материал раздела 4.2). При невозможности выполнить перевод (цепочка не принадлежит исходному языку) необходимо выводить на экран соответствующее сообщение. Исходный преобразователь вводить с клавиатуры в соответствии с определённым форматом. Ввод цепочек также производить с клавиатуры, выполняя его до тех пор, пока не возникнет желание закончить работу. Процесс перевода цепочки в виде последовательной смены конфигураций отображать на экране.

На вход программы подаётся ДМП-преобразователь (множество состояний, алфавиты входного и выходного языков, алфавит магазина, начальное состояние, начальное содержимое стека, множество заключительных состояний, функция переходов в виде списка правил) и проверяемая цепочка символов (аналогично лаб. раб. №2).

Рекомендуется за основу взять программу лабораторной работы №2, дополнив исходные данные алфавитом стека, начальным содержимым стека и выходным алфавитом, функцию переходов – в соответствии с определением преобразователя, а конфигурации – содержимым памяти и выходными цепочками.

Выход: отображение на экране процесса перевода цепочки в виде последовательной смены конфигураций преобразователя, результат перевода – полученная цепочка.

Проверить работу программы на примерах из лекций и задачах из контрольных вопросов.

**Контрольные вопросы:**

1. Как поведёт себя преобразователь, если поданная на его вход цепочка не может быть распознана (не принадлежит к заданному языку)?
2. Как соотносятся алфавиты исходного языка и того, на который выполняется перевод (должны совпадать, различаться…)? Поясните ответ.
3. Как поведёт себя Ваша программа при некорректном вводе? Например, функция переходов задана не в том формате, определена не для того количества параметров, использует алфавит, отличный от заданных…