Теория языков программирования и методы трансляции

Задание на курсовую работу для студентов факультета ИВТ

Каждый студент выполняет задание согласно своему номеру варианта по журналу. Номер задания – число в скобках, выделенное жирным шрифтом. Выполнение задания включает разработку программного средства, тестирование его на наборе данных и написание отчёта о работе. Отчёт должен содержать:

1) титульный лист;
2) номер варианта и текст задания;
3) описание алгоритма решения задачи с иллюстрацией на примере;
4) описание основных блоков программы;
5) распечатку текста программы;
6) результаты тестирования.

Вместе с отчётом необходимо сдать преподавателю электронный вариант программы – все исходники, exe-модуль, файлы данных, файлы результатов.

Задание выдаётся на 13-й – 14-й неделе. Срок сдачи готовой программы – не позднее 16-й недели. Сдача курсовой работы происходит по этапам:

1. Демонстрация преподавателю работающей программы.
2. Оформление и сдача отчёта и электронной версии программы.
3. Защита курсовой работы.

В некоторых случаях защита может быть совмещена с этапом 1.

Разработанное программное приложение должно иметь графический интерфейс. Выбор конкретного средства разработки оставляется за студентом.

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

Программа должна управляться посредством меню с пунктами "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 другими, определяемыми конкретным заданием. При вводе данных с клавиатуры использовать соответствующую форму, а также предусмотреть возможность вызова справки с примером формата данных. Во всех вариантах заданий все результаты расчётов должны отображаться на экране и выводиться в файл (по требованию пользователя).

Замечание: Необходимо предусмотреть обработку ошибок. Программа не должна "зависать" или прекращать выполнение по неизвестной причине - обязательна выдача соответствующей диагностики.

При введении автором каких-либо ограничений на исходные данные (размер алфавита и т.п.) они должны быть описаны в пояснительной записке и в справке.

ВНИМАНИЕ!!! Наиболее подготовленные и интересующиеся предметом студенты (минимальный критерий – выполнение всех текущих заданий в срок) могут по собственному выбору и индивидуальной договоренности с преподавателем СВОЕВРЕМЕННО заменить своё задание на другое, не входящее в число предложенных вариантов, но относящееся к изучаемой тематике.

Тема 1 Построение конструкций, задающих язык. Варианты 1 – 10.

1 Написать программу, которая по предложенному описанию языка построит детерминированный конечный автомат, распознающий этот язык, и проверит вводимые с клавиатуры цепочки на их принадлежность языку. Предусмотреть возможность поэтапного отображения на экране процесса проверки цепочек. Функция переходов ДКА может изображаться в виде таблицы и графа (выбор вида отображения посредством меню). Варианты задания языка:

(1) Алфавит, обязательная конечная подцепочка всех цепочек языка и кратность вхождения выбранного символа алфавита в любую цепочку языка.

(2) Алфавит, начальная и конечная подцепочки всех цепочек языка.

(3) Алфавит, обязательная фиксированная подцепочка и кратность длины всех цепочек языка.

2 Написать программу, которая по предложенному описанию языка построит регулярную грамматику (ЛЛ или ПЛ – по заказу пользователя), задающую этот язык, и позволит сгенерировать с её помощью все цепочки языка в заданном диапазоне длин. Предусмотреть возможность поэтапного отображения на экране процесса генерации цепочек. Варианты задания языка:

(4) Алфавит, начальная и конечная подцепочки и кратность длины всех цепочек языка.

(5) Алфавит, обязательная фиксированная подцепочка и кратность длины всех цепочек языка.

(6) Алфавит, кратность вхождения некоторого символа алфавита и обязательная фиксированная подцепочка, на которую заканчиваются все цепочки языка.

3 Написать программу, которая по предложенному описанию языка построит регулярное выражение, задающее этот язык, и сгенерирует с его помощью все цепочки языка в заданном диапазоне длин. Предусмотреть также возможность генерации цепочек по введённому пользователем РВ (в рамках варианта). Варианты задания языка:

(7) Алфавит, кратность длины и заданная фиксированная подцепочка всех цепочек языка.

(8) Алфавит, заданные начальная и конечная подцепочки и кратность длины всех цепочек языка.

(9) Алфавит, кратность вхождения некоторого символа алфавита во все цепочки языка и заданная подцепочка всех цепочек языка.

(10) Алфавит, заданные начальная и конечная подцепочки и кратность вхождения некоторого символа алфавита во все цепочки языка.

Тема 2 Преобразование конструкций, задающих язык. Варианты 11 – 15.

(11) Написать программу, которая по заданному детерминированному конечному автомату построит эквивалентную регулярную грамматику (ЛЛ или ПЛ по желанию пользователя). Функцию переходов ДКА задавать в виде таблицы, но предусмотреть возможность автоматического представления её в графическом виде. Программа должна сгенерировать по построенной грамматике несколько цепочек в указанном диапазоне длин и проверить их допустимость заданным автоматом. Процессы построения цепочек и проверки их выводимости отображать на

экране (по требованию). Предусмотреть возможность проверки автоматом цепочки, введённой пользователем.

(12) Написать программу, которая по заданной регулярной грамматике (ЛЛ или ПЛ по желанию пользователя) построит эквивалентное регулярное выражение. Программа должна сгенерировать по заданной грамматике и построенному регулярному выражению множества всех цепочек в указанном диапазоне длин, проверить их на совпадение. Процесс построения цепочек отображать на экране. Для подтверждения корректности выполняемых действий предусмотреть возможность генерации цепочек по введённому пользователем регулярному выражению. При обнаружении несовпадения в элементах множеств должна выдаваться диагностика различий – где именно несовпадения и в чём они состоят.

(13) Написать программу, которая по заданной регулярной грамматике (грамматика может быть НЕ автоматного вида!, ЛЛ или ПЛ) построит эквивалентный ДКА (представление функции переходов в виде таблицы). Программа должна сгенерировать по исходной грамматике несколько цепочек в заданном диапазоне длин и проверить их допустимость построенным автоматом. Процессы построения цепочек и проверки их выводимости отображать на экране (по требованию). Предусмотреть возможность проверки цепочки, введённой пользователем.

(14) Написать программу, которая по заданному регулярному выражению построит эквивалентную грамматику (по желанию разработчика – грамматика может быть контекстно-свободной или регулярной). Программа должна сгенерировать по построенной грамматике и регулярному выражению множества всех цепочек в указанном диапазоне длин, проверить их на совпадение. Процесс построения цепочек отображать на экране. Для подтверждения корректности выполняемых действий предусмотреть возможность корректировки любого из построенных множеств пользователем (изменение цепочки, добавление, удаление…). При обнаружении несовпадения в элементах множеств должна выдаваться диагностика различий – где именно несовпадения и в чём они состоят.

(15) Написать программу, которая по заданному регулярному выражению построит эквивалентный ДКА. Функция переходов ДКА может изображаться в виде таблицы и графа (выбор вида отображения посредством меню). Программа должна сгенерировать по РВ несколько цепочек в заданном диапазоне длин и проверить их допустимость построенным автоматом. Процесс разбора цепочек автоматом отображать на экране (по требованию). Предусмотреть возможность разбора цепочки, введённой пользователем. В качестве исходных данных допускаются РВ, порождающие цепочки, имеющие определенное количество циклических повторений всех символов алфавита или некоторой их части, заканчивающиеся на заданную цепочку. Например, (a+b+c)*aaca, или ((a+b)(a+b))*aacb, и т.п.

Тема 3 Преобразования КС-грамматик и виды разбора. Варианты 16 – 20.

(16) Написать программу, которая будет принимать на вход контекстно-свободную грамматику в каноническом виде (проверить корректность задания и при отрицательном результате выдать соответствующее сообщение) и приведёт её к нормальной форме Хомского. Программа должна проверить построенную грамматику (БНФ) на эквивалентность исходной: по обеим грамматикам сгенерировать множества всех цепочек в заданном пользователем диапазоне длин и проверить их на идентичность. Для подтверждения корректности выполняемых действий

предусмотреть возможность корректировки любого из построенных множеств пользователем (изменение цепочки, добавление, удаление…). При обнаружении несовпадения должна выдаваться диагностика различий – где именно несовпадения и в чём они состоят. Построить дерево вывода для любой выбранной цепочки из числа сгенерированных.

(17) Написать программу, которая будет принимать на вход произвольную контекстно-свободную грамматику и выполнит преобразование её к каноническому виду. Преобразование осуществлять поэтапно, отображая результат на каждом из этапов. Программа должна проверить построенную грамматику на эквивалентность исходной: по обеим грамматикам сгенерировать множества всех цепочек в заданном пользователем диапазоне длин и проверить их на идентичность. Для подтверждения корректности выполняемых действий предусмотреть возможность корректировки любого из построенных множеств пользователем (изменение цепочки, добавление, удаление…). При обнаружении несовпадения должна выдаваться диагностика различий – где именно несовпадения и в чём они состоят.

(18) Написать программу, которая для языка, заданного контекстно-свободной грамматикой в требуемой форме (проверить корректность задания и при отрицательном результате произвести соответствующее преобразование), построит детерминированный распознаватель со стековой памятью, используя алгоритм нисходящего анализа с возвратами. Программа должна сгенерировать по исходной грамматике несколько цепочек в указанном пользователем диапазоне длин и проверить их допустимость построенным ДМПА. Процессы построения цепочек и проверки их выводимости отображать на экране (по требованию). Предусмотреть возможность проверки цепочки, введённой пользователем.

(19) Написать программу, которая для языка, заданного контекстно-свободной грамматикой в требуемой форме (проверить корректность задания и при отрицательном результате выдать соответствующее сообщение), построит детерминированный распознаватель с магазинной памятью, используя алгоритм восходящего анализа с возвратами («сдвиг-свертка»). Программа должна сгенерировать по исходной грамматике несколько цепочек в указанном диапазоне длин и проверить их допустимость построенным ДМПА. Процессы построения цепочек и проверки их выводимости отображать на экране (по требованию). Предусмотреть возможность проверки цепочки, введённой пользователем.

(20) Написать программу, которая для языка, заданного контекстно-свободной грамматикой в требуемой форме (проверить корректность задания и при отрицательном результате выдать соответствующее сообщение), построит модель табличного распознавателя (алгоритм Кока-Янгера-Касами). Программа должна сгенерировать по исходной грамматике несколько цепочек в указанном диапазоне длин и проверить их допустимость, построить цепочку вывода. Полученные цепочки и проверку их выводимости (включая построение промежуточной таблицы T) отображать на экране. Предусмотреть возможность проверки цепочки, введённой пользователем.

****

Рекомендуемая литература:

1. А. Ахо, Дж. Хопкрофт, Дж. Ульман. Построение Структуры данных и алгоритмы. - М.- Мир. - 2003 г.

2. А.Ахо, Дж. Ульман. Теория синтаксического анализа, перевода и компиляции. Т.1,2. - М. - Мир.-1978.
3. А.В. Гордеев, А.Ю. Молчанов. Системное программное обеспечение. – СПБ. – Питер. – 2001.
4. Молчанов А.Ю. Системное программное обеспечение: Учебник для вузов. - 2003.
5. Хантер Р. Основные концепции компиляторов. - 2002.
6. Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. – 2006.