**Теория языков программирования и методы трансляции СИБГУТИ**

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.

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.

**Вариант 10**

Написать программу для автоматического построения регулярной грамматики (леволинейной или праволинейной) по словесному описанию языка.

Вход программы: алфавит языка, обязательная конечная подцепочка, кратность длины всех цепочек языка, тип грамматики (ЛЛ либо ПЛ), 2 числа – диапазон длин для генерации цепочек.

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

Подробно:

Язык задан своим алфавитом, обязательной конечной подцепочкой, которая должна присутствовать во всех цепочках языка, и указанием кратности длины всех цепочек языка. В конечной цепочке не должно находиться символов, не содержащихся в алфавите. В крайнем случае она может быть и пустой. Кратность длины цепочек задаётся любым натуральным числом. Кратность, равная единице, равносильна отсутствию ограничений на длину цепочек.

Программа должна:

1.            по предложенному описанию регулярного языка 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.

Задан язык: алфавит {a,b,c}, обязательная конечная цепочка ‘aab’, кратность длины 2. Анализируем задание: язык будет состоять из цепочек чётной длины, заканчивающихся на ‘aab’, например {сaab, аaab, bааb, caaааb, сabaab, …}.

Для примера выберем праволинейную грамматику (ПЛ). В правой части её правил вывода может быть только один нетерминал, и он должен располагаться справа от цепочек терминальных символов. Для соблюдения кратности длины цепочек будет использоваться чередование нетерминалов. Длина кратна двум, следовательно, потребуется два нетерминала:  S→aA|bA|cA,  А→aS|bS|cS.  Выход должен происходить по обязательной конечной цепочке ‘aab’, в силу чётности длины цепочек перед ней должен быть как минимум один символ. Следовательно, эта цепочка должна порождаться нетерминалом А: А→aab.

Итак, построенная грамматика имеет вид:
G({a,b,c},{S,A},P,S),  Р:  S→aA|bA|cA;  А→aS|bS|cS|aab.