**[Открыть Курсовая работа в формате Microsoft Word](file:///C%3A%5C%5CUsers%5C%5Caleks%5C%5CDownloads%5C%5Ccourse271%5C%5Ckurs.doc%22%20%5Ct%20%22_blank)**

**Курсовая работа**

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 4**

Написать программу для автоматического построения детерминированного конечного автомата (ДКА), эквивалентного заданной регулярной грамматике.

Вход программы: терминальный и нетерминальный алфавиты грамматики, целевой символ, правила грамматики, цепочки для распознавания.

Выход: построенный ДКА (все 5 элементов), результат проверки цепочек.

Подробно:

Язык задан регулярной грамматикой, причём она может быть не автоматного вида. При написании программы разработчику разрешается выбрать один из двух типов регулярной грамматики (ЛЛ или ПЛ) и следует информировать об этом пользователя. Терминальный алфавит грамматики может включать в себя любые символы, в нетерминальном алфавите могут использоваться заглавные буквы латинского алфавита или (на усмотрение разработчика) слова. Правила задаваемой грамматики должны соответствовать выбранному типу. Для того чтобы в исходной грамматике можно было использовать пустое правило, необходимо предусмотреть поле ввода для символа, которым пользователь может обозначить пустую цепочку.

Программа должна:

1.            по заданной регулярной грамматике строить эквивалентный ДКА, распознающий этот же язык, в том виде, как он рассматривался в теории, раздел 2.2.2;

2.            с помощью построенного ДКА проверять вводимые пользователем цепочки на их принадлежность этому языку.

ДКА должен распознавать язык, задаваемый исходной грамматикой, т.е. являться эквивалентной конструкцией. Функция переходов ДКА может изображаться в виде таблицы или графа, вариант вида её представления выбирается разработчиком. Для удобства построения автомата рекомендуется предварительно привести заданную грамматику к автоматному виду (в соответствии с лекционным разделом 2.2.1).

При выборе такого способа построения ДКА, когда сначала по заданной грамматике строится эквивалентный НКА, а затем он приводится к детерминированному виду, промежуточный результат в виде НКА необходимо также отображать на экране по просьбе пользователя.

После построения ДКА пользователь может вводить произвольные цепочки для проверки их на принадлежность исходному языку. Разбор цепочек автоматом следует поэтапно отображать на экране в виде последовательной смены конфигураций в соответствии с лабораторной работой №2.

Рассмотрим пример построения ДКА.

Язык задан праволинейной грамматикой: G({a,b,c},{S,A},P,S),
Р: S→aA|bA|cA; А→aS|bS|cS|aab.

Сначала следует привести грамматику к автоматному виду, добавив 2 нетерминала для разделения на символы цепочки ‘aab’. Правила примут вид:
Р`: S→aA|bA|cA; А→aS|bS|cS|aB; B→aC; C→b. Теперь нужно по правилам грамматики построить конечный автомат, в котором состояния будут получены из нетерминалов грамматики, а переходы определяются терминальными символами. Заключительным будет являться то состояние, в которое происходит переход по одному символу (C→b), следовательно, придётся добавить ещё одно состояние D. Функцию переходов сначала будем строить графически:

|  |
| --- |
|  |
|  | C:\Users\aleks\Downloads\course271\img\p03.png |

Как видно из графа переходов, автомат получился недетерминированный (из состояния А есть два перехода по символу ‘a’). Далее можно применить алгоритм преобразования НКА к ДКА, рассмотренный подробно в лекциях, раздел 2.2.2.

Построим таблицу переходов нашего автомата и преобразуем его в ДКА согласно вышеупомянутому алгоритму.

|  |  |  |
| --- | --- | --- |
|  | ***вход*** | Исходную таблицу переходов отделим от остальной части жирной линией. Для упрощения процесса будем создавать не все возможные сочетания исходных состояний, а только те, которые реально возникают при построении. Сначала занесём в таблицу SB. Затем появляются AC, SD. Состояния исходного автомата B, C, D (выделены синим) оказались недостижимыми. Удалим их.  |
| ***состояние*** | *а* | *b* | *c* |
| S | {A} | {A} | {A} |
| A | {S,B} | {S} | {S} |
| B | {C} | – | – |
| C | – | {D} | – |
| D | – | – | – |
| SB | B | {AC} | {A} | {A} |
| AC | C | {SB} | {SD} | {S} |
| SD | D | {A} | {A} | {A} |

Новые состояния для удобства переобозначим B, C, D. Заключительным состоянием станет состояние D (поскольку в состояние SD входит то состояние D, которое было заключительным в исходном автомате). Новая таблица переходов примет следующий вид:

|  |  |  |
| --- | --- | --- |
|  | ***вход*** | Построение графа переходов по таблице:  |
| ***состояние*** | *a* | *b* | *c* |
| S | {A} | {A} | {A} |
| A | {B} | {S} | {S} |
| B | {C} | {A} | {A} |
| C | {B} | {D} | {S} |
| D | {A} | {A} | {A} |

 

Итак, ДКА построен и имеет вид: M({S,A,B,C,D},{a,b,с},,S,{D}), где функция переходов  получена выше.