Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра вычислительной техники

Отчет по лабораторной работе № 2

по дисциплине «Теория формальных языков и компиляторов»

на тему «Лексика языков программирования. Конечные автоматы  
без памяти для обнаружения слов в тексте программы»

Студент: Резниченко М. К.

Группа: АВТ-709

Преподаватель: Малявко А.А.

Вариант: 33143423

Новосибирск, 2020

# Цель работы

Изучение конечных автоматов (КА) без памяти, способов определения КА – канонического, графового и табличного, методов построения недетерминированного КА по системе регулярных выражений, методов эквивалентных преобразований недетерминированных КА в оптимальные полностью определенные КА – лексические акцепторы.

# Постановка задачи

1. Используя пакет ВебТрансЛаб, освоить:

- создание лексических правил на языке регулярных выражений (РВ);

- использование операций «+, \*, ?, конкатенации и выбора» языка РВ для построения сложных регулярных выражений;

- преобразование простой системы РВ в одноавтоматный лексический акцептор;

- добавление правил и действий в систему РВ для построе-ния мультиавтоматного лексического акцептора;

2. Разработать (доработать разработанный при выполнении работы №1) фрагмент системы регулярных выражений для всех (или выбранной самостоятельно части) групп слов языка, определенного заданием на курсовую работу. Построить по этому фрагменту:

- программный модуль, управляемый графом состояний и переходов;

- программный модуль, управляемый таблично;

3. Изучить структуру программных модулей, построенных ВебТрансЛабом, изучить алгоритмы работы лексического акцептора для графового и табличного способов реализации КА, сравнить реализации конечных автоматов, управляемых различными способами, между собой, оценить их затраты времени в шагах на распознавание слов;

4. Изучить по тексту программного модуля способ реализации вызова действий, определенных в лексических правилах и алгоритм работы формирователя лексем.

5. Проверить функционирование конечных автоматов, построенных ВебТрансЛабом (подготовить тестовый пример программы на языке, заданном на курсовую работу, запустить каждый автомат на выполнение, протрассировать вручную работу лексического акцептора в графовой и табличной реализации, убедиться в работоспособности автоматов, в противном случае – доработать систему РВ и добиться правильного функционирования лексического акцептора).

Лексика языка программирования, заданного вариантом 33143423:

1. Идентификаторы:  
   <б><пЦ><пБ> (a13KL, w8U, …)
2. Константы:  
   целые по основаниям 4, 8 и 10, вещественные, строковые.
3. Объявления примитивных типов:  
   – целое – long, long\_u  
   – вещественное – number  
   – строковые – char
4. Оператор присваивания:  
   put <В> to <И>;
5. Условный оператор:  
   when <ЛВ> then <ОБ> [other <ОБ>];
6. Оператор цикла:  
   exec <ОБ> with <И> from <К> to <К> [step <К>];
7. Оператор переключателя:  
   case <В> { when <К> then <ОБ> [exit;] … [other <ОБ> ]}

# Описание лексики языка

Исходный код программы может состоять из нескольких файлов и содержать следующие элементы в заданном порядке: указание пакета программы (опционально), указание импортов других пакетов с функциями (опционально), объявления переменных верхнего уровня и/или объявления функций.

**Пакет программы** указывается в следующем виде: *package <name>;*. Имя пакета <*name>* должно состоять их двух непустых последовательностей, состоящих из латинских символов и символа \_, разделенных точкой. Пакет программы указывает, как функции из других файлов могут обратиться к функциям в данном файле, и используется в импортах.

**Импорты функций** указываются в следующем виде: *import <package>.<function>;*, где *<package>* - имя пакета, *<function>* - идентификатор функции. Импорты функций позволяют функциям в данном файле использовать функции из других файлов, обозначенных пакетами.

**Блок кода** – набор команд, заключенных в фигурные скобки *{* *}*.

**Объявления переменных** указываются в следующем виде: *<type> <name>;*, где *<type>* - тип переменной, *<name>* - идентификатор переменной. Переменные могут использоваться только после их объявления, обращения к переменным возможны только в данном блоке кода или во вложенных блоках кода, в том числе внутри функций (для переменных, объявленных вне функций) и внутри операторов. Повторное объявление переменной с тем же идентификатором внутри одного блока кода запрещено. Обращение к переменной, объявленной вне конкретного блока кода, невозможно, если внутри этого блока кода до обращения объявлена переменная с таким же идентификатором.

**Типы переменных:**

**– целые по основанию 4** – константы данного типа должны начинаться с символов *0q* и содержать цифры от 0 до 3,

**– целые по основанию 8** – константы данного типа должны начинаться с символов *0c* и содержать цифры от 0 до 7,

**– целые по основанию 10** – константы данного типа должны содержать цифры от 0 до 9,

– **вещественные** – константы данного типа должны содержать последовательность цифры от 0 до 9, точку и еще одну последовательность цифр от 0 до 9, причем одна из этих двух последовательностей может быть пустой,

– **строковые** – константы данного типа должны содержать последовательность символов, состоящую из любых символов кроме двойных кавычек и заключенную в двойные кавычки.

Все целые типы обозначаются как *long* (и положительные, и отрицательные значения) или *long\_u* (только положительные значения), вещественный тип – как *number*, строковый тип – как *char*.

**Идентификаторы переменных** и **идентификаторы функций** должны начинаться с любой строчной латинской буквы, затем содержать непустую последовательность цифр от 0 до 9, затем содержать непустую последовательно латинских букв (строчных и/или прописных).

**Объявления функций** указываются в следующем виде:*<returnType> <name>(<params>) { <body> }*, где *<returnType>* - тип возвращаемого значения, *<name>* - идентификатор функции, *<params>* - возможно пустая последовательность пар *<paramType> <paramName>*, разделенных запятой, *<paramType>* - тип переменной-параметра, *<paramName>* - идентификатор переменной-параметра, *<body>* - тело функции, состоящее из последовательности команд. Команды могут состоять из ключевых слов, идентификаторов, констант, вызовов операторов и/или вызовов других функций. Каждая команда должна заканчиваться символом *;* . Если функция имеет возвращаемое значение, то внутри функции должно использоваться ключевое слово *return* с указанием этого значения. Если возвращаемого значения у функции нет, то в качестве типа возвращаемого значения нужно указывать ключевое слово *void*.

**Оператор присваивания** указывается в следующем виде: put *<value> to <name>;*, где *<value>* - присваиваемое значение или идентификатор переменной с этим значением, *<name>* - идентификатор переменной, куда нужно присвоить это значение.

**Логические операторы** ==, !=, >, >=, <, <= используются в **логическом выражении**, котороеуказывается в следующем виде: *<value1> <operator> <value2>*, где *<value1>* и *<value2>* - значения для сравнения и/или идентификаторы переменных с этими значениями, *<operator>* - один из логических операторов.

**Условный оператор** указывается в следующем виде: *when <exp> then <body1> [other <body1>];*, где <exp> - проверяемое логическое выражение, *<body1>* - оператор или блок кода, выполняемые в случае правильности логического выражения, *<body2>* - оператор или блок кода, выполняемые в случае неправильности логического выражения.

**Оператор цикла** указывается в следующем виде: *exec <body> with <variable> from <start> to <end> [step <step>];*, где *<body>* - выполняемый оператор или блок кода, *<variable>* - идентификатор переменной цикла, *<start>* - начальное значение этой переменной, *<end>* - конечное значение этой переменной, *<step>* - значение, на которое изменяется переменная после одного выполнения *<body>*.

**Оператор переключателя** указывается в следующем виде: *case <variable> { when <value> then <body> [exit]; … [other <end>;] }*, где *<variable>* - идентификатор проверяемой переменной, *<value>* - константа, с которой сравнивается значение переменной, *<body>* - оператор или блок кода, выполняемый в случае совпадения значений *<variable>* и соответствующего *<value>*, *<end>* - оператор или блок кода, выполняемый в случае отсутствия совпадений. Внутри оператор может быть указано несколько блоков *when <value> then <body>;*. По умолчанию осуществляется переход от одного такого блока к другому, пока оператор переключения не закончится. Чтобы отменить переход и выйти из оператора заранее, можно использовать ключевое слово *exit*.

**Арифметические операторы** могут использовать для изменения значения переменной. К ним относятся символы и наборы символов:

+ и += (сложение)

- и -= (вычитание),

\* и \*= (умножение),

/ и /= (деление),

++ (инкремент)

- - (декремент).

**Знаки пунктуации** включают в себя следующие символы: *; , ( ) { }*.

**Комментарии** используются для указания текста, не являющегося программным кодом и игнорируемым компилятором. Они указываются в следующем виде: *// <comment> //*, где *<comment>* - текст комментария.

Таким образом, в заданном языке существуют следующие **ключевые слова**: *long*, *long\_u*, *number*, *char*, *package*, *import*, *when*, *then*, *other*, *with*, *from*, *to*, *step*, *case*, *when*, *exit*.

**Запуск программы** возможен, только если в коде программы объявлена функция *void m4in() {…}*. Выполнение программы начинается с выполнения кода внутри данной функции.

# Тестовая программа

package my.program;

//single import//

import system.output.a0print;

long\_u f4ctorial(long\_u x0x) {

when x0x == 0

then

return 0;

other {

long\_u a0f;

put 1 to a0f;

exec a0f \*= i0i;

with i0i from 1 tox0x;

return a0f;

}

}

void s4yNumber(long a0a) {

char n1n;

case a0a {

when 0 then {

put "zero" to n1n;

exit;  
 }  
 when 1 then {

put "one " to n1n;

exit;  
 }

other put "unknown" to n1n;

}

}

void m4in() {

long\_u a0a = 0q10;

a0print(f4ctorial(a0a));

s4yNumber(0c1);

}

# Вывод

В ходе лабораторной работы были описана структура программы и операторы заданного языка программирования, составлена и проверена на корректность тестовая программа, использующая этот язык.

# Приложение

Текст xml-файла с системой правил:

<?xml version="1.0" encoding="windows-1251"?>

<transLab>

<lexic>

<automat name='main'>

<rule groupWordsName='formatWord'>

<expression>[ \t\r\n]+</expression>

<action>ignoreLastWord=true;</action>

<comment>Слова для форматирования текста</comment>

</rule>

<rule groupWordsName='braces'>

<expression>[{}]</expression>

<comment>Фигурные скобки</comment>

</rule>

<rule groupWordsName='parentheses'>

<expression>[()]</expression>

<comment>Круглые скобки</comment>

</rule>

<rule groupWordsName='delimiter'>

<expression>[;,]</expression>

<comment>Разделитель команд</comment>

</rule>

<rule groupWordsName='id'>

<expression>[a-z][0-9]+[a-zA-Z]+</expression>

<comment>Идентификаторы</comment>

</rule>

<rule groupWordsName='quaterneryConst'>

<expression>[0][q][0-3]+</expression>

<comment>Целая четвертичная константа</comment>

</rule>

<rule groupWordsName='octalConst'>

<expression>[0][c][0-7]+</expression>

<comment>Целая восьмеричная константа</comment>

</rule>

<rule groupWordsName='decimalConst'>

<expression>[0-9]+</expression>

<comment>Целая десятичная константа</comment>

</rule>

<rule groupWordsName='floatConst'>

<expression>([0-9]+[.][0-9]\*)|([.][0-9]+)</expression>

<comment>Вещественная константа</comment>

</rule>

<rule groupWordsName='stringConst'>

<expression>["][]\*["]</expression>

<comment>Строковая константа</comment>

</rule>

<rule groupWordsName='operationSign'>

<expression>[-+\*/]</expression>

<comment>Знаки операций</comment>

</rule>

<rule groupWordsName='logicalOperator'>

<expression>([=][=])|([!=])|([>])|([>][=])|([<])|([<][=])</expression>

<comment>Логический оператор</comment>

</rule>

<rule groupWordsName='keyword'>

<expression>[a-z\_]+</expression>

<comment>Ключевое слово</comment>

</rule>

<rule groupWordsName='packageId'>

<expression>[a-zA-Z\_]+[.][a-zA-Z\_]+</expression>

<comment>Пакет приложения</comment>

</rule>

<rule groupWordsName='comment'>

<expression>[/][/][]\*[/][/]</expression>

<comment>Комментарий</comment>

</rule>

<rule groupWordsName='importId'>

<expression>[a-zA-Z\_]+[.][a-zA-Z\_]+[.][a-z][0-9]+[a-zA-Z]+</expression>

<comment>Идентификатор импорта</comment>

</rule>

</automat>

</lexic>

</transLab>