

**Министерство образования и науки Российской Федерации**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение высшего профессионального образования**

**«Московский государственный технический университет имени Н.Э. Баумана»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Робототехника и комплексная автоматизация» (РК)**

**Кафедра «Системы автоматизированного проектирования» (РК6)**

****

**Отчет по лабораторной работе \_\_\_\_\_A18 (N)\_\_\_\_\_\_\_ по курсу**

**«САПР»**

**Студент: Сергеева Диана**

**Группа:** РК6-36Б

**Преподаватель:** Волосатова Т.М.

Проверил:

Дата:

2020

**Задание**

**Текст задания:**

Разработать конечный автомат для распознавания записей целых чисел в системе счисления по основанию 16, где все цифры расположены в неубывающем порядке. Записи всех целых чисел должны передаваться конечному автомату через строки потока стандартного ввода. Результаты их распознавания должны отображаться строками потока стандартного вывода. Программная реализация конечного автомата должна формироваться генератором YACC по правилам регулярной грамматики.

**Введение**

Настоящий документ определяет техническое задание на разработку программы лексического анализатора, реализующего конечный автомат записей чисел в системе счисления по основанию 16 в формате системы программирования C.

**Основания для разработки**

Программа HEXSET разрабатывается в рамках лабораторной работы по курсу "Лингвистическое обеспечение САПР" для практического изучения этапов синтаксического анализа в процедурах трансляции формальных языков.

**Назначение разработки**

Программа HEXSET предназначается для реализации конечного автомата, распознающего необходимые записи шестнадцатеричных наборов, в системах и модулях обработки файлов.

**Требования к программе**

1. Требования к функциональным характеристикам
   1. Программа HEXSET должна в интерактивном режиме распознавать шестнадцатеричные наборы.
   2. Шестнадцатеричные наборы должны содержать произвольное натуральное количество разрядов.
   3. Шестнадцатеричные наборы должны быть в неубывающем порядке.
   4. Шестнадцатеричные наборы должны передаваться строками стандартного ввода.
   5. Программа HEXSET обнаруживает соответствие или несоответствие шестнадцатеричных последовательностей заданным правилам.
   6. При нарушении ввода или несоответствии последовательности заданным правилам, программа HEXSET должна вывести сообщение об ошибке.
   7. Результат распознавания программой HEXSET шестнадцатеричных последовательностей, соответствующей заданным правилам, должен отображаться в поток стандартного вывода.
2. Требования к надежности

Программа HEXSET не должна иметь каких-либо ограничений по числу символов в анализируемой шестнадцатеричной последовательности, кроме внутренних ограничений инструментальных средств, использованных для ее реализации.

1. Условия к составу и параметрам технических средств

Программа HEXSET должна быть разработана исходя из возможности реализации на стандартном составе технических средств компьютеров любой архитектуры, после соответствующей трансляции исходного кода.

1. Условия эксплуатации
   1. Программа HEXSET должна быть ориентирована на эксплуатацию в среде OS UNIX.
   2. Программа HEXSET должна быть реализована в виде выполняемого файла, по которому она должна вызываться средствами любого командного процессора OS UNIX.
2. Требования к информационной и программной совместимости
   1. При разработке программы HEXSET необходимо использовать генератор синтаксических анализаторов из состава инструментальных средств OS UNIX, который ориентирован на обработку файла спецификаций проектируемого конечного автомата.
   2. Синтаксический анализатор программы HEXSET должен обеспечивать грамматический разбор шестнадцатеричных последовательностей с целью установить соответствие или несоответствие содержащей их строки потока стандартного ввода заданным правилам.
   3. Для выполнения грамматического разбора синтаксический анализатор программы HEXSET должен реализовывать однозначную контекстно-свободную грамматику, которая ориентирована на обработку строки заданной шестнадцатеричной последовательности из потока стандартного ввода.
   4. Грамматику синтаксического анализатора программы HEXSET должны составлять следующие элементы:

* терминальные символы (терминалы), соответствующие структурным единицам (лексемам) входного выражения;
* начальный нетерминальный символ (начальный нетерминал), к которому приводится входное выражение;
* нетерминальные символы (нетерминалы), обозначающие допустимые варианты комбинации лексем во входном скобочном выражении;
* система продукций (правил вывода), обеспечивающая грамматический разбор входного выражения.
  1. Терминальные символы грамматики синтаксического анализатора программы HEXSET должны представляться лексемами, специфицированными следующими литералами: ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘А’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘\n’.
  2. Начальный нетерминал грамматики синтаксического анализатора программ HEXSET должен обозначаться именем s. Он должен выводиться из любой корректной последовательности.
  3. Система продукций грамматики синтаксического анализатора программ HEXSET должна обеспечивать грамматический разбор произвольной входной строки потока стандартного ввода путем приведения терминалов и нетерминалов к начальному нетерминалу с помощью левостороннего вывода. Также отдельно должна быть создана программа с аналогичными функциями, реализующая разбор с помощью правостороннего вывода.
  4. Для разработки синтаксического анализатора программ HEXSET, необходимо использовать генератор синтаксических анализаторов из состава OS UNIX, инструментальные средства которого ориентированы на обработку файла спецификаций проектируемого синтаксического анализатора.
  5. При разработке синтаксического анализатора программ HEXSET необходимо составить два файла спецификации, отражающих специфику грамматического разбора с помощью правостороннего и левостороннего выводов, соответственно в выбранном доступном рабочем каталоге файловой системы OS UNIX.
  6. Проектируемые файлы должны состоять из 3-х секций: деклараций, правил и функций. Разделителем секций должны быть символические пары %%, расположенные в начальных позициях содержащих их строк файла.
  7. Секция деклараций файлов должна включать: спецификацию блока внешних описаний, ограниченную директивами %{ и %}, в которой необходимо подключить библиотеку ввода/вывода stdio.h.
  8. В секции правил файлов должны быть приведены описания продукций приведения нетерминалов грамматики.
  9. Каждая продукция секции правил файлов должна быть задана в нотации, близкой к форме Бэкуса-Наура, где в левой части указывается приводимый нетерминал, а в правой - последовательность терминалов и/или нетерминалов грамматики, которые перечисляются через пробел. Для разделения частей продукции должен использоваться символ двоеточия (:). Каждую продукцию нужно начинать с новой строки и завершать либо символом точки с запятой (;), либо блоком действий в фигурных скобках.
  10. Продукции секции правил файлов, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой фигурных скобок. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C.
  11. Продукции секции правил файлов, необходимые для приведения нетерминалов, должны быть специфицированы с помощью альтернативных правил. В каждом из них альтернативы свертки различных нетерминалов правой части должны быть объединены с помощью оператора ИЛИ, который обозначается символом вертикальной черты ‘|’.
  12. Секция функций файлов должна содержать исходный код, оформленный по правилам системы программирования C, для функции yylex() которая должна иметь целочисленный код возврата.
  13. Для достижения целей лексического анализа, исходный код функции yylex() должен предусматривать чтение и возврат очередного символа.
  14. Исходный код секции функций файлов образует исходный код программ HEXSET, который должен формироваться генератором синтаксических анализаторов YACC в файлах в текущем рабочем каталоге файловой системы OS UNIX. Выполняемые модули программ HEXSET должны быть созданы по файлам исходного кода в выполняемых файлах средствами компилирующей системы программирования C.

**Стадии и этапы разработки**

В процессе разработки программы HEXSET необходимо выполнить следующие 3 этапы:

* составить файлы в выбранном рабочем каталоге файловой системы OS UNIX, используя любой текстовый редактор;
* получить исходный код программы в файлах, обработав файлы спецификации командой yacc.
* сформировать выполняемые модули в файлах текущего каталога файловой системы OS UNIX, компилируя исходный код синтаксических анализаторов.

Результаты разработки программ HEXSET должны содержать описание грамматики и файлы спецификаций для генератора синтаксических анализаторов YACC.

**Порядок контроля и приемки**

1. Для проверки функционирования программы должны быть предложены контрольные примеры, предусматривающий стандартный ввод корректных и некорректных данных.

2. Для приемки программы должен быть организован вызов выполняемого файла в консольном режиме работы OS.

**Результаты разработки**

Программа HEXSET выполняет построение конечного автомата на основе правил регулярной грамматики. Далее при запуске программы, в течение её выполнения, происходит процесс считывания информации из входного потока и последующий вывод требуемых шестнадцатеричных наборов в случае их обнаружения в строке.

Описание грамматики:

* Терминальный алфавит: Σ = {‘0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’\n’}.
* Нетерминальный алфавит: {l1, l2, l3, l4, l5, l6, l7, l8, l9, lA, lB, lC, lD, lE, lF, s}.

**Текст программы:**

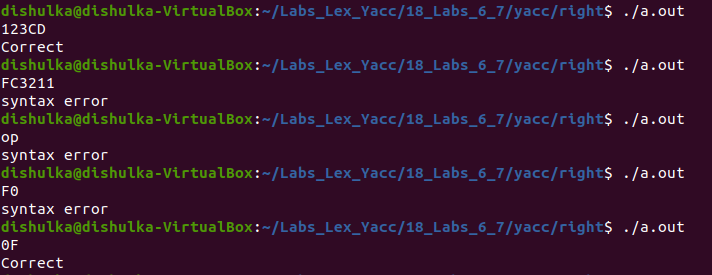
**Левосторонний вывод:**

%start s  
  
%{  
[#include](https://vk.com/im?sel=63386677&st=%23include) <stdio.h>  
int yylex();  
int yyerror(char\* s);  
%}  
  
%%  
  
s: |  
s num {return 0;} ;  
  
num: l0 '\n' | l1 '\n' | l2 '\n' | l3 '\n' | l4 '\n' | l5 '\n' | l6 '\n' | l7 '\n' | l8 '\n' | l9 '\n' | lA '\n' | lB '\n' | lC '\n' | lD '\n' | lE '\n' | lF '\n' ;  
  
lF: l1 'F' | l2 'F' | l3 'F' | l4 'F' | l5 'F' | l6 'F' | l7 'F' | l8 'F' | l9 'F' | lA 'F' | lB 'F' | lC 'F' | lD 'F' | lE 'F' | lF 'F' | 'F' | '-' 'F';  
lE: l1 'E' | l2 'E' | l3 'E' | l4 'E' | l5 'E' | l6 'E' | l7 'E' | l8 'E' | l9 'E' | lA 'E' | lB 'E' | lC 'E' | lD 'E' | lE 'E' | 'E' | '-' 'E';  
lD: l1 'D' | l2 'D' | l3 'D' | l4 'D' | l5 'D' | l6 'D' | l7 'D' | l8 'D' | l9 'D' | lA 'D' | lB 'D' | lC 'D' | lD 'D' | 'D' | '-' 'D';  
lC: l1 'C' | l2 'C' | l3 'C' | l4 'C' | l5 'C' | l6 'C' | l7 'C' | l8 'C' | l9 'C' | lA 'C' | lB 'C' | lC 'C' | 'C' | '-' 'C';  
lB: l1 'B' | l2 'B' | l3 'B' | l4 'B' | l5 'B' | l6 'B' | l7 'B' | l8 'B' | l9 'B' | lA 'B' | lB 'B' | 'B' | '-' 'B';  
lA: l1 'A' | l2 'A' | l3 'A' | l4 'A' | l5 'A' | l6 'A' | l7 'A' | l8 'A' | l9 'A' | lA 'A' | 'A' | '-' 'A';  
l9: l1 '9' | l2 '9' | l3 '9' | l4 '9' | l5 '9' | l6 '9' | l7 '9' | l8 '9' | l9 '9' | '9' | '-' '9';  
l8: l1 '8' | l2 '8' | l3 '8' | l4 '8' | l5 '8' | l6 '8' | l7 '8' | l8 '8' | '8' | '-' '8';  
l7: l1 '7' | l2 '7' | l3 '7' | l4 '7' | l5 '7' | l6 '7' | l7 '7' | '7' | '-' '7';  
l6: l1 '6' | l2 '6' | l3 '6' | l4 '6' | l5 '6' | l6 '6' | '6' | '-' '6';  
l5: l1 '5' | l2 '5' | l3 '5' | l4 '5' | l5 '5' | '5' | '-' '5';  
l4: l1 '4' | l2 '4' | l3 '4' | l4 '4' | '4' | '-' '4';  
l3: l1 '3' | l2 '3' | l3 '3' | '3' | '-' '3';  
l2: l1 '2' | l2 '2' | '2' | '-' '2';  
l1: l1 '1' | '1' | '-' '1';  
l0: '0';  
  
%%  
  
int yylex()  
{  
return getchar();  
}  
  
int yyerror(char\* s)  
{  
printf("%s\n", s);  
return 1;  
}  
  
int main()  
{  
int ret = yyparse();  
if (ret == 0)  
puts("Correct");  
return ret;  
}

**Правосторонний вывод:**

%start s  
%{  
[#include](https://vk.com/im?sel=63386677&st=%23include) <stdio.h>  
int yylex();  
int yyerror(char\* s);  
%}  
  
%%  
  
s: |  
s l1 {return 0;} |  
'-' l1 {return 0;} |  
s '0' {return 0;} ;   
  
l1: '1' l1 | '2' l2 | '3' l3 | '4' l4 | '5' l5 | '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l2: '2' l2 | '3' l3 | '4' l4 | '5' l5 | '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l3: '3' l3 | '4' l4 | '5' l5 | '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l4: '4' l4 | '5' l5 | '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l5: '5' l5 | '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l6: '6' l6 | '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l7: '7' l7 | '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l8: '8' l8 | '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
l9: '9' l9 | 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
lA: 'A' lA | 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
lB: 'B' lB | 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
lC: 'C' lC | 'D' lD | 'E' lE | 'F' lF | '\n' ;  
lD: 'D' lD | 'E' lE | 'F' lF | '\n' ;  
lE: 'E' lE | 'F' lF | '\n' ;  
lF: 'F' lF | '\n' ;  
  
%%  
  
int yylex()  
{  
return getchar();  
}  
  
int yyerror(char\* s)  
{  
printf("%s\n", s);  
return 1;  
}  
  
int main()  
{  
int ret = yyparse();  
if (ret == 0)  
puts("Correct");  
return ret;  
}

**Тесты:**

****

**Используемая литература:**

1. Кристиан К. Введение в операционную систему UNIX, M.: Финансы bстатистика, 1985.
2. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС: инструментальные средства программирования, М.: Финансы и статистика, 1988.
3. SCO XENIX, Development System, LEX Programmer Guide, SCO Inc., 1986.
4. bigor.bmstu.ru – база и генератор образовательных ресурсов
5. Лекции по курсу САПР
6. Автоматизация Проектирования Лексических Анализаторов (Т.М. Волосатова, С.В. Родионов)