ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Вывод фразы простейшего языка и проверка нашего компилятора

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по специальностям
230105-«Программное обеспечение вычислительной техники и автоматизированных систем»
010503-«Математическое обеспечение и администрирование информационных систем»

Автор

Коледов Леонид Викторович

Ростов-на-Дону, 2013

Аннотация

Данная разработка может быть использована в качестве основного учебного материала по дисциплинам: «Теория языков программирования и методы трансляции» и «Теория вычислительных процессов»

Автор

Коледов Леонид Викторович, к. ф.-м. н., доцент, профессор кафедры

Область научных интересов

Информационные технологии, системы искусственного интеллекта

Оглавление

ЛАБОРАТОРНАЯ РАБОТА 3

Задание 1.

Собрать и откомпилировать описанный ниже лексический анализатор. Провести его исполнение в режиме отладки.

Таким образом, можно (в некоторых языках) представить и действительное число:

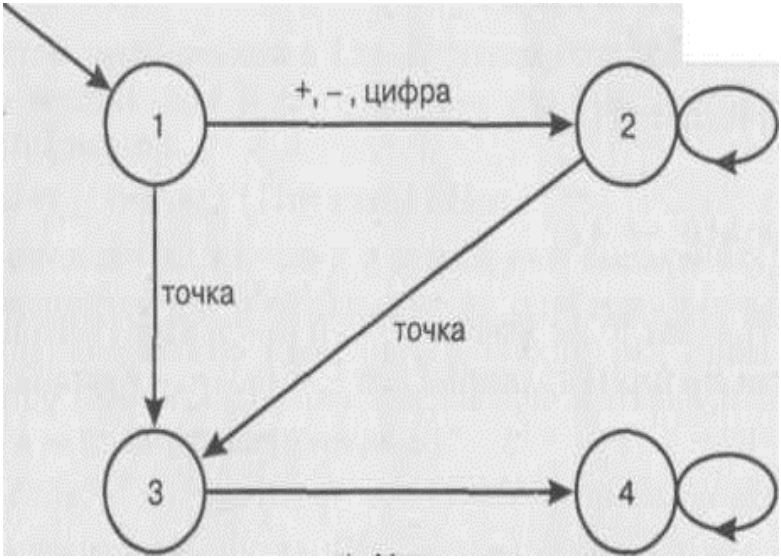
(+\-\)digit*.digit digit'

Можно написать программу для распознавания действительных чисел:

```
#include <stdio.h>
#include <ctype.h>
main ()
{char in;
in = getchar();
if (in=='+'| | in-'-' ) in = getchar();
while (isdigit(in)) in = getchar(); if (in==1.') in = getchar(); else error();
if (isdigit(in)) in = getchar(); else error();
while (isdigit(in)) in = getchar();
printf("ok\n"); }
```

Обратите внимание, что возможны три ситуации и три способа их представления в программе.
1. Необязательные знаки ("+", "-"). Если их нет - это не ошибка, просто переходим к считыванию следующего символа.
2. Обязательные знаки (десятичная точка и одна цифра после нее). Если их нет - вызывается функция error.
3. Знаки, которые могут появиться нуль или большее число раз (цифра перед точкой или после первой цифры за точкой) - инициируется цикл while для проверки каждого знака без обращения к функции error.

Действительное число, определенное в этом разделе как регулярное выражение, можно представить с помощью конечного автомата и запрограммировать способом, подобным приведенному выше.


1
+, – , цифра
2
точка
точка
3
4

Действительное число, определенное в этом разделе как регулярное выражение, можно представить с помощью конечного автомата (см. рис) и запрограммировать способом, подобным приведенному выше.

```
#include <stdio.h>
#include <ctype.h>

int issign (char sign)
{return (sign == '+' || sign == '-');
}

int main()

{int state;
char in;

state = 1;
in = getchar();

while (isdigit(in)||issign(in)||in =='.')
{switch (state)
{
case 1:   if (isdigit(in)||issign(in))
          state = 2;
          else if (in == ('.')
          state = 3;break;
case 2:   if (isdigit(in))
          state = 2;
          else if (in == '.')
          state = 3;
          else error();
          break;
case 3:   if (isdigit(in))
          state = 4;
          else error();
          break;
case 4:   if (isdigit(in))
          state = 4;
          else error();
          break;
```

```
}
  in = getchar()
return(state == 4);
}
```

Здесь error () имеет то же значение, что и ранее; sign () принимает знчение true, если его параметр равен + или -, и false - в остальных случаях.

Задание 2.

Собрать и откомпилировать описанный ниже лексический анализатор. Провести его исполнение в режиме отладки.

Идентификатор может быть представлен следующим образом:
letter (letter \ digit)*

Можно написать программу распознавания символов. Для идентификатора она будет иметь следующий вид:

```
#include <stdio.h>      "
#include  <ctype.h>
main ()
{char  in;
in =  getchar();
if   (isalpha(in))
in =  getchar();
else error();
while   (isalpha(in)    ||   isdigit(in))
in  =  getchar();
}
```

Здесь in - значение только что считанного знака; функции isalpha() и isdigit() осуществляют проверку аргумента на предмет принадлежности к буквам и цифрам, соответственно;
error () выполняет некоторые операции при возникновении ошибки.
Написать код довольно просто: проверять поступающие символы и использовать цикл while для реализации оператора *.

Конечный автомат распознавателя:

При создании распознавателей удобно использовать представление конечных автоматов. Например, следующая программа осуществляет распознавание идентификаторов.

```
 #include <stdio.h>
 #include <ctype.h>

int main()
{int state;
char in,
state = 1;
in = getchar();

while (isalpha (in) || isdigit (in) )
{switch (state)



{case 1: if (isalpha(in))
         state = 2;
         else error();
         break;

case 2: state = 2;
        break;
} in = getchar();
}

return (state == 2);
}
```

Цикл while обеспечивает прекращение работы программы при считывании любого знака, который не является буквой или цифрой. Оператор switch имеет по элементу для каждого состояния автомата, причем в каждом элементе представлены все возможные переходы из данного состояния. Во втором элементе оператора switch уже не нужно проверять вход, поскольку (из-за условия цикла while) переход к нему невозможен, если последний считанный знак не является буквой или цифрой. Присвоение этому состоянию значения 2 не является обязательным - оно просто делает переход явным.

Задание 3.

Решите задачу, номер которой сравним по модулю 8 с двузначным числом, образованному последними двумя цифрами номера Вашей зачетки.

Задача 1.

Токен задается следующим регулярным выражением:

d. dd*

Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 2.

Токен задается следующим регулярным выражением:

(a|b|c)xx*(f|g|h)
Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 3.

Токен задается следующим регулярным выражением:

(a|b|c)xx*(f|g|h)
Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 4.

Токен задается следующим регулярным выражением:

(ab*)xy*(g|h*)
Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 5.

Токен задается следующим регулярным выражением:

a. (s|dd*)

Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 6.

Токен задается следующим регулярным выражением:

(a|b)x*,s*

Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 7.

Токен задается следующим регулярным выражением:

(a|b),s*(g|h)
Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Задача 8.

Токен задается следующим регулярным выражением:

b*,x(g|h*)xx*
Сконструируйте конечный автомат распознавателя.

Написать программу распознавания токенов.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%