Міністерство освіти і науки України

Національний технічний університет України

«Київський Політехнічний Інститут»

Факультет прикладної математики

Кафедра Спеціалізованих комп’ютерних систем

Домашня контрольна робота

З дисципліни «Системне програмне забезпечення»

Виконав:

студент групи КВ-91

Федай Г. Г.

Перевірив:

Симоненко В. П.

Київ 2012

Разработать инфраструктуру (структуры данных, константы, набор функций, методов, классов, порядок работы и т. п.) для построения парсера на основе работы автоматов.

Необходимые условия:

1. Символы входного файла могут быть в любой кодировке (один символ занимает не обязательно 1 байт), входной файл состоит из любого количества символов;
2. Один парсер может состоять из нескольких автоматов, всегда есть один главный автомат, из главного автомата можно переходить в другие автоматы парсера и т. д. вниз по дереву;
3. Каждый автомат управляется одним входным символом, текущим состоянием автомата и возможно текущим контекстом разбора;
4. Должна быть функция/метод получения очередной синтаксической конструкции из входного файла или получение информации об ошибке разбора.

Используя разработанную инфраструктуру создать парсеры для следующих входных файлов:

* 1. Входной файл состоит из email адресов разделённых пробелами или символами перевода строки, синтаксическая конструкция – это имя и домен. Email адрес состоит из имени, символа ‘@’ и доменного имени. Имя состоит из букв английского алфавита, цифр и символа подчёркивания. Доменное имя состоит из одного или нескольких имён поддоменов, разделённых символом ‘.’ (точка). Имя поддомена состоит из одной или нескольких букв английского алфавита, цифр и символа ‘-‘ (минус), должно начинаться и заканчиваться буквой или цифрой.

<signal-program> --> <program>

<program> --> PROGRAM <procedure-identifier> ; <block>.

<block> --> BEGIN <statements-list> END

<statements-list> --> <statement> <statements-list> |  
<empty>

<statement> --> <condition-statement> ENDIF ; |  
WHILE <conditional-expression> DO <statements-list> ENDWHILE ;

<condition-statement> --> <incomplete-condition-statement><alternative-part>

<incomplete-condition-statement> --> IF <conditional-expression> THEN <statements-list>

<alternative-part> --> ELSE <statements-list> |  
<empty>

<conditional-expression> --> <expression><comparison-operator><expression>

<comparison-operator> --> < |  
<= |  
= |  
<> |  
>= |  
>

<expression> --> <variable-identifier> |  
<unsigned-integer>

<variable-identifier> --> <identifier>

<procedure-identifier> --> <identifier>

<identifier> --> <letter><string>

<string> --> <letter><string> |  
<digit><string> |  
<empty>

<unsigned-integer> --> <digit><digits-string>

<digits-string> --> <digit><digits-string> |  
<empty>

<digit> --> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> --> A | B | C | D | ... | Z

Лістинг програми:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace mail\_pars

{

class lexem

{

string[] letters;

string[] digits;

string[] symb;

public lexem()

{

digits = new string[11];

for (int i = 0; i < 10; i++)

digits[i] = i.ToString();

letters = new string[54];

for (char i = 'a'; i <= 'z'; i++)

{

letters[i-'a'] = i.ToString();

letters[i + 26 - 'a'] = i.ToString().ToUpper();

}

letters[52] = "\_";

letters[53] = "-";

symb = new string[2];

symb[0] = "@";

symb[1] = ".";

}

public int RetCode(string it)

{

if (digits.ToList().Contains(it))

return 0;

if (letters.ToList().Contains(it))

return 1;

if (symb.ToList().Contains(it))

return 2;

return 3;

}

public void PrintLex()

{

for (int i = 0; i < letters.Length; i++)

Console.WriteLine("{0}", letters[i]);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace mail\_pars

{

class lex\_analizer

{

public string[] lexems;

public int len;

public lex\_analizer()

{

len = 0;

lexems = new string[50];

}

public void Automat(string line)

{

string str = line;

string buf = "";

int strlen = 0, code;

lexem lex = new lexem();

while (strlen < line.Length)

{

code = lex.RetCode(str[strlen].ToString());

switch (code)

{

case 0:

Console.WriteLine("error: name starts with number");

while (strlen < line.Length)

{

code = lex.RetCode(str[strlen].ToString());

if (code != 0)

break;

strlen++;

}

break;

case 1:

buf = "";

while (strlen < line.Length)

{

code = lex.RetCode(str[strlen].ToString());

if (code != 0 && code != 1)

break;

buf += str[strlen];

strlen++;

}

lexems[len] = buf;

len++;

break;

case 2:

lexems[len] = str[strlen].ToString();

len++;

strlen++;

break;

case 3:

Console.WriteLine("error: used forbidden symbol {0}", str[strlen]);

strlen++;

break;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace mail\_pars

{

class parser

{

lex\_analizer lex;

public string line;

int num\_lex;

int max\_len;

public parser()

{

line = "";

lex = new lex\_analizer();

}

void NickName()

{

if (lex.lexems[num\_lex] == "@" || lex.lexems[num\_lex] == ".")

{

Console.WriteLine("error: expected nick name");

return;

}

if (lex.lexems[num\_lex].Contains("-"))

Console.WriteLine("error: name contains symbol - ");

num\_lex++;

}

void DomenString()

{

if (num\_lex < lex.len && lex.lexems[num\_lex].Contains("\_"))

Console.WriteLine("error: domen contains symbol \_ ");

lexem check = new lexem();

int code = check.RetCode(lex.lexems[num\_lex].Last().ToString());

if (num\_lex < lex.len && (code > 1 || lex.lexems[num\_lex][lex.lexems[num\_lex].Length - 1] == '-' || lex.lexems[num\_lex][lex.lexems[num\_lex].Length - 1] == '\_'))

Console.WriteLine("error: domen name ends with forbidden symbol");

num\_lex++;

}

void DomenName()

{

if (num\_lex >= lex.len)

{

Console.WriteLine("error1: expected domen name");

}else

DomenString();

if (num\_lex >= lex.len)

{

Console.WriteLine("error2: expected domen name");

return;

}else

SubDomenName();

SubDomenName();

if (num\_lex < lex.len)

{

Console.WriteLine("error4: unexpected end of e-mail");

}

}

void SubDomenName()

{

if (num\_lex >= lex.len)

return;

if ( lex.lexems[num\_lex] == ".")

num\_lex++;

else

{

Console.WriteLine("error: expected '.')");

}

DomenString();

// if (num\_lex < max\_len && lex.lexems[num\_lex] == ".")

}

public void MainTitle()

{

lex.Automat(line);

max\_len = lex.lexems.Length;

NickName();

if (lex.lexems[num\_lex] == "@")

{

num\_lex++;

}

else

Console.WriteLine("error: expected @");

DomenName();

Console.WriteLine("--------------------------------------------");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace mail\_pars

{

class Program

{

static void Main(string[] args)

{

parser pars = new parser();

pars.line = "hello@narod.ru";

try

{

StreamReader rd = new StreamReader(new FileStream("email.txt", FileMode.Open));

string curr\_em = "";

while ( null != ( curr\_em = rd.ReadLine()))

{

Console.WriteLine(curr\_em);

pars.line = curr\_em;

pars.MainTitle();

}

rd.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

}

}

hello\_world@com.ua

--------------------------------------------

somenick@net.com.ua

--------------------------------------------

somenick@net.com&.ua

error: used forbidden symbol &

--------------------------------------------

@dot.net

error: expected nick name

--------------------------------------------

sdhghg.ua

error: expected @

error: domen name ends with forbidden symbol

error: expected '.')

--------------------------------------------

Для продолжения нажмите любую клавишу . . .