Міністерствоосвіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет Прикладної математики

Кафедра спеціалізованих комп’ютерних систем

Лабораторна робота №1

*з дисципліни* Основи проектування трансляторів

Розробка лексичного аналізатора (сканеру, scanner )

Виконав:

студент групи КВ-92

Степанюк Михайло

Перевірив:

Марченко О. І.

Київ 2011

***Постановка задачі:***

Розробити програму лексичного аналізатора (ЛА) для підмножини мови програмування SIGNAL.

Програма має забезпечувати наступне (якщо це передбачається граматикою варіанту):

* згортання ідентифікаторів;
* згортання ключових слів;
* згортання цілих десяткових констант;
* згортання дійсних десяткових констант;
* згортання строкових констант, якщо вони визначені в заданій мові;

Також у всіх варіантах необхідно забезпечити:

* видалення коментарів, заданих у вигляді (\*<текст коментарю>\*)

Для кодування лексем необхідно використовувати числові діапазони, вказані в Таблиці 1.

Діапазони кодування лексем

|  |  |
| --- | --- |
| Вид лексеми | Числовий діапазон |
| односимвольні роздільники та знаки операцій (: / ; + і т.д.) | 0 – 255,  Тобто коди ASCII |
| Багато символьні роздільники ( := ,<= ,<=, і т.д.) | 301 - 400 |
| цілі десяткові константи | 401 - 500 |
| символьні константи | 501 - 600 |
| рядкові константи | 601 - 700 |
| ключові слова (BEGIN, END, FOR) та ідентифікатори | 701 – 1000 |

**Входом** ЛА має бути наступне:

* вихідна програма, написана підмножиною мови SIGNAL відповідно до варіанту;
* таблиця кодів ASCII з атрибутами для визначення токенів;
* таблиця багато символьних роздільників;
* таблиця ідентифікаторів, в яку попередньо занесені ключові слова з атрибутом ключового слова;

**Вихід** ЛА має бути наступним:

* закодований рядок лексем;
* таблиці ідентифікаторів, числових, символьних та рядкових констант, сформовані для конкретної програми;

Варіант 15

1. <signal-program> --> <program>
   1. <program> --> PROGRAM <procedure-identifier> ; <block> ;
   2. <block> --> BEGIN <statements-list> END
   3. <statements-list> --> <statement> <statements-list> |  
      <empty>
   4. <statement> --> <unsigned-integer> : <statement> |  
      <variable-identifier> := <unsigned-integer> ; |  
      <procedure-identifier><actual-arguments> ; |  
      GOTO <unsigned-integer> ; |  
      LINK <variable-identifier> , <unsigned-integer> ; |  
      IN <unsigned-integer>; |  
      OUT <unsigned-integer>; |  
      RETURN ; |  
      ; |  
      ($ <assembly-insert-file-identifier> $)
   5. <actual-arguments> --> ( <variable-identifier> <actual-arguments-list> ) |  
      <empty>
   6. <actual-arguments-list> --> ,<variable-identifier> <actual-arguments-list> |  
      <empty>
   7. <variable-identifier> --> <identifier>
   8. <procedure-identifier> --> <identifier>
   9. <assembly-insert-file-identifier> --> <identifier>
   10. <identifier> --> <letter><string>
   11. <string> --> <letter><string> |  
       <digit><string> |  
       <empty>
   12. <unsigned-integer> --> <digit><digits-string>
   13. <digits-string> --> <digit><digits-string> |  
       <empty>
   14. <digit> --> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
   15. <letter> --> A | B | C | D | ... | Z

**S**

**IDN**

**OUT**

**lit**

**lit,dig**

**other**

**NUM**

**OUT**

**dig**

**dig**

**other**

**EXIT**

**WS**

**ws**

**other**

**ws**

**OUT**

**INP**

**dm**

**ERR**

**other**

**INP**

**eof**

**BCOM**

**COM**

**(**

**other**

**\***

**ECOM**

**ERRL**

**\***

**eof**

**)**

**eof**

**other**

**other**

**INP**

**Вихідна програма:**

PROGRAM PROG ;

BEGIN

WHILE ( VR >= 10 DO )

IF VR2 <>6 THEN TY = 7; ENDIF;

(\* cooment

anothere line \*)

ENDWHILE;

END.

**Текст програми:**

**analizer.h**

#include <iostream>

#include <vector>

#include <string>

#pragma warning ( disable: 4018 )

#pragma warning ( disable: 4996 )

using namespace std;

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

//містить інформацію про вхідний символ та його атрибут

#pragma once

class EnterAlph {

private:

int attr;

char \* id;

public:

EnterAlph () { attr = 0; };

EnterAlph ( const EnterAlph &l ) {

id = new char [ strlen ( l.id ) + 1 ];

strcpy(id,l.id);

attr = l.attr;

};

~EnterAlph () {

delete [] id;

};

char \*GetId() { return id ; };

int GetAttr() { return attr; };

void SetAttr ( int \_attr ) {

attr = \_attr;

};

void SetId ( char \* \_id ) {

id = new char [ strlen ( \_id ) + 1 ];

strcpy( id, \_id );

};

};

//містить інформацію про ідентифікатор та його місце в таблиці //ідентифікаторі/констант/роздільників

class id {

private:

int number;

char \*ident;

int const\_num;

public:

id () {ident = NULL;};

id ( const id &\_id ) {

if ( \_id.ident == NULL )

ident = NULL; else {

ident = new char [ strlen ( \_id.ident ) + 1 ];

strcpy(ident, \_id.ident);

}

number = \_id.number;

const\_num = \_id.const\_num;

};

~id () { delete []ident;};

char \* GetId() { return ident; };

int GetNum () {return number;};

int GetConst () {return const\_num;};

void SetId ( char \*name ) {

ident = new char [ strlen ( name ) + 1 ];

strcpy ( ident, name );

};

void SetNum ( int num ) { number = num; };

friend class Table;

};

//містить масив attr\_arr з вхідними символами та атрибутами

class Attr

{

public:

vector<EnterAlph> attr\_arr;

Attr () {

};

~Attr () {

};

//заповнення таблиці вхідних символів

void Fill ( ) ;

void PrintId();

//функція повертає значення атрибуту для вхідного символу ident

int RetAttr ( char ident );

friend class Table;

};

//клас містить в собі масиви ключових слів та ідентифікаторів користувача (tab\_id ), констант //(tab\_const) та роздільників ( tab\_sep ), а також методи для їх попередньої ініціалізації та //обробки

class Table {

private:

int limit;

int lex\_size;

int limit\_id;

public:

vector <EnterAlph> tab\_sep;

vector <id> tab\_const;

vector <id> tab\_id;

Table () {

lex\_size = 87;

limit = 100;

limit\_id = 0;

}

~Table () {};

//додавання константи в таблицю констант

void push\_const ( int \_const );

//додавання ідентифікатора в таблицю ідентифікаторів

void push\_id ( char \*\_id );

//заповнення таблиці ключових слів

void FillId () ;

//заповнення таблиці роздільників

void SetSep();

void print\_id () ;

//повертає значення про кількість елементів в таблиці ключових слів

int GetLexSize (){ return lex\_size;};

//якщо заданий ідентифікатор/константа/ключове слово/роздільник міститься в //таблиці, то повертає номер лексеми, якщо ні, то додає новий //ідентифікатор/константу в відповідну таблицю, присвоює лексемі номер та повертає //його

int GetLexCode ( Attr lex , char \*\_id );

friend void PrintTab ( Table tab, Attr atr );

};

void concat (char \*\*str, char ch );

void Automat (); //процедура-сканер

void strtochar( String ^orig, char \*&out);

void PrintTab ( Table tab, Attr atr ); //вивід таблиць на екран

#include "analizer.h"

#include <stdio.h>

#pragma once

using namespace std;

void Attr::Fill ( ) {

char \*buf ;

buf = NULL;

int i = 40, j = 0;

EnterAlph l;

buf = new char [4];

buf [ 0 ] = '\n';

buf[1] = '\0';

l.SetId(buf);

l.SetAttr(0);

attr\_arr.push\_back(l);

buf [ 0 ] = 9;

l.SetId(buf);

attr\_arr.push\_back(l);

buf [ 0 ] = 32;

l.SetId(buf);

attr\_arr.push\_back(l);

while ( i < 123 )

{

buf[0] = i;

l.SetId(buf);

if ( i == 40 ) {

l.SetAttr(3);

attr\_arr.push\_back(l);

}

if (( i==63 || i == 64 ) || ( i>90 && i!=95 && i<97 ) )

l.SetAttr(5);

if ( i>='0' && i<='9' )

l.SetAttr(1);

if ( ( i>57 && i<63) || ( i > 39 && i < 48) )

l.SetAttr(4);

if ( i == 63 ) {

l.SetId(":=");

cout<<l.GetId()<<endl;

l.SetAttr(4);

attr\_arr.push\_back(l);

l.SetId(">=");

attr\_arr.push\_back(l);

l.SetId("<=");

attr\_arr.push\_back(l);

l.SetId("<>");

//i++;

}

if ( (i>= 'A' && i<= 'Z') || i == '\_' )

l.SetAttr(2);

i++;

attr\_arr.push\_back(l);

}

delete [] buf;

};

void Attr::PrintId () {

for ( int i = 0 ; i< 86; i++ ) {

cout<<i<<" "<<attr\_arr[i].GetAttr()<<" "<<attr\_arr[i].GetId()<<endl;

}

}

int Attr::RetAttr ( char ident ) {

for ( int i = 0; i < attr\_arr.size(); i++ )

if ( ident == attr\_arr[i].GetId()[0] )

return attr\_arr[i].GetAttr();

return -1;

}

void Table::push\_const ( int \_const ) {

id \_id;

\_id.number = tab\_const.size() + limit+ lex\_size + tab\_id.size();

\_id.const\_num = \_const;

\_id.ident = NULL;

tab\_const.push\_back( \_id );

}

void Table::push\_id ( char \*\_id ) {

id d;

d.number = tab\_id.size()+ lex\_size ;

d.SetId ( \_id );

tab\_id.push\_back( d );

}

void Table::FillId () {

char \*buf[] = { "PROGRAM","BEGIN","END","WHILE","DO","ENDWHILE","IF","THEN","ELSE","ENDIF" };

id d;

d.ident = new char [10];

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

strcpy( d.ident, buf[i] );

d.number = i + lex\_size;

tab\_id.push\_back( d );

}

limit\_id +=tab\_id.size();

}

void Table::SetSep () {

Attr atr;

atr.Fill();

for ( int i = 0; i < atr.attr\_arr.size(); i++ ) {

if ( (i>=3 && i<11) || ( i>21 && i<31 ) )

tab\_sep.push\_back ( atr.attr\_arr[i] );

}

}

int Table:: GetLexCode ( Attr lex , char \*\_id ) {

int i;

if ( lex.RetAttr( \_id[0] ) == 4 ) {

for ( int i = 0; i< tab\_sep.size(); i++ )

if ( strcmp ( \_id, tab\_sep[i].GetId() )== 0 )

return i;

return -1;

}

for ( int i = 0; i<tab\_id.size(); i++ )

if ( strcmp ( \_id, tab\_id[i].ident ) == 0 )

return tab\_id[i].number;

for ( int i = 0; i < tab\_const.size(); i++ )

if ( atoi( \_id) == tab\_const[i].const\_num )

return tab\_const[i].number;

if ( strcmp( \_id,"0" ) == 0 || atoi ( \_id ) > 0 ) {

push\_const ( atoi ( \_id ) );

return (tab\_const[ tab\_const.size() - 1 ].number) ;

}else {

push\_id ( \_id );

return ( tab\_id[ tab\_id.size() - 1 ].number );

}

return -1;

}

void Table::print\_id () {

for ( int i = 0; i<tab\_id.size(); i++) {

cout<<tab\_id[i].number<<" "<<tab\_id[i].ident<<endl;

}

}

void strtochar( String ^orig, char \*&out) {

int length = orig->Length;

out = new char[length+1];

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

out[i] = (char) orig[i];

out[length] = '\0';

}

void concat ( char \*\*str, char ch ) {

char \*buf , \*str1 ;

int i;

if ( ch == 9 || ch == 32 || ch == '\n' )

return;

if ( \*str == NULL ) {

\*str = new char [2];

strcpy( \*str, &ch);

return;

}

str1 = new char [ strlen ( \*str ) + 1];

strcpy( str1, \*str );

buf = new char [ strlen ( \*str ) + 2 ];

for ( i = 0; i< strlen ( \*str ) ; i++ )

buf [i] = str1[i];

buf [i] = ch;

buf [i+1] = '\0';

delete [] \*str;

\*str = new char [ strlen (buf) + 1 ];

strcpy(\*str,buf);

delete []buf;

return;

}

namespace lexan\_shell {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для Form1

///

/// Внимание! При изменении имени этого класса необходимо также изменить

/// свойство имени файла ресурсов ("Resource File Name") для средства компиляции управляемого ресурса,

/// связанного со всеми файлами с расширением .resx, от которых зависит данный класс. В противном случае,

/// конструкторы не смогут правильно работать с локализованными

/// ресурсами, сопоставленными данной форме.

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::RichTextBox^ result;

protected:

protected:

private: System::Windows::Forms::RichTextBox^ TabId;

private: System::Windows::Forms::Button^ Start;

private: System::Windows::Forms::RichTextBox^ Source;

private: System::Windows::Forms::RichTextBox^ error;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::RichTextBox^ User;

private: System::Windows::Forms::RichTextBox^ consts;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::TextBox^ textBox6;

private: System::Windows::Forms::TextBox^ textBox7;

private: System::Windows::Forms::RichTextBox^ Sep;

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

/\* оголошення елементів форми \*/

}

void PrintTab ( Table tab, Attr atr ) {

for ( int i = 1 ; i < tab.tab\_sep.size(); i++ ) {

String ^s = i.ToString();

String ^s1 = gcnew String ( tab.tab\_sep[i].GetId() );

Sep->Text += s;

Sep->Text += " ";

Sep->Text += s1;

Sep->Text +="\n ";

}

for ( int i = 0 ; i< tab.tab\_id.size(); i++ ) {

int j=tab.tab\_id[i].GetNum();

String ^s = j.ToString();

String ^s1 = gcnew String ( tab.tab\_id[i].GetId() ) ;

if ( j>=97 ) {

User->Text += s;

User->Text += " ";

User->Text += s1;

User->Text +="\n ";

}

else {

TabId->Text += s;

TabId->Text += " ";

s = gcnew String ( tab.tab\_id[i].GetId() );

TabId->Text += s;

TabId->Text +="\n ";

}

}

for ( int i = 0; i<tab.tab\_const.size(); i++ ){

int j=tab.tab\_const[i].GetNum();

String ^s = j.ToString();

consts->Text += s;

consts->Text += " ";

j=tab.tab\_const[i].GetConst();

s = j.ToString();

consts->Text += s;

consts->Text +="\n ";

}

}

void Automat () {

FILE \*f;

Attr lex;

Table tab;

int attr, lexcode;

char symbol;

bool inp\_fl ;

char \*cons = NULL, \*buf = NULL;

if ( NULL == ( f = fopen ("test2.txt","rt") ) ) {

cout<<"File doesn't exist "<<endl;

return;

}

lex.Fill();

tab.FillId();

tab.SetSep();

tab.print\_id();

lex.PrintId();

symbol = fgetc(f);

while ( symbol != EOF ) {

attr = lex.RetAttr(symbol);

cons = NULL;

lexcode = 0;

inp\_fl = true;

switch ( attr ) {

case 0: //whitespace

while ( EOF != ( symbol = fgetc(f) ) ) {

if ( lex.RetAttr ( symbol ) != 0 )

break;

}

inp\_fl = false;

break;

case 1: //const

do {

concat(&cons, symbol);

symbol = fgetc(f);

attr = lex.RetAttr(symbol);

}while ( symbol != EOF && attr == 1 );

lexcode = tab.GetLexCode ( lex, cons );

delete []cons;

cons = NULL;

break;

case 2:

//identifier

do {

concat(&cons, symbol);

symbol = fgetc(f);

attr = lex.RetAttr(symbol);

}while ( symbol != EOF && ( attr == 2 || attr == 1 ) );

if ( attr == 5 ) {

inp\_fl = false;

break;

}

lexcode = tab.GetLexCode ( lex, cons );

delete []cons;

cons = NULL;

break;

case 3:

symbol = fgetc(f);

if ( symbol == EOF ) {

cout<<"error: unexpexted end of file"<<endl;

error->Text +="error: unexpexted end of file\n";

lexcode = tab.GetLexCode ( lex, "(" ); //code of the opened paranthes

} else {

if ( symbol == '\*' ) {

symbol = fgetc(f);

if ( symbol == EOF )

error->Text +="unexpected end of file\n";

else {

do {

while ( symbol != EOF && symbol != '\*' )

symbol = fgetc(f);

if ( symbol == EOF ) {

error->Text +="expected '\*) but end of file appeared\n";

break;

}else

symbol = fgetc(f);

}while ( symbol != ')' );

if ( symbol == ')' )

inp\_fl = false;

symbol = fgetc(f);

}

}else lexcode = 1;

}

break;

case 4:

lexcode = tab.GetLexCode ( lex, &symbol );

concat( &cons, symbol );

symbol = fgetc(f);

if ( symbol != EOF && lex.RetAttr(symbol) == 4 ) {

concat( &cons, symbol );

if ( tab.GetLexCode ( lex, cons ) != -1 ) {

lexcode = tab.GetLexCode ( lex, cons ) ;

symbol = fgetc(f);

}

}

delete []cons;

cons = NULL;

break;

case 5:

String ^s = gcnew String (&symbol);

error->Text +="error: forbidden symbol ";

error->Text += s;

error->Text +="\n";

symbol = fgetc(f);

while ( lex.RetAttr(symbol ) == 1 || lex.RetAttr(symbol ) == 2 )

symbol = fgetc(f);

inp\_fl = false;

break;

}

if ( inp\_fl == true ) {

cout<<lexcode<<" ";

int i=lexcode;

String ^s = i.ToString();

result->Text += s;

result->Text +=" ";

}

}

PrintTab ( tab, lex );

}

#pragma endregion

private: System::Void richTextBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

result->Text = "";

error->Text = "";

consts->Text="";

User->Text="";

Sep->Text = "";

TabId->Text = "";

Automat();

}

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e) {

FILE \*f ;

char \*buf;

f = fopen("test.txt","r");

buf = new char [100];

while ( fgets(buf,100,f) != NULL ) {

String ^s = gcnew String(buf);

Source->Text+=s;

}

fclose (f);

}

private: System::Void Source\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void Source\_TextChanged\_1(System::Object^ sender, System::EventArgs^ e) {

FILE \*f;

char \*buf;

strtochar( Source->Text,buf);

f = fopen ("test2.txt","w");

fputs(buf,f);

fclose(f);

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void TabId\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void Sep\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

}