НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №5

з дисципліни **«**Системне програмування**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Київ – 2014 р.

## **Лабораторна робота 2.5**

**Тема роботи**: **Створення і на­стро­ювання низхід­них синтаксич­них аналізато­рів на базі використання метамов Бекуса**

**Мета роботи**: Одержання навичок створення механізмів синтаксичного розбору методом рекурсив­ного спуска і створення обробника синтаксичних помилок вхідного тексту.

**Завдання на лабораторну роботу**

1) Підготовка програми

а) Проаналізувати конструкції (задані згідно з варіантом): визначити які саме правила граматики треба довизначити

а) Реалізувати тіла процедур граматичного розбору аналогічно тому, як це показано в розділі «Приклади реалізації правил розбору граматики»

2) Створити тестовий приклад (згідно з варіантом)

3) Налагодити і запустити програму. Получить дерево грама-тичного розбору. Скопіювати результати роботи програми і вставити їх в протокол.

4) Внести помилку в код, що аналізується, і розглянути реак-цію програми на неї. Результати внести до протоколу.

5) Зробити висновки по роботі

Варіант завдання:

|  |  |
| --- | --- |
| 12 | for i :=1 to n+m do begin d:=j; if (n<m) then begin end; end; |

// spLb6.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "..\spLb3\token.h"

#include "..\spLb3\visgrp.h"

#include "..\spLb4\tables.h"

#include "..\spLb4\lexan.h"

#include "..\spLb4\langio.h"

#include <stdio.h>

#include <string.h>

#include <iostream>

#include "synan.h"

using namespace std;

extern struct recrdKWD \*tablKWD;

struct lxNode nodes[200]= // масив приймач вузлів дерева

{{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},{\_nil,NULL,NULL,0,0,0,0,0,NULL,0},

};

char file\_name[20];

extern enum ltrType ltClsC[256];

extern enum ltrType ltClsP[256];

extern enum tokType dlCdsC[256];

extern enum tokType dlCdsP[256];

extern enum ltrType ltClsC[256];

extern enum ltrType ltClsP[256];

enum ltrType \*ltCls = ltClsC;

enum tokType \*dlCds = dlCdsC;

int main(int argc, char\* argv[]) {

int nn = -1;//np,

if (argc > 1) {

strcpy(file\_name, argv[1]);

printf("Processing file -- %s\n", file\_name);

} else {

printf("Please enter file Name: ");

scanf("%s", file\_name);

strcat(file\_name, ".h");

}

opFls(file\_name);

//opFls("v1.h");

LxAnInit('P');

srtBin(tablKWD, 67);

do {//np=nn;

nn = LxAnlzr();

} while (nodes[nn].ndOp != \_EOF);

//Начало синтаксического анализа

synNode\* mainNode = synAnalysis(nodes);

//Конец синтаксического анализа

mainNode->toString(0);

//prLaTxt(nodes, nn);

cin.get();

cin.get();

return 0;

}

#include "stdafx.h"

#include "automat.h"

#include "langio.h"

#include "..\spLb3\token.h"

#include "tables.h"

#include "index.h"

#include <stdlib.h>

extern enum ltrType \*ltCls; // уточнюються для версій та режимів

extern enum autStat nxtStsC[Eo+1][ltrcode+1];

extern enum autStat nxtStsP[Eo+1][ltrcode+1];

extern char \*oprtrC[], \*oprtrP[], \*oprtrV[],

\*cprC[], \*cprP[], \*cprV[];

char \*\*oprtr, \*\*cpr;

char modeP=0, // тип роздільника операторних дужок для Паскаля

modeC=1, // тип роздільника операторних дужок для С

modeL;

extern int x, y, f;

extern unsigned nImBg, nImCr; // Початковий та поточний номери образів в буфері

extern char imgBuf[]; // буфер вхідних образів

extern enum tokType \*dlCds;

extern enum ltrType \*ltCls;

unsigned nNode=0;

extern struct lxNode nodes[100];

extern unsigned nNdxNds;

extern struct indStrUS \*pRtNdx, ndxNds[50];

struct recrdKWD \*tablKWD;

extern struct recrdKWD tablKWDC[67];

extern struct recrdKWD tablKWDP[67];

enum autStat //\*nxtStsR=&nxtStsC[0][0],

nxtSts[Eo+1][ltrcode+1];//=&nxtStsR;//(enum autStat\*\*)nxtStsC;

void LxAnInit(char nl)

{char i=0;

switch (nl)

{case 'P':modeL=modeP;

oprtr=oprtrP, cpr=cprP;

tablKWD = tablKWDP;

for(i=0;i<=Eo;i++)for(char j=0;j<=ltrcode;j++)

nxtSts[i][j]=nxtStsP[i][j];

break;

case 'V':

default:

case 'C':modeL=modeC;

oprtr=oprtrC, cpr=cprC;

tablKWD = tablKWDC;

for(i=0;i<=Eo;i++)for(char j=0;j<=ltrcode;j++)

nxtSts[i][j]=nxtStsC[i][j];

}

}

// функція лексичного аналізу чергової лексеми

int cntMdB=0;

int LxAnlzr(void)

{//static int lxNmb=0;

static enum autStat s=S0, sP; // поточний та попередній стан лексеми

enum autStat SP;

static enum ltrType c; // клас чергової літери

static char l=1; // чергова літера (початок фыктивний)

struct recrdKWD\* pRt;

int s1, c1;

char l1, l0; // чергова літера

lxInit(nNode,c);

do {SP=

sP=s; l1=l; // запам'ятовування стану

l0=l=ReadLtr(); // читання літери

c1=c=ltCls[l]; // визначення класу літери

if(s==Scl&&c!=dlmeorml)continue;

s=nxtSts[s][c];//[c<dlmaux?c:dlmaux];// стан лексеми

s1=s;

if(s==Scr)continue;

if((sP==S2||sP==S3)&&(c>nc&&c<dlmeormr))

{// пошук в таблиці групових роздільників

imgBuf[nImCr]=0;

pRt=selBin(imgBuf+nImBg, tablKWD, 67);

if(pRt!=NULL)

{nodes[nNode].ndOp=pRt->func;

if(pRt->func==\_remL)

{s=Scr; nodes[nNode].pstNd=(struct lxNode \*)(imgBuf+nImCr);

continue;}

if(pRt->func==\_rem)

{s=Scl;} continue;

break;}else

{if(sP!=S3)

{nodes[nNode].ndOp=/\*(enum tokType)\*/dlCds[l1];

imgBuf[nImBg]=imgBuf[nImBg+1];

}else{imgBuf[nImBg]=imgBuf[nImBg+1]; nImCr--;}

nImCr--;

sP=S0;

s=nxtSts[sP][c];//[c<dlmaux?c:dlmaux];// стан лексеми

return nNode++;

}

}

s1=s;}while(s!=S0&&s!=S2&&!((sP==S0||sP==S2||sP==S3)&&s<S2)); // перевірка кінця лексеми

s1=sP;

switch(sP)

{case Scr: case Scl:

imgBuf[nImCr++]=0;

// ((char\*)(nodes[nNode].prvNd))--;

nImBg=nImCr;

break;

case S2:

case S0:

/\* if(s==S0)

// dGroup(nNode);// аналіз групових роздільників

{//imgBuf[nImBg]=l1;

imgBuf[++nImCr]=0;//nImCr++;

}\*/

if(nodes[nNode].ndOp!=\_nil)

{nImCr=nImBg; l=' '; return nNode++;}

// if(sP!=S0)

nodes[nNode].ndOp=/\*(enum tokType)\*/dlCds[l1];//dlCds[l1];

if(nodes[nNode].ndOp==\_opbr&&(nodes[nNode-1].ndOp==\_ass||cntMdB))

{cntMdB++;nodes[nNode].ndOp=\_tdbr;}

if(nodes[nNode].ndOp==\_ocbr&&cntMdB)

{nodes[nNode].ndOp=\_tcbr;cntMdB--;}

if(nodes[nNode].ndOp!=\_nil)

// &&imgBuf[nImBg]==)

{nodes[nNode].prvNd=NULL;

if(nImBg+1!=nImCr)

// if(dlCdsC[l0]!=\_nil||ltClsC[l0]==dlmaux||ltClsC[l0]==dlmeormr)

{imgBuf[nImBg]=imgBuf[nImBg+1];

if(s!=S0){nImCr--;

imgBuf[nImBg]=imgBuf[nImCr];nImCr=nImBg+1;}// 04.07.07

else nImCr=nImBg;}

return nNode++;}

else if(ltCls[imgBuf[nImBg]]==dlmaux/\*&&ltClsC[imgBuf[nImBg]]>nc\*/)

{imgBuf[nImBg]=imgBuf[nImBg+1];

nImCr--;}

return nNode;

case S1n:// пошук ключових слів та імен

imgBuf[nImCr-1]=0;

// пошук у таблиці ключів;

if(\*(imgBuf+nImBg)==13)nImBg++;

pRt=selBin(imgBuf+nImBg, tablKWD, 67);

if(pRt){nodes[nNode].ndOp=pRt->func;

// якщо знайдено

nodes[nNode].prvNd=NULL;

nImCr=nImBg;

if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;

return nNode++;}

// якщо не знайдено

nodes[nNode].ndOp=\_nam;

insBTr(nodes+nNode,pRtNdx);

nImBg=nImCr;

if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;

break;

default: // не дійшли до класифікованих помилок

case Eu: case Ec: case Ep: case Eq: case En: case Eo:// обробка помилок

eNeut(nNode); // фіксація помилки

case S1c: case S2c: case S1p: case S2s: // формування констант

imgBuf[nImCr-1]=0;

nodes[nNode].resLength=sP;// frmCns(sP, nNode); break;

nodes[nNode].ndOp=\_srcn; nodes[nNode].dataType=sP;

insBTr(nodes+nNode,pRtNdx);

nImBg=nImCr;

if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;

break;

case S3: nImCr=nImBg;

nodes[nNode].prvNd=NULL;

imgBuf[nImBg]=imgBuf[nImBg+2];

s1=s;

if(s!=S0)nImCr=nImBg+1;//else nImCr--;

}

return nNode++;

}

#include "stdafx.h"

#include "..\spLb3\token.h"

#include "automat.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SCNST 1024 // граничний обсяг символьних образів

struct fileInp

{char \* src\_name;

FILE \* stream;

} include\_files[20];

int x=0, y=0, f=0, cl;

//extern

unsigned nImBg=0, nImCr=0; // Початковий та поточний номери образів в буфері

//extern

char imgBuf[MAX\_SCNST]; // буфер вхідних образів

extern struct lxNode nodes[MAX\_SCNST];

FILE \*fi\_progr;

void opFls(char\* fn)

{ //search in startup dir

fi\_progr=fopen(fn,"r+b");

if (!fi\_progr)

{printf("\nImpossible to open file: %s\n",fn);exit(2);}

include\_files[f].stream=fi\_progr;

include\_files[f].src\_name=fn;

f++;

}

FILE \*fi\_progrC;

void opFlsC(char\* fn)

{/\* //search in startup dir

char \*ps, s[30]="";

ps=strcat(s,"..\\spLb9c\\");//"");

ps=strcat(ps,fn);\*/

fi\_progrC=fopen(fn,"wb");

if (!fi\_progrC)

{printf("\nImpossible to open file: %s\n",fn);exit(2);}

}

void WriteCod(char\*cd)

{fprintf(fi\_progrC,cd);

}

char ReadLtr(void)

{static char c=0;

cl=fgetc(fi\_progr);

if (c==13||c==10)

{x=0; y++;

if(cl==10)cl=fgetc(fi\_progr);

}

x++;

c=char(cl);

if (cl==EOF)c=0; //c=13;

imgBuf[nImCr++]=c;//cl;

return c;

}

void lxInit(int lxNmb,enum ltrType cl)

{nodes[lxNmb].x=x;

nodes[lxNmb].y=y;

nodes[lxNmb].f=f;

nodes[lxNmb].ndOp=\_nil;

nodes[lxNmb].prnNd=NULL;

nodes[lxNmb].prvNd=(struct lxNode \*)(imgBuf+nImCr);

if(cl<nc/\*&&nImCr!=0\*/)nodes[lxNmb].prvNd=(struct lxNode \*)(imgBuf+nImCr-1);

nodes[lxNmb].pstNd=NULL;

nodes[lxNmb].dataType=0;

nodes[lxNmb].resLength=0;

nodes[lxNmb].stkLength=0;

}

void frmCns(enum autStat s, int lxNmb)

{;

}

void eNeut(int lxNmb)

{;

}

void dGroup(int lxNmb)

{;

}