НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

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

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Київ – 2014 р.

Завдання на роботу

Завдання на підготовку до роботи на комп’ютері:

1. Визначити варіант завдання для основних задач за таб-ли¬цею 5.1. Визна¬чи¬ти приклади лексем через файл в папці spLb5 модуля тестування spLb5.cpp.

2. Відповісти на контрольні запитання.

3. Підготувати настройки вхідної мови програмування.

4. Використати структуру елементу struct lxNode шаб¬лону програмного проекту spLb5 для побудови елементу індексу таблиць лексем і визначити початковий стан індексу.

5. Підготувати програмний модуль контрольної задачі, який виконує заданий варіант з таблиці 4.1 і дозволяє перевіри-ти коректність виконання програм.

Завдання на роботу на комп’ютері

6. Побудувати програмний проект, ввівши програмні мо-дулі у відповідні файли проекту і налагодити синтаксис.

7. Побудувати виконуваний модуль тестової програми і налагодити змістовне виконання програми для перевірки ре-зультатів контрольних прикладів.

8. Одержати результати виконання програми висхідного розбору, зробивши наступні дії:

- вставити контрольну точку в модулі spLb5.cpp перед ви-кликом функції prLxTxt(nodes+nr);

- проглянути в режимі налагодження масив лексем struct lxNode nodes [200] і проаналізувати в ре-жимі налагодження поля типів лексем enum tokType ndOp, координат лексем у вхідному тесті int x, y, f та полів покажчиків struct lxNode\* prvNd і struct lxNode\* pstNd для базових лексем вхідного тексту;

- проглянути реконструкцію тексту викликом функції prLxTxt(nodes+ nr);

- зробити висновки про роботу програми та настройки таблиць.

9. Продемонструвати результати викладачам

Порядок вибору варіанту:

За останньою цифрою номера залікової книжки або за порядко-вим номером студента в списку підгрупи з доданим номером групи визначте за табл. 4.1 варіант оброблюваних даних та на-стройки програм за прикладом.

|  |  |  |
| --- | --- | --- |
| 12 | b:=2\*a[n]; b:=d; | Pascal |

Лістинг програми

// spLb5.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 "syntaxP.h"

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

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

extern struct recrdKWD tablKWD[];

extern struct lxNode nodes[MAX\_NODES]; // масив приймач вузлів дерева

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;

char file\_name[20];

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

{int nn=-1, nr=0, nc=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);

LxAnInit('C');

// srtBin(tablKWD, 67);

do{//np=nn;

nn=LxAnlzr();

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

prLaTxt(nodes,nn);

printf("\n");

SxAnInit('C');

nr=0; nc=1; nodes[0].prnNd=-1;

do nr=nxtProd(nodes,nr,nc);

while(++nc<nn);

// конверсія до семантичної обробки

prLxTxt(nodes+nr);

// nr=prCmpr(nodes,nn,nr);// компресія для скорочення графа

// prLxTxt(nodes+nr);

printf("\n");

cin.get();

system("pause");

return 0;

}

#include "stdafx.h"

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

#include "syntaxP.h"

extern enum tokPrec opPrFC[256];

extern enum tokPrec opPrGC[256];

extern enum tokPrec opPrFP[256];

extern enum tokPrec opPrGP[256];

enum tokPrec \*opPrF,\*opPrG;

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

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

extern char \*\*oprtr, \*\*cpr,

modeP, // тип роздільника операторних дужок для Паскаля

modeC, // тип роздільника операторних дужок для С

modeL;

void SxAnInit(char nl)

{char i=0;

switch (nl)

{case 'P': opPrF=opPrFP; opPrG=opPrGP; modeC=0;

modeL=modeP; modeP=1; oprtr=oprtrP; cpr=cprP;

break;

case 'V':

default:

case 'C':opPrF=opPrFC; opPrG=opPrGC; modeP=0;

modeL=modeC; modeC=1; oprtr=oprtrC; cpr=cprC;

}

}

int nxtProd(struct lxNode\*nd, // вказівник на початок масиву вузлів

int nR, // номер кореневого вузла

int nC) // номер поточного вузла

{int n=nC-1; // номер попереднього вузла

enum tokPrec pC = opPrF[nd[nC].ndOp],// передування поточного вузла

\*opPr=opPrG;//F;// nd[nC].prvNd = nd+n;

while(n!=-1) // цикл просування від попереднього вузла до кореню

{if(opPr[nd[n].ndOp]<pC//)// порівняння функцій передувань

&&nd[n].ndOp</\*\_ctbz\*/\_frkz)

{if(n!=nC-1&&nd[n].pstNd!=0) // перевірка необхідності вставки

{nd[nC].prvNd = nd[n].pstNd; // підготовка зв’язків

nd[nC].prvNd->prnNd=/\*nd+\*/nC;} // для вставки вузла

if(opPrF[nd[n].ndOp]==pskw&&nd[n].prvNd==0)nd[n].prvNd = nd+nC;

else nd[n].pstNd = nd+nC;

nd[nC].prnNd=/\*nd+\*/n; // додавання піддерева

return nR;}

if(opPrG[nd[n].ndOp]==pC&&

(nd[n].ndOp==\_brkt||nd[n].ndOp==\_ixbr||nd[n].ndOp==\_opbr||nd[n].ndOp==\_tdbr))

{nd[n].ndOp=(enum tokType)((nd[n].ndOp-\_fork)/2+\_frkz);//09.04.07

nd[nC]=nd[n];

if(nd[nC].prnNd==-1){nR=nC; nd[nR].prnNd=-1;}

else if(opPrF[nd[nd[nC].prnNd].ndOp]==pskw&&nd[nC].ndOp<\_frkz)

nd[nd[nC].prnNd].prvNd = nd+nC;

else if(opPrF[nd[nd[nC].prnNd].ndOp]==pekw&&nd[nC].ndOp==\_opbz)

{nd[nd[nC].prnNd].prvNd =nd+nC;nd[nd[nC].prnNd].pstNd=0;}

return nR;}

/\* if(nd[n].ndOp==\_brkt||nd[n].ndOp==\_ixbr||nd[n].ndOp==\_opbr||nd[n].ndOp==\_tdbr)

{nd[nC].prnNd=n; nd[nC].prvNd=nd[n].pstNd;

nd[n].pstNd->prnNd=nC; nd[n].pstNd= nd+nC;

return nR;}\*/

n=nd[n].prnNd;

opPr=opPrG;} // просування до кореню

// if(n<=) else

nd[nC].prvNd = nd+nR; nd[nR].prnNd=/\*nd+\*/nC; nR = nC; nd[nR].prnNd=-1;

return nR;}

int prCmpr(struct lxNode\*nd, int nn, int nr) //компресія для скорочення графа

{int nR, nN=0, nC=0;

do{

if((nd+nN)->ndOp==\_remL||(nd+nN)->ndOp==\_rem)

{if(nd[nN].pstNd>&nd[nN]&&nd[nN].prnNd!=-1)

{if(nC-nN>1)nd[nN]=nd[nC];

// nN++;

nC++; continue;}

}

// if(nR<-1){nC++; continue;}

if(nr==nC)nr=nN;

if(nd[nC].ndOp==\_brkz&&nd[nC].prvNd==0)

{if(nd[nd[nC].prnNd].prvNd==&nd[nC])

{nd[nC].pstNd->prnNd=nd[nC].prnNd;

nd[nd[nC].prnNd].prvNd=nd[nC].pstNd;}

if(nd[nd[nC].prnNd].pstNd==&nd[nC]) // b 02/06/07

{nd[nC].pstNd->prnNd=nd[nC].prnNd;

nd[nd[nC].prnNd].pstNd=nd[nC].pstNd;}// e 02/06/07

nC++;

}

if(nN!=nC)

{nR=nd[nC].prnNd;

nd[nN]=nd[nC];

//зв'язок з батьківським вузлом

if(nd[nR].prvNd==&nd[nC])

nd[nR].prvNd=&nd[nN];

if(nd[nR].pstNd==&nd[nC])

nd[nR].pstNd=&nd[nN];

if(nd[nN].ndOp>\_cnst){

//зв'язок з лівим наступником

if(nd[nN].prvNd)

nd[nN].prvNd->prnNd=nN;

//зв'язок з правим наступником

if(nd[nN].pstNd)

nd[nN].pstNd->prnNd=nN;

}

// if(nR<nN)nd[nN].prnNd=nR;

// nd[nR].prvNd

}

// else nC

//}else

nN++; nC++;

}while(nC<nn);

return nr;

}

#include "stdafx.h"

#include "tables.h"

#include <string.h>

// порівняння рядків

int cmpStr(unsigned char\* s1, unsigned char\* s2)

{unsigned n=0;

while(s1[n]==s2[n]&&s1[n]!=0)n++;

return s1[n]-s2[n];

}

// порівняння за відношенням порядку

int cmpKys(char \*k0, char \*kArg)

{int i=cmpStr((unsigned char\*)k0,

(unsigned char\*)kArg);

//if(i)

return i;

}

//------------------------------------------------------

// вибірка за двійковим пошуком

struct recrdKWD\*selBin(char\* kArg, struct recrdKWD\*tb, int ln)

{int i, nD=-1, nU=ln, n=(nD+nU)>>1;

while(i=cmpKys(tb[n].key,kArg))

{if(i>0)nU=n;else nD=n;

n=(nD+nU)>>1;

if(n==nD)return NULL;

}

return &tb[n];

}

// сортування для двійкового пошуку

struct recrdKWD\*srtBin(struct recrdKWD\*tb, int ln)

{int n=0, n1;

struct recrdKWD el;

for(;n<ln;n++)for(n1=n+1;n1<ln;n1++)

if(cmpKys(tb[n].key,tb[n1].key)>0)

{el=tb[n];tb[n]=tb[n1];tb[n1]=el;}

return tb;

}