|  |
| --- |
| Київський національний університет  «Київський політехнічний інститут» |
| Факультет інформатики та обчислювальної техніки |
| Кафедра обчислювальної техніки |
| Лабораторна робота №6 |
| з дисципліни «Системне програмування» |
| *Залікова книжка № 4213* |
|  |
| Виконав студент 3-го курсу |
| групи ІО-42 |
| Кочетов Данило |

|  |
| --- |
| 2016 |

Мета: Одержання навичок настройки таблиць для семантичних аналізаторів для обробки для визначення описів даних програм і дерев підлеглості з запам’ятовуванням типів даних для результатів кожного графа внутрішнього подання програми, в тому числі таблиць відповідності операндів і операцій. Вивчення процедур розпізнавання типів і формування внутрішнього подання констант у відповідній машинній формі інструментальної машини.

|  |  |  |
| --- | --- | --- |
| № вар. | Вираз, який відтворюється в дереві розбору | Мова відтворення |
| 13 | float \*b, a[3]; short n,d; b=0;b+=a[n]; | C |

# Лістинг програми

#include "token.h"

#include "visgrp.h"

#include "tables.h"

#include "lexan.h"

#include "syntaxP.h"

#include "langio.h"

#include "seman.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

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');

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);

printf("\n");

SmAnlzr(nodes+nr,nr);

prLxTxt(nodes+nr);

printf("\n");

system("pause");

return 0;

}

#include "token.h"

#include "index.h"

#include "automat.h"

#include "seman.h"

#include "lexcalc.h"

#include <stdlib.h>

#include <stdio.h>

unsigned nbBlk=0;

extern struct recrdSMA ftTbl[353];// таблиця припустимості типів для операцій

struct recrdSMA ftImp= // таблиця припустимості типів для операцій

{\_nil,\_v,0,\_v,0,\_v,0};

extern int nInCr, lnCod[]; // вектор довжин типів

extern enum datType tpLx[];// масив кодів типів констант від типів лексем

extern struct recrdTPD tpTbl[]; // таблиця модифікованих типів

enum tokType lPrv[3]={\_void,\_void,\_void};// масив для накопичення ключових слів типа

extern struct indStrUS ndxNds[50];

// порівняння рядків

int cmpStr(struct recrdSMA \*s1, struct recrdSMA \*s2)

{unsigned n=0;

if(s1->oprtn!=s2->oprtn)return (int)(s1->oprtn)-(int)(s2->oprtn);

if(s1->oprd1-s2->oprd1)return s1->oprd1-s2->oprd1;

if(s1->ln1-s2->ln1)return s1->ln1-s2->ln1;

if(s1->oprd2-s2->oprd2)return s1->oprd2-s2->oprd2;

return s1->ln2-s2->ln2;

}

// вибірка за двійковим пошуком

struct recrdSMA\*selBin(struct recrdSMA \*kArg, struct recrdSMA\*tb, int ln)

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

while(i=cmpStr(tb+n,kArg))

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

n=(nD+nU)>>1;

if(n==nD)return NULL;

}

return &tb[n];

}

// порівняння рядків

int cmpStr(enum tokType s1[3], enum tokType s2[3])

{unsigned n=0;

while(s1[n]==s2[n]&&n<2)n++;

return (int)(s1[n])-(int)(s2[n]);

}

// вибірка за двійковим пошуком

struct recrdTPD\*selBin(enum tokType kArg[3], struct recrdTPD\*tb, int ln)

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

while(i=cmpStr(tb[n].kTp,kArg))

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

n=(nD+nU)>>1;

if(n==nD)return NULL;

}

return &tb[n];

}

void prDtLst(struct lxNode \* nd)

{if(nd->ndOp==\_comma)

{prDtLst(nd->prvNd);

nd->pstNd->dataType=tpLx[nd->pstNd->dataType];

convNum(nd->pstNd); //nInCr++;//, enum ltrTypeS ltrCls[256]);

}else if(nd->ndOp==\_srcn)

{nd->dataType=tpLx[nd->dataType];

convNum(nd); //nInCr++;//, enum ltrTypeS ltrCls[256]);

}

}

enum datType //int

SmAnDcl(int tpCod, struct lxNode \* nd)

{if(nd->ndOp==\_comma)

{SmAnDcl(tpCod,nd->prvNd);

SmAnDcl(tpCod,nd->pstNd);

}else if(nd->ndOp==\_ass)

{SmAnDcl(tpCod,nd->prvNd);

if(nd->pstNd->ndOp==\_srcn)

{nd->pstNd->dataType=tpLx[nd->pstNd->dataType];

convNum(nd->pstNd); //nInCr++;//, enum ltrTypeS ltrCls[256]);

// nd->pstNd->resLength=lnCod[nd->pstNd->dataType];

}else if(nd->pstNd->ndOp==\_tdbz)

{prDtLst(nd->pstNd->pstNd);}

}else if(nd->ndOp==\_refU)//\_mul)//унарна \*

{if(nd->prnNd)SmAnDcl(tpCod+cdPtr,nd->pstNd);

}else if(nd->ndOp==\_ixbz)

{SmAnDcl((tpCod+cdPtr)|cdArr,nd->prvNd);

if(nd->pstNd->ndOp==\_srcn)

{nd->pstNd->dataType=\_ui;

convNum(nd->pstNd); //nInCr++;//, enum ltrTypeS ltrCls[256]);

// nd->pstNd->resLength=lnCod[nd->pstNd->dataType];

}

// SmAnDcl(\_ui,nd->pstNd);

nd->dataType=tpCod;

nd->resLength=lnCod[tpCod&0x7FF];

}else if(nd->ndOp==\_nam)

{nd->dataType=tpCod;

if(tpCod&msPtr)nd->resLength=lnFPtr;

else nd->resLength=lnCod[tpCod&0x7fff];

}

return (enum datType)tpCod;

}

//enum datType

int SmAnlzr(struct lxNode \* nd, // покажчик на початок масиву вузлів

int nR) // номер кореневого вузла

{//enum datType

int tPrv, tPst; // типи вузлів попередника та наступника

int lnPrv, lnPst; // довжини попередника та наступника

char \*name; // робочий покажчик на і'мя

struct recrdTPD\*pRc;

struct indStrUS \*pRtNdx;// робочий покажчик вузла двійкового дерева імен

if(nd->ndOp>=\_void&&nd->ndOp<=\_string)

{lPrv[0]=nd->ndOp;

if(nd->prvNd/\*&&nd->prvNd->ndOp<=\_const\*/)// якщо не одне слово визначає тип

{lPrv[1]=nd->prvNd->ndOp;

if(nd->prvNd->prvNd)//якщо не два слова задають тип

lPrv[2]=nd->prvNd->prvNd->ndOp;

else lPrv[2]=\_void;

}

else {lPrv[1]=\_void; lPrv[2]=\_void;}

pRc=selBin(lPrv, tpTbl, 126);//пошук складеного типа

if(pRc) // якщо тип існує

{tPrv=pRc->dTp;

lnPrv=pRc->ln;

if(nd->ndOp>=\_enum&&nd->ndOp<=\_union)// якщо тип визначено користувачем

{nd->prvNd->dataType=nd->ndOp;

tPrv=(enum datType)(tPrv+(++nbBlk));

}

// if(nd->ndOp==\_enum)

}

SmAnDcl(tPrv,nd->pstNd);// визначити тип

}

else if(nd->ndOp==\_nam)//якщо термінал-ім'я

{pRtNdx=selBTr(nd,ndxNds);//пошук імені

// якщо не знайдено - неописане ім'я

name=(char\*)pRtNdx->pKyStr->prvNd;

/\* nd->pstNd=pRtNdx->pKyStr->pstNd;\*/

nd->dataType=tPrv=(enum datType)pRtNdx->pKyStr->dataType;

nd->resLength=(int)pRtNdx->pKyStr->resLength;

}

else if(nd->ndOp==\_srcn)//якщо термінал-константа

{nd->dataType=tpLx[nd->dataType];

tPrv=(enum datType)nd->dataType;

convNum(nd/\*->pstNd\*/); //перетворення константи на внутрішню

//nInCr++;//, enum ltrTypeS ltrCls[256]);

// nd->resLength=lnCod[tPrv];

}

else{if(nd->ndOp==\_remL)return \_v;

if(nd->prvNd&&nd->ndOp!=\_nam&&nd->ndOp!=\_srcn)

{if(nd->ndOp!=\_brkz&&nd->prnNd)

{tPrv=SmAnlzr(nd->prvNd,nR);

lnPrv=nd->prvNd->resLength;}

}else{ftImp.oprd1=tPrv=\_v;ftImp.ln1=0;}

if(nd->pstNd&&nd->ndOp!=\_nam&&nd->ndOp!=\_srcn)

{tPst=SmAnlzr(nd->pstNd,nR);

lnPst=nd->pstNd->resLength;}

else{ftImp.oprd2=tPst=\_v;ftImp.ln2=0;}

if(nd->ndOp==\_EOS)

{nd->dataType=\_v; nd->resLength=0;

}else if(nd->ndOp==\_brkz)

{if(nd->prvNd==0)

{nd->dataType=nd->pstNd->dataType; nd->resLength=nd->pstNd->resLength;

tPrv=(enum datType)nd->dataType;}

else

{nd->dataType=nd->prvNd->dataType;nd->resLength=nd->prvNd->resLength;

if(nd->prvNd&&nd->prvNd->ndOp==\_refU)

{tPrv=tPst-cdPtr; nd->dataType=tPrv;

if(tPrv>=cdPtr)nd->resLength=32;

else nd->resLength=lnCod[tPrv&0x7FF];

}

else tPrv=(enum datType)nd->dataType;}

}

else

{ftImp.oprd1=tPrv&0xffff7fff;

if(ftImp.oprtn>=\_asOr&&ftImp.oprtn<=\_ass)tPrv&=0xfff7ffff;

if(tPrv!=\_v)ftImp.ln1=lnPrv;else ftImp.ln1=lnPrv=0;

ftImp.oprd2=tPst&0xffff7fff;

if(tPst!=\_v)ftImp.ln2=lnPst;else ftImp.ln2=lnPst=0;

ftImp.oprtn=nd->ndOp;

struct recrdSMA\*

pftImp = selBin(&ftImp, ftTbl, 361);

if(pftImp)

{nd->dataType=pftImp->res;tPrv=(enum datType)pftImp->res;

nd->resLength=pftImp->lnRes;

}}

}

return tPrv;

}

