НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6

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

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Київ – 2014 р.

**Тема роботи**: **Побудова, на­строювання та використання се­ман­тич­ного аналі­за­то­ра в транслято­рах**

**Мета роботи**: 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.

**Завдання на роботу**

Завдання на підготовку до роботи на комп’ютері:

1. Визначити варіант завдання для основних задач за табли­цею 7.1. Визна­чи­ти приклади лексем через файл в папці spLb7 модуля тестування spLb7.cpp.

2. Відповісти на контрольні запитання.

3. Підготувати настройки вхідної мови програмування.

4. Використати структуру елементу struct lxNode з файлу index.h шаб­лону програмного проекту spLb7 для побудови елементу індексу таблиць лексем і визначити поля, що заповнюються при семантичному аналізі.

5. Підготувати програмний модуль контрольної задачі, який виконує заданий варіант з таблиці 6.1 і дозволяє перевірити коректність виконання програм. Для цього доповнити таблиці відповідності типів результатів типам операндів struct recrdSMA ftTbl[179] в модулі реалізації таблиць semanT.cpp для семантичного аналізу, доповнивши її потрібними для варіанта елементами.

Порядок вибору варіанту:

За останньою цифрою номера залікової книжки або за порядковим номером студента в списку підгрупи з доданим номером групи визначте за табл. 6.1 варіант оброблюваних даних та настройки програм за прикладом.

Завдання на роботу на комп’ютері

6. Побудувати програмний проект, ввівши програмні модулі у відповідні файли проекту і налагодити синтаксис.

7. Побудувати виконуваний модуль тестової програми і налагодити змістовне виконання програми для перевірки результатів контрольних прикладів, заданих в варіантах з таблиці 6.1, що включають помилкові покажчики в описах даних.

8. Одержати результати виконання, проаналізувати виникнення діагностичних повідомлень в них в режимі налагодження і зробити висновки.

9. Одержати результати виконання, проаналізувати коректність сформованих типів результатів в режимі налагодження і зробити висновки.

10. Продемонструвати результати викладачам

Варіант завдання:

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

// spLb7.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 "..\spLb5\syntaxP.h"

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

#include "seman.h"

#include <stdio.h>

#include <string.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=ltClsP;

enum tokType \*dlCds=dlCdsP;

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);

printf("\n");

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

SmAnlzr(nodes+nr,nr);

prLxTxt(nodes+nr);

printf("\n");

getchar();

getchar();

return 0;

}

#include "stdafx.h"

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

#include "..\spLb4\index.h"

#include "..\spLb4\automat.h"

#include "seman.h"

#include "lexcalc.h"

#include <stdlib.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=0, 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;

}

#include "stdafx.h"

#include "syntaxP.h"

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

#include <stdio.h>

struct lxNode nodes[MAX\_NODES]= // 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},

};

#include "stdafx.h"

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

#include "seman.h"

struct recrdSMA ftTbl[]= // таблиця припустимості типів для операцій

{{\_ass,\_ui,32,\_si,32,\_ui,32},

{\_dcr,\_v,0,\_ui,32,\_ui,32},

{\_add,\_ui,32,\_ui,32,\_ui,32},

{\_mul,\_si,32,\_ui,32,\_si,32},

{\_mul,\_si,32,\_si,32,\_si,32},

{\_ixbz,\_si+cdPtr|cdCns,32,\_ui,32,\_si,32},

};

int lnCod[]= // вектор довжин типів

{0, 0, 0, 0,

8,16,32,64,

8,16,32,64,//

32,64,80,48,

\_lbl,//

\_enm,\_str,\_unn,

};

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

{\_v, //0 Eu - Некласифікований об'єкт

\_v, //1 S0 - Роздільник

\_v, //2 S1g - Знак числової константи

\_ui, //3 S1c - Ціле число

\_f, //4 S2c - Число з точкою

\_v, //5 S1e - Літера "e" або "E"

\_v, //6 S1q - Знак "-" або "+"

\_f, //7 S1p - Десяткові цифри порядку

\_v, //8 S1n - Елементи імені

\_v, //9 S1s - Літери рядка або символьної константи

\_v, //10 S1t - Елементи констант, які перетворюються

\_strn, //11 S2s - Ознака закінчення константи

\_v, //12 S2 - Початковий елемент групового роздільника

\_v, //13 S3 - Наступний елемент групового роздільника

\_ui, //14?S3c - Ціле число з недесятковою основою

\_v, //15?S0p - Ознака типу константи

\_v, //16 Soc- Вісімковий код

\_v, //17 Scr- Коментар-рядок

\_v, //18 Scl- Обмежений коментар

\_v, //19 Ec - Неправильна константа

\_v, //20 Ep - Неправильна константа з точкою

\_v, // Eq - Неправильна константа з порядком

\_v, // En - Неправильне ім'я

\_v // Eo - Неприпустиме сполучення операцій

};

struct recrdTMD tpLxMd[]=

// масив кодів та ознак ключових слів типів

{{\_v, 0, 0}, //0 \_void

{\_v, 0, 0}, //1 \_extern

{\_v, 0, 0}, //2 \_var

{\_v,cdCns,0}, //3 \_const

{\_enm, 0,32}, //4 \_enum

{\_str, 0, 0}, //5 \_struct/\*\_record\*/

{\_unn, 0, 0}, //6 \_union

{\_v,cdReg,0}, //7 \_register

{\_ui,0,32}, //8 \_unsigned

{\_si,0,32}, //9 \_signed

{\_si,0, 8}, //10 \_char

{\_si,0,16}, //11 \_short

{\_si,0,32}, //12 \_int

{\_si,0,32}, //13 \_long

{\_si,0,64}, //14 \_sint64

{\_ui,0,64}, //15 \_uint64

{\_f, 0,32}, //16 \_float

{\_d, 0,64}, //17 \_double

};

struct recrdTPD tpTbl[]= // таблиця модифікованих типів

{{{\_void,\_void,\_void},\_v,0},

{{\_void,\_extern,\_void},\_v+cdExt,0},

{{\_void,\_const,\_void},\_v+cdCns,0},

{{\_void,\_register,\_void},\_v+cdReg,0},

{{\_void,\_auto,\_void},\_v+cdAut,0},

{{\_void,\_static,\_void},\_v+cdStt,0},

{{\_enum,\_void,\_void},\_enm,32},

{{\_enum,\_extern,\_void},\_enm+cdExt,32},

{{\_enum,\_const,\_void},\_enm+cdCns,32},

{{\_enum,\_register,\_void},\_enm+cdReg,32},

{{\_enum,\_auto,\_void},\_enm+cdAut,32},

{{\_enum,\_static,\_void},\_enm+cdStt,32},

{{\_struct,\_void,\_void},\_str,0},

{{\_struct,\_extern,\_void},\_str+cdExt,0},

{{\_struct,\_const,\_void},\_str+cdCns,0},

{{\_struct,\_register,\_void},\_str+cdReg,0},

{{\_struct,\_auto,\_void},\_str+cdAut,0},

{{\_struct,\_static,\_void},\_str+cdStt,0},

{{\_union,\_void,\_void},\_unn,0},

{{\_union,\_extern,\_void},\_unn+cdExt,0},

{{\_union,\_const,\_void},\_unn+cdCns,0},

{{\_union,\_register,\_void},\_unn+cdReg,0},

{{\_union,\_auto,\_void},\_unn+cdAut,0},

{{\_union,\_static,\_void},\_unn+cdStt,0},

{{\_unsigned,\_void,\_void},\_ui,32},

{{\_unsigned,\_extern,\_void},\_ui+cdExt,32},

{{\_unsigned,\_const,\_void},\_ui+cdCns,32},

{{\_unsigned,\_register,\_void},\_ui+cdReg,32},

{{\_unsigned,\_auto,\_void},\_ui+cdAut,32},

{{\_unsigned,\_static,\_void},\_ui+cdStt,32},

{{\_signed,\_void,\_void},\_si,32},

{{\_signed,\_extern,\_void},\_si+cdExt,32},

{{\_signed,\_const,\_void},\_si+cdCns,32},

{{\_signed,\_register,\_void},\_si+cdReg,32},

{{\_signed,\_auto,\_void},\_si+cdAut,32},

{{\_signed,\_static,\_void},\_si+cdStt,32},

{{\_char,\_unsigned,\_void},\_uc,8},

{{\_char,\_unsigned,\_extern},\_uc+cdExt,8},

{{\_char,\_unsigned,\_const},\_uc+cdCns,8},

{{\_char,\_unsigned,\_register},\_uc+cdReg,8},

{{\_char,\_unsigned,\_auto},\_uc+cdAut,8},

{{\_char,\_unsigned,\_static},\_uc+cdStt,8},

{{\_char,\_signed,\_void},\_sc,8},//4

{{\_char,\_signed,\_extern},\_sc+cdExt,8},//4

{{\_char,\_signed,\_const},\_sc+cdCns,8},//4

{{\_char,\_signed,\_register},\_sc+cdReg,8},//4

{{\_char,\_signed,\_auto},\_sc+cdAut,8},//4

{{\_char,\_signed,\_static},\_sc+cdStt,8},//4

{{\_char,\_void,\_void},\_sc,8},

{{\_char,\_extern,\_void},\_sc+cdExt,8},

{{\_char,\_const,\_void},\_sc+cdCns,8},

{{\_char,\_register,\_void},\_sc+cdReg,8},

{{\_char,\_auto,\_void},\_sc+cdAut,8},

{{\_char,\_static,\_void},\_sc+cdStt,8},

{{\_short,\_void,\_void},\_si,16},

{{\_short,\_extern,\_void},\_si+cdExt,16},

{{\_short,\_const,\_void},\_si+cdCns,16},

{{\_short,\_register,\_void},\_si+cdReg,16},

{{\_short,\_auto,\_void},\_si+cdAut,16},

{{\_short,\_static,\_void},\_si+cdStt,16},

{{\_short,\_unsigned,\_void},\_ui,16},

{{\_short,\_unsigned,\_extern},\_ui+cdExt,16},

{{\_short,\_unsigned,\_const},\_ui+cdCns,16},

{{\_short,\_unsigned,\_register},\_ui+cdReg,16},

{{\_short,\_unsigned,\_auto},\_ui+cdAut,16},

{{\_short,\_unsigned,\_static},\_ui+cdStt,16},

{{\_short,\_signed,\_void},\_si,16},

{{\_short,\_signed,\_extern},\_si+cdExt,16},

{{\_short,\_signed,\_const},\_si+cdCns,16},

{{\_short,\_signed,\_register},\_si+cdReg,16},

{{\_short,\_signed,\_auto},\_si+cdAut,16},

{{\_short,\_signed,\_static},\_si+cdStt,16},

{{\_int,\_void,\_void},\_si,32},//9

{{\_int,\_extern,\_void},\_si+cdExt,32},//9

{{\_int,\_const,\_void},\_si+cdCns,32},//9

{{\_int,\_register,\_void},\_si+cdReg,32},//9

{{\_int,\_auto,\_void},\_si+cdAut,32},//9

{{\_int,\_static,\_void},\_si+cdStt,32},//9

{{\_int,\_unsigned,\_void},\_ui,32},

{{\_int,\_unsigned,\_extern},\_ui+cdExt,32},

{{\_int,\_unsigned,\_const},\_ui+cdCns,32},

{{\_int,\_unsigned,\_register},\_ui+cdReg,32},

{{\_int,\_unsigned,\_auto},\_ui+cdAut,32},

{{\_int,\_unsigned,\_static},\_ui+cdStt,32},

{{\_int,\_signed,\_void},\_si,32},

{{\_int,\_signed,\_extern},\_si+cdExt,32},

{{\_int,\_signed,\_const},\_si+cdCns,32},

{{\_int,\_signed,\_register},\_si+cdReg,32},

{{\_int,\_signed,\_auto},\_si+cdAut,32},

{{\_int,\_signed,\_static},\_si+cdStt,32},

{{\_int,\_long,\_void},\_si,32},

{{\_int,\_long,\_extern},\_si+cdExt,32},

{{\_int,\_long,\_const},\_si+cdCns,32},

{{\_int,\_long,\_register},\_si+cdReg,32},

{{\_int,\_long,\_auto},\_si+cdAut,32},

{{\_int,\_long,\_static},\_si+cdStt,32},

{{\_long,\_void,\_void},\_si,32},

{{\_long,\_extern,\_void},\_si+cdExt,32},

{{\_long,\_const,\_void},\_si+cdCns,32},

{{\_long,\_register,\_void},\_si+cdReg,32},

{{\_long,\_auto,\_void},\_si+cdAut,32},

{{\_long,\_const,\_void},\_si+cdStt,32},

{{\_float,\_void,\_void},\_f,32},//14

{{\_float,\_extern,\_void},\_f+cdExt,32},//14

{{\_float,\_const,\_void},\_f+cdCns,32},//14

{{\_float,\_register,\_void},\_f+cdReg,32},//14

{{\_float,\_auto,\_void},\_f+cdAut,32},//14

{{\_float,\_static,\_void},\_f+cdStt,32},//14

{{\_double,\_void,\_void},\_d,64},

{{\_double,\_extern,\_void},\_d+cdExt,64},

{{\_double,\_const,\_void},\_d+cdCns,64},

{{\_double,\_register,\_void},\_d+cdReg,64},

{{\_double,\_auto,\_void},\_d+cdAut,64},

{{\_double,\_static,\_void},\_d+cdStt,64},

{{\_double,\_long,\_void},\_ld,80},

{{\_double,\_long,\_extern},\_ld+cdExt,80},

{{\_double,\_long,\_const},\_ld+cdCns,80},

{{\_double,\_long,\_register},\_ld+cdReg,80},

{{\_double,\_long,\_auto},\_ld+cdAut,80},

{{\_double,\_long,\_static},\_ld+cdStt,80},

{{\_class,\_void,\_void},\_cls,0},

{{\_class,\_extern,\_void},\_cls+cdExt,0},

{{\_class,\_const,\_void},\_cls+cdCns,0},

{{\_class,\_register,\_void},\_cls+cdReg,0},

{{\_class,\_auto,\_void},\_cls+cdAut,0},

{{\_class,\_static,\_void},\_cls+cdStt,0},

{{\_class,\_volatile,\_void},\_cls+cdVlt,0},

};