{\* 注释者：周美廷—76066002 \*}

{\* 程序名 PASCALS \*}

program PASCALS(INPUT,OUTPUT,PRD,PRR);

{ author:N.Wirth, E.T.H. CH-8092 Zurich,1.3.76 }

{ modified by R.E.Berry

Department of computer studies

University of Lancaster

Variants of this program are used on

Data General Nova,Apple,and

Western Digital Microengine machines. }

{ further modified by M.Z.Jin

Department of Computer Science&Engineering BUAA,0ct.1989

}

{\* 全局变量定义 \*}

{\* 常数变量 \*}

const nkw = 27; { no. of key words } {\* 关键字数量 \*}

alng = 10; { no. of significant chars in identifiers } {\* 标识符中的重要字符数\*}

llng = 121; { input line length } {\* 输入长度\*}

emax = 322; { max exponent of real numbers } {\* 实数的最大指数 \*}

emin = -292; { min exponent } {\* 最小指数 \*}

kmax = 15; { max no. of significant digits } {\* 最大有效位数 \*}

tmax = 100; { size of table } {\* 表的大小 \*}

bmax = 20; { size of block-talbe } {\* 块表的大小 \*}

amax = 30; { size of array-table } {\* 数组表的大小 \*}

c2max = 20; { size of real constant table } {\* 实常数表的大小 \*}

csmax = 30; { max no. of cases } {\* 最大案例数 \*}

cmax = 800; { size of code } {\* 代码大小 \*}

lmax = 7; { maximum level } {\* 最高嵌套层次 \*}

smax = 600; { size of string-table } {\* 字符串表的大小 \*}

ermax = 58; { max error no. } {\* 最大错误数 \*}

omax = 63; { highest order code } {\* 最高序列代码 \*}

xmax = 32767; { 2\*\*15-1 } {\* 2\*\*15-1 \*}

nmax = 32767; { 2\*\*15-1 } {\* 2\*\*15-1 \*}

lineleng = 132; { output line length } {\* 输出长度 \*}

linelimit = 200; {\* 行最长度 \*}

stacksize = 1450; {\* 堆栈大小 \*}

{\* 类定义 \*}

{\* Enumerated \*}

type symbol = ( intcon, realcon, charcon, stringcon,

notsy, plus, minus, times, idiv, rdiv, imod, andsy, orsy,

eql, neq, gtr, geq, lss, leq,

lparent, rparent, lbrack, rbrack, comma, semicolon, period,

colon, becomes, constsy, typesy, varsy, funcsy,

procsy, arraysy, recordsy, programsy, ident,

beginsy, ifsy, casesy, repeatsy, whilesy, forsy,

endsy, elsesy, untilsy, ofsy, dosy, tosy, downtosy, thensy);

{\* Subrange \*}

index = -xmax..+xmax;

alfa = packed array[1..alng]of char;

{\* Enumerated \*}

objecttyp = (konstant, vvariable, typel, prozedure, funktion );

types = (notyp, ints, reals, bools, chars, arrays, records );

symset = set of symbol;

typset = set of types;

item = record

typ: types;

ref: index;

end;

order = packed record

f: -omax..+omax;

x: -lmax..+lmax;

y: -nmax..+nmax

end;

{\* 变量 \*}

{\* 从源程序读取的最后一个字符： 字符 \*}

var ch: char; { last character read from source program }

{\* 来自insymbol的实数： 实数 \*}

rnum: real; { real number from insymbol }

{\* 来自insymbol的整数： 整数 \*}

inum: integer; { integer from insymbol }

{\* 字符串长度： 整数 \*}

sleng: integer; { string length }

{\* 字符计数： 整数 \*}

cc: integer; { character counter }

{\* 程序位置计数： 整数 \*}

lc: integer; { program location counter }

{\* 当前行的长度： 整数 \*}

ll: integer; { length of current line }

{\* 错误的位置： 整数 \*}

errpos: integer;

{\* 表的索引： 整数 \*}

t,a,b,sx,c1,c2:integer; { indices to tables }

{\* 布尔 \*}

iflag, oflag, skipflag, stackdump, prtables: boolean;

{\* 由insymbol读取的最后一个符号： 符号 \*}

sy: symbol; { last symbol read by insymbol }

{\* 错误数 \*}

errs: set of 0..ermax;

{\* 来自insymbol的标识符： alfa \*}

id: alfa; { identifier from insymbol }

progname: alfa;

stantyps: typset;

constbegsys, typebegsys, blockbegsys, facbegsys, statbegsys: symset;

{\* 行： 字符数组（1-llng）\*}

line: array[1..llng] of char;

{\* 键： alfa数组（1-nkw）\*}

key: array[1..nkw] of alfa;

{\* 符号数组（1-nkw）\*}

ksy: array[1..nkw] of symbol;

{\* 特殊符号： 符号数组（字符）\*}

sps: array[char]of symbol; { special symbols }

{\* 整数数组（0-lmax）\*}

display: array[0..lmax] of integer;

{\* 标识符标签 \*}

tab: array[0..tmax] of { indentifier lable }

packed record

name: alfa;

link: index;

obj: objecttyp;

typ: types;

ref: index;

normal: boolean;

lev: 0..lmax;

adr: integer

end;

{\* 阵列表 \*}

atab: array[1..amax] of { array-table }

packed record

inxtyp,eltyp: types;

elref,low,high,elsize,size: index

end;

{\* 块表 \*}

btab: array[1..bmax] of { block-table }

packed record

last, lastpar, psize, vsize: index

end;

{\* 字符串表 \*}

stab: packed array[0..smax] of char; { string table }

rconst: array[1..c2max] of real;

code: array[0..cmax] of order;

{\* pascal p中的默认值 \*}

psin,psout,prr,prd:text; { default in pascal p }

inf, outf, fprr: string;

{\* 过程： 错误信息 \*}

procedure errormsg;

{\* 变量定义 \*}

var k : integer;

msg: array[0..ermax] of alfa;

{\* 局部变量定义 \*}

begin

msg[0] := 'undef id '; msg[1] := 'multi def ';

msg[2] := 'identifier'; msg[3] := 'program ';

msg[4] := ') '; msg[5] := ': ';

msg[6] := 'syntax '; msg[7] := 'ident,var ';

msg[8] := 'of '; msg[9] := '( ';

msg[10] := 'id,array '; msg[11] := '( ';

msg[12] := '] '; msg[13] := '.. ';

msg[14] := '; '; msg[15] := 'func. type';

msg[16] := '= '; msg[17] := 'boolean ';

msg[18] := 'convar typ'; msg[19] := 'type ';

msg[20] := 'prog.param'; msg[21] := 'too big ';

msg[22] := '. '; msg[23] := 'type(case)';

msg[24] := 'character '; msg[25] := 'const id ';

msg[26] := 'index type'; msg[27] := 'indexbound';

msg[28] := 'no array '; msg[29] := 'type id ';

msg[30] := 'undef type'; msg[31] := 'no record ';

msg[32] := 'boole type'; msg[33] := 'arith type';

msg[34] := 'integer '; msg[35] := 'types ';

msg[36] := 'param type'; msg[37] := 'variab id ';

msg[38] := 'string '; msg[39] := 'no.of pars';

msg[40] := 'real numbr'; msg[41] := 'type ';

msg[42] := 'real type '; msg[43] := 'integer ';

msg[44] := 'var,const '; msg[45] := 'var,proc ';

msg[46] := 'types(:=) '; msg[47] := 'typ(case) ';

msg[48] := 'type '; msg[49] := 'store ovfl';

msg[50] := 'constant '; msg[51] := ':= ';

msg[52] := 'then '; msg[53] := 'until ';

msg[54] := 'do '; msg[55] := 'to downto ';

msg[56] := 'begin '; msg[57] := 'end ';

msg[58] := 'factor';

{\* 输出在psout文件 \*}

writeln(psout);

{\* 输出'key words\n'在psout文件 \*}

writeln(psout,'key words');

{\* 以k为计数变量，如果k不在errs数组里，继续寻找，直到找到，然后输出跟k对应的错误信息 \*}

{\* k初始值为0 \*}

k := 0;

{\* while errs数组不为空 \*}

while errs <> [] do

begin

{\* 如果k不在errs数组里，k = k+1 \*}

while not( k in errs )do k := k + 1;

{\* 输出k+ ' '+ msg[k]\n在psout文件 \*}

writeln(psout, k, ' ', msg[k] );

{\* errs – [k] \*}

errs := errs - [k]

end { while errs }

end { errormsg } ;

{\* 过程： endskip（被跳过部分的输入画下划线）\*}

procedure endskip;

begin { underline skipped part of input }

{\* while errpos小于cc \*}

while errpos < cc do

begin

{\* 输出'-'在psout文件 \*}

write( psout, '-');

{\* errpos + 1 \*}

errpos := errpos + 1

end;

{\* 重启skipflag（false） \*}

skipflag := false

end { endskip };

{\* 过程： 读下一个字; 生产线结束 \*}

procedure nextch; { read next character; process line end }

begin

{\* 如果计数变量等于当前行结尾 \*}

if cc = ll

then begin

{\* 如果已达到psin的文件结尾，那么就是错误（错误处理） \*}

if eof( psin )

then begin

writeln( psout );

writeln( psout, 'program incomplete' );

errormsg;

exit;

end;

{\* 如果errpos不等于0 \*}

if errpos <> 0

then begin

{\* 如果skipflag有效，那代表有被跳过的输入，要处理（执行endskip） \*}

if skipflag then endskip;

writeln( psout );

{\* errpos清零 \*}

errpos := 0

end;

write( psout, lc: 5, ' ');

{\* ll和cc都清零（重启） \*}

ll := 0;

cc := 0;

{\* while 还没到psin行的结束 \*}

while not eoln( psin ) do

begin

{\* 当前行加1 \*}

ll := ll + 1;

{\* 读取psin中的字符放在ch \*}

read( psin, ch );

{\* 在psout文件输出ch \*}

write( psout, ch );

{\* 当前行的字符值为ch的值\*}

line[ll] := ch

end;

{\* 当前行加1 \*}

ll := ll + 1;

{\* 读取psin的当前一行，鼠标放在下一行 \*}

readln( psin );

{\* line[ll]的字符值为空格 \*}

line[ll] := ' ';

{\* 输出新行在psout \*}

writeln( psout );

end;

{\* 计数变量加1 \*}

cc := cc + 1;

{\* ch赋值line[cc]的值 \*}

ch := line[cc];

end { nextch };

{\* 过程： error （参数：n为整数）\*}

procedure error( n: integer );

begin

{\* 如果errpos等于0，输出psout+'\*\*\*\*' \*}

if errpos = 0

then write ( psout, '\*\*\*\*' );

{\* 如果cc大于errpos \*}

if cc > errpos

then begin

{\* 在psout文件输出' ': cc-errpos+'^'+ n:2 \*}

write( psout, ' ': cc-errpos, '^', n:2);

{\* errpos = cc+3 \*}

errpos := cc + 3;

{\* errs = errs+[n] \*}

errs := errs +[n]

end

end { error };

{\* 过程： fatal（参数：n为整数） \*}

procedure fatal( n: integer );

{\* 变量定义 \*}

var msg : array[1..7] of alfa;

begin

{\* 输出新行在psout \*}

writeln( psout );

{\* 执行errormsg \*}

errormsg;

{\* 局部变量定义 \*}

msg[1] := 'identifier'; msg[2] := 'procedures';

msg[3] := 'reals '; msg[4] := 'arrays ';

msg[5] := 'levels '; msg[6] := 'code ';

msg[7] := 'strings ';

{\* 在psout文件输出'compiler table for'+msg[n]+' is too small'\n \*}

writeln( psout, 'compiler table for ', msg[n], ' is too small');

exit; {terminate compilation }

end { fatal };

{\* 过程： insymbol（读下一个符号） \*}

procedure insymbol; {reads next symbol}

{\* 标签定义，用于goto \*}

label 1,2,3;

{\* 变量定义 \*}

var i,j,k,e: integer;

{\* 过程： readscale \*}

procedure readscale;

{\* 变量定义 \*}

var s,sign: integer;

begin

{\* 执行nextch \*}

nextch;

{\* sign = 1，s = 0 \*}

sign := 1;

s := 0;

{\* 如果ch='+'执行nextch \*}

if ch = '+'

then nextch

{\* 而如果ch等于'-'，执行nextch，以及sign-1 \*}

else if ch = '-'

then begin

nextch;

sign := -1

end;

{\* 如果ch<'0'以及ch>'9'，那么执行error（参数为40）\*}

if not(( ch >= '0' )and (ch <= '9' ))

then error( 40 )

{\* 不然，重复以下程序直到ch<'0'以及ch>'9' \*}

else repeat

{\* s = 10\*s + ord( ord(ch)-ord('0') \*}

s := 10\*s + ord( ord(ch)-ord('0'));

{\* 执行nextch \*}

nextch;

until not(( ch >= '0' ) and ( ch <= '9' ));

e := s\*sign + e

end { readscale };

{\* 过程： adjustscale \*}

procedure adjustscale;

{\* 变量定义 \*}

var s : integer;

d, t : real;

begin

{\* 如果 k + e > emax，那么执行error（参数为21） \*}

if k + e > emax

then error(21)

{\* 而如果 k + e < emin，那么rnum=0 \*}

else if k + e < emin

then rnum := 0

{\* 不然执行以下程序 \*}

else begin

{\* s = abs(e)，t = 1.0，d = 10.0 \*}

s := abs(e);

t := 1.0;

d := 10.0;

{\* 重复以下程序，直到s等于0 \*}

repeat

{\* while s不是奇书，做一下程序 \*}

while not odd(s) do

begin

{\* s = s div 2，d = sqr(d) \*}

s := s div 2;

d := sqr(d)

end;

{\* s = s - 1，t = d \* t \*}

s := s - 1;

t := d \* t

until s = 0;

{\* 如果e大于等于0，那么rnum = rnum \* t \*}

if e >= 0

then rnum := rnum \* t

{\* 不然rnum = rnum / t \*}

else rnum := rnum / t

end

end { adjustscale };

{\* 过程： options \*}

procedure options;

{\* 过程： switch（参数b为布尔） \*}

procedure switch( var b: boolean );

begin

{\* 如果ch等于'+'那么b有效 \*}

b := ch = '+';

{\* b不是'+'的情况下，若也不是'-' \*}

if not b

then if not( ch = '-' )

then begin { print error message }

{\* while ch不是'\*'也不是','，那么读取下一个字符 \*}

while( ch <> '\*' ) and ( ch <> ',' ) do

nextch;

end

else nextch

else nextch

end { switch };

begin { options }

{\* 重复以下程序直到ch不等于',' \*}

repeat

nextch;

{\* 如果ch不是'\*'，而是't'读下一个字符然后switch(prtables)，而如果是's'，读下一个字符然后switch(stackdump) \*}

if ch <> '\*'

then begin

if ch = 't'

then begin

nextch;

switch( prtables )

end

else if ch = 's'

then begin

nextch;

switch( stackdump )

end;

end

until ch <> ','

end { options };

begin { insymbol }

{\* 如果ch是空格或htab，执行以下程序 \*}

1: while( ch = ' ' ) or ( ch = chr(9) ) do

nextch; { space & htab }

{\* 如果ch是字母 \*}

case ch of

'a','b','c','d','e','f','g','h','i',

'j','k','l','m','n','o','p','q','r',

's','t','u','v','w','x','y','z':

begin { identifier of wordsymbol }

{\* 变量定义 \*}

k := 0;

id := ' ';

{\* 重复以下程序直到ch不为任何字母或数字 \*}

repeat

{\* 如果k的长度小于重要字符数，那么继续读取下一个字符放到id数组里 \*}

if k < alng

then begin

k := k + 1;

id[k] := ch

end;

nextch

until not((( ch >= 'a' ) and ( ch <= 'z' )) or (( ch >= '0') and (ch <= '9' )));

{\* 变量定义 \*}

i := 1;

j := nkw; { binary search }

{\* 重复以下程序直到i > j \*}

repeat

k := ( i + j ) div 2;

if id <= key[k]

then j := k - 1;

if id >= key[k]

then i := k + 1;

until i > j;

if i - 1 > j

then sy := ksy[k]

else sy := ident

end;

{\* 如果ch是数字 \*}

'0','1','2','3','4','5','6','7','8','9':

begin { number }

{\* 变量定义 \*}

k := 0;

inum := 0;

sy := intcon;

{\* 重复以下程序直到ch不为数字 \*}

repeat

inum := inum \* 10 + ord(ch) - ord('0');

k := k + 1;

nextch

until not (( ch >= '0' ) and ( ch <= '9' ));

if( k > kmax ) or ( inum > nmax )

then begin

error(21);

inum := 0;

k := 0

end;

{\* 如果ch是'.'那继续读下一个字符，如果那字符是'.'，那给ch赋值':' \*}

if ch = '.'

then begin

nextch;

if ch = '.'

then ch := ':'

else begin

{\* 变量定义 \*}

sy := realcon;

rnum := inum;

e := 0;

{\* 执行以下程序直到ch不是数字 \*}

while ( ch >= '0' ) and ( ch <= '9' ) do

begin

e := e - 1;

rnum := 10.0 \* rnum + (ord(ch) - ord('0'));

nextch

end;

{\* 如果e是0，那就是第40错误 \*}

if e = 0

then error(40);

{\* 如果ch是e，那执行readscale \*}

if ch = 'e' then readscale;

{\* 如果e不是0，那执行adjustscale \*}

if e <> 0 then adjustscale

end

end

{\* 如果ch是'e'，那定义以下变量 \*}

else if ch = 'e'

then begin

sy := realcon;

rnum := inum;

e := 0;

readscale;

{\* 如果e不是0，那执行adjustscale \*}

if e <> 0

then adjustscale

end;

end;

{\* 如果ch是':'，然后下一个ch是'='，那么符号是becomes然后继续读下一个字符，

不然sy是colon \*}

':':

begin

nextch;

if ch = '='

then begin

sy := becomes;

nextch

end

else sy := colon

end;

{\* 如果ch是'<'，然后下一个ch是'='，那么符号是leq然后继续读下一个字符，

而如果下一个是'>'，那么符号是neq然后继续读下一个字符，不然sy是lss \*}

'<':

begin

nextch;

if ch = '='

then begin

sy := leq;

nextch

end

else

if ch = '>'

then begin

sy := neq;

nextch

end

else sy := lss

end;

{\* 如果ch是'>'，然后下一个ch是'='，那么符号是geq然后继续读下一个字符，

不然sy是gtr \*}

'>':

begin

nextch;

if ch = '='

then begin

sy := geq;

nextch

end

else sy := gtr

end;

{\* 如果ch是'.'，然后下一个ch是'.'，那么符号是colon然后继续读下一个字符，

不然sy是period \*}

'.':

begin

nextch;

if ch = '.'

then begin

sy := colon;

nextch

end

else sy := period

end;

{\* 如果ch是'''' \*}

'''':

begin

{\* 清零k \*}

k := 0;

{\* 读下一个字符 \*}

2: nextch;

{\* 如果ch是''''，然后下一个ch是不是''''，goto 3 \*}

if ch = ''''

then begin

nextch;

if ch <> ''''

then goto 3

end;

{\* 如果sx+k=smax那就是第7fatal \*}

if sx + k = smax

then fatal(7);

stab[sx+k] := ch;

k := k + 1;

{\* 如果cc是1，k要清零，不然goto 2 \*}

if cc = 1

then begin { end of line }

k := 0;

end

else goto 2;

3: if k = 1

then begin

sy := charcon;

inum := ord( stab[sx] )

end

else if k = 0

then begin

error(38);

sy := charcon;

inum := 0

end

else begin

sy := stringcon;

inum := sx;

sleng := k;

sx := sx + k

end

end;

{\* 如果ch是'('，然后下一个ch不是'\*'，那么符号是lparent，

不然如果下一个ch是'$'，那么执行options \*}

'(':

begin

nextch;

if ch <> '\*'

then sy := lparent

else begin { comment }

nextch;

if ch = '$'

then options;

{\* 重复以下程序直到ch为')'' \*}

repeat

{\* while ch不是'\*'，读下一个字符 \*}

while ch <> '\*' do nextch;

{\* 读下一个字符 \*}

nextch

until ch = ')';

{\* 读下一个字符后goto 1 \*}

nextch;

goto 1

end

end;

{\* 如果ch是'{'，然后下一个ch是'$'，那么执行options，while ch不是花括号，

读下一个字符，不然读下一个字符然后goto 1 \*}

'{':

begin

nextch;

if ch = '$'

then options;

while ch <> '}' do

nextch;

nextch;

goto 1

end;

{\* 如果ch是以下字符，那么符号是sps[ch]然后读下一个字符 \*}

'+', '-', '\*', '/', ')', '=', ',', '[', ']', ';':

begin

sy := sps[ch];

nextch

end;

{\* 如果ch是以下字符，那么是第24错误然后读下一个字符然后goto 1 \*}

'$','"' ,'@', '?', '&', '^', '!':

begin

error(24);

nextch;

goto 1

end

end { case }

end { insymbol };

{\* 过程： enter（参数x0，x1，x2，x3） \*}

procedure enter(x0:alfa; x1:objecttyp; x2:types; x3:integer );

begin

t := t + 1; { enter standard identifier }

with tab[t] do

begin

name := x0;

link := t - 1;

obj := x1;

typ := x2;

ref := 0;

normal := true;

lev := 0;

adr := x3;

end

end; { enter }

{\* 过程： enterarray（参数tp，l，h） \*}

procedure enterarray( tp: types; l,h: integer );

begin

{\* 如果l大于h，就是第27错误 \*}

if l > h

then error(27);

{\* 若是以下情况，那就是第27错误然后清零l和h \*}

if( abs(l) > xmax ) or ( abs(h) > xmax )

then begin

error(27);

l := 0;

h := 0;

end;

{\* 如果a是amax，就是第4 fatal \*}

if a = amax

then fatal(4)

else begin

a := a + 1;

with atab[a] do

begin

inxtyp := tp;

low := l;

high := h

end

end

end { enterarray };

{\* 过程： enterblock \*}

procedure enterblock;

begin

{\* 如果b等于块表的大小（已经到表结尾），那就是第2 fatal \*}

if b = bmax

then fatal(2)

{\* 不然给块表中的每个b元素赋值0 \*}

else begin

b := b + 1;

btab[b].last := 0;

btab[b].lastpar := 0;

end

end { enterblock };

{\* 过程： enterreal（参数x） \*}

procedure enterreal( x: real );

begin

{\* 如果c2已经到实常数结尾，那就是第3 fatal \*}

if c2 = c2max - 1

then fatal(3)

else begin

{\* 把x放在实常数表里，然后c1为1 \*}

rconst[c2+1] := x;

c1 := 1;

{\* 如果表里第c1元素不是x，c1要挪一个，如果c1大于c2，那c2就是那个值 \*}

while rconst[c1] <> x do

c1 := c1 + 1;

if c1 > c2

then c2 := c1

end

end { enterreal };

{\* 过程： emit（参数fct） \*}

procedure emit( fct: integer );

begin

if lc = cmax

then fatal(6);

code[lc].f := fct;

lc := lc + 1

end { emit };

{\* 过程： emit1（参数fct，b） \*}

procedure emit1( fct, b: integer );

begin

{\* 如果代码计数变量跟代码大小一样，那就是第6 fatal \*}

if lc = cmax

then fatal(6);

with code[lc] do

begin

f := fct;

y := b;

end;

{\* 代码计数变量加1 \*}

lc := lc + 1

end { emit1 };

{\* 过程： emit2（参数fct，a，b） \*}

procedure emit2( fct, a, b: integer );

begin

{\* 如果代码计数变量跟代码大小一样，那就是第6 fatal \*}

if lc = cmax then fatal(6);

with code[lc] do

begin

f := fct;

x := a;

y := b

end;

{\* 代码计数变量加1 \*}

lc := lc + 1;

end { emit2 };

{\* 过程： printtables （输出表内容）\*}

procedure printtables;

{\* 局部变量定义 \*}

var i: integer;

o: order;

mne: array[0..omax] of

packed array[1..5] of char;

begin

{\* 局部变量定义 \*}

mne[0] := 'LDA '; mne[1] := 'LOD '; mne[2] := 'LDI ';

mne[3] := 'DIS '; mne[8] := 'FCT '; mne[9] := 'INT ';

mne[10] := 'JMP '; mne[11] := 'JPC '; mne[12] := 'SWT ';

mne[13] := 'CAS '; mne[14] := 'F1U '; mne[15] := 'F2U ';

mne[16] := 'F1D '; mne[17] := 'F2D '; mne[18] := 'MKS ';

mne[19] := 'CAL '; mne[20] := 'IDX '; mne[21] := 'IXX ';

mne[22] := 'LDB '; mne[23] := 'CPB '; mne[24] := 'LDC ';

mne[25] := 'LDR '; mne[26] := 'FLT '; mne[27] := 'RED ';

mne[28] := 'WRS '; mne[29] := 'WRW '; mne[30] := 'WRU ';

mne[31] := 'HLT '; mne[32] := 'EXP '; mne[33] := 'EXF ';

mne[34] := 'LDT '; mne[35] := 'NOT '; mne[36] := 'MUS ';

mne[37] := 'WRR '; mne[38] := 'STO '; mne[39] := 'EQR ';

mne[40] := 'NER '; mne[41] := 'LSR '; mne[42] := 'LER ';

mne[43] := 'GTR '; mne[44] := 'GER '; mne[45] := 'EQL ';

mne[46] := 'NEQ '; mne[47] := 'LSS '; mne[48] := 'LEQ ';

mne[49] := 'GRT '; mne[50] := 'GEQ '; mne[51] := 'ORR ';

mne[52] := 'ADD '; mne[53] := 'SUB '; mne[54] := 'ADR ';

mne[55] := 'SUR '; mne[56] := 'AND '; mne[57] := 'MUL ';

mne[58] := 'DIV '; mne[59] := 'MOD '; mne[60] := 'MUR ';

mne[61] := 'DIR '; mne[62] := 'RDL '; mne[63] := 'WRL ';

{\* 在psout文件输出新行 \*}

writeln(psout);

writeln(psout);

writeln(psout);

{\* 输出表头 \*}

writeln(psout,' identifiers link obj typ ref nrm lev adr');

writeln(psout);

{\* 输出表内容 \*}

for i := btab[1].last to t do

with tab[i] do

writeln( psout, i,' ', name, link:5, ord(obj):5, ord(typ):5,ref:5, ord(normal):5,lev:5,adr:5);

{\* 在psout文件输出新行 \*}

writeln( psout );

writeln( psout );

writeln( psout );

{\* 输出表头 \*}

writeln( psout, 'blocks last lpar psze vsze' );

writeln( psout );

{\* 输出表内容 \*}

for i := 1 to b do

with btab[i] do

writeln( psout, i:4, last:9, lastpar:5, psize:5, vsize:5 );

{\* 在psout文件输出新行 \*}

writeln( psout );

writeln( psout );

writeln( psout );

{\* 输出表头 \*}

writeln( psout, 'arrays xtyp etyp eref low high elsz size');

writeln( psout );

{\* 输出表内容 \*}

for i := 1 to a do

with atab[i] do

writeln( psout, i:4, ord(inxtyp):9, ord(eltyp):5, elref:5, low:5, high:5, elsize:5, size:5);

{\* 在psout文件输出新行 \*}

writeln( psout );

writeln( psout );

writeln( psout );

writeln( psout, 'code:');

writeln( psout );

{\* 输出表内容 \*}

for i := 0 to lc-1 do

begin

write( psout, i:5 );

o := code[i];

write( psout, mne[o.f]:8, o.f:5 );

if o.f < 31

then if o.f < 4

then write( psout, o.x:5, o.y:5 )

else write( psout, o.y:10 )

else write( psout, ' ' );

writeln( psout, ',' )

end;

writeln( psout );

writeln( psout, 'Starting address is ', tab[btab[1].last].adr:5 )

end { printtables };

{\* 过程： block（参数fsys，isfun，level） \*}

procedure block( fsys: symset; isfun: boolean; level: integer );

{\* 变量定义 \*}

type conrec = record

case tp: types of

ints, chars, bools : ( i:integer );

reals :( r:real )

end;

var dx : integer ; { data allocation index }

prt: integer ; { t-index of this procedure }

prb: integer ; { b-index of this procedure }

x : integer ;

{\* 过程：skip（参数fsys，n）（跳过代码段）\*}

procedure skip( fsys:symset; n:integer);

begin

{\* 第n 错误，把skipflag改成true，代表有被跳过的 \*}

error(n);

skipflag := true;

{\* 只要sy不在fsys里，那就要继续读符号 \*}

while not ( sy in fsys ) do

insymbol;

if skipflag then endskip

end { skip };

{\* 过程：test（参数s1，s2，n）（试试看sy是否在s1里，不然就是不合法，跳过） \*}

procedure test( s1,s2: symset; n:integer );

begin

{\* 如果sy不在s1里，那就要skip \*}

if not( sy in s1 )

then skip( s1 + s2, n )

end { test };

{\* 过程：testsemicolon（检查';'是否合法）\*}

procedure testsemicolon;

begin

{\* 如果sy为semicolon，那就要读取符号，不然就是第14 error，

如果sy在[comma, colon]那么读取符号 \*}

if sy = semicolon

then insymbol

else begin

error(14);

if sy in [comma, colon]

then insymbol

end;

test( [ident] + blockbegsys, fsys, 6 )

end { testsemicolon };

{\* 过程：enter（参数id，k） （把数据放在标识符标签表里）\*}

procedure enter( id: alfa; k:objecttyp );

var j,l : integer;

begin

{\* 如果t已到表结尾，那就是第1 fatal \*}

if t = tmax

then fatal(1)

else begin

tab[0].name := id;

j := btab[display[level]].last;

l := j;

while tab[j].name <> id do

j := tab[j].link;

if j <> 0

then error(1)

else begin

t := t + 1;

with tab[t] do

begin

{\* 表参数的赋值 \*}

name := id;

link := l;

obj := k;

typ := notyp;

ref := 0;

lev := level;

adr := 0;

normal := false { initial value }

end;

{\* 当前btab的last变为t \*}

btab[display[level]].last := t

end

end

end { enter };

{\* 函数：loc（参数id）（表里寻找id） \*}

function loc( id: alfa ):integer;

var i,j : integer; { locate if in table }

begin

i := level;

tab[0].name := id; { sentinel }

{\* 重复以下程序直到i大于等于0或j不是0 \*}

repeat

j := btab[display[i]].last;

while tab[j].name <> id do

j := tab[j].link;

i := i - 1;

until ( i < 0 ) or ( j <> 0 );

{\* 如果j为0，代表id不存在，那就是第0错误 \*}

if j = 0

then error(0);

loc := j

end { loc } ;

{\* 过程：entervariable（变量录入表里的过程）\*}

procedure entervariable;

begin

{\* 如果sy是ident，那么录入变量到表里 \*}

if sy = ident

then begin

enter( id, vvariable );

insymbol

end

else error(2)

end { entervariable };

{\* 过程：constant（参数fsys，c）（检查常量，类型和值由c返回） \*}

procedure constant( fsys: symset; var c: conrec );

var x, sign : integer;

begin

{\* 如果c的类型是notyp，那么c中的i为0，然后检查constbegsys是否合法 \*}

c.tp := notyp;

c.i := 0;

test( constbegsys, fsys, 50 );

if sy in constbegsys

then begin

{\* 若sy是字符常量，那c类型是字符，值为inum然后继续读下一个字符 \*}

if sy = charcon

then begin

c.tp := chars;

c.i := inum;

insymbol

end

else begin

sign := 1;

if sy in [plus, minus]

then begin

{\* 如果sy是minus，那sign是-1 \*}

if sy = minus

then sign := -1;

insymbol

end;

{\* 如果sy是标识符，那么在表里找id的位置放在x \*}

if sy = ident

then begin

x := loc(id);

{\* 如果x不是0，就是找到了 \*}

if x <> 0

then

{\* 如果tab表里对应的符号不是konstant，那就是第25错误 \*}

if tab[x].obj <> konstant

then error(25)

{\* 不然常量类为tab表里对应符号的类型 \*}

else begin

c.tp := tab[x].typ;

{\* 如果c类型为实数 \*}

if c.tp = reals

then c.r := sign\*rconst[tab[x].adr]

else c.i := sign\*tab[x].adr

end;

insymbol

end

{\* 如果sy是整数 \*}

else if sy = intcon

then begin

{\* c类型为整数 \*}

c.tp := ints;

c.i := sign\*inum;

insymbol

end

{\* 如果sy是实数 \*}

else if sy = realcon

then begin

{\* c类型为实数 \*}

c.tp := reals;

c.r := sign\*rnum;

insymbol

end

{\* 不然，跳过（没用的符号） \*}

else skip(fsys,50)

end;

test(fsys,[],6)

end

end { constant };

{\* 过程：typ（参数fsys，tp，rf，sz）（处理类型说明） \*}

procedure typ( fsys: symset; var tp: types; var rf,sz:integer );

{\* 变量定义 \*}

var eltp : types;

elrf, x : integer;

elsz, offset, t0, t1 : integer;

{\* 过程：arraytyp（参数aref，arsz）（处理数组类型） \*}

procedure arraytyp( var aref, arsz: integer );

{\* 变量定义 \*}

var eltp : types;

low, high : conrec;

elrf, elsz: integer;

begin

{\* 检查常量 \*}

constant( [colon, rbrack, rparent, ofsy] + fsys, low );

{\* 若下界的类型为实数那就是第27错误，重启下界的类型和值 \*}

if low.tp = reals

then begin

error(27);

low.tp := ints;

low.i := 0

end;

{\* 如果sy是colon，读下一个符号，不然是第13错误 \*}

if sy = colon

then insymbol

else error(13);

{\* 检查常量 \*}

constant( [rbrack, comma, rparent, ofsy ] + fsys, high );

{\* 如果上界和下界的类型不一样，那就是第27错误，把下界的值赋给上界 \*}

if high.tp <> low.tp

then begin

error(27);

high.i := low.i

end;

{\* 把数据放在atab里 \*}

enterarray( low.tp, low.i, high.i );

aref := a;

{\* 如果sy是逗号，读下一个符号 \*}

if sy = comma

then begin

insymbol;

eltp := arrays;

arraytyp( elrf, elsz )

end

else begin

{\* 如果sy是右中括号，读下一个符号，不然就是第12错误，若sy是右括号，读下一个符号 \*}

if sy = rbrack

then insymbol

else begin

error(12);

if sy = rparent

then insymbol

end;

{\* 若sy是of关键字，读下一个符号，不然就是第8错误 \*}

if sy = ofsy

then insymbol

else error(8);

typ( fsys, eltp, elrf, elsz )

end;

with atab[aref] do

begin

arsz := (high-low+1) \* elsz;

size := arsz;

eltyp := eltp;

elref := elrf;

elsize := elsz

end

end { arraytyp };

begin { typ }

{\* 类型tp为notyp，清零rf和sz，然后检查typebegsys \*}

tp := notyp;

rf := 0;

sz := 0;

test( typebegsys, fsys, 10 );

if sy in typebegsys

then begin

{\* 若sy是标识符，在表里找id的位置放在x里 \*}

if sy = ident

then begin

x := loc(id);

{\* 若x不是0那就是找到 \*}

if x <> 0

then with tab[x] do

{\* 如果表里第x对应的元素不是typel就是第29错误 \*}

if obj <> typel

then error(29)

else begin

{\* 读取表里的数据 \*}

tp := typ;

rf := ref;

sz := adr;

{\* 若对应数据的类型是无类型，那就是第30错误 \*}

if tp = notyp

then error(30)

end;

insymbol

end

{\* 若sy是数组元素，那么读下一个符号 \*}

else if sy = arraysy

then begin

insymbol;

{\* 若sy是左中括号，读下一个符号，不然就是第11错误然后

若sy是左括号，读下一个符号 \*}

if sy = lbrack

then insymbol

else begin

error(11);

if sy = lparent

then insymbol

end;

tp := arrays;

arraytyp(rf,sz)

end

else begin { records }

{\* 读下一个符号然后执行enterblock \*}

insymbol;

enterblock;

{\* 类信号为records，rf只想b \*}

tp := records;

rf := b;

{\* 若嵌套层次已达到最高层次那就是第5 fatal \*}

if level = lmax

then fatal(5);

level := level + 1;

display[level] := b;

offset := 0;

while not ( sy in fsys - [semicolon,comma,ident]+ [endsy] ) do

begin { field section }

{\* 如果sy是标识符 \*}

if sy = ident

then begin

t0 := t;

{\* 把变量放入表里 \*}

entervariable;

{\* while sy是逗号，读下一个符号，放入表里 \*}

while sy = comma do

begin

insymbol;

entervariable

end;

{\* 若sy是colon，读下一个符号，不然就是第5 错误 \*}

if sy = colon

then insymbol

else error(5);

t1 := t;

typ( fsys + [semicolon, endsy, comma,ident], eltp, elrf, elsz );

while t0 < t1 do

begin

t0 := t0 + 1;

with tab[t0] do

begin

typ := eltp;

ref := elrf;

normal := true;

adr := offset;

offset := offset + elsz

end

end

end; { sy = ident }

{\* 若sy不是endsy \*}

if sy <> endsy

then begin

{\* 若sy是';'，读下一个符号 \*}

if sy = semicolon

then insymbol

{\* 不然就是第14错误，然后若sy是逗号那读下一个符号 \*}

else begin

error(14);

if sy = comma

then insymbol

end;

test( [ident,endsy, semicolon],fsys,6 )

end

end; { field section }

btab[rf].vsize := offset;

sz := offset;

btab[rf].psize := 0;

{\* 读下一个符号 \*}

insymbol;

level := level - 1

end; { record }

test( fsys, [],6 )

end;

end { typ };

{\* 过程：parameterlist （处理程序中的参数放到表里）\*}

procedure parameterlist; { formal parameter list }

{\* 变量定义 \*}

var tp : types;

valpar : boolean;

rf, sz, x, t0 : integer;

begin

{\* 读下一个符号 \*}

insymbol;

{\* 初始化tp，rf，sz \*}

tp := notyp;

rf := 0;

sz := 0;

test( [ident, varsy], fsys+[rparent], 7 );

while sy in [ident, varsy] do

begin

{\* 若sy是var关键字那valpar为true \*}

if sy <> varsy

then valpar := true

{\* 不然读下一个符号，valpar为false \*}

else begin

insymbol;

valpar := false

end;

t0 := t;

{\* 把变量放入表中 \*}

entervariable;

{\* while sy是逗号，读下一个符号然后放入表里 \*}

while sy = comma do

begin

insymbol;

entervariable;

end;

{\* 若sy是colon，读下一个符号 \*}

if sy = colon

then begin

insymbol;

{\* 若sy是标识符，就是第2错误，不然 \*}

if sy <> ident

then error(2)

else begin

{\* 在表里寻找id的位置放在x中，然后读下一个符号 \*}

x := loc(id);

insymbol;

{\* 若x不为0，那就是找到了 \*}

if x <> 0

{\* 那就对表里第x对应的数据做以下操作 \*}

then with tab[x] do

if obj <> typel

then error(29)

else begin

tp := typ;

rf := ref;

if valpar

then sz := adr

else sz := 1

end;

end;

test( [semicolon, rparent], [comma,ident]+fsys, 14 )

end

{\* 不然就是第5错误 \*}

else error(5);

while t0 < t do

begin

t0 := t0 + 1;

{\* 对表里第t0对应的数据做以下操作 \*}

with tab[t0] do

begin

typ := tp;

ref := rf;

adr := dx;

lev := level;

normal := valpar;

dx := dx + sz

end

end;

{\* 若sy不是右括号 \*}

if sy <> rparent

then begin

{\* 若sy是';'，读下一个符号，不然就是第14错误 \*}

if sy = semicolon

then insymbol

else begin

error(14);

{\* 若sy是逗号，读下一个符号 \*}

if sy = comma

then insymbol

end;

test( [ident, varsy],[rparent]+fsys,6)

end

end { while };

{\* 若sy是右括号 \*}

if sy = rparent

then begin

{\* 读下一个符号 \*}

insymbol;

test( [semicolon, colon],fsys,6 )

end

{\* 不然就是第4错误 \*}

else error(4)

end { parameterlist };

{\* 过程：constdec（处理常量声明）\*}

procedure constdec;

var c : conrec;

begin

{\* 读下一个符号 \*}

insymbol;

test([ident], blockbegsys, 2 );

{\* while sy是标识符 \*}

while sy = ident do

begin

{\* 把数据放入表中然后读下一个符号 \*}

enter(id, konstant);

insymbol;

{\* 若sy是等号，读下一个符号 \*}

if sy = eql

then insymbol

{\* 不然就是第16错误，然后若sy是becomes，读下一个符号 \*}

else begin

error(16);

if sy = becomes

then insymbol

end;

{\* 检查常量 \*}

constant([semicolon,comma,ident]+fsys,c);

{\* 把数据放入表中对应的元素数据 \*}

tab[t].typ := c.tp;

tab[t].ref := 0;

{\* 若常量类型是实数 \*}

if c.tp = reals

then begin

enterreal(c.r);

tab[t].adr := c1;

end

else tab[t].adr := c.i;

testsemicolon

end

end { constdec };

{\* 过程：typedeclaration（处理类声明）\*}

procedure typedeclaration;

var tp: types;

rf, sz, t1 : integer;

begin

{\* 读下一个符号 \*}

insymbol;

test([ident], blockbegsys,2 );

{\* while sy是标识符 \*}

while sy = ident do

begin

{\* 把数据放入表中然后读下一个符号 \*}

enter(id, typel);

t1 := t;

{\* 读下一个符号 \*}

insymbol;

{\* 若sy是等号，读下一个符号 \*}

if sy = eql

then insymbol

{\* 不然就是第16错误，然后若sy是becomes，读下一个符号 \*}

else begin

error(16);

if sy = becomes

then insymbol

end;

typ( [semicolon,comma,ident]+fsys, tp,rf,sz );

{\* 对表中第t1对应的数据进行以下操作 \*}

with tab[t1] do

begin

typ := tp;

ref := rf;

adr := sz

end;

testsemicolon

end

end { typedeclaration };

{\* 过程：variabledeclaration（处理变量声明） \*}

procedure variabledeclaration;

var tp : types;

t0, t1, rf, sz : integer;

begin

{\* 读下一个符号 \*}

insymbol;

{\* while sy是标识符 \*}

while sy = ident do

begin

t0 := t;

{\* 把变量放入表中 \*}

entervariable;

{\* while sy是逗号，读下一个符号然后把变量放入表中 \*}

while sy = comma do

begin

insymbol;

entervariable;

end;

{\* 若sy是colon，读下一个符号，不然就是第5错误 \*}

if sy = colon

then insymbol

else error(5);

t1 := t;

typ([semicolon,comma,ident]+fsys, tp,rf,sz );

while t0 < t1 do

begin

t0 := t0 + 1;

{\* 对表中第t0对应的元素数据进行以下操作 \*}

with tab[t0] do

begin

typ := tp;

ref := rf;

lev := level;

adr := dx;

normal := true;

dx := dx + sz

end

end;

testsemicolon

end

end { variabledeclaration };

{\* 过程：procdeclaration（处理过程声明） \*}

procedure procdeclaration;

var isfun : boolean;

begin

{\* 若sy是funcsy（函数），那isfun就有效，然后读下一个符号 \*}

isfun := sy = funcsy;

insymbol;

{\* 若sy不是标识符，那么就是第2错误 \*}

if sy <> ident

then begin

error(2);

id :=' '

end;

{\* 若是函数，把id,funktion数据放入表里，不然把id,prozedure放入表里 \*}

if isfun

then enter(id,funktion)

else enter(id,prozedure);

tab[t].normal := true;

{\* 读下一个符号 \*}

insymbol;

block([semicolon]+fsys, isfun, level+1 );

{\* 若sy是';'，读下一个符号，不然就是第14错误 \*}

if sy = semicolon

then insymbol

else error(14);

emit(32+ord(isfun)) {exit}

end { proceduredeclaration };

{\* 过程：statement（参数fsys） \*}

procedure statement( fsys:symset );

var i : integer;

{\* 过程：expression（参数fsys，x）（处理表达式） \*}

procedure expression(fsys:symset; var x:item); forward;

{\* 过程：selector（参数fsys，v）（处理结构变量） \*}

procedure selector(fsys:symset; var v:item);

var x : item;

a,j : integer;

begin { sy in [lparent, lbrack, period] }

{\* 重复以下程序直到sy不是左括号，左括号开头，句号 \*}

repeat

{\* 若sy是句号，读下一个符号 \*}

if sy = period

then begin

insymbol; { field selector }

{\* 若sy是标识符，那就是第2错误 \*}

if sy <> ident

then error(2)

else begin

{\* 若v的类型不是records，那就是第31错误 \*}

if v.typ <> records

then error(31)

{\* 不然搜索字段标识符 \*}

else begin { search field identifier }

j := btab[v.ref].last;

tab[0].name := id;

{\* while表中第j对应的元素名字不是id，做以下操作 \*}

while tab[j].name <> id do

j := tab[j].link;

{\* 若j是0（不存在），那就是第0错误 \*}

if j = 0

then error(0);

v.typ := tab[j].typ;

v.ref := tab[j].ref;

a := tab[j].adr;

if a <> 0

{\* 调用第9号指令 \*}

then emit1(9,a)

end;

{\* 读下一个符号 \*}

insymbol

end

end

else begin { array selector }

{\* 若sy不是左括号开头，那就是第11错误 \*}

if sy <> lbrack

then error(11);

{\* 重复以下程序直到sy不是逗号 \*}

repeat

{\* 读下一个符号 \*}

insymbol;

expression( fsys+[comma,rbrack],x);

{\* 若类型不是数组，那就是第28错误 \*}

if v.typ <> arrays

then error(28)

else begin

a := v.ref;

if atab[a].inxtyp <> x.typ

then error(26)

else if atab[a].elsize = 1

{\* 调用第20号指令 \*}

then emit1(20,a)

{\* 不然调用第21号指令 \*}

else emit1(21,a);

v.typ := atab[a].eltyp;

v.ref := atab[a].elref

end

until sy <> comma;

{\* 若sy是右中括号，读下一个符号 \*}

if sy = rbrack

then insymbol

{\* 不然就是第12错误，然后若sy右括号，读下一个符号 \*}

else begin

error(12);

if sy = rparent

then insymbol

end

end

until not( sy in[lbrack, lparent, period]);

test( fsys,[],6)

end { selector };

{\* 过程：call（参数fsys，i） \*}

procedure call( fsys: symset; i:integer );

var x : item;

lastp,cp,k : integer;

begin

{\* 调用第18号指令 \*}

emit1(18,i); { mark stack }

lastp := btab[tab[i].ref].lastpar;

cp := i;

{\* 若sy是左括号 \*}

if sy = lparent

then begin { actual parameter list }

{\* 重复以下程序直到sy不是逗号 \*}

repeat

{\* 读下一个符号 \*}

insymbol;

if cp >= lastp

then error(39)

else begin

cp := cp + 1;

if tab[cp].normal

then begin { value parameter }

expression( fsys+[comma, colon,rparent],x);

if x.typ = tab[cp].typ

then begin

{\* 若变量跟表里对应的元素ref不一样，那就是第36错误 \*}

if x.ref <> tab[cp].ref

then error(36)

{\* 若x的类型是数组，调用第22号指令 \*}

else if x.typ = arrays

then emit1(22,atab[x.ref].size)

{\* 若x的类型是records，调用第22号指令 \*}

else if x.typ = records

then emit1(22,btab[x.ref].vsize)

end

else if ( x.typ = ints ) and ( tab[cp].typ = reals )

then emit1(26,0)

{\* 若x有类型定义，那就是第36错误 \*}

else if x.typ <> notyp

then error(36);

end

else begin { variable parameter }

{\* 若sy是标识符，那就是第2错误 \*}

if sy <> ident

then error(2)

else begin

{\* 不然在表里寻找id的位置放在k中，然后读下一个符号 \*}

k := loc(id);

insymbol;

{\* 若k不是0，那就是找到了 \*}

if k <> 0

then begin

{\* 若表中第k对应的元素不是vvariable，那就是第37错误 \*}

if tab[k].obj <> vvariable

then error(37);

{\* 获取表中第k元素的数据 \*}

x.typ := tab[k].typ;

x.ref := tab[k].ref;

if tab[k].normal

then emit2(0,tab[k].lev,tab[k].adr)

else emit2(1,tab[k].lev,tab[k].adr);

{\* 若sy是左括号开头，左括号，或句号 \*}

if sy in [lbrack, lparent, period]

then selector(fsys+[comma,colon,rparent],x);

if ( x.typ <> tab[cp].typ ) or ( x.ref <> tab[cp].ref )

then error(36)

end

end

end {variable parameter }

end;

test( [comma, rparent],fsys,6)

until sy <> comma;

{\* 若sy是右括号，读下一个符号，不然是第4错误 \*}

if sy = rparent

then insymbol

else error(4)

end;

if cp < lastp

then error(39); { too few actual parameters }

emit1(19,btab[tab[i].ref].psize-1 );

if tab[i].lev < level

then emit2(3,tab[i].lev, level )

end { call };

{\* 函数：resulttype（参数a，b） \*}

function resulttype( a, b : types) :types;

begin

{\* 若a或b大于实数上限，那就是第33错误，返回无类型 \*}

if ( a > reals ) or ( b > reals )

then begin

error(33);

resulttype := notyp

end

{\* 而如果a或b无类型，返回无类型 \*}

else if ( a = notyp ) or ( b = notyp )

then resulttype := notyp

{\* 而如果a是整数，若b也是整数，返回整数 \*}

else if a = ints

then if b = ints

then resulttype := ints

else begin

{\* 不然返回实数，然后调用第26指令 \*}

resulttype := reals;

emit1(26,1)

end

else begin

{\* 不然返回实数 \*}

resulttype := reals;

{\* 若b是整数，调用第26指令 \*}

if b = ints

then emit1(26,0)

end

end { resulttype } ;

{\* 过程：expression（参数fsys，x） \*}

procedure expression( fsys: symset; var x: item );

var y : item;

op : symbol;

{\* 过程：simpleexpression（参数fsys，x） \*}

procedure simpleexpression( fsys: symset; var x: item );

var y : item;

op : symbol;

{\* 过程：term（参数fsys，x） \*}

procedure term( fsys: symset; var x: item );

var y : item;

op : symbol;

{\* 过程：factor（参数fsys，x） \*}

procedure factor( fsys: symset; var x: item );

var i,f : integer;

{\* 过程：standfct（参数n） \*}

procedure standfct( n: integer );

var ts : typset;

begin { standard function no. n }

{\* 若sy是左括号，读下一个符号，不然就是第9错误 \*}

if sy = lparent

then insymbol

else error(9);

{\* 若标准函数的编号小于17 \*}

if n < 17

then begin

expression( fsys+[rparent], x );

case n of

{\* 若n是0，2，就是求绝对值和平方 \*}

{ abs, sqr } 0,2: begin

ts := [ints, reals];

tab[i].typ := x.typ;

if x.typ = reals

then n := n + 1

end;

{\* 若n是4，5，就是求奇书判断和ascii码转化成字符 \*}

{ odd, chr } 4,5: ts := [ints];

{\* 若n是6，允许的类型是整数，布尔或字符 \*}

{ odr } 6: ts := [ints,bools,chars];

{\* 若n是7，8，允许的类型是整数，布尔或字符，然后把类型记录到表里 \*}

{ succ,pred } 7,8 : begin

ts := [ints, bools,chars];

tab[i].typ := x.typ

end;

{\* 若n是以下这几个操作好，那就是数学运算 \*}

{ round,trunc } 9,10,11,12,13,14,15,16:

{ sin,cos,... } begin

ts := [ints,reals];

if x.typ = ints

{\* 调用第26号指令 \*}

then emit1(26,0)

end;

end; { case }

if x.typ in ts

then emit1(8,n)

{\* 若x有类型定义，那就是第48错误 \*}

else if x.typ <> notyp

then error(48);

end

else begin { n in [17,18] }

{\* 若sy不是标识符，那就是第2错误 \*}

if sy <> ident

then error(2)

{\* 若id不是'input '，那就是第0错误 \*}

else if id <> 'input '

then error(0)

{\* 不然读下一个符号 \*}

else insymbol;

{\* 调用第8指令 \*}

emit1(8,n);

end;

{\* 获取表中对应元素的类型 \*}

x.typ := tab[i].typ;

{\* 若sy是右括号，读下一个符号，不然是第4错误 \*}

if sy = rparent

then insymbol

else error(4)

end { standfct } ;

begin { factor }

x.typ := notyp;

x.ref := 0;

test( facbegsys, fsys,58 );

while sy in facbegsys do

begin

{\* 若sy是标识符 \*}

if sy = ident

then begin

{\* 在表中寻找id的位置放在i然后读下一个符号 \*}

i := loc(id);

insymbol;

{\* 对表中第i对应的数据进行以下操作 \*}

with tab[i] do

case obj of

{\* 若obj是konstant \*}

konstant: begin

x.typ := typ;

x.ref := 0;

if x.typ = reals

then emit1(25,adr)

else emit1(24,adr)

end;

{\* 若obj是vvariable \*}

vvariable:begin

x.typ := typ;

x.ref := ref;

{\* 若sy是左括号开头，左括号，或句号 \*}

if sy in [lbrack, lparent,period]

then begin

if normal

then f := 0

else f := 1;

emit2(f,lev,adr);

selector(fsys,x);

if x.typ in stantyps

then emit(34)

end

else begin

if x.typ in stantyps

then if normal

then f := 1

else f := 2

else if normal

then f := 0

else f := 1;

emit2(f,lev,adr)

end

end;

{\* 若obj是typel，prozedure那就是第44错误 \*}

typel,prozedure: error(44);

{\* 若obj是funktion \*}

funktion: begin

x.typ := typ;

if lev <> 0

then call(fsys,i)

else standfct(adr)

end

end { case,with }

end

{\* 而若sy是字符或整数或实数 \*}

else if sy in [ charcon,intcon,realcon ]

then begin

{\* 若sy是realcon，x的类型是实数，把实数放入表中 \*}

if sy = realcon

then begin

x.typ := reals;

enterreal(rnum);

emit1(25,c1)

end

{\* 而若sy是charcon，x类型是字符，不然x类型是整数 \*}

else begin

if sy = charcon

then x.typ := chars

else x.typ := ints;

emit1(24,inum)

end;

x.ref := 0;

{\* 读下一个符号 \*}

insymbol

end

{\* 而若sy是左括号，读下一个符号 \*}

else if sy = lparent

then begin

insymbol;

expression(fsys + [rparent],x);

{\* 若sy是右括号，读下一个符号，不然就是第4错误 \*}

if sy = rparent

then insymbol

else error(4)

end

{\* 而若sy是notsy，读下一个符号 \*}

else if sy = notsy

then begin

insymbol;

factor(fsys,x);

{\* x的类型定义为布尔 \*}

if x.typ = bools

then emit(35)

{\* 而若x有类型定义，那就是第32错误 \*}

else if x.typ <> notyp

then error(32)

end;

test(fsys,facbegsys,6)

end { while }

end { factor };

begin { term }

factor( fsys + [times,rdiv,idiv,imod,andsy],x);

{\* while sy 属于下面几个其中一个 \*}

while sy in [times,rdiv,idiv,imod,andsy] do

begin

{\* 运算符是sy，然后读下一个符号 \*}

op := sy;

insymbol;

factor(fsys+[times,rdiv,idiv,imod,andsy],y );

{\* 若运算符是times \*}

if op = times

then begin

x.typ := resulttype(x.typ, y.typ);

case x.typ of

{\* 若x无类型，什么也不做 \*}

notyp: ;

{\* 若x是整数，执行第57指令 \*}

ints : emit(57);

{\* 若x是实数，执行第60指令 \*}

reals: emit(60);

end

end

{\* 若运算符是rdiv \*}

else if op = rdiv

then begin

{\* 若x是整数，执行第26指令然后类型改成实数 \*}

if x.typ = ints

then begin

emit1(26,1);

x.typ := reals;

end;

{\* 若y是整数，执行第26指令然后类型改成实数 \*}

if y.typ = ints

then begin

emit1(26,0);

y.typ := reals;

end;

{\* 若x和y是整数，执行第61指令 \*}

if (x.typ = reals) and (y.typ = reals)

then emit(61)

{\* 不然若x和y有类型定义，那就是第33错误，x定义为无类型 \*}

else begin

if( x.typ <> notyp ) and (y.typ <> notyp)

then error(33);

x.typ := notyp

end

end

{\* 若运算符是andsy \*}

else if op = andsy

then begin

{\* 若x和y是布尔，执行第56指令 \*}

if( x.typ = bools )and(y.typ = bools)

then emit(56)

{\* 而若x和y有类型定义，那就是第32错误，x定义为无类型 \*}

else begin

if( x.typ <> notyp ) and (y.typ <> notyp)

then error(32);

x.typ := notyp

end

end

else begin { op in [idiv,imod] }

{\* 若x和y是整数 \*}

if (x.typ = ints) and (y.typ = ints)

{\* 若运算符是idiv，执行第58指令，不然执行第59指令 \*}

then if op = idiv

then emit(58)

else emit(59)

else begin

{\* 若x和y有类型定义，那就是第34错误，x定义为无类型 \*}

if ( x.typ <> notyp ) and (y.typ <> notyp)

then error(34);

x.typ := notyp

end

end

end { while }

end { term };

begin { simpleexpression }

{\* 若sy是plus或minus \*}

if sy in [plus,minus]

then begin

{\* 运算符是sy，然后读下一个符号 \*}

op := sy;

insymbol;

term( fsys+[plus,minus],x);

{\* 若类型是bools, chars, arrays, records，那就是第33错误 \*}

if x.typ > reals

then error(33)

{\* 而若运算符是minus，执行第36指令 \*}

else if op = minus

then emit(36)

end

else term(fsys+[plus,minus,orsy],x);

{\* while sy是plus，minus，或orsy \*}

while sy in [plus,minus,orsy] do

begin

{\* 运算符是sy，然后读下一个符号 \*}

op := sy;

insymbol;

term(fsys+[plus,minus,orsy],y);

{\* 若运算符是orsy \*}

if op = orsy

then begin

{\* 若x和y是布尔，执行第51指令 \*}

if ( x.typ = bools )and(y.typ = bools)

then emit(51)

else begin

{\* 而若x和y有类型定义，那就是第32错误，x定义为无类型 \*}

if( x.typ <> notyp) and (y.typ <> notyp)

then error(32);

x.typ := notyp

end

end

else begin

x.typ := resulttype(x.typ,y.typ);

case x.typ of

{\* 若x无类型，什么都不做 \*}

notyp: ;

{\* 若x是整数，若运算符是plus，执行第52指令，不然执行第53指令 \*}

ints: if op = plus

then emit(52)

else emit(53);

{\* 若x是实数，若运算符是plus，执行第54指令，不然执行第55指令 \*}

reals:if op = plus

then emit(54)

else emit(55)

end { case }

end

end { while }

end { simpleexpression };

begin { expression }

simpleexpression(fsys+[eql,neq,lss,leq,gtr,geq],x);

{\* 若sy属于下面几个其中一个 \*}

if sy in [ eql,neq,lss,leq,gtr,geq]

then begin

{\* 运算符是sy，然后读下一个符号 \*}

op := sy;

insymbol;

simpleexpression(fsys,y);

if(x.typ in [notyp,ints,bools,chars]) and (x.typ = y.typ)

{\* 对于sy不一样的属性，执行不一样的指令 \*}

then case op of

eql: emit(45);

neq: emit(46);

lss: emit(47);

leq: emit(48);

gtr: emit(49);

geq: emit(50);

end

else begin

{\* 若x类型是整数，x类型改成实数，执行第26指令 \*}

if x.typ = ints

then begin

x.typ := reals;

emit1(26,1)

end

{\* 若y类型是整数，y类型改成实数，执行第26指令 \*}

else if y.typ = ints

then begin

y.typ := reals;

emit1(26,0)

end;

{\* 若x和y是实数，对于sy不一样的属性，执行不一样的指令 \*}

if ( x.typ = reals)and(y.typ=reals)

then case op of

eql: emit(39);

neq: emit(40);

lss: emit(41);

leq: emit(42);

gtr: emit(43);

geq: emit(44);

end

{\* 不然就是第35错误 \*}

else error(35)

end;

{\* x类型定义为布尔 \*}

x.typ := bools

end

end { expression };

{\* 过程：assignment（参数lv，ad） \*}

procedure assignment( lv, ad: integer );

var x,y: item;

f : integer;

begin { tab[i].obj in [variable,prozedure] }

x.typ := tab[i].typ;

x.ref := tab[i].ref;

if tab[i].normal

then f := 0

else f := 1;

emit2(f,lv,ad);

{\* 若sy是左括号开头，左括号，或句号 \*}

if sy in [lbrack,lparent,period]

then selector([becomes,eql]+fsys,x);

{\* 若sy是becomes，读下一个符号 \*}

if sy = becomes

then insymbol

{\* 不然就是第51错误，若sy是eql，读下一个符号 \*}

else begin

error(51);

if sy = eql

then insymbol

end;

expression(fsys,y);

{\* 若x和y的类型是一样 \*}

if x.typ = y.typ

{\* 若x的类型是stantyps，执行第38指令 \*}

then if x.typ in stantyps

then emit(38)

{\* 若x和y的ref不一样，那就是46错误 \*}

else if x.ref <> y.ref

then error(46)

{\* 若x类型为数组，执行第23指令 \*}

else if x.typ = arrays

then emit1(23,atab[x.ref].size)

{\* 不然执行第23指令 \*}

else emit1(23,btab[x.ref].vsize)

{\* 若x是实数而y是整数，执行第26指令 \*}

else if(x.typ = reals )and (y.typ = ints)

then begin

emit1(26,0);

emit(38)

end

{\* 若x和y有类型定义，那就是第46错误 \*}

else if ( x.typ <> notyp ) and ( y.typ <> notyp )

then error(46)

end { assignment };

{\* 过程：compoundstatement \*}

procedure compoundstatement;

begin

{\* 读下一个符号 \*}

insymbol;

statement([semicolon,endsy]+fsys);

while sy in [semicolon]+statbegsys do

begin

{\* 若sy是';'，读下一个符号，不然是第14错误 \*}

if sy = semicolon

then insymbol

else error(14);

statement([semicolon,endsy]+fsys)

end;

{\* 若sy是endsy，读下一个符号，不然就是第57错误 \*}

if sy = endsy

then insymbol

else error(57)

end { compoundstatement };

{\* 过程：ifstatement \*}

procedure ifstatement;

var x : item;

lc1,lc2: integer;

begin

{\* 读下一个符号 \*}

insymbol;

expression( fsys+[thensy,dosy],x);

{\* 若x不属于bools或notyp，那就是第17错误 \*}

if not ( x.typ in [bools,notyp])

then error(17);

lc1 := lc;

emit(11); { jmpc }

{\* 若sy是thensy，读下一个符号 \*}

if sy = thensy

then insymbol

{\* 不然就是第52错误，若sy是dosy，读下一个符号 \*}

else begin

error(52);

if sy = dosy

then insymbol

end;

statement( fsys+[elsesy]);

{\* 若sy是elsesy，读下一个符号 \*}

if sy = elsesy

then begin

insymbol;

lc2 := lc;

emit(10);

code[lc1].y := lc;

statement(fsys);

code[lc2].y := lc

end

else code[lc1].y := lc

end { ifstatement };

{\* 过程：casestatement \*}

procedure casestatement;

var x : item;

i,j,k,lc1 : integer;

casetab : array[1..csmax]of

packed record

val,lc : index

end;

exittab : array[1..csmax] of integer;

{\* 过程：caselabel \*}

procedure caselabel;

var lab : conrec;

k : integer;

begin

constant( fsys+[comma,colon],lab );

{\* 若lab和x的类型不一样，那就是第47错误 \*}

if lab.tp <> x.typ

then error(47)

{\* 若i是csmax，那就是第6 fatal \*}

else if i = csmax

then fatal(6)

else begin

i := i+1;

k := 0;

casetab[i].val := lab.i;

casetab[i].lc := lc;

repeat

k := k+1

until casetab[k].val = lab.i;

if k < i

then error(1); { multiple definition }

end

end { caselabel };

{\* 过程：onecase \*}

procedure onecase;

begin

if sy in constbegsys

then begin

caselabel;

{\* while sy是逗号，执行以下程序 \*}

while sy = comma do

begin

{\* 读下一个符号 \*}

insymbol;

caselabel

end;

{\* 若sy是colon，读下一个符号，不然就是第5错误 \*}

if sy = colon

then insymbol

else error(5);

statement([semicolon,endsy]+fsys);

j := j+1;

exittab[j] := lc;

emit(10)

end

end { onecase };

begin { casestatement }

{\* 读下一个符号 \*}

insymbol;

i := 0;

j := 0;

expression( fsys + [ofsy,comma,colon],x );

{\* 若x类型不属于整数，布尔，字符，无类型，那就是第23错误 \*}

if not( x.typ in [ints,bools,chars,notyp ])

then error(23);

lc1 := lc;

emit(12); {jmpx}

{\* 若sy是ofsy，读下一个符号，不然就是第8错误 \*}

if sy = ofsy

then insymbol

else error(8);

onecase;

{\* while sy是semicolon \*}

while sy = semicolon do

begin

{\* 度下一个符号 \*}

insymbol;

onecase

end;

code[lc1].y := lc;

for k := 1 to i do

begin

emit1( 13,casetab[k].val);

emit1( 13,casetab[k].lc);

end;

emit1(10,0);

for k := 1 to j do

code[exittab[k]].y := lc;

{\* 若sy是endsy，读下一个符号，不然就是第57错误 \*}

if sy = endsy

then insymbol

else error(57)

end { casestatement };

{\* 过程：repeatstatement \*}

procedure repeatstatement;

var x : item;

lc1: integer;

begin

lc1 := lc;

{\* 读下一个符号 \*}

insymbol;

statement( [semicolon,untilsy]+fsys);

while sy in [semicolon]+statbegsys do

begin

{\* 若sy是semicolon，读下一个符号，不然就是第14错误 \*}

if sy = semicolon

then insymbol

else error(14);

statement([semicolon,untilsy]+fsys)

end;

{\* 若sy是untilsy \*}

if sy = untilsy

then begin

{\* 读下一个符号 \*}

insymbol;

expression(fsys,x);

{\* 若x不是布尔，无类型，那就是第17错误 \*}

if not(x.typ in [bools,notyp] )

then error(17);

{\* 执行第11指令 \*}

emit1(11,lc1);

end

{\* 不然就是第53错误 \*}

else error(53)

end { repeatstatement };

{\* 过程：whilestatement \*}

procedure whilestatement;

var x : item;

lc1,lc2 : integer;

begin

{\* 读下一个符号 \*}

insymbol;

lc1 := lc;

expression( fsys+[dosy],x);

{\* 若x不是布尔，无类型，那就是第17错误 \*}

if not( x.typ in [bools, notyp] )

then error(17);

lc2 := lc;

emit(11);

{\* 若sy是dosy，读下一个符号，不然就是第54错误 \*}

if sy = dosy

then insymbol

else error(54);

statement(fsys);

{\* 执行第10指令 \*}

emit1(10,lc1);

code[lc2].y := lc

end { whilestatement };

{\* 过程：forstatement \*}

procedure forstatement;

var cvt : types;

x : item;

i,f,lc1,lc2 : integer;

begin

{\* 读下一个符号 \*}

insymbol;

{\* 若sy是标识符 \*}

if sy = ident

then begin

{\* 表中寻找id的位置放在i中，读下一个符号 \*}

i := loc(id);

insymbol;

{\* 若i是0，那就是没找到 \*}

if i = 0

then cvt := ints

else if tab[i].obj = vvariable

then begin

cvt := tab[i].typ;

if not tab[i].normal

then error(37)

else emit2(0,tab[i].lev, tab[i].adr );

{\* 若cvt不属于notyp, ints, bools, chars，那就是第18错误 \*}

if not ( cvt in [notyp, ints, bools, chars])

then error(18)

end

{\* 不然就是第37错误 \*}

else begin

error(37);

cvt := ints

end

end

{\* 不然，跳过 \*}

else skip([becomes,tosy,downtosy,dosy]+fsys,2);

{\* 若sy是becomes，读下一个符号 \*}

if sy = becomes

then begin

insymbol;

expression( [tosy, downtosy,dosy]+fsys,x);

{\* 若x类型不是cvt，那就是第19错误 \*}

if x.typ <> cvt

then error(19);

end

{\* 不然，跳过 \*}

else skip([tosy, downtosy,dosy]+fsys,51);

f := 14;

{\* 若sy属于tosy，downtosy \*}

if sy in [tosy,downtosy]

then begin

{\* 若sy是downtosy，f赋值16 \*}

if sy = downtosy

then f := 16;

{\* 读下一个符号 \*}

insymbol;

expression([dosy]+fsys,x);

{\* 若x的类型不是cvt，那就是第19错误 \*}

if x.typ <> cvt

then error(19)

end

{\* 不然，跳过 \*}

else skip([dosy]+fsys,55);

lc1 := lc;

emit(f);

{\* 若sy是dosy，读下一个符号，不然就是第54错误 \*}

if sy = dosy

then insymbol

else error(54);

lc2 := lc;

statement(fsys);

emit1(f+1,lc2);

code[lc1].y := lc

end { forstatement };

{\* 过程：standproc（参数n） \*}

procedure standproc( n: integer );

var i,f : integer;

x,y : item;

begin

case n of

{\* 读取 \*}

1,2 : begin { read }

{\* 若iflag无效，那就是第20错误，然后iflag定义为true \*}

if not iflag

then begin

error(20);

iflag := true

end;

{\* 若sy是左括号 \*}

if sy = lparent

then begin

{\* 重复以下程序直到sy不是逗号 \*}

repeat

{\* 读下一个符号 \*}

insymbol;

{\* 若sy不是标识符，那就是第2错误 \*}

if sy <> ident

then error(2)

else begin

{\* 不然在表中寻找id的位置放在i，读下一个符号 \*}

i := loc(id);

insymbol;

{\* 若i不是0，那就是找到了 \*}

if i <> 0

then if tab[i].obj <> vvariable

then error(37)

else begin

x.typ := tab[i].typ;

x.ref := tab[i].ref;

if tab[i].normal

then f := 0

else f := 1;

emit2(f,tab[i].lev,tab[i].adr);

{\* 若sy是左括号开头，左括号，或句号 \*}

if sy in [lbrack,lparent,period]

then selector( fsys+[comma,rparent],x);

{\* 若x的类型属于ints,reals,chars,notyp \*}

if x.typ in [ints,reals,chars,notyp]

then emit1(27,ord(x.typ))

else error(41)

end

end;

test([comma,rparent],fsys,6);

until sy <> comma;

{\* 若sy是右括号，读下一个符号，那就是第4错误 \*}

if sy = rparent

then insymbol

else error(4)

end;

if n = 2

then emit(62)

end;

{\* 写 \*}

3,4 : begin { write }

{\* 若sy是左括号 \*}

if sy = lparent

then begin

{\* 重复以下程序直到sy不是逗号 \*}

repeat

{\* 读下一个符号 \*}

insymbol;

{\* 若sy是stringcon \*}

if sy = stringcon

then begin

emit1(24,sleng);

emit1(28,inum);

insymbol

end

else begin

expression(fsys+[comma,colon,rparent],x);

if not( x.typ in stantyps )

then error(41);

{\* 若sy是colon，读下一个符号 \*}

if sy = colon

then begin

insymbol;

expression( fsys+[comma,colon,rparent],y);

{\* 若y类型不是整数，那就是第43错误 \*}

if y.typ <> ints

then error(43);

{\* 若sy是colon \*}

if sy = colon

then begin

{\* 若x的类型不是实数，那就是第42错误，读下一个符号 \*}

if x.typ <> reals

then error(42);

insymbol;

expression(fsys+[comma,rparent],y);

{\* 若y的类型不是整数，那就是第43错误 \*}

if y.typ <> ints

then error(43);

emit(37)

end

else emit1(30,ord(x.typ))

end

else emit1(29,ord(x.typ))

end

until sy <> comma;

{\* 若sy是右括号，读下一个符号，那就是第4错误 \*}

if sy = rparent

then insymbol

else error(4)

end;

if n = 4

then emit(63)

end; { write }

end { case };

end { standproc } ;

begin { statement }

if sy in statbegsys+[ident]

then case sy of

ident : begin

i := loc(id);

insymbol;

if i <> 0

then case tab[i].obj of

konstant,typel : error(45);

vvariable: assignment( tab[i].lev,tab[i].adr);

prozedure: if tab[i].lev <> 0

then call(fsys,i)

else standproc(tab[i].adr);

funktion: if tab[i].ref = display[level]

then assignment(tab[i].lev+1,0)

else error(45)

end { case }

end;

beginsy : compoundstatement;

ifsy : ifstatement;

casesy : casestatement;

whilesy : whilestatement;

repeatsy: repeatstatement;

forsy : forstatement;

end; { case }

test( fsys, [],14);

end { statement };

begin { block }

dx := 5;

prt := t;

if level > lmax

then fatal(5);

test([lparent,colon,semicolon],fsys,14);

enterblock;

prb := b;

display[level] := b;

tab[prt].typ := notyp;

tab[prt].ref := prb;

if ( sy = lparent ) and ( level > 1 )

then parameterlist;

btab[prb].lastpar := t;

btab[prb].psize := dx;

if isfun

{\* 若sy是colon，读下一个符号 \*}

then if sy = colon

then begin

insymbol; { function type }

{\* 若sy是标识符 \*}

if sy = ident

then begin

{\* 在表中寻找id的位置放在x，然后读下一个符号 \*}

x := loc(id);

insymbol;

{\* 若x不是0，那就是找到了 \*}

if x <> 0

then if tab[x].typ in stantyps

then tab[prt].typ := tab[x].typ

else error(15)

end

else skip( [semicolon]+fsys,2 )

end

{\* 不然就是第5错误 \*}

else error(5);

{\* 若sy是semicolon，读下一个符号，那就是第14错误 \*}

if sy = semicolon

then insymbol

else error(14);

repeat

if sy = constsy

then constdec;

if sy = typesy

then typedeclaration;

if sy = varsy

then variabledeclaration;

btab[prb].vsize := dx;

while sy in [procsy,funcsy] do

procdeclaration;

test([beginsy],blockbegsys+statbegsys,56)

until sy in statbegsys;

tab[prt].adr := lc;

{\* 读下一个符号 \*}

insymbol;

statement([semicolon,endsy]+fsys);

while sy in [semicolon]+statbegsys do

begin

{\* 若sy是semicolon，读下一个符号，那就是第14错误 \*}

if sy = semicolon

then insymbol

else error(14);

statement([semicolon,endsy]+fsys);

end;

{\* 若sy是endsy，读下一个符号，那就是第57错误 \*}

if sy = endsy

then insymbol

else error(57);

test( fsys+[period],[],6 )

end { block };

{\* 过程： interpret \*}

procedure interpret;

{\* 变量定义 \*}

var ir : order ; { instruction buffer }

pc : integer; { program counter }

t : integer; { top stack index }

b : integer; { base index }

h1,h2,h3: integer;

lncnt,ocnt,blkcnt,chrcnt: integer; { counters }

ps : ( run,fin,caschk,divchk,inxchk,stkchk,linchk,lngchk,redchk );

fld: array [1..4] of integer; { default field widths }

display : array[0..lmax] of integer;

s : array[1..stacksize] of { blockmark: }

record

case cn : types of { s[b+0] = fct result }

ints : (i: integer ); { s[b+1] = return adr }

reals :(r: real ); { s[b+2] = static link }

bools :(b: boolean ); { s[b+3] = dynamic link }

chars :(c: char ) { s[b+4] = table index }

end;

{\* 过程： dump \*}

procedure dump;

{\* 变量定义 \*}

var p,h3 : integer;

begin

{\* 输出内容到psout文件 \*}

h3 := tab[h2].lev;

writeln(psout);

writeln(psout);

writeln(psout,' calling ', tab[h2].name );

writeln(psout,' level ',h3:4);

writeln(psout,' start of code ',pc:4);

writeln(psout);

writeln(psout);

writeln(psout,' contents of display ');

writeln(psout);

for p := h3 downto 0 do

writeln(psout,p:4,display[p]:6);

writeln(psout);

writeln(psout);

writeln(psout,' top of stack ',t:4,' frame base ':14,b:4);

writeln(psout);

writeln(psout);

writeln(psout,' stack contents ':20);

writeln(psout);

for p := t downto 1 do

writeln( psout, p:14, s[p].i:8);

writeln(psout,'< = = = >':22)

end; {dump }

{\* 过程：inter0 \*}

procedure inter0;

begin

case ir.f of

{\* 加载地址 \*}

0 : begin { load addrss }

t := t + 1;

if t > stacksize

then ps := stkchk

else s[t].i := display[ir.x]+ir.y

end;

{\* 负载值 \*}

1 : begin { load value }

t := t + 1;

if t > stacksize

then ps := stkchk

else s[t] := s[display[ir.x]+ir.y]

end;

{\* 加载间接 \*}

2 : begin { load indirect }

t := t + 1;

if t > stacksize

then ps := stkchk

else s[t] := s[s[display[ir.x]+ir.y].i]

end;

{\* 更新显示 \*}

3 : begin { update display }

h1 := ir.y;

h2 := ir.x;

h3 := b;

repeat

display[h1] := h3;

h1 := h1-1;

h3 := s[h3+2].i

until h1 = h2

end;

8 : case ir.y of

{\* 整数x求绝对值 \*}

0 : s[t].i := abs(s[t].i);

{\* 实数x求绝对值 \*}

1 : s[t].r := abs(s[t].r);

{\* 整数x求平方 \*}

2 : s[t].i := sqr(s[t].i);

{\* 实数x求平方 \*}

3 : s[t].r := sqr(s[t].r);

{\* 整数x判奇偶性,计数返回1 \*}

4 : s[t].b := odd(s[t].i);

{\* ascii码x转化为字符char \*}

5 : s[t].c := chr(s[t].i);

{\* 字符x转化为ascii码 \*}

6 : s[t].i := ord(s[t].c);

{\* 求字符x的后继字符,比如'a'的后继是'b' \*}

7 : s[t].c := succ(s[t].c);

{\* 求字符x的前导字符 \*}

8 : s[t].c := pred(s[t].c);

{\* 求x的四舍五入 \*}

9 : s[t].i := round(s[t].r);

{\* 求实数x的整数部分 \*}

10 : s[t].i := trunc(s[t].r);

{\* 求正弦sin(x),注意x为实数弧度 \*}

11 : s[t].r := sin(s[t].r);

{\* 求余弦sin(x),注意x为实数弧度 \*}

12 : s[t].r := cos(s[t].r);

{\* 求e^x,x为实数 \*}

13 : s[t].r := exp(s[t].r);

{\* 求自然对数ln(x),x为实数 \*}

14 : s[t].r := ln(s[t].r);

{\* 实数x开方 \*}

15 : s[t].r := sqrt(s[t].r);

{\* 反三角函数arctan(x) \*}

16 : s[t].r := arcTan(s[t].r);

17 : begin

t := t+1;

if t > stacksize

then ps := stkchk

else s[t].b := eof(prd)

end;

18 : begin

t := t+1;

if t > stacksize

then ps := stkchk

else s[t].b := eoln(prd)

end;

end;

9 : s[t].i := s[t].i + ir.y; { offset }

end { case ir.y }

end; { inter0 }

{\* 过程：inter1 \*}

procedure inter1;

var h3, h4: integer;

begin

case ir.f of

{\* 跳过 \*}

10 : pc := ir.y ; { jump }

{\* 有条件跳过 \*}

11 : begin { conditional jump }

if not s[t].b

then pc := ir.y;

t := t - 1

end;

{\* 切换 \*}

12 : begin { switch }

h1 := s[t].i;

t := t-1;

h2 := ir.y;

h3 := 0;

repeat

if code[h2].f <> 13

then begin

h3 := 1;

ps := caschk

end

else if code[h2].y = h1

then begin

h3 := 1;

pc := code[h2+1].y

end

else h2 := h2 + 2

until h3 <> 0

end;

{\* 若1 up \*}

14 : begin { for1up }

h1 := s[t-1].i;

if h1 <= s[t].i

then s[s[t-2].i].i := h1

else begin

t := t - 3;

pc := ir.y

end

end;

{\* 若2 up \*}

15 : begin { for2up }

h2 := s[t-2].i;

h1 := s[h2].i+1;

if h1 <= s[t].i

then begin

s[h2].i := h1;

pc := ir.y

end

else t := t-3;

end;

{\* 若1 down \*}

16 : begin { for1down }

h1 := s[t-1].i;

if h1 >= s[t].i

then s[s[t-2].i].i := h1

else begin

pc := ir.y;

t := t - 3

end

end;

{\* 若2 down \*}

17 : begin { for2down }

h2 := s[t-2].i;

h1 := s[h2].i-1;

if h1 >= s[t].i

then begin

s[h2].i := h1;

pc := ir.y

end

else t := t-3;

end;

{\* 标记堆栈 \*}

18 : begin { mark stack }

h1 := btab[tab[ir.y].ref].vsize;

if t+h1 > stacksize

then ps := stkchk

else begin

t := t+5;

s[t-1].i := h1-1;

s[t].i := ir.y

end

end;

{\* 叫 \*}

19 : begin { call }

{\* h1指向base \*}

h1 := t-ir.y; { h1 points to base }

{\* h2指向tab \*}

h2 := s[h1+4].i; { h2 points to tab }

h3 := tab[h2].lev;

display[h3+1] := h1;

h4 := s[h1+3].i+h1;

s[h1+1].i := pc;

s[h1+2].i := display[h3];

s[h1+3].i := b;

for h3 := t+1 to h4 do

s[h3].i := 0;

b := h1;

t := h4;

pc := tab[h2].adr;

if stackdump

then dump

end;

end { case }

end; { inter1 }

{\* 过程：inter2 \*}

procedure inter2;

begin

case ir.f of

{\* 索引1 \*}

20 : begin { index1 }

{\* h1只想atab \*}

h1 := ir.y; { h1 points to atab }

h2 := atab[h1].low;

h3 := s[t].i;

if h3 < h2

then ps := inxchk

else if h3 > atab[h1].high

then ps := inxchk

else begin

t := t-1;

s[t].i := s[t].i+(h3-h2)

end

end;

{\* 索引 \*}

21 : begin { index }

{\* h1只想atab \*}

h1 := ir.y ; { h1 points to atab }

h2 := atab[h1].low;

h3 := s[t].i;

if h3 < h2

then ps := inxchk

else if h3 > atab[h1].high

then ps := inxchk

else begin

t := t-1;

s[t].i := s[t].i + (h3-h2)\*atab[h1].elsize

end

end;

{\* 加载块 \*}

22 : begin { load block }

h1 := s[t].i;

t := t-1;

h2 := ir.y+t;

if h2 > stacksize

then ps := stkchk

else while t < h2 do

begin

t := t+1;

s[t] := s[h1];

h1 := h1+1

end

end;

{\* 复制块 \*}

23 : begin { copy block }

h1 := s[t-1].i;

h2 := s[t].i;

h3 := h1+ir.y;

while h1 < h3 do

begin

s[h1] := s[h2];

h1 := h1+1;

h2 := h2+1

end;

t := t-2

end;

{\* 文字 \*}

24 : begin { literal }

t := t+1;

if t > stacksize

then ps := stkchk

else s[t].i := ir.y

end;

{\* 加载实数 \*}

25 : begin { load real }

t := t+1;

if t > stacksize

then ps := stkchk

else s[t].r := rconst[ir.y]

end;

{\* 浮点数 \*}

26 : begin { float }

h1 := t-ir.y;

s[h1].r := s[h1].i

end;

{\* 读取 \*}

27 : begin { read }

if eof(prd)

then ps := redchk

else case ir.y of

1 : read(prd, s[s[t].i].i);

2 : read(prd, s[s[t].i].r);

4 : read(prd, s[s[t].i].c);

end;

t := t-1

end;

{\* 写字符串 \*}

28 : begin { write string }

h1 := s[t].i;

h2 := ir.y;

t := t-1;

chrcnt := chrcnt+h1;

if chrcnt > lineleng

then ps := lngchk;

repeat

write(prr,stab[h2]);

h1 := h1-1;

h2 := h2+1

until h1 = 0

end;

{\* 写1 \*}

29 : begin { write1 }

chrcnt := chrcnt + fld[ir.y];

if chrcnt > lineleng

then ps := lngchk

else case ir.y of

1 : write(prr,s[t].i:fld[1]);

2 : write(prr,s[t].r:fld[2]);

3 : if s[t].b

then write('true')

else write('false');

4 : write(prr,chr(s[t].i));

end;

t := t-1

end;

end { case }

end; { inter2 }

{\* 过程：inter3 \*}

procedure inter3;

begin

case ir.f of

{\* 写2 \*}

30 : begin { write2 }

chrcnt := chrcnt+s[t].i;

if chrcnt > lineleng

then ps := lngchk

else case ir.y of

1 : write(prr,s[t-1].i:s[t].i);

2 : write(prr,s[t-1].r:s[t].i);

3 : if s[t-1].b

then write('true')

else write('false');

end;

t := t-2

end;

31 : ps := fin;

{\* 退出程序 \*}

32 : begin { exit procedure }

t := b-1;

pc := s[b+1].i;

b := s[b+3].i

end;

{\* 退出函数 \*}

33 : begin { exit function }

t := b;

pc := s[b+1].i;

b := s[b+3].i

end;

34 : s[t] := s[s[t].i];

35 : s[t].b := not s[t].b;

36 : s[t].i := -s[t].i;

37 : begin

chrcnt := chrcnt + s[t-1].i;

if chrcnt > lineleng

then ps := lngchk

else write(prr,s[t-2].r:s[t-1].i:s[t].i);

t := t-3

end;

{\* 储存 \*}

38 : begin { store }

s[s[t-1].i] := s[t];

t := t-2

end;

39 : begin

t := t-1;

s[t].b := s[t].r=s[t+1].r

end;

end { case }

end; { inter3 }

{\* 过程：inter4 \*}

procedure inter4;

begin

case ir.f of

40 : begin

t := t-1;

s[t].b := s[t].r <> s[t+1].r

end;

41 : begin

t := t-1;

s[t].b := s[t].r < s[t+1].r

end;

42 : begin

t := t-1;

s[t].b := s[t].r <= s[t+1].r

end;

43 : begin

t := t-1;

s[t].b := s[t].r > s[t+1].r

end;

44 : begin

t := t-1;

s[t].b := s[t].r >= s[t+1].r

end;

45 : begin

t := t-1;

s[t].b := s[t].i = s[t+1].i

end;

46 : begin

t := t-1;

s[t].b := s[t].i <> s[t+1].i

end;

47 : begin

t := t-1;

s[t].b := s[t].i < s[t+1].i

end;

48 : begin

t := t-1;

s[t].b := s[t].i <= s[t+1].i

end;

49 : begin

t := t-1;

s[t].b := s[t].i > s[t+1].i

end;

end { case }

end; { inter4 }

{\* 过程：inter5 \*}

procedure inter5;

begin

case ir.f of

50 : begin

t := t-1;

s[t].b := s[t].i >= s[t+1].i

end;

51 : begin

t := t-1;

s[t].b := s[t].b or s[t+1].b

end;

52 : begin

t := t-1;

s[t].i := s[t].i+s[t+1].i

end;

53 : begin

t := t-1;

s[t].i := s[t].i-s[t+1].i

end;

54 : begin

t := t-1;

s[t].r := s[t].r+s[t+1].r;

end;

55 : begin

t := t-1;

s[t].r := s[t].r-s[t+1].r;

end;

56 : begin

t := t-1;

s[t].b := s[t].b and s[t+1].b

end;

57 : begin

t := t-1;

s[t].i := s[t].i\*s[t+1].i

end;

58 : begin

t := t-1;

if s[t+1].i = 0

then ps := divchk

else s[t].i := s[t].i div s[t+1].i

end;

59 : begin

t := t-1;

if s[t+1].i = 0

then ps := divchk

else s[t].i := s[t].i mod s[t+1].i

end;

end { case }

end; { inter5 }

{\* 过程：inter6 \*}

procedure inter6;

begin

case ir.f of

60 : begin

t := t-1;

s[t].r := s[t].r\*s[t+1].r;

end;

61 : begin

t := t-1;

s[t].r := s[t].r/s[t+1].r;

end;

62 : if eof(prd)

then ps := redchk

else readln;

63 : begin

writeln(prr);

lncnt := lncnt+1;

chrcnt := 0;

if lncnt > linelimit

then ps := linchk

end

end { case };

end; { inter6 }

begin { interpret }

{\* 变量赋值 \*}

s[1].i := 0;

s[2].i := 0;

s[3].i := -1;

s[4].i := btab[1].last;

display[0] := 0;

display[1] := 0;

t := btab[2].vsize-1;

b := 0;

pc := tab[s[4].i].adr;

lncnt := 0;

ocnt := 0;

chrcnt := 0;

ps := run;

fld[1] := 10;

fld[2] := 22;

fld[3] := 10;

fld[4] := 1;

{\* 重复以下程序直到ps不为run \*}

repeat

ir := code[pc];

pc := pc+1;

ocnt := ocnt+1;

{\* 对于不同的ir，执行不一样的interpret \*}

case ir.f div 10 of

0 : inter0;

1 : inter1;

2 : inter2;

3 : inter3;

4 : inter4;

5 : inter5;

6 : inter6;

end; { case }

until ps <> run;

{\* 如果ps不为fin \*}

if ps <> fin

then begin

writeln(prr);

write(prr, ' halt at', pc :5, ' because of ');

case ps of

caschk : writeln(prr,'undefined case');

divchk : writeln(prr,'division by 0');

inxchk : writeln(prr,'invalid index');

stkchk : writeln(prr,'storage overflow');

linchk : writeln(prr,'too much output');

lngchk : writeln(prr,'line too long');

redchk : writeln(prr,'reading past end or file');

end;

h1 := b;

blkcnt := 10; { post mortem dump }

{\* 重复以下程序直到h1<0 \*}

repeat

writeln( prr );

blkcnt := blkcnt-1;

if blkcnt = 0

then h1 := 0;

h2 := s[h1+4].i;

if h1 <> 0

then writeln( prr, '',tab[h2].name, 'called at', s[h1+1].i:5);

h2 := btab[tab[h2].ref].last;

{\* while h2不是0 \*}

while h2 <> 0 do

{\* 对于表中第h2对应的元素执行以下操作 \*}

with tab[h2] do

begin

if obj = vvariable

then if typ in stantyps

then begin

write(prr,'',name,'=');

if normal

then h3 := h1+adr

else h3 := s[h1+adr].i;

case typ of

ints : writeln(prr,s[h3].i);

reals: writeln(prr,s[h3].r);

bools: if s[h3].b

then writeln(prr,'true')

else writeln(prr,'false');

chars: writeln(prr,chr(s[h3].i mod 64 ))

end

end;

h2 := link

end;

h1 := s[h1+3].i

until h1 < 0

end;

writeln(prr);

writeln(prr,ocnt,' steps');

end; { interpret }

{\* 过程：setup \*}

procedure setup;

begin

{\* 局部变量定义 \*}

key[1] := 'and ';

key[2] := 'array ';

key[3] := 'begin ';

key[4] := 'case ';

key[5] := 'const ';

key[6] := 'div ';

key[7] := 'do ';

key[8] := 'downto ';

key[9] := 'else ';

key[10] := 'end ';

key[11] := 'for ';

key[12] := 'function ';

key[13] := 'if ';

key[14] := 'mod ';

key[15] := 'not ';

key[16] := 'of ';

key[17] := 'or ';

key[18] := 'procedure ';

key[19] := 'program ';

key[20] := 'record ';

key[21] := 'repeat ';

key[22] := 'then ';

key[23] := 'to ';

key[24] := 'type ';

key[25] := 'until ';

key[26] := 'var ';

key[27] := 'while ';

ksy[1] := andsy;

ksy[2] := arraysy;

ksy[3] := beginsy;

ksy[4] := casesy;

ksy[5] := constsy;

ksy[6] := idiv;

ksy[7] := dosy;

ksy[8] := downtosy;

ksy[9] := elsesy;

ksy[10] := endsy;

ksy[11] := forsy;

ksy[12] := funcsy;

ksy[13] := ifsy;

ksy[14] := imod;

ksy[15] := notsy;

ksy[16] := ofsy;

ksy[17] := orsy;

ksy[18] := procsy;

ksy[19] := programsy;

ksy[20] := recordsy;

ksy[21] := repeatsy;

ksy[22] := thensy;

ksy[23] := tosy;

ksy[24] := typesy;

ksy[25] := untilsy;

ksy[26] := varsy;

ksy[27] := whilesy;

sps['+'] := plus;

sps['-'] := minus;

sps['\*'] := times;

sps['/'] := rdiv;

sps['('] := lparent;

sps[')'] := rparent;

sps['='] := eql;

sps[','] := comma;

sps['['] := lbrack;

sps[']'] := rbrack;

sps[''''] := neq;

sps['!'] := andsy;

sps[';'] := semicolon;

end { setup };

{\* 过程：enterids \*}

procedure enterids;

begin

{\* 局部变量定义 \*}

enter(' ',vvariable,notyp,0); { sentinel }

enter('false ',konstant,bools,0);

enter('true ',konstant,bools,1);

enter('real ',typel,reals,1);

enter('char ',typel,chars,1);

enter('boolean ',typel,bools,1);

enter('integer ',typel,ints,1);

enter('abs ',funktion,reals,0);

enter('sqr ',funktion,reals,2);

enter('odd ',funktion,bools,4);

enter('chr ',funktion,chars,5);

enter('ord ',funktion,ints,6);

enter('succ ',funktion,chars,7);

enter('pred ',funktion,chars,8);

enter('round ',funktion,ints,9);

enter('trunc ',funktion,ints,10);

enter('sin ',funktion,reals,11);

enter('cos ',funktion,reals,12);

enter('exp ',funktion,reals,13);

enter('ln ',funktion,reals,14);

enter('sqrt ',funktion,reals,15);

enter('arctan ',funktion,reals,16);

enter('eof ',funktion,bools,17);

enter('eoln ',funktion,bools,18);

enter('read ',prozedure,notyp,1);

enter('readln ',prozedure,notyp,2);

enter('write ',prozedure,notyp,3);

enter('writeln ',prozedure,notyp,4);

enter(' ',prozedure,notyp,0);

end;

begin { main }

{\* 执行setup \*}

setup;

{\* 变量初始化定义 \*}

constbegsys := [ plus, minus, intcon, realcon, charcon, ident ];

typebegsys := [ ident, arraysy, recordsy ];

blockbegsys := [ constsy, typesy, varsy, procsy, funcsy, beginsy ];

facbegsys := [ intcon, realcon, charcon, ident, lparent, notsy ];

statbegsys := [ beginsy, ifsy, whilesy, repeatsy, forsy, casesy ];

stantyps := [ notyp, ints, reals, bools, chars ];

lc := 0;

ll := 0;

cc := 0;

ch := ' ';

errpos := 0;

errs := [];

writeln( 'NOTE input/output for users program is console : ' );

writeln;

{\* 让用户输入Source input file名放在psin变量 \*}

write( 'Source input file ?');

readln( inf );

assign( psin, inf );

reset( psin );

{\* 让用户输入Source listing file名放在psout变量 \*}

write( 'Source listing file ?');

readln( outf );

assign( psout, outf );

rewrite( psout );

assign ( prd, 'con' );

{\* 让用户输入result file名放在prr变量 \*}

write( 'result file : ' );

readln( fprr );

assign( prr, fprr );

reset ( prd );

rewrite( prr );

{\* 表的索引的定义 \*}

t := -1;

a := 0;

b := 1;

sx := 0;

c2 := 0;

display[0] := 1;

iflag := false;

oflag := false;

skipflag := false;

prtables := false;

stackdump := false;

{\* 执行insymbol \*}

insymbol;

{\* 如果insymbol最后读取的符号不是programsy，那么就是第3错误 \*}

if sy <> programsy

then error(3)

else begin

{\* 不然再执行insymbol \*}

insymbol;

{\* 如果insymbol最后读取的符号不是ident，那么就是第2错误 \*}

if sy <> ident

then error(2)

else begin

{\* 不然再执行insymbol \*}

progname := id;

insymbol;

{\* 如果insymbol最后读取的符号不是lparent，那么就是第9错误 \*}

if sy <> lparent

then error(9)

{\* 不然重复以下程序直到insymbol最后读取的符号不是comma \*}

else repeat

{\* 再执行insymbol \*}

insymbol;

{\* 如果insymbol最后读取的符号不是ident，那么就是第2错误 \*}

if sy <> ident

then error(2)

else begin

if id = 'input '

then iflag := true

else if id = 'output '

then oflag := true

else error(0);

{\* 再执行insymbol \*}

insymbol

end

until sy <> comma;

{\* 如果insymbol最后读取的符号是rparent那么执行insymbol，不然就是第4错误 \*}

if sy = rparent

then insymbol

else error(4);

if not oflag then error(20)

end

end;

enterids;

with btab[1] do

begin

last := t;

lastpar := 1;

psize := 0;

vsize := 0;

end;

block( blockbegsys + statbegsys, false, 1 );

if sy <> period

then error(2);

emit(31); { halt }

if prtables

then printtables;

{\* 如果errs数组为空，那么interpret \*}

if errs = []

then interpret

{\* 不然把错误信息输出在psout文件 \*}

else begin

writeln( psout );

writeln( psout, 'compiled with errors' );

writeln( psout );

errormsg;

end;

writeln( psout );

close( psout );

close( prr )

end.