《编译技术》课程设计文 档

学号：\_\_\_\_\_13061049\_\_\_\_\_\_\_

姓名：\_\_\_\_\_\_\_季豪\_\_\_\_\_\_\_\_\_

2016年1月6日

**目录**

[一．需求说明 3](#_Toc32622)

[1．文法说明 3](#_Toc17079)

[2．目标代码说明 3](#_Toc13052)

[二．详细设计 5](#_Toc8573)

[1．程序结构 5](#_Toc3831)

[2．类/方法/函数功能 6](#_Toc30403)

[3．调用依赖关系 7](#_Toc23845)

[4．符号表管理方案 7](#_Toc1376)

[5．存储分配方案 8](#_Toc7295)

[6. 解释执行程序\* 8](#_Toc480)

[7. 目标代码生成方案\* 11](#_Toc757)

[8. 出错处理 11](#_Toc6080)

[三．操作说明 12](#_Toc20015)

[1．运行环境 12](#_Toc21919)

[2．操作步骤 12](#_Toc25476)

[四．测试报告 13](#_Toc8165)

[1．测试程序及测试结果 13](#_Toc127)

[2．测试结果分析 20](#_Toc1453)

[五．总结感想 21](#_Toc2211)

## 一．需求说明

### 1．文法说明

**1) 获取的文法（低难度PL/0）**

1.<程序>::=<分程序>.

2.<分程序>::=[<常量说明部分>][<变量说明部分>][<过程说明部分>]<复合语句>

3.<常量说明部分>::=const<常量定义>{,<常量定义>};

4.<常量定义>::=<标识符>=<无符号整数>

5.<无符号整数>::=<非零数字>{<数字>}

6.<标识符>::=<字母>{<字母>|<数字>}

7.<变量说明部分>::=var<标识符>{, <标识符>};

8.<过程说明部分>::=<过程首部><分程序>{;<过程说明部分>};

9.<过程首部>::=procedure<标识符>;

10.<语句>::=<赋值语句>|<条件语句>|<当循环语句>|<过程调用语句>|<复合语句>|<读语句>|<写语句>|<空>

11.<赋值语句>::=<标识符> := <表达式>

12.<表达式>::= [+|-]<项>{<加法运算符><项>}

13.<项>::=<因子>{<乘法运算符><因子>}

14.<因子>::=<标识符>|<无符号整数>| '('<表达式>')'

15.<加法运算符>::=+|-

16.<乘法运算符>::=\*|/

17.<条件>::=<表达式><关系运算符><表达式>|odd<表达式>

18.<关系运算符>::==|<>|<|<=|>|>=

19.<条件语句>::=if<条件>then<语句>[else<语句>]

20.<当循环语句>::=while<条件>do<语句>

21.<过程调用语句>::=call<标识符>

22.<复合语句>::=begin<语句>{;<语句>}end

23.<读语句>::=read '('<标识符>{, <标识符>}')'

24.<写语句>::=write '('<表达式>{, <表达式>}')'

25.<字母>::=a|b|c|d…x|y|z|A|B...|Z

26.<数字>::=0|<非零数字>

27.<非零数字>::=1|2|3…8|9

附加说明：标识符不区分大小写字母

1. **改写扩展**

由于原文法不存在左递归，因此不对文法进行改写

### 2．目标代码说明

**1)文法生成的目标代码是p-code码，由interpret()函数解释执行**

enum fct

{

lit, opr, lod, sto, cal, Int, jmp, jpc

};

typedef struct

{

enum fct f;

long l;

long a;

} instruction;

**2)目标代码指令集**

|  |  |
| --- | --- |
| 指令 | 主要功能 |
| **lit** | 把常量值放到运行栈栈顶 |
| **opr** | 运算操作（包括读写） |
| **lod** | 找到变量的地址，并存入栈顶 |
| **sto** | 将数据栈顶内容保存入变量 |
| **cal** | 过程调用 |
| **int** | 开辟新的内存单元 |
| **jmp** | 无条件跳转 |
| **jpc** | 有条件跳转 |

其中：opr 0 a

//a=0 return

//a=1 取反

//a=2 加法

//a=3 减法

//a=4 乘法

//a=5 除法

//a=6 判断奇数odd

//a=8 判断等于

//a=9 判断不等

//a=10 小于

//a=11 大于等于

//a=12 大于

//a=13 小于等于

//a=14 输出

//a=15 换行

//a=16 输入

## 二．详细设计

### 1．程序结构

**1)函数定义层次结构**

main()

    error()//错误处理

    getsym()//词法分析

        getch()

    gen()//生成P-CODE

    test()//测试当前单词符号是否合法

    block()//分程序分析

        enter()

        position()

        constdeclaration()

        vardeclaration()

        listcode()

        statement()//语句分析处理

            expression()

                term()

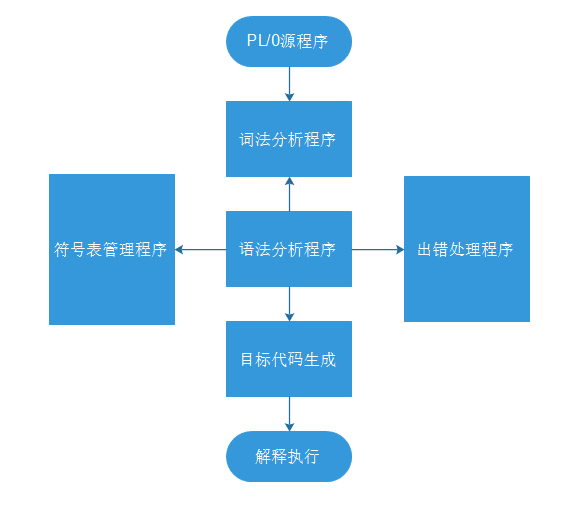
                    factor()

            condition()

    interpret()//P-code解释执行

        base()

**2)系统层次结构**



### 2．类/方法/函数功能

**1)函数定义解释**

|  |  |
| --- | --- |
| 函数 | 主要功能 |
| main | 主程序 |
| error | 错误处理 |
| getsym | 词法分析 |
| getch | 取字符 |
| gen | 生成P-code指令 |
| test | 测试当前单词符号是否合法 |
| block | 分程序分析处理 |
| enter | 登记符号表 |
| position | 查找标识符在符号表中的位置 |
| constdeclaration | 常量定义处理 |
| vardeclaration | 变量定义处理 |
| listcode | 列出p-code指令清单 |
| statement | 语句分析处理 |
| expression | 表达式分析处理 |
| term | 项分析处理 |
| factor | 因子分析处理 |
| condition | 条件分析处理 |
| interpret | P-code解释执行 |
| base | 通过静态链求出数据区的基地址 |

**2)关键算法**

////////////////////词法分析////////////////////////////////////////////

getsym(); 二分折半查找保留字

i = 0;

j = norw - 1;

do

{

k = (i + j) / 2;

if (strcmp(id, word[k])==0) break;

else if (strcmp(id,word[k])<0)

{

j = k - 1;

}

else if (strcmp(id,word[k])>0)

{

i = k + 1;

}

} while (i <= j);

if (i>j)

{

strcpy\_s(sym,al+1,ident);

}

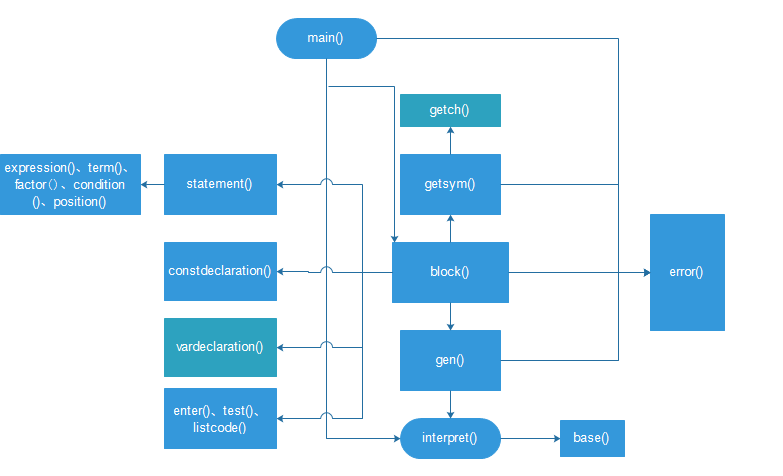
else

{

strcpy\_s(sym, al+1, wsym[k]);

}

### 3．调用依赖关系



### 4．符号表管理方案

**1)符号表table定义**

#define txmax       100

enum object {

constant, variable, proc

};

struct {

char name[al+1];

enum object kind;

int val;

int level;

int addr;

}table[txmax+1]

1. **符号表管理**

|  |  |
| --- | --- |
| 方法 | 作用 |
| enter() | 登记符号表 |
| position() | 查找标识符在符号表中的位置 |

### 5．存储分配方案

**1)存储组织**

运行栈以数组的形式存储、运行。

通过栈顶指针的移动，栈顶数据的加载和保存来运行程序。

**2)运行栈**

#define stacksize 500

int s[stacksize];

int p,b,t;

// p为程序指令指针，指向下一条要运行的代码

//b为基址指针，指向每个过程被调用时数据区中分配给它的局部变量数据段基址

//t记录这个计算机的当前栈顶位置

instructioni; // i变量中存放当前在运行的指令

### 6. 解释执行程序\*

**1)解释执行根据产生的p-code代码解释执行程序，用case语句确定指令并执行**

1. **关键算法：通过静态链求出数据区基地址的函数base()**

long base(long b, long l)

{

long b1;

b1 = b;

while (l>0)

{

b1 = s[b1];

l--;

}

return b1;

}

1. **解释执行程序**

void interpret()

{

long p, b, t; // program-, base-, topstack-registers

instruction i; // instruction register

t = 0; b = 1; p = 0;

s[1] = 0; s[2] = 0; s[3] = 0;

do

{

i = code[p];

p = p + 1;

switch (i.f)

{

case lit:

t = t + 1; s[t] = i.a;

break;

case opr:

switch (i.a) // operator

{

case 0: // return

t = b - 1; p = s[t + 3]; b = s[t + 2];

break;

case 1:

s[t] = -s[t];

break;

case 2:

t = t - 1; s[t] = s[t] + s[t + 1];

break;

case 3:

t = t - 1; s[t] = s[t] - s[t + 1];//减法

break;

case 4:

t = t - 1; s[t] = s[t] \* s[t + 1];

break;

case 5:

t = t - 1; s[t] = s[t] / s[t + 1];

break;

case 6:

s[t] = s[t] % 2;

break;

case 8:

t = t - 1; s[t] = (s[t] == s[t + 1]);

break;

case 9:

t = t - 1; s[t] = (s[t] != s[t + 1]);

break;

case 10:

t = t - 1; s[t] = (s[t]<s[t + 1]);

break;

case 11:

t = t - 1; s[t] = (s[t] >= s[t + 1]);

break;

case 12:

t = t - 1; s[t] = (s[t]>s[t + 1]);

break;

case 13:

t = t - 1; s[t] = (s[t] <= s[t + 1]);

break;

case 14:

printf("%10d", s[t]); t = t - 1;

break;

case 15:

printf("\n");

break;

case 16:

t = t + 1; printf("input:"); scanf("%d", &s[t]);

}

break;

case lod:

t = t + 1; s[t] = s[base(b, i.l) + i.a];

break;

case sto:

s[base(b, i.l) + i.a] = s[t]; t = t - 1;

break;

case cal:

s[t + 1] = base(b, i.l); s[t + 2] = b; s[t + 3] = p;

b = t + 1; p = i.a;

break;

case Int:

t = t + i.a;

break;

case jmp:

p = i.a;

break;

case jpc:

if (s[t] == 0)

{

p = i.a;

}

t = t - 1;

}

} while (p != 0);

}

### 7. 目标代码生成方案\*

**1)目标代码生成gen()**

enumfct {

lit, opr, lod, sto, cal, Int, jmp, jpc

};

void gen(enum fct x, long y, long z)

{

if (cx > cxmax)

{

error(26);

exit(1);

}

code[cx].f = x;

code[cx].l = y;

code[cx].a = z;

cx++;

}

### 8. 出错处理

char\* err\_msg[] =

{

"错误0：非正常结束！",

"错误1：需要’=’而不是’:=’！",

"错误2：需要数字！",

"错误3：需要等号/赋值号！",

"错误4：需要标识符！",

"错误5：需要分号’;’！",

"错误6：过程说明末尾出现非法字符",

"错误7：说明部分之后应该为复合语句",

"错误8：分程序结尾有非法字符",

"错误9：需要句号’.’",

"错误10：需要分号’;’！",

"错误11：未定义的标识符！",

"错误12：标识符不是变量！",

"错误13：需要赋值号’:=’！",

"错误14：call后跟的不是标识符！",

"错误15：call后跟的不是过程名！",

"错误16：if后没有then！",

"错误17：需要end",

"错误18：需要do",

"错误19：语句末尾出现非法字符",

"错误20：缺少二元逻辑运算符！",

"错误21：错误使用标识符，标识符是过程名！",

"错误22：需要右括号’)’！",

"错误23：因子之间应该为'\*'或'/'",

"错误24：因子开头应该为标识符/数字/左括号",

"错误25：无符号整数的首字符应该为非零数字！",

"错误26：生成代码过多！",

"错误27：不合法的同名标识符",

"错误28：",

"错误29：",

"错误30：数字太大！",

"错误31：数字位数太长！",

"错误32：嵌套层数大于最大允许的套层数！",

"错误33：",

"错误34：",

"错误35：read()中应是声明过的变量名！",

"错误36：read语句的标识符应该为变量！",

"错误37：",

"错误38：",

"错误39：",

"错误40：需要左括号’(’"

};

## 三．操作说明

### 1．运行环境

Windows VS Express 2013

### 2．操作步骤

运行编译程序。

输入需要编译的源文件名称，回车。

若程序无误，将会产生正确的目标代码，并在控制台解释执行；

否则，在控制台打印错误信息。

## 四．测试报告

### 1．测试程序及测试结果

**1.1（正确）**

const a=10;

var b,c;

procedure p;

begin

c:=a+b;

end;

begin

read(b);

while b<>100 do

begin

call p;

write(2\*c);

read(b);

end;

end.

/////////////////////

input:5

30

**1.2（正确）**

const true=1,false=2;

var x,y,m,n,pf;

procedure prime;

var i,f;

procedure mod;

begin

x:=x-x/y\*y;

end;

begin

f:=true;

i:=3;

while i<m do

begin

x:=m;

y:=i;

call mod;

if x=1-1 then f:=false;

i:=i+2

end;

if f=true then

begin

write(m);

pf:=true

end

end;

begin

pf:=false;

read(n);

while n>=2 do

begin

write(2);

if n=2 then pf:=true;

m:=3;

while m<=n do

begin

call prime;

m:=m+2

end;

read(n)

end;

if pf=false then write(1-1)

end.

/////////////////////

input:12

2

3

5

7

11

**1.3（正确）**

const a=45,b=27;

var x,y,g,m;

procedure swap;

var temp;

begin

temp:=x;

x:=y;

y:=temp

end;

procedure mod;

begin

x:=x-x/y\*y

end;

begin

x:=a;

y:=b;

call mod;

while x<>1-1 do

begin

call swap;

call mod

end;

g:=y;

m:=a\*b/g;

write(g,m)

end.

/////////////////////

9 135

**1.4（正确）**

const a=3,b=4,c=5,m=199;

var x1x,s,a1,b1,c1;

procedure circle;

const m=555;

begin

if a1=1 then

begin

s:=1;

write(s,m)

end

else

begin

a1:=a1-1;

call Circle

end

end;

begin

a1:=a;

b1:=b;

c1:=c;

read(x1x);

write(x1x);

if x1x<2 then

begin

b1:=2;

c1:=3;

call circle;

end

else

begin

s:=2;

write (s,m);

while x1x>=2 do

x1x:=x1x-1

end;

write(a1,b1,c1,m)

end.

/////////////////////

input:1

1

1 555

1 2 3 199

input:2

2

2 199

3 4 5 199

**1.5（正确）**

const z=2;

var head,foot,cock,rabbit,n;

begin

n:=1-1;

read(head,foot);

cock:=1;

while cock<=head do

begin

rabbit:=head-cock;

if cock\*2+rabbit\*4=foot then

begin

write(cock,rabbit);

n:=n+1

end;

cock:=cock+1;

end;

if n=1-1 then write(1-1,1-1)

end.

/////////////////////

input:20

input:50

15 5

**1.6（错误）**

1 const a=3,b=4,c=5,m=199,yyy=500;

2 var x1x,s,a1,b1,c1,yyy;

Error=>(2,22)|错误27：非法同名标识符(27)

3 procedure circle;

4 const m=555;

5 begin

6 if a1=1 then

7 begin

8 s:=1;

9 write(s,m)

10 end

11 else

12 begin

13 a1:=a1-1;

14 call Circle

15 end

16 end;

17

18 begin

19 a1:=a;

20 b1:=b;

21 c1:=c;

22 read(x1x);

23 write(x1x);

24 if x1x<2 then

25 begin

26 b1:=2;

27 c1:=3;

28 call circle;

29 end

30 else

31 begin

32 s:=2;

33 write (s,m);

34 while x1x>=2 do

35 x1x:=x1x-1

36 end;

37 write(a1,b1,c1,m)

38 end

Error=>(39,3)|错误0：非正常结束！(0)

**1.7（错误）**

1 const b=2;

2 var c;

3 procedure p;

4 begin

5 c:=a+b;

Error=>(5,8)|错误11：未定义的标识符！(11)

6 end;

7 begin

8 read(b);

Error=>(8,8)|错误36：read语句的标识符应该为变量！(36)

9 while b<>100 do

10 begin

11 call p;

12 write(2\*c);

13 read(b);

Error=>(13,6)|错误36：read语句的标识符应该为变量！(36)

14 end;

15 end.

**1.8（错误）**

1 const a=045,b=27;

Error=>(1,8)|错误25：无符号整数的首字符应该为非零数字！(25)

2 var x,y,g,m;

3 procedure swap;

4 var temp;

5 begin

6 temp:=x;

7 x:=y;

8 y:=temp

9 end;

10 procedure mod;

11 x:=x-x/y\*y;

Error=>(11,3)|错误7：说明部分之后应该为复合语句(7)

12 begin

13 x:=a;

14 y:=b;

15 call mod;

16 while x<>1-1 do

17 begin

18 call swap;

19 call mod

20 end;

21 g:=y;

22 m:=a\*b/g;

23 write(g,m)

24 end.

**1.9（错误）**

1 const true=1,false=2;

2 var x,y,m,n,pf;

3 procedure prime;

4 var i,f;

5 procedure mod;

6 begin

7 x=x-x/y\*y;

Error=>(7,6)|错误13：需要赋值号’:=’！(13)

8 end;

9 begin

10 f:=true;

11 i:=3;

12 while i<m

13 begin

Error=>(13,9)|错误18：需要do(18)

14 x:=m;

15 y:=i;

16 call mod;

17 if x=1-1 then f:=false;

18 i:=i+2

19 end;

20 if f=true then

21 begin

22 write(m;

Error=>(22,14)|错误22：需要右括号’)’！(22)

23 pf:=true

24 end

25 end;

26

27 begin

28 pf:=false;

29 read(n);

30 while n>=2 do

31 begin

32 write(2);

33 if n=2 then pf:=true;

34 m:=3;

35 while m<=n do

36 begin

37 call prime;

38 m:=m+2

39 end;

40 read(n)

41 end;

42 if pf=false then write(1-1)

43 end.

**1.10（错误）**

1 const z=2;

2 var head,foot,cock,rabbit,n;

3 begin

4 n:=1-1;

5 read(head,foot);

6 cock:=1;

7 while cock:=head do

Error=>(7,14)|错误20：缺少二元逻辑运算符！(20)

8 begin

9 rabbit:=head-cock;

10 if cock\*2+rabbit\*4=foot

11 begin

Error=>(11,9)|错误16：if后没有then！(16)

12 write(cock,rabbit);

13 n:=n+1

14 end;

15 cock:=cock+1;

16 end;

17 if n=1-1 then write(1-1,1-1)

18 end.

### 2．测试结果分析

**2.1（正确）**

本程序主要测试了过程调用，while循环，同时还有输入输出语句，算术运算等。

程序的功能是输入一个数b，输出c=2\*(10+b)，若b=100结束，结果正确。

**2.2（正确）**

本程序主要测试了多层过程嵌套，常量、变量定义，条件语句，赋值语句，循环语句，输入输出等。

程序功能为打印小于输入数字的素数，结果正确。

**2.3（正确）**

本程序主要测试了多过程调用，复合语句，输出语句，赋值语句，四则运算等。

程序功能为打印两个数字的最小公约数和最大公倍数，结果正确。

**2.4（正确）**

本程序综合测试了常量、变量定义，if-else，递归调用，读写语句，while分支等。

程序功能为递归调用Circle对改变变量的值并打印，结果正确。

**2.5（正确）**

本程序主要测试了四则运算，if-else，复合语句，赋值语句，都写语句等。

程序功能为解决鸡兔同笼问题，输入头、脚的数量，输出鸡、兔的数量，结果正确。

**2.6（错误）**

本程序主要测试声明变量，常量时的错误，以及结尾缺少”.”的错误。

1)Error=>(2,22)|错误27：非法同名标识符(27)

第2行的yyy和第一行yyy在相同作用域同名，这是语义错误；

2)Error=>(39,3)|错误0：非正常结束！(0)

程序最后缺少”.”，这是语法错误。

**2.7（错误）**

本程序主要测试使用未定义的标识符错误以及对常量进行赋值的错误。

1)Error=>(5,8)|错误11：未定义的标识符！(11)

a是未定义的标识符，这是语义错误；

2)Error=>(8,8)|错误36：read语句的标识符应该为变量！(36)

b是一个常量，不能用于read，这是一个语义错误。

**2.8（错误）**

本程序主要测试PL0文法修改部分无符号整数、复合语句的错误。

1)Error=>(1,8)|错误25：无符号整数的首字符应该为非零数字！(25)

045首字符不能为0，这是语法错误；

2)Error=>(11,3)|错误7：说明部分之后应该为复合语句(7)

说明部分之后必须是复合语句，这是语法错误。

**2.9（错误）**

本程序主要测试声明变量，常量时的错误，以及结尾缺少”.”的错误。

1)Error=>(7,6)|错误13：需要赋值号’:=’！(13)

赋值号应该用:=而不是=，这是语法错误；

2)Error=>(13,9)|错误18：需要do(18)

while后面应该跟有 do，这是语法错误。

1. Error=>(22,14)|错误22：需要右括号’)’！(22)

缺少右括号，这是语法错误。

**2.10（错误）**

本程序主要测试声明变量，常量时的错误，以及结尾缺少”.”的错误。

1)Error=>(7,14)|错误20：缺少二元逻辑运算符！(20)

没有”:=”这个二元逻辑运算符，这是语法错误；

2)Error=>(11,9)|错误16：if后没有then！(16)

if后面缺少then，这是语法错误。

## 五．总结感想

这次完成了编译器的课程设计，也是完成了“计算机组成-操作系统-编译-数据库”一个体系的学习，我对计算机底层的工作有了更直观的理解和认识。PL/0编译器的编写也锻炼了我C语言的运用，其中也用到了一些比较精巧的设计，整体来说还算顺利，一些BUG和不妥当的地方渐渐完善，最后得出的成品虽然还不够尽善尽美，但也是非常有成就感的。还有一点比较好的是认真完成编译课程设计，同时也有助于编译原理的考核，真正动手去做去感受，总是会比死记硬背的要强很多。