**北京科技大学 计算机与通信工程学院**

**实 验 报 告**

**实验名称**： 编译原理实验

**学生姓名**： 李启巍

**专 业**： 计算机科学与技术

**班 级**： 计 1404

**学 号**： 41455097

**指导教师**： 班晓娟

**实验成绩**：

**实验地点**： 机电楼304

**实验时间**：\_\_2017\_\_年\_\_\_6\_\_\_月 7 日

# 一、实验目的

实验1：词法分析器

通过本实验的设计更具体的理解词法分析器的工作机制。同时更理解C语言的结构体系。从而更深刻的透析编译原理过程。

实验2：语法分析器

了解掌握算符优先分析的基本方法、内容；

学会科学思考并解决问题，提高程序设计能力。

# 二、实验设备（环境）及要求

实验硬件设备：PC机一台

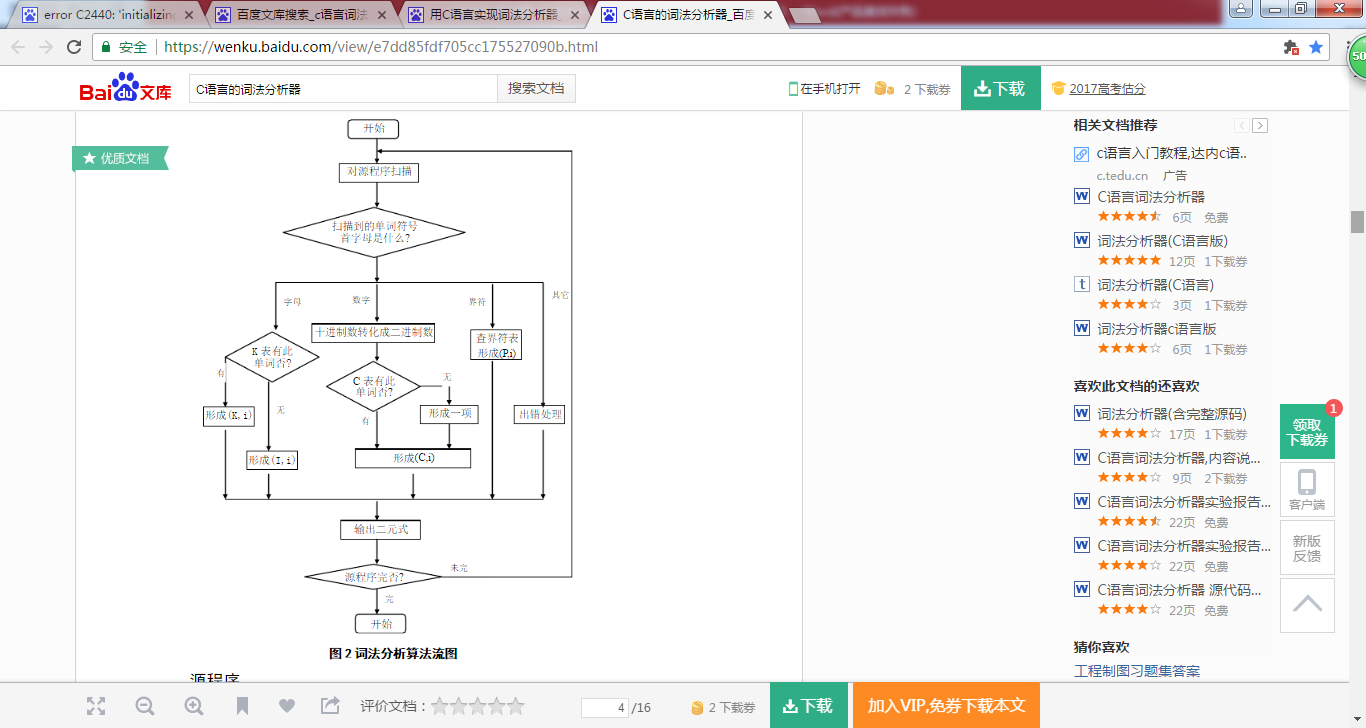
实验软件要求：VC++6.0

# 三、实验原理与内容

1、实验1

**（1）实验原理（算法流程）**

设计c语言的词法分析器，结合状态转换图的原理完成对c语言源程序的基本单词的分析及提取，并设计相应的数据结构保存提取出来的单词。以及对c语言中的保留字的处理策略，实现一个完整的C语言的词法分析器的编写。



**（2）实验内容**

**源代码**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#include <assert.h>

#define LENGTH1 10 //定义保留字的大小

FILE \* fp=NULL; //输出流指针

FILE \* fw=NULL; //输入流指针

char character; //字符

char token[16]; //字符数组，用来存放已读入的字符序列

//编码表

char\* CODE[]={"identifier"/\*标识符\*/,"constant"/\*常数\*/,"keyword"/\*保留字\*/,"+","-", "\*","/","<","<=",">",">=","!=","==","=","(",")",",",":",";","{","}"};

//保留字表

char \*k[]={"for","while","do","else","if","static","int","sizeof","break","continue"};

//标识符结构体

typedef struct { char \* I[256]; //标识符数组

int len;//标识符数量

}

identifier;

//常量结构体

typedef struct { int cont[300];//存放常量的数组

int len;//常量的数目

}

constnumber;

//读入一个字符，从输入流中读入一个字符到变量character中。

void getNextChar(FILE \*ifp)

{ if((character=getc(ifp))==EOF) exit(1); } //读入非空白字符，检查变量character中的字符是否为空白字符或回车或换行符。若是,则调用getNextChar()读入下一个字符，直到character中的字符满足条件.

void getnbc(FILE \*ifp)

{

while(character==' '|| character=='\n'||character==9)

{

getNextChar(ifp);

}

} //连接字符串，把character中的字符连接到token数组的结尾。

void concat()

{

char \* ct=&character;

strcat(token,ct);

} //判断是否为字母。

int letter()

{

return isalpha(character);

} //判断是否为数字

int digit()

{

return isdigit(character);

} //回退字符，将刚读入的character中的字符回退到输入流中。并把character中的值置为空。

void retract(FILE \*ifp)

{

(ifp->\_cnt)++;

(ifp->\_ptr)--;

character=' ';

} //处理保留字，对存放在token中的字符串查保留字，若查到，则返回该保留字的类别编码，否则返回0.

int reserve(char \*\*k)

{

int i;

for(i=0;i<LENGTH1;i++)

if(strcmp(token,k[i])==0)

return i+1;

return 0;

} //处理标识符，对存放在token中的字符串查找符号表，若查到，则返回它在符号表的位置， //存入常数表中，并返回它在常数表中的位置编号。

int symbol(identifier \* id)

{

int i;

for(i=0;i<id->len;i++)

if(strcmp(token,id->I[i])==0)

return i+1;

if(id->len>256)

assert(0);

id->I[id->len]=token;

id->len++;

return id->len;

}

//将数字字符串转化为整数。

int strtonumber()

{

int i;

int sum=0;

for(i=0;i<strlen(token);i++)

{

sum=10\*sum+(token[i]-'0');

}

return sum;

}

//常数存入常数表的函数，将token中的数字串（实际上是字符串），转化成标准的二进制值（整数值）

//存入常数表中，并返回它在常数表中的位置编号。

int constant(constnumber \* con)

{

con->cont[con->len]=strtonumber();

con->len++;

return con->len;

} //将整数值转化为字符串

char \* numbertoString(int num)

{

char s[3];

int i=num/10;

while(i>0)

{

char c=i+'0';

strcat(s,&c);

}

return s;

} //将结果写入到文件并且输出到屏幕。

void returntofile(int num,int val,identifier \*id,constnumber \*con)

{

int i;

int \_num=num;

char c;

c='('; putc(c,fw);

printf("%c",c);

i=\_num/10;

while(i>0)

{

\_num=\_num-10\*i;

c=(i+'0');

printf("%c",c);

putc(c,fw);

i=\_num/10;

}

c=\_num+'0';

printf("%c",c);

putc(c,fw);

printf(",");

putc(',',fw); //如果是标识符或常数 则放入括号内。

if(num==1) //处理标识符

{

printf("%s",id->I[val-1]);

printf(")"); printf("\n");

fputs(id->I[val-1],fw);

putc(')',fw);

putc('\n',fw);

}

if(num==2) //处理常数

{

\_num=con->cont[val-1];

i=\_num/10;

while(i>0)

{

\_num=\_num-10\*i;

c=(i+'0');

printf("%c",c);

putc(c,fw);

i=\_num/10;

}

c=\_num+'0';

printf("%c",c);

printf(")");

printf("\n");

putc(c,fw);

fputs(" ",fw);

putc('#',fw);

fputs(k[val-1],fw);

putc('#',fw);

putc('\n',fw); }

if(num>3) //处理界符

{

printf("-");

printf(")");

printf(" ");

printf("#");

printf("%s",CODE[num-1]);

printf("#");

printf("\n");

putc('-',fw);

putc(')',fw);

fputs(" ",fw);

putc('#',fw);

fputs(CODE[num-1],fw);

putc('#',fw);

putc('\n',fw); } } //将错误写入到文件或输出到屏幕

void error()

{

printf("(ERROR,");

printf("%c",character);

printf(")");

printf("\n");

fputs("(ERROR,",fw);

putc(character,fw);

putc(')',fw);

putc('\n',fw);

} //词法分析函数

void LexAnalyze(char \*\*k,char \*\*CODE,identifier \*id,constnumber \*con,FILE \*fp,FILE \*fw)

{

int num,val;

strcpy(token,"");

getNextChar(fp);

getnbc(fp);

switch(character)

{

case 'a':

case 'b':

case 'c':

case 'd':

case 'e':

case 'f':

case 'g':

case 'h':

case 'i':

case 'j':

case 'k':

case 'l':

case 'm':

case 'n':

case 'o':

case 'p':

case 'q':

case 'r':

case 's':

case 't':

case 'u':

case 'v':

case 'w':

case 'x':

case 'y':

case 'z':

case 'A':

case 'B':

case 'C':

case 'D':

case 'E':

case 'F':

case 'G':

case 'H':

case 'I':

case 'J':

case 'K':

case 'L':

case 'M':

case 'N':

case 'O':

case 'P':

case 'Q':

case 'R':

case 'S':

case 'T':

case 'U':

case 'V':

case 'W':

case 'X':

case 'Y':

case 'Z':

while(letter()||digit())

{

concat();

getNextChar(fp);

}

retract(fp);

num=reserve(k);//保留字

if(num!=0)

returntofile(3,num,id,con);

else

{

val=symbol(id);

returntofile(1,val,id,con);

}

break;

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

while(digit())

{

concat();

getNextChar(fp);

} retract(fp);

val=constant(con);

returntofile(2,val,id,con);

break;

case '<':

getNextChar(fp);

if(character=='=')

returntofile(9,0,id,con);

else

{

retract(fp);

returntofile(8,0,id,con);

}

break;

case '>':

getNextChar(fp);

if(character=='=')

returntofile(11,0,id,con);

else

{

retract(fp);

returntofile(10,0,id,con);

}

break;

case '=':

getNextChar(fp);

if(character=='=')

returntofile(13,0,id,con);

else

{

retract(fp);

returntofile(14,0,id,con);

}

break;

case '!':

getNextChar(fp);

if(character=='=')

returntofile(12,0,id,con);

else

error();

break;

case '+':

returntofile(4,0,id,con);

break;

case '-':

returntofile(5,0,id,con);

break;

case '\*':

returntofile(6,0,id,con);

break;

case '/':

returntofile(7,0,id,con);

break;

case '(':

returntofile(15,0,id,con);

break;

case ')':

returntofile(16,0,id,con);

break;

case ',':

returntofile(17,0,id,con);

break;

case ':':

returntofile(18,0,id,con);

break;

case ';':

returntofile(19,0,id,con);

break;

case '{':

returntofile(20,0,id,con);

break;

case '}':

returntofile(21,0,id,con);

break;

default:

error();

}

}

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

{ //初始化标识符和常数结构体

identifier\* id=(identifier\*)malloc(sizeof(identifier));

constnumber \* con=(constnumber\*)malloc(sizeof(constnumber));

con->len=0;

id->len=0;

argc=3;

argv[1]="E:\\file1.txt";//待分析的文件

argv[2]="E:\\file2.txt";//保存分析结果的文件

//从打开目标文件流

if((fp=fopen(argv[1],"r"))==NULL)

{

printf("cat: can't open %s\n",\*argv);

return 1; } //打开要写二元式的文件流

if((fw=fopen(argv[2],"w"))==NULL)

{

printf("cat:can't open %s\n",argv[2]);

return 1;

}

while(!feof(fp))

{

LexAnalyze(k,CODE,id,con,fp,fw);//执行词法分析

} //关闭流

fclose(fp);

fclose(fw);

return 0;

}

2、实验2

1. 实验原理

语法分析的任务: 把单词符号作为基本单位,分析程序是否为合法的程序.

算符优先分析法是自下而上的语法分析方法，即根据文法,对输入字串进行归约,若能正确地归约为文法的初始符号,则表示输入字串是合法的.

该试验主要研究对输入的赋值语句、输出语句、清除语句进行词法分析、语法分析、表达式求值并存储于指定变量中；若存在错误，提示错误相关信息。

1. 实验内容

#include <stdio.h>

#include <string.h>

char prog[800], token[20];

char ch;

int syn,p,m,n,sum;

char \* rwtab[6]= {"begin","if","then","while","do","end"};

main()

{

p=0;

printf("\n 请输入字符串: \n");

do{

ch=getchar();

prog[p++]=ch;

}while (ch!='#');

p=0;

do{

scaner();

switch(syn)

{

case 11: printf("\n(%d,%d)",syn,sum); break;

case -1: printf("\n(%s,出错!)",token);break;

default: printf("\n(%d,%s)",syn, token);

}

}while (syn!=0);

printf("\n 分析成功 \n");

p=0;

scaner();

E();

if (syn==0)

printf("\n 没有错误. \n");

else printf("\n 出现错误. \n");

}

scaner()

{

for (n=0;n<20;n++) token[n]=NULL;

m=0;

sum=0;

ch=prog[p++];

while (ch==' ') {ch=prog[p++];}

if (ch>='a'&& ch<='z')

{while (ch>='a'&& ch<='z'||ch>='0' && ch<='9')

{

token[m++]=ch;

ch=prog[p++];

}

syn=10;p--;

for (n=0;n<6;n++)

if(strcmp(token,rwtab[n])==0) {syn=n+1;break;}

}

else

if(ch>='0' && ch<='9')

{while (ch>='0' && ch<='9') {sum=sum\*10+(ch-'0'); ch=prog[p++];}

syn=11;p--;

}

else

switch(ch)

{

case '<': token[m++]=ch;

ch=prog[p++];

if (ch=='>') {syn=21;token[m++]=ch;}

else if (ch=='=') {syn=22;token[m++]=ch;}

else {syn=20;p--;}

break;

case '>': m=0; token[m++]=ch;

ch=prog[p++];

if (ch=='='){syn=24;token[m++]=ch;}

else {syn=23;p--;}

break;

case ':': m=0; token[m++]=ch;

ch=prog[p++];

if (ch=='='){syn=18;token[m++]=ch;}

else {syn=17;p--;}

break;

case '+': syn=13;token[0]=ch;break;

case '-': syn=14;token[0]=ch;break;

case '\*': syn=15;token[0]=ch;break;

case '/': syn=16;token[0]=ch;break;

case '=': syn=25;token[0]=ch;break;

case ';': syn=26;token[0]=ch;break;

case '(': syn=27;token[0]=ch;break;

case ')': syn=28;token[0]=ch;break;

case '#': syn=0; token[0]=ch;break;

default: syn=-1;token[0]=ch;

}

}

E()

{printf("E ");

T();

E1();

}

E1()

{printf("E1 ");

if (syn==13) {

scaner();

T();

E1();

}

else {

if (syn!=28 && syn!=0) error();

}

}

T()

{printf("T ");

F();

T1();

}

T1()

{printf("T1 ");

if (syn==15) {

scaner();

F();

T1();

}

else {

if (syn!=28 && syn!=0 && syn!=13) error();

}

}

F()

{printf("F ");

if (syn==27) {

scaner();

E();

if(syn==28) scaner();

else error();

}

else if (syn==11 || syn==10) scaner();

}

error()

{

printf("\n (%d,%s)错误地方! \n",syn, token);

}

# 四、实验结果与分析

实验1

要分析的C语言程序（1）

int i=0,sum=0;

while(i<10)

{

um=sum+i;

i=i+1;

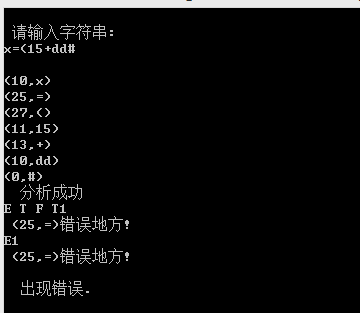
}

**结果：**

(3,-) #int# (1,i) (14,-) #=# (2,0) (17,-) #,# (1,sum) (14,-) #=# (2,0) (19,-) #;# (3,-) #while# (15,-) #(# (1,i) (8,-) #<# (2,10) (16,-) #)# (20,-) #{# (1,sum) (14,-) #=# (1,sum) (4,-) #+# (1,i) (19,-) #;# (1,i) (14,-) #=# (1,i) (4,-) #+# (2,1) (19,-) #;# (21,-) #}#

实验2

**结果**



**五、结论与讨论**

这次的实验对我的帮助非常大，让我对词法分析以及语法分析进行实践，实验进行的非常成功，通过两上机让我对上课所学到的东西有了更加深刻的理解，同时通过两次的编程实践让我的编程能力有了一定的提高。感谢这门课程对我的帮助。

**六、教师评审**

|  |  |
| --- | --- |
| **教师评语** | **实验成绩** |
| 签名：  日期： |  |