**编译原理实验报告—词法分析**

|  |  |
| --- | --- |
| 姓名 | 张志伟 |
| 专业/班级 | 计算机一班 |
| 学号 | 320160940531 |

# 一、实验目的

设计、编制并测试一个词法分析程序，加深对词法分析原理的理解。

# 二、实验原理

1. **词法分析器原理**

将文件输入的字符流，通过语法分析器切割为一个个的单词（记号流），用作语法分析的输入。

1. **词法分析主程序**

主程序示意图如图1所示。

读入文件

调用扫描子程序得到单词节点

分析得到的结点

是

存入词串节点

否

输入结束

输出分析结果

图1 词法分析主程序示意图

1. **词法分析子程序**

文件输入字符

忽略空格和注释等

是否文件结束

是

返回

运算符,界符等符号

否

字母

数字

得到符号（分隔符/运算符）

组装成单词结点

单词节点分析

返回

拼数

拼字符串

图2 词法分析程序流程

# 三、实验内容

1. 源文件结构:

main.c 词法分析器主程序

Lex\_Head.h 词法分析头文件

Table.h 编译器表文件

Lex\_Function.c 词法分析定义文件

Table.c 表操作函数定义文件

1. 数据结构

//词串结点

struct WordNode

{

int type;//单词类型

int Varnum;//单词在相关表中的编号

int number;//单词在词串中的编号

char\* Word;//单词

double value;//变量或常量的值

struct WordNode\* next;//下一结点指针

int row;//行号

};

//词串头

struct WordHead

{

struct WordNode \* first;//词串第一个单词结点

struct WordNode \* last;//词串最后一个单词结点

int nodecount;//词串单词计数

int rwo;//行号

};

//关键字表

static char\* Key[32] =

{

"auto", "break", "case", "char", "const", "continue",

"default", "do", "double", "else", "enum", "extern",

"float", "for", "goto", "if", "int", "long",

"register", "return", "short", "signed", "sizeof", "static",

"struct", "switch", "typedef", "union", "unsigned", "void",

"volatile", "while"

};

//分隔符表

static char\* Sparator[15] =

{

"[", "]", "{", "}", ";", "\"", "\'", "#", "(", ")", ".", ":", ",",

"\\", "\?"

};

//运算符表

static char\* Operator[26] =

{

"+", "-", "\*", "/", "<", ">", "=", "^", "&", "|", "%", "~","!",

">>", "<<", "++", "--", "&&", "||", "!=", "==", "+=", "-=", "<=", ">=", "->"

};

//变量表结点

struct VariableNode

{

int index;//索引

double Value;//值

char\* name;//变量名

int Type;//变量类型

struct VariableNode\* next;//下一结点

int keeper;//保留

};

1. 主函数介绍

int main()

{

FILE\* filepointer;//文件指针

struct WordHead Head;//词串链表头

struct WordNode\* node;//词串链表结点

Head.first = Head.last = NULL;

Head.keeper = Head.nodecount = 0;//初始化

filepointer = openfile();//打开文件

while (!feof(filepointer))//在读取到文件尾之前,循环获取单词

{

node = getWord(filepointer);//读取一个单词节点

if(!node)break;

AnalyseWord(node);//分析单词

addNode(&Head, node);//将当初添加到词串链表

}

fclose(filepointer);//关闭文件

OutPutNode(&Head,fopen("LEX\_List.txt","w"));//输出词法分析结果

OutPutVar(fopen("VAR\_Table.txt","w"));//输出变量表

return 0;

}

1. 子函数介绍

//扫描子程序，单词读取函数 读取一个单词 输出 词串结点

struct WordNode\* getWord(FILE\* filepointer)

{

char in = 0;//暂存读入字符

char word[127];//单词

int i = 0;//单词中字符指针计数

int Type = 0;//单词类型

int Value = 0;//常量值

word[i++] = in = getc(filepointer);//读取第一个字符

//根据第一个字符判断单词可能的类型

//文件结束

if(in==EOF)return NULL;

//空格,返回下一个单词

if (in==' ')

{ return getWord(filepointer); }

//换行符

if (in==10)

{ rowcount++;return getWord(filepointer); }

//可能是注释语句

if (in == '/')

{

in = getc(filepointer);//读取一个字,符进一步判断

//行注释

if (in == '/')

{

while ((in = getc(filepointer)) != EOF)//抛弃本行所有内容

{

if (in == 10)//直到读取到换行符

{ rowcount++;return getWord(filepointer); }//抛掉注释后,返回一个单词

}

}

//段注释

else if (in == '\*')

{

while ((in = getc(filepointer)) != EOF)//抛掉注释段内所有内容

{

if(in==10)rowcount++;

if (in == '\*')//遇到 \* 符号,可能是段注释结束

{

if ((in = getc(filepointer)) == '/' )//再读取一个一个字符,进一步判断是否结束

{

return getWord(filepointer);//抛掉注释后,返回一个单词

}

else ungetc(in,filepointer);//段注释未结束,放回字符----防止\*\*/结束的情况

}

}

}

//不是注释,放回字符.恢复状态

else

{

ungetc(in, filepointer);

in = word[i - 1];

}

}

//是标识符/关键字/常量 isalnum 数字字母判断 isalpha英文字母判断

if (isalnum(in) || in == '\_')

{

//标识符/关键字

if (in == '\_' || isalpha(in))

{

Type = VARIABLE;//假设是标识符

while ((in = getc(filepointer)) != EOF)//读取到标识符结束

{

if (isalnum(in) || in == '\_')//可能是标识符/关键字的组成

{

word[i++] = in;

continue;

}

else//结束,非(数字/字母/'\_')

{

ungetc(in, filepointer);

break;

}

}

}

//常量

else

{

Type = VALUE;

Value = in - '0';//计算值

while ((in = getc(filepointer)) != EOF)

{

if (isdigit(in))//读取以后所有连续数字

{

Value = Value \* 10 + in - '0';//计算值

continue;

}

else//遇到非数字字符

{

ungetc(in, filepointer);

break;

}

}

}

}

//分隔符/运算符

else

{

Type = SPARATOR;//假设是分隔符

while ((in = getc(filepointer)) != EOF)

{

if (isDoubleOpe(word[i - 1], in))//判断是否为双字符分隔符/运算符

{

word[i++] = in;

break;

}

else

{

ungetc(in, filepointer);

break;

}

}

}

//将读取到的单词组装成词串结点

struct WordNode\* node = malloc(sizeof(struct WordNode));

node->type = Type;//单词类型

node->row=rowcount;

node->number = WordCount++;//词串计数加一

node->Varnum = 0;//单词在表中的序号

node->next = NULL;//下一结点

word[i++] = '\0';

node->Word = malloc(i \* sizeof(char));

strcpy(node->Word, word);

node->value = Value;

return node;//返回结点

}

//单词分析函数 进一步分析单词类型 将变量写到表中

void AnalyseWord(struct WordNode\* node)

{

if (node->type == VARIABLE)

{

if ((node->Varnum = Search(node, KEY)) != -1)

{ node->type = KEY; }

else { node->Varnum = SearchVar(node); }

}

else

if (node->type == SPARATOR)

{

if ((node->Varnum = Search(node, OPERATOR)) != -1)

{

node->type = OPERATOR;

}

else { node->Varnum = Search(node, SPARATOR); }

}

}

# 四、实验过程演示记录

真确测试用例

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i=2;

i=i++;

//行注释

printf("Hello World!!!\n");

/\*

段注释

\*/

return 0;

}

错误测试用例

#include <stdio.h>

#include <stdlib.h>

int main()

{

innt i=2;///int 多一个 n

i=i+;///运算符错误

//行注释

printf("Hello World!!!\n") ///缺少分号

/\*

段注释

\*/

return 0;

}

# 五、实验结果及分析

变量表

类型 序号 值 变量名

4 10 0.000000 n

4 9 0.000000 World

4 8 0.000000 Hello

4 7 0.000000 printf

4 6 0.000000 i

4 5 0.000000 main

4 4 0.000000 stdlib

4 3 0.000000 h

4 2 0.000000 stdio

4 1 0.000000 include

词串表

序号 行号 类型 表序号 单词

0 1 分隔符 7 #

1 1 标识符 1 include

2 1 运算符 4 <

3 1 标识符 2 stdio

4 1 分隔符 10 .

5 1 标识符 3 h

6 1 运算符 5 >

7 2 分隔符 7 #

8 2 标识符 3 include

9 2 运算符 4 <

10 2 标识符 4 stdlib

11 2 分隔符 10 .

12 2 标识符 3 h

13 2 运算符 5 >

14 4 关键字 16 int

15 4 标识符 5 main

16 4 分隔符 8 (

17 4 分隔符 9 )

18 5 分隔符 2 {

19 6 分隔符 -1

20 6 关键字 16 int

21 6 标识符 6 i

22 6 运算符 6 =

23 6 常 量 0 2.000000

24 6 分隔符 4 ;

25 7 分隔符 -1

26 7 标识符 6 i

27 7 运算符 6 =

28 7 标识符 6 i

29 7 运算符 15 ++

30 7 分隔符 4 ;

31 8 分隔符 -1

32 9 分隔符 -1

33 9 标识符 7 printf

34 9 分隔符 8 (

35 9 分隔符 5 "

36 9 标识符 8 Hello

37 9 标识符 9 World

38 9 运算符 12 !

39 9 运算符 12 !

40 9 运算符 12 !

41 9 分隔符 13 \

42 9 标识符 10 n

43 9 分隔符 5 "

44 9 分隔符 9 )

45 9 分隔符 4 ;

46 10 分隔符 -1

47 13 分隔符 -1

48 13 关键字 19 return

49 13 常 量 0 0.000000

50 13 分隔符 4 ;

51 14 分隔符 3 }

# 六、总结

词法分析是计算机科学中将字符序列转换为单词序列的过程。词法分析器一般以函数的形式存在，供语法分析器调用，是编译过程的第一个阶段，是编译的基础。

本词法分析器任存在许多不足：

1、字符、字符串常量识别

2、关键字拼写错误检查

3、非法字符输入等