语言词法分析器实验报告

班级：2009211315 学号：09211541姓名：李根

1. **实验目的：**

通过C++实现词法分析器，更加深入的掌握词法分析；同时也可以更加了解词法分析的原理。

1. 实验环境

PC微机；

Windows操作系统；

本次实验采用visual C++程序集成环境，采用C语言编写。

1. **实验过程：**

**1、词法分析过程**：

根据输入单词的第一个有效字符（有时还需读第二个字符），判断单词类，产生类号。对于标识符和常数，需分别与标识符表和常数表中已登记的元素相比较，比较之后输出结果。

Main函数中设计一个计数器，用于，语句，单词数，字符个数的计数

**3、实验过程中的问题**：

一开始实验很生疏，很难把理论知识融汇到编程上，后来通过多次看PPT和书本内容，渐渐把握到诀窍；LEX软件工具用的不熟，所以这道题我采用C语言来写

和以前编其他东西很不一样，一开始不知道如何下手；通过网上的成熟代码学习了下。

**4、实验代码：**

//头文件

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#define LEN sizeof(struct Node)

//定义动态链表数据结构

struct Node{

char data;

struct Node \*next;

};

void output(struct Node \*);//扫描链表缓冲区函数声明

void scaner(); //词法分析函数声明

void getbc(); //若ch中是空白字符，则不停调用getch()直到读入的不是空白字符为止函数的声明

void getch(); //从缓冲区读入一字符函数的声明

void concat(); //将ch中的字符连接到token的后面函数的声明

int letter(char ch); //判断ch中的是否是字母函数声明

int digit(char ch); //判断ch中的是否是数字函数声明

int reserve(); //判断token中的字符串是否是关键字或是标识符函数声明

void retract(); //指针回退一个字符函数的声明

struct Node \* ReturnPoint(struct Node \*p); //返回链表指针（即返回符号表入口指针）函数声明

void error(); //错误处理函数声明

void count(); //非字母数字子符数的统计函数声明

struct Node \*head,\*p;

char ch; //全局变量，存放当前字符

char \*key[]={"auto","break","case","char","const","continue","default","do","double","else","enum","extern","float","for","goto","include","if","int","long","main","pause","printf","register","return","scanf","short","signed","static","sizeof","struct","switch","system","typedef","union","unsigned","void","volatile","while","Bool","Complex","Imaginary","inline","restrict"};//关键字表

char token[20];//字符数组，存放构成单词的字符串

int num=0;

//主函数

int main()

{

int k=0,h=0,word=0;

head=(struct Node \*)malloc(LEN); //分配头节点存储空间

if(!head) { printf("error.\n"); exit(1); }

head->next=NULL;

head->data=' ';

p=head;

printf("When input a '@' at the beigining of an line,this programe will be over.\n");

printf("The programe will output what you've already inputed.\nPlease input your codes:\n"); //显示提示信息

while(1) //不断以行为单位读入源代码，直到@在行首出现源程序输入结束，跳出本循环

{

int i=0;

char temp[255];//每行长度不超过256个字符

gets(temp); //以行为单位读入源程序

if(temp[0]=='@') break;//当输入的第一个字符为$时表示输入源代码结束

k++; //源程序行数统计变量

p->next=(struct Node \*)malloc(LEN);

if(!(head->next)) { printf("error.\n"); exit(1); }

p=p->next;

while((temp[i]!='\0') && (i<256)) //将输入的代码以行为单位存入缓冲区

{

p->data=temp[i];

p->next=(struct Node \*)malloc(LEN);

if(!(p->next)) { printf("error.\n"); exit(1); }

p=p->next;

i++; //判断每行字符结束并不超过行边界的局部变量

++h; //除回车外的全文字符总数统计变量

}

p->data='\n';

p->next=NULL; //尾结点

}

output(head); //扫描缓冲区，输出结果

p=head->next; //p指针重新指向头字符所在结点

while(p->next!=NULL)

{

word++;

scaner(); //词法分析

}

printf("源程序共有%d行。\n",k);

printf("除注释外，源程序共有%d个单词。\n",word-num);

printf("源程序共有%d个字符。\n",h+k);//输出显示提示信息

system("pause");

return 0;

}

//扫描链表缓冲区函数定义

void output(struct Node \*head)

{

if(!head) { printf("error.\n"); exit(1); }

p=head->next;//指向本行C语言代码第一个字符

while(p->next!=NULL) //循环直到本行打印结束

{

printf("%c",p->data);

p=p->next; //从头至尾遍历链表

}

printf("\n"); //显示最后一个回车

}

//词法分析函数定义

void scaner()

{

int c,j=0;

for(j=0;j<20;j++)

token[j]='\0'; //将token清空

getch(); //读入一个字符

getbc(); //读入一个单词

if(letter(ch))//处理字符

{

while((letter(ch)) || (digit(ch)))//以字母开头的字母数字串

{

concat(); //字符放入token数组中

getch(); //指针再往后读一个字符

}

retract(); //当出现除字母、数字外的字符时，指针回退一步

c=reserve(); //字母数字串与保留字匹配

if(c!=50)

printf("(%s,-)\n",key[c]); //输出保留字记号

else

{

printf("(%s,%x)\n",token,ReturnPoint(p));

/\* int m=0; //输出标示符记号及其属性（符号表入口指针）

printf("(");

for(m=0;token[m]!='\0';m++)

printf("%c",token[m]);

printf(",-)\n"); \*/

}

}

else if(digit(ch)) //处理数字开头的串

{

while(digit(ch))//数字串则依次放入token数组中

{

concat(); //字符放入token数组中

getch(); //指针再往后读一个字符

}

retract(); //当出现除字母、数字外的字符时，指针回退一步

printf("(num,%d)\n",atoi(token)); //输出数字的记号及其属性

}

else{ //处理字母、数字外的特殊符号

count();

switch(ch){

case'+': getch(); //需要超前扫描

if(ch=='+')

printf("(++,-)\n");

else if(ch=='=')

printf("(+=,-)\n");

else

{

retract(); //指针回退一步

printf("(+,-)\n");

}

break;

case'-': getch(); //需要超前扫描

if(ch=='-')

printf("(--,-)\n");

else if(ch=='=')

printf("(-=,-)\n");

else if(ch=='>')

printf("(->,-)\n");

else

{

retract(); //指针回退一步

printf("(-,-)\n");

}

break;

case'\*': if(ch=='=')

printf("(\*=,-)\n");

else

{

retract(); //指针回退一步

printf("(\*,-)\n");

}

break;

case'/': getch(); //需要超前扫描

if(ch=='/')

{

getch();

printf("注释为：");

while(ch!='\n')

{

printf("%c",ch);

getch();

}

printf("\n");

}

else if(ch=='\*')

{

getch();

printf("注释为：");

while(ch!='\*')

{

printf("%c",ch);

getch();

}

printf("\n");

getch(); //过滤掉注释结束符

}

else if(ch=='=')

{

printf("(/=,-)\n"); break;

}

else

{

retract(); //指针回退一步

printf("(/,-)\n");

}

break;

case'%': getch();

if(ch=='=')

printf("(%=,-)\n");

else

{

retract(); //指针回退一步

printf("(%,-)\n");

}

break;

case'<': getch(); //需要超前扫描

if(ch=='=')

printf("(relop,LE)\n"); //关系运算符

else

{

retract(); //指针回退一步

printf("(relop,LT)\n"); //关系运算符

}

break;

case'>': getch(); //需要超前扫描

if(ch=='=')

printf("(relop,GE)\n");//关系运算符

else

{

retract(); //指针回退一步

printf("(relop,GT)\n"); //关系运算符

}

break;

case'=': getch(); //需要超前扫描

if(ch=='=')

printf("(relop,EQ)\n");//关系运算符

else

{

retract(); //指针回退一步

printf("(assign-op,-)\n");//赋值运算符

}

break;

case'!': getch();

if(ch=='=')

printf("(relop,NE)\n");//关系运算符

else

{

retract(); //指针回退一步

printf("(logic,NOT)\n");//逻辑运算符

}

break;

case'&': getch();

if(ch=='&')

printf("(logic,AND)\n");//逻辑运算符

else

{

retract(); //指针回退一步

printf("(&,-)\n");

}

break;

case'|': getch();

if(ch=='|')

printf("(logic,OR)\n");//逻辑运算符

else

{

retract(); //指针回退一步

printf("(|,-)\n");

}

break;

//case'\': printf("(\,-)\n");break;

case'\n': printf("(enter-op,-)\n");break;

//标点符号

case';': printf("(;,-)\n");break;

case':': printf("(:,-)\n");break;

case'{': printf("({,-)\n");break;

case'}': printf("(},-)\n");break;

case'(': printf("((,-)\n");break;

case')': printf("(),-)\n");break;

case'[': printf("([,-)\n");break;

case']': printf("(],-)\n");break;

case'.': printf("(.,-)\n");break;

case',': printf("(,,-)\n");break;

case'?': printf("(?,-)\n");break;

//case'"': printf("("",-)\n");break;

//case''': printf("(',-)\n");break;

//case'\n': printf("(enter-op,-)\n");break;

default: error(); break;

}

}

}

//从缓冲区读入一字符函数的定义

void getch()

{

ch=p->data;

p=p->next;

}

//若ch中是空白字符，则不停调用getch()直到读入的不是空白字符为止函数的定义

void getbc()

{

while(ch==' ')

getch();

}

int letter(char ch) { return isalpha((int)ch);} //强制转换后 利用库函数，判断ch中的是否是字母

int digit(char ch) { return isdigit((int)ch);} //强制转换后 利用库函数，判断ch中的是否是数字

//将ch中字符连接到token后面的函数定义

void concat()

{

unsigned int i;

i=strlen(token); //求字符串长度库函数调用

token[i]=ch;

token[i+1]='\0';

}

//指针回退一个字符函数的定义

void retract()

{

struct Node \*Q;

Q=head->next;

while(Q->next!=p)

Q=Q->next;

p=Q; //指针回退一步操作

}

//判断token中的字符串是否是关键字或是标识符函数的定义

int reserve()

{

int k=0;

for(k=0;k<=42;k++)

if(strcmp(key[k],token)==0) //调用字符串比较库函数

return k; //是关键字

return 50; //不是关键字

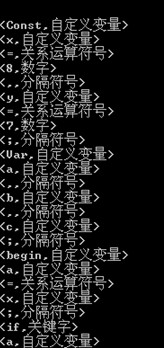
}

struct Node \* ReturnPoint(struct Node \*p) { return p; } //返回链表指针（即返回符号表入口指针）函数定义

void error() { printf("error\n"); } //错误处理函数定义

//非字母数字子符数的统计函数定义

void count() { num++; }



1. **实验心得：**
2. 这次实验让我对词法分析器有了一定的了解，把理论知识应用于实验中。

2、实现功能有限，比如 识别和读取源程序中的注释，在这个程序中没有能够实现，以后会继续改进的。

3．通过此次编译原理词法分析器的实验，激发了我对学习编译原理的积极性，培养了我独立发现问题、思考问题，解决问题的能力。通过与老师和同学的探讨更深刻的掌握了实验原理。

4．体会到了编译原理与其他学科知识之间的融合渗透。将以前学的VC++的知识应用于此次实验，更锻炼了对VC++编程的操作。