**《编译原理》实验报告**

|  |  |
| --- | --- |
| **实验名称** | 一个简单的C--词法分析器 |
| 1. **实验目的：**   设计、编制并调试一个自定义语言C--的词法分析程序，加深对词法分析原理的理解。 | |
| 1. **实验过程及步骤：** 2. 将需要分析的源程序字符串从.txt文件中读出并去除回车和空格 3. 构造一个结构体Word，Word的属性有 char type[50] 和 char w[20]，分别用来存放预分析词的类型和词本身。将读出的字符串或者字符一次存入Word 的 w 属性中。 4. 构造分析数组，将词法记号和含义定义一一对应 5. 依次分析Word 中的w 属性，和分析数组一一比对，将比对结果存入相应的type属性中 6. 如果检测到w是一个数字，则继续检测是八进制或者十进制或者十六进制，将判断出来的结果存入type属性 7. 如果检测到w是单个字符，则继续检测是id还是ch   **程序源代码如下：**  #include<stdio.h>  #include<stdlib.h>  #include<string.h>  typedef struct Word  {  char type[50];//词所属类型  char w[20];//词  } Word;  int main()  {  char ch,word\_temp[20]="";  int i=0,j=0,key=0;  char ktt[38][20]= {"int","char","void","if","else","switch","case","default","while","do","for","break","continue","return",  "+","-","\*","/","%","++","--","!","&&","||","=",">",">=","<","<=","==","!=",  ",",":",";","(",")","{","}"  };  char ktt1[38][20]= {"kw\_int","kw\_char","kw\_void","kw\_if","kw\_else","kw\_switch","kw\_case","kw\_default","kw\_while","kw\_do","kw\_for","kw\_break","kw\_continue","kw\_return",  "add","sub","mul","div","mod","inc","dec","not","and","or","assign","gt","ge","lt","le","equ","nequ",  "comma","colon","simcon","lparen","rparen","lbrac","rbrac"  };  FILE \*fp;  Word word[500];  fp = fopen("D:/1.c","r");  if(!fp)  {  printf("can't open file\n");  exit(1);  }  while((ch=fgetc(fp))!= EOF)  {  putchar(ch);  if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')||(ch>='0'&&ch<='9'))  {  word\_temp[key]=ch;//连续几个字母的连成单词  word\_temp[key+1]='\0';  key++;  continue;  }  if(strcmp(word\_temp,"")!=0)  {  strcpy(word[i].w,word\_temp);//将单词拷贝到结构数组中  strcpy(word\_temp,"");  key=0;//回到临时数组的开始位置  i++;//结构数组的下标加1  continue;  }  else if(ch==' '||ch==10||ch==' ') //去除空格，回车和tab  {  continue;  }  else  {  word\_temp[0]=ch;  word\_temp[1]='\0';  strcpy(word[i].w,word\_temp);//将非字母数字符号拷贝到结构数组中  strcpy(word\_temp,"");  key=0;//回到临时数组的开始位置  i++;  }  }  printf("\n词法分析结果如下:\n");  int k = 0;  for(j=0; j<i; j++)  {  for(k=0; k<38; k++)  {  if((strcmp(word[j].w,ktt[k]))==0)  {  if(k>=0&&k<14)  strcpy(word[j].type,ktt1[k]);//关键字  else if(k>=14&&k<31)  strcpy(word[j].type,ktt1[k]);//运算符  else if(k>=31&&k<38)  strcpy(word[j].type,ktt1[k]);//分界符  break;  }  else if(word[j].w[0]>='0'&&word[j].w[0]<='9')  {  if(word[j].w[0] == '0')  {  strcpy(word[j].type,"八进制数字");  if(word[j].w[1] == 'x')  {  strcpy(word[j].type,"十六进制数字");  }  }  else  {  strcpy(word[j].type,"十进制数字");  }  }  else if(strlen(word[j].w) == 1)  {  strcpy(word[j].type,"ch");//字符  }  else  {  strcpy(word[j].type,"str");//字符串  }  }  }  for(j=0; j<i; j++)  {  printf("(%s,'%s')\n",word[j].type,word[j].w);  }  fclose(fp);  return 0;  }  **测试程序如下：**  int main()  {  int i = 232;  i = 057;  i = 0x25aF;  string s = "str";  char a ='a';  printf ( "hello");  } | |
| 1. **实验结果：** | |
| 1. **实验总结：**   本次实验，在很大程度上，提高了我对编译程序的理解。程序实现了词法分析的功能，能对用户输入的程序段进行词法分析，达到了实验要求。通过实验，加深了对词法分析过程的理解，了解了词法分析程序的工作原理。但该程序有一些不足，比如实现比较简单。刚开始的时候写出来的程序还不能准确的识别八进制和十六进制，幸好后来经过老师的指导和同学的帮助，我改正了这一错误，完善了程序。 | |