**软件学院实践报告（编译原理）**

**课程编号:** 3152100131 **实践课程名称：编译原理 学年：2018-2019 学期：秋**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学生姓名** | | 田宇 | **学号** | 2016212011 |
| **指导教师姓名** | | 修佳鹏 | 完成时间 | 2018年11月20日 |
| **实验名称** | | 词法编译器 | | |
| **实验目的** | 通过对词法的分析，以及状态机的构造知识的学习，编写一个词法编译器，能够简单的分析代码的标识符、数字和分隔符等，并且能够将存在的错误进行简单的提示。 | | | |
| **实验要求** | 1. 根据附录给定的文法，从输入的类 C 语言源程序中，识别出各个具有独立意义的单词，即标识符、运算符、分隔符等 2. 词法分析后可查看输出的符号表 3. 保存符号表和 TOKEN 串表（如：文本文件） 4. 遇到错误时可显示提示信息 | | | |
| **实验环境** | C语言编写  Windows10  Dev C++ | | | |
| **设计方案** | 通过对词法分析器的状态分析，用switch结构对其进行实现，整体上用c代码进行编写。  实现了统计行数、统计词个数，判断数字、判断字符、判断分界符（包含有：. { } [ ] # \\ \ \ \? , ; :) 等）、判断关键字（包含有：break case char const continue default double do else enum extern float for goto if int long return short signed sizeof static struct switch typedef union unsigned void while等）、错误提示以及定位等功能 | | | |
| **成源程序** | /\*  \*词法编译器  \*by 田宇 2018 11  \*  \*通过文件将源程序代码导入进行分析  \*最后将输出的符号表导出到out.txt文件当中  \*  \*/  #include<stdio.h>  #include<stdlib.h>  #include<string.h>  #define MAX 100  FILE \*infile,\*outfile; //stdio.h  char strToken[MAX]; //存放构成单词符号的字符串  char ch; //存放最新读进的源程序字符  int pos=0; //当前字符串的位置指针  int numberOflines=1; //统计行数  int numberOfwords=0; //统计源程序中单词个数,标点和空格不计为单词  int numberOfchars=-1; //统计源程序中字符个数 要去掉EOF  char GetChar()  {  ch = fgetc(infile);  if(ch == EOF)  printf("\n source.txt 的词法分析已完成,分析结果记录在 out.txt\n\n");  numberOfchars++;  return ch;  }  void GetBC()//检查ch中字符是否空白 若是继续调用GetChar()直至非空  {  while(ch==' '||ch=='\t'||ch=='\b'||ch=='\n')  {  if(ch=='\n')  numberOflines++;  GetChar();  }  }  void Concat(char ch) //将ch中的字符连接到strToken后  {  strToken[pos++]=ch;  }    int IsLetter(char ch) //判断当前字符是否为字母  {  if((ch>='a' && ch<='z')||(ch>='A' && ch<='Z'))  return 1;  else return 0;  }    int IsDigit(char ch) //判断当前字符是否为数字  {  if(ch>='0' && ch<='9')  return 1;  else return 0;  }  int IsBound(char ch) //判断当前字符是否为分界符  {  if(ch=='.' || ch=='{' || ch=='}' || ch=='[' || ch==']' || ch=='(' || ch==')' || ch==',' || ch==';' ||  ch=='#' || ch=='\\' || ch=='\'' || ch=='\"' || ch=='\?' || ch==':')  return 1;  else return 0;  }  void Retract() //strToken回退，将ch置空  {  pos--;  ch=' ';  }    int KeyList(char strToken[]) //判断strToken中是否为关键字  {  if(strcmp(strToken,"auto")==0)  return 1;  else if(strcmp(strToken,"break")==0)  return 1;  else if(strcmp(strToken,"case")==0)  return 1;  else if(strcmp(strToken,"char")==0)  return 1;  else if(strcmp(strToken,"const")==0)  return 1;  else if(strcmp(strToken,"continue")==0)  return 1;  else if(strcmp(strToken,"default")==0)  return 1;  else if(strcmp(strToken,"double")==0)  return 1;  else if(strcmp(strToken,"do")==0)  return 1;  else if(strcmp(strToken,"else")==0)  return 1;  else if(strcmp(strToken,"enum")==0)  return 1;  else if(strcmp(strToken,"extern")==0)  return 1;  else if(strcmp(strToken,"float")==0)  return 1;  else if(strcmp(strToken,"for")==0)  return 1;  else if(strcmp(strToken,"goto")==0)  return 1;  else if(strcmp(strToken,"if")==0)  return 1;  else if(strcmp(strToken,"int")==0)  return 1;  else if(strcmp(strToken,"long")==0)  return 1;  else if(strcmp(strToken,"register")==0)  return 1;  else if(strcmp(strToken,"return")==0)  return 1;  else if(strcmp(strToken,"short")==0)  return 1;  else if(strcmp(strToken,"signed")==0)  return 1;  else if(strcmp(strToken,"sizeof")==0)  return 1;  else if(strcmp(strToken,"static")==0)  return 1;  else if(strcmp(strToken,"struct")==0)  return 1;  else if(strcmp(strToken,"switch")==0)  return 1;  else if(strcmp(strToken,"typedef")==0)  return 1;  else if(strcmp(strToken,"union")==0)  return 1;  else if(strcmp(strToken,"unsigned")==0)  return 1;  else if(strcmp(strToken,"void")==0)  return 1;  else if(strcmp(strToken,"volatile")==0)  return 1;  else if(strcmp(strToken,"while")==0)  return 1;  else return 0;  }        main()  {  if((infile=fopen("source.txt","r"))==NULL)  {  printf("Open source file error!\n");  exit(0);  }  if((outfile=fopen("out.txt","w"))==NULL)  {  printf("Open dest file error!\n");  exit(0);  }  GetChar();  GetBC();  while(ch!=EOF)  {  //识别数字  if(IsDigit(ch))  {  pos=0;  while(IsDigit(ch))  {  Concat(ch);  GetChar();  }  //while循环在读到 数字开头+字母时报错  if(IsLetter(ch))  {  strToken[0]='\0';  printf("第%d行有非法字符！继续分析......\n",numberOflines);  while(IsLetter(ch))  GetChar();  }  strToken[pos++]='\0';  if(strToken[0]!='\0')  fprintf(outfile,"%s 数字\n",strToken);  numberOfwords++;  GetBC();  }    //识别关键字、标识符  else if(IsLetter(ch))  {  pos=0;  while(IsLetter(ch)||IsDigit(ch)||ch=='\_')  {  Concat(ch);  GetChar();  }  strToken[pos++]='\0';  if(KeyList(strToken)==1)  fprintf(outfile,"%s 关键字\n",strToken);  else fprintf(outfile,"%s 标识符\n",strToken);    numberOfwords++;  GetBC();  }    //识别运算符(算术运算符 关系运算符 逻辑运算符 赋值运算符 不包括位运算符)  else if(ch=='+') //+,++,+=  {  pos=0;  Concat(ch);  GetChar();  if(ch=='+'||ch=='=')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else  {  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetBC();//此情况下ch不应该用GetChar()  }  }    else if(ch=='-') //-,--,-=  {  pos=0;  Concat(ch);  GetChar();  if(ch=='-'||ch=='=')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else  {  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetBC();  }  }    else if(ch=='\*' || ch=='%' || ch=='!' || ch=='>' || ch=='<' || ch=='=')  //\*,\*=,%,%=!,!=,>,>=,<,<=,=,==  {  pos=0;  Concat(ch);  GetChar();  if(ch=='=')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else  {  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetBC();  }  }    else if(ch=='&') // &&  {  pos=0;  Concat(ch);  GetChar();  if(ch=='&')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else  {  printf("第%d行有非法字符！继续分析......\n",numberOflines);  GetChar();  GetBC();  }  }    else if(ch=='|') // ||  {  pos=0;  Concat(ch);  GetChar();  if(ch=='|')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else  {  printf("第%d行有非法字符！继续分析......\n",numberOflines);  GetChar();  GetBC();  }  }    else if(ch=='/') // /,/=,//  {  pos=0;  Concat(ch);  GetChar();  if(ch=='=')  {  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetChar();  GetBC();  }  else if(ch=='/') //注释的处理(跳过)，只处理//型的注释，不处理/\*\*/型的注释  // //型注释读到本行末尾  {  Retract();  while(ch!='\n'&&ch!=EOF) //可能是最后一行所以考虑EOF  GetChar();  GetBC(); //新加的  }  else  {  strToken[pos++]='\0';  fprintf(outfile,"%s 运算符\n",strToken);  GetBC();  }  }  //识别分界符  else if(IsBound(ch))  {  pos=0;  Concat(ch);  strToken[pos++]='\0';  fprintf(outfile,"%s 分界符\n",strToken);  GetChar();  GetBC();  }  else  {  printf("第%d行有非法字符！继续分析......\n",numberOflines);  GetChar();  GetBC();  }  }    //打印统计结果  printf("\n共计%d行\n",numberOflines);  printf("\n单词个数为%d个(即关键字、标识符和数字个数)\n",numberOfwords);  printf("\n字符个数为%d个(即包括空格、换行、注释中的字符在内的所有字符)\n\n\n",numberOfchars);  system("pause");  return 0;  } | | | |
| **实验结果** |  | | | |
| **实验分析** | 基本功能可以实现并且能够完整的输出错误信息以及符号表 | | | |
| **心得体会** | 在这次实验中，我深切地体会到词法编译器的具体实现是如何进行的，将大量的不同的字符串进行分析是一件辛苦的事情，很难想象早期计算机科学家在对第一个代码程序进行分析时，经历过怎样的苦苦思索。 | | | |