编译原理课程词法分析实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **专业班级** | | **计算机科学与技术05版** | **姓名** | | **龙俊桦** | **学号** | **20174302** |
| **实验题目** | 词法分析程序的设计与实现 | | | | | | |
| **实验时间** | **2020.9.24** | | | **实验地点** | **A主410** | | |
| **实验成绩** |  | | | **实验性质** | **□验证性 □设计性 ■综合性** | | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | | |
| 一、实验目的  🡪了解编译器中词法分析环节的作用和意义  🡪掌握词法分析基础  🡪实现实验要求中C语言子集的词法分析器程序 | | | | | | | |
| 二、实验项目内容   1. 实验内容   设计并实现给定C语言子集的词法分析程序，并实现功能要求:   1. 必做项： 2. 可以识别出用C语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号。（基本数据类型至少包含整数、布尔类型；至少包含赋值语句、变量申明语句、算数加/减法表达式、布尔表达式、循环语句、分支语句） 3. 可以识别并且跳过源程序中的空白。 4. 检查源程序中存在的词法错误，并报告错误所在的位置。 5. 可选项:   需在给定C语言子集的基础上进行扩充,实现下列要求：   1. 识别带else的if语句 2. 能识别8进制/16进制数。 3. 可以统计源程序中的语句行数、各类单词的个数，以及字符总数，并输出统计结果 4. 实验要求： 5. 自选程序设计语言（java，python，c/c++）作为实现语言，手工编写词法分析程序。 6. 提交实验报告及源代码。实验报告需严格遵循学校文档规范，内容包含对应文法、词类编码表、词法分析测试用例。 7. 给定C语言子集:   数据类型: **int**, 无符号整数, 取值范围0-9999  int a;  int a,b;  int a = 1;  算术运算符: **+**,**-**  a = b + 1;  a = b + c;  赋值运算符: **=**  a = 1;  关系运算符: **==** ,**>**,**<,!=,>=,<=**  a = (b==c);  a = (b>c);  a = (b<c);  逻辑运算符: **&&**, **||, !**  a = (b&&c);  a = (b||c);  a = (!b);  条件语句: **if**  if(a==b)  {  };  循环语句: **while**  while(a==b)  {  };  输入,输出: **get**,**put**  get(a);  put(a);  语句结束符: **;**  **加分项：**  条件语句 **if else**  if(a==b)  {  };  else  {  }; | | | | | | | |
| 三、实验过程或算法 | | | | | | | |
| 1. 程序实现内容介绍   本程序简单实现了词法分析，通过构建出状态转换图以及确定符号表以及相关代码的编写，能够将一个文本分析化为由一个个词组成的序列。实现了所有的必做内容，同时也实现了所有的选做内容(能够识别if else，处理8进制/16进制，统计行数、字符数、单词个数)。   1. 状态转换图的绘制   状态图的绘制如下图所示     1. 词类编码  |  |  | | --- | --- | | 单词 | 编码 | | int | 1 | | if | 2 | | else | 3 | | get | 4 | | put | 5 | | ( | 6 | | ) | 7 | | > | 8 | | < | 9 | | = | 10 | | >= | 11 | | <= | 12 | | == | 13 | | ! | 14 | | != | 15 | | && | 16 | | || | 17 | | while | 18 | | ; | 19 | | , | 20 | | + | 21 | | - | 22 | | { | 23 | | } | 24 | | | | 25 | | 数字 | 66 | | 变量 | 77 |  1. 错误处理说明   本词法分析器只针对部分错误进行处理，例如变量中包含非法字符，或者说变量以数字开头的情况。当一个地方被判定为错误时，会打印出错误的行数以及相关位置。  例如，当输入为：    错误打印的结果如下：     1. 程序说明   程序中主要部分是状态机的设计，这里列举几个状态的具体编写思路。   1. 初态(state=0)   其中初态(state=0)的编写如下所示。  在初态读取下一个字符，并根据字符的不同跳转到不同的下一个状态。  case 0:{                  if(s[pos] != ' ' && s[pos]!='\n') curString += s[pos];                  if(isLetter(s[pos])) state = 1;                  if(isDigit(s[pos])) state = 2;                  if(s[pos] == ' ') state = 98;                  if(s[pos] == '+' || s[pos] == '-') state =3;                  if(s[pos] == '=') state = 4;                  if(s[pos] == '>') state=6;                  if(s[pos] == '!') state = 8;                  if(s[pos] == '&') state = 10;                  if(s[pos] == '|') state = 12;                  if(s[pos] == '(') state = 14;                  if(s[pos] == ')') state = 15;                  if(s[pos] == '{') state = 16;                  if(s[pos] == '}') state = 17;                  if(s[pos] == ';') state = 18;                  if(s[pos] == '<') state = 19;                  if(s[pos] == '\n') state = 97;                  if(s[pos] == ',') state  = 22;                  pos++;                  break;              }   1. 处理变量的状态(state=1)   当在初态(state=0)输入的字符为字母时，会跳转到state=1的状态。此时，如果输入仍为字母或数字或者下划线，下一个状态仍为state=1；如果输入了非法字符，则跳转到错误态(state=100)；如果输入为其它字符，则跳转到输出态(state=99)。  case 1:{                  if(isLetter(s[pos]) || isDigit(s[pos]) || s[pos] == '\_'){                      state = 1;                      curString += s[pos];                      pos++;                  }else if(isIllegal(s[pos])){                      curString += s[pos];                      state = 100;                  }                  else{                      state = 99;                  }                    break;              }   1. 输出态(state=99)   输出态将当前存储的字符串存储到结果中，如果当前的字符串不在词类编码中，那么要么是变量，要么是数字，要么就是发生了错误。如果是数字，将答案存储为<数字,66>;如果是变量，将答案存储为<变量，77>  case 99:{                  //输出态                  if(keywords[curString] == 0 ){                      if(isNumber(curString)){                          ans.push\_back(make\_pair(curString,66)); //如果是一个数字，就存为<数字,66>                      }else if(isVariables(curString)){                          ans.push\_back(make\_pair(curString,77)); //如果是一个变量，就存为<变量,77>                      }                      else{                          state = 100;                          break;                      }                  }else{                      ans.push\_back(make\_pair(curString,keywords[curString]));                  }                  cnt[curString]++;                  curString = "";                  state = 0;                  break;              } | | | | | | | |
| 四、实验测试结果   1. 必选测试样例: 2. 简易计算器   //输入数据num1,num2,op，根据op确定操作进行运算，最后输出运算结果ans  int num1,num2,op,ans;  get(num1,num2,op);  if(op==0)  {  ans = num1 + num2;  };  if(op==1)  {  ans = num1 - num2;  };  if(op==2)  {  ans = num1 & num2;  };  if(op==3)  {  ans = num1 | num2;  };  put(ans);  输出结果如下：   1. 跑马灯   //循环输入op，改变输出结果out，输入0则结束程序  int num0,num1,out,op;  num1 = 3333;  num2 = 6666;  num3 = 9999;  op = 1;  while(op>0)  {  if(op==1)  {  out = num1;  };  if(op==2)  {  out = num2;  };  if(op==2)  {  out = num3;  };  put(out);  get(op);  };  输出结果如下：     1. 自定义测试样例(包含自定义的词法错误类型) 2. 测试if else语句   int a = 1;  int b = 1;  int c = 0;  if(a == b) c=1;  else c=2;  输出结果如下：     1. 测试8进制/16进制数   int a = 031;  int b = 0xFF;  输出结果如下：     1. 测试16进制的错误   int a = 0xKK;  输出结果如下：     1. 测试变量字符打头的错误   int 1b = 2;  输出结果如下：     1. 测试变量中有非法字符的错误   int b@a = 2;  输出结果如下： | | | | | | | |
| 五、实验总结  实验过程中遇到的问题以及解决办法：   1. 状态机的设计   在这个实验中，状态机的设计是非常重要的，但是一开始设计的状态机到后面可能会发现与现实情况有些许出入，此时将全部状态推翻重写又太过困难，想要解决就只有另想办法。比如为状态机添加一个新状态，让以前的状态跳转到这个新状态进行处理，或者在状态机以外做一些其他的处理。 | | | | | | | |