|  |
| --- |
| 编译原理  实验报告  （第 二 周）  **指导教师：杨冬**  **年 级：2019级**  **班 级：五班**  **小组编号： 2**  **组长学号姓名：55190521 季圣鹏**  **组员学号姓名：55190516 孟令震**  **组员学号姓名：55190512 贾鹏程**  **2022年 5 月 27 日**  **软件学院** |

|  |  |  |
| --- | --- | --- |
| **完成实验内容** | | |
| 阅读实验指导书  掌握SNL语言的词法，语法，语义  完成词法分析器的开发 | | |
| **小组成员任务完成情况** | | |
| 姓名 | 具体完成任务 | 工作量  百分比 |
| 季圣鹏 | 学习SNL语言  完成词法分析器的开发 | 90% |
| 孟令震 | 学习SNL语言 | 5% |
| 贾鹏程 | 学习SNL语言 | 5% |
| **小组成员协作情况** | | |
| 季圣鹏负责开发词法分析器  孟令震辅助开发  贾鹏程辅助开发 | | |
| **实验中出现的问题及解决方案** | | |
| 1.出现变量名不明确的错误  Std名词空间中含有size，得重新命名  2.发现词法分析器中“”和>的token定义不明确对其重新进行定义 | | |
| **程序界面及运行截图** | | |
|  | | |
| **已完成部分的源程序代码**  //完成词法分析器  #include <iostream>  #include <vector>  using namespace std;  //定义token序列的结构  struct Token {  string value1;  string value2;  };  //全局变量  int size1= 0;//token序列的长度  vector<Token> token;//token序列  //分界符的命名+ | - | \*| / | ( | ) | [ | ] | ; | . | < | : | = | ' | := | > | "  char SingleDelimiter[17][20]= {"PLUS","MINUS","TIMES","OVER","LPAREN","RPAREN",  " LMIDPAREN","RMIDPAREN", "SEMI","DOT","LT","COLON","EQ","COMMA","ASSIGN","RT","SY"};  // 保留字的命名  string reservedWords[21] = { "program","type","var","procedure","begin","end","array","of","record","if","then","else","fi","while","do","endwh","read","write","return","integer","char" };  //打印出错函数  void printwrong() {  cout << "你输入的程序段有词法错误" << endl;  }  //打印token序列  void printtoken(vector<Token>token) {  cout << "输入程序段经过词法分析器之后的Token序列为:" << endl;  for (int i = 0; i < size1-1; i++) {  cout << token[i].value1 << "," << token[i].value2 << endl;  }  cout << token[size1 - 1].value1 << "," << token[size1 - 1].value2 << endl<<"EOF";    }  //通过一个连续字符串,提取token序列  void gettoken(vector<char> input, int len) {  int i = 0;  while (i < len) {  //如果遇到'\n'和空格  if (input[i] == ' ' || input[i] == '\n') {  i = i + 1;  continue;  }  //如果遇到数字  if (input[i] >= '0' && input[i] <= '9') {  string tmp = "";  tmp += input[i];  int j = i + 1;  while (j < len) {  if (input[j] >= '0' && input[j] <= '9') {  tmp += input[j];  j++;  }  else {  i = j;  break;  }  }  //下面要判断他是数学下标还是数字，这个地方的下标已经是i+1了，所以不用再变  if (i >= len) {  Token token1;  token1.value1 = "NUM";  token1.value2 = "val=" + tmp;  size1 = size1 + 1;  token.push\_back(token1);  }  else {  //如果这个右边是]，则判断他是数组类型的下表  if (input[i] == ']') {  Token token1;  token1.value1 = "UNDERANGE";  token1.value2 = "val=" + tmp;  size1 = size1 + 1;  token.push\_back(token1);  }  else {  Token token1;  token1.value1 = "NUM";  token1.value2 = "val=" + tmp;  size1 = size1 + 1;  token.push\_back(token1);  }  }  continue;  }  //如果遇到字符  if ((input[i] >= 'a' && input[i] <= 'z') || (input[i] >= 'A' && input[i] <= 'Z')) {  string tmp = "";  tmp += input[i];  int j = i + 1;  while (j < len) {  if ((input[j] >= 'a' && input[j] <= 'z') || (input[j] >= 'A' && input[j] <= 'Z') || (input[j] >= '0' && input[j] <= '9')) {  tmp += input[j];  j = j + 1;  }  else {  i = j;  break;  }  }  //cout << "\n" << "\*\*\*\*\*\*" << tmp << endl;  if (tmp.length()==1) { //如果是char类型的  Token token1;  token1.value1 = "CHAR";  token1.value2 = "char=" + tmp;  size1 = size1 + 1;  token.push\_back(token1);  }  else {  //判断他个是保留字  bool flag2 = false;  for (int k = 0; k < 21; k++) {  if (tmp == reservedWords[k]) {  Token token1;  token1.value1 = "reserved word";  token1.value2 = tmp;  size1 = size1 + 1;  token.push\_back(token1);  flag2 = true;  break;  }  }  //如果他就是普通字符串  if (!flag2) {  Token token1;  token1.value1 = "ID";  token1.value2 = tmp;  size1 = size1 + 1;  token.push\_back(token1);  }  }  continue;  }  //如果遇到左注释和出错处理  if (input[i] == '{') {  int j = i; //通过双指针找到下一个}  bool flag = false; //标识是否找到了}  while (j < len) {  if (input[j] == '}') {  i = j + 1;  flag = true;  break;  }  j = j + 1;  }  if (!flag) {  printwrong();  }  continue;  }  //如果遇到右注释,出错处理  if (input[i] == '}') {  i = i + 1;  continue;  }  //如果遇到单分界符+  if (input[i] == '+') {  Token token1;  token1.value1 = SingleDelimiter[0];  token1.value2 = '+';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符-  if (input[i] == '-') {  Token token1;  token1.value1 = SingleDelimiter[1];  token1.value2 = '-';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符\*  if (input[i] == '\*') {  Token token1;  token1.value1 = SingleDelimiter[2];  token1.value2 = '\*';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符/  if (input[i] == '/') {  Token token1;  token1.value1 = SingleDelimiter[3];  token1.value2 = '/';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符(  if (input[i] == '(') {  Token token1;  token1.value1 = SingleDelimiter[4];  token1.value2 = '(';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符)  if (input[i] == ')') {  Token token1;  token1.value1 = SingleDelimiter[5];  token1.value2 = ')';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符[  if (input[i] == '[') {  Token token1;  token1.value1 = SingleDelimiter[6];  token1.value2 = '[';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符]  if (input[i] == ']') {  Token token1;  token1.value1 = SingleDelimiter[7];  token1.value2 = ']';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符:  if (input[i] == ';') {  Token token1;  token1.value1 = SingleDelimiter[8];  token1.value2 = ';';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符.  if (input[i] == '.') {  Token token1;  token1.value1 = SingleDelimiter[9];  token1.value2 = '.';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符<  if (input[i] == '<') {  Token token1;  token1.value1 = SingleDelimiter[10];  token1.value2 = '<';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符:这个地方得考虑一下是不是:=  if (input[i] == ':') {  int j = i + 1;  if (j >= len) {  Token token1;  token1.value1 = SingleDelimiter[11];  token1.value2 = ':';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  }  else {  if (input[j] == '=') {  Token token1;  token1.value1 = SingleDelimiter[14];  token1.value2 = ":=";  size1 = size1 + 1;  token.push\_back(token1);  i = i + 2;  }  else {  Token token1;  token1.value1 = SingleDelimiter[11];  token1.value2 = ':';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  }  }  continue;  }  //如果遇到单分界符=  if (input[i] == '=') {  Token token1;  token1.value1 = SingleDelimiter[12];  token1.value2 = '=';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符'  if (input[i] == '\'') {  Token token1;  token1.value1 = SingleDelimiter[13];  token1.value2 = '\'';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符>  if (input[i] == '>') {  Token token1;  token1.value1 = SingleDelimiter[15];  token1.value2 = '>';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  //如果遇到单分界符  if (input[i] == '"') {  Token token1;  token1.value1 = SingleDelimiter[16];  token1.value2 = '"';  size1 = size1 + 1;  token.push\_back(token1);  i = i + 1;  continue;  }  printwrong();  i = i + 1;  }  }  int main() {  //存放输入字符串  vector<char> input;  char tmp = ' ';  cout << "请输入SNL程序片段：" << endl;  while (tmp != '.') {  tmp = getchar();  input.push\_back(char(tmp));    }  //打印看一下  int len = input.size();  //for (int i = 0; i < len; i++) {  // cout << input[i];  //}  gettoken(input, len); //将字符串变成token序列  //system("cls");  printtoken(token); //将token进行输出  return 0;    } | | |