|  |
| --- |
| 编译原理课程设计  实验报告  （第 2 周）  **指导教师：申春**  **年 级： 2117**  **班 级：211711**  **小组编号： 2117-01**  **组长学号姓名：21171102许俊杰**  **组员学号姓名：21171110付昊翀**  **组员学号姓名：21171135周智勇**  **2020年 5 月 6 日**  **计算机科学与技术学院** |

|  |  |  |
| --- | --- | --- |
| **完成实验内容** | | |
| 将33个关键字，12个运算符以及11个分界符分别存入数组中以便于对程序中出现的所有字符进行扫描转换  定义标识符展示格式  **typedef struct{**  **int 标识符所在表的下标**  **int 标识符的类型**  **int 标识符所在表的行数**  **char []标识符所在表的名称**  **};**  输出词法错误类型（暂时三种）的展示功能：  **error type=1--------变量的长度超过限制（define 256）**  **error type=2--------小数点错误**  **error type=3--------常量的长度超过限制（define 256）**  设计begin-end 嵌套关系的分析  设计字符扫描函数（语法分析）  功能分别为：**{**   1. **处理空格和tab：**   **忽略空格和tab**   1. **处理换行符：**   **遇到换行符，行数加1**   1. **标识符：**   **if 以字母、下划线开头，其余是字母下划线数字，识别字符串**  **判断长度if 标识符超过规定长度，报错处理**  **else 与关键字匹配**  **if是关键字 写入table表中**  **else不是关键字**   1. **常数：**   **else if 以数字开头**  **判断小数点**  **flag表记小数点的个数**  **0个小数点为整数**  **1个小数点为小数**  **2个及以上小数点出错**  **常量标识符超过规定长度，报错处理**  **置标识符类型（整数/小数）**   1. **运算符：**   **for循环依次与运算符设定数组匹配 若匹配成功则是运算符**   1. **界符：**   **for循环依次与分界符设定数组匹配 若匹配成功则是分界符**   1. **其他无法识别字符：**   **开头的不是字母、数字、运算符、界符**  **设置错误flag**  }  初步设想语法错误类型的展示功能  使用switch-case对不同errorType进行不同输出 | | |
| **小组成员任务完成情况** | | |
| 姓名 | 具体完成任务 | 工作量  百分比 |
| 许俊杰 | 定义标识符展示格式  **typedef struct{};**  **共同完成字符扫描函数的六个功能的编写和测试**  **共同设计语法分析的多种错误类型及其输出格式** | 33.3% |
| 周智勇 | 将33个关键字，12个运算符以及11个分界符分别存入数组中以便于对程序中出现的所有字符进行扫描转换  **共同完成字符扫描函数的六个功能的编写和测试**  **共同设计语法分析的多种错误类型及其输出格式** | 33.3% |
| 付昊翀 | 输出词法错误类型（暂时三种）的展示功能：  **变量的长度超过限制 小数点错误 常量的长度超过限制**  **共同完成字符扫描函数的六个功能的编写和测试**  **共同设计语法分析的多种错误类型及其输出格式** | 33.3% |
| **小组成员协作情况** | | |
| **字符扫描函数的六个功能**  **处理空格和tab，处理换行符，判断关键字，判断常数，判断分界符，判断运算符**  **互相依靠，联系紧密，都对标识符展示表结构体的类型（int），下标（int），名称（char[]）,有频繁操作，而所有类型的标识符都用统一的标识符表结构体表示，因此这一部分内容，代码量较大，逻辑复杂，容易出错，三个人共同讨论完成** | | |
| **实验中出现的问题及解决方案** | | |
| 关键字数量经常有所缺少  经过查找和分析多个程序，完善33个关键字的信息存储  对标识符表的下标和行数等信息经常发生错误的处理，  进行缜密的思考，理清逻辑思路，进行修改  常数没有考虑到小数点>2时的情况，导致意外输入3.2.45时发生崩溃  经检测后，完善了常数判断功能，能够对数字分类判断为整数或小数或错误  词法分析错误类型  应该分析常量与变量名称超界问题  （尽管现实中出现的可能性不大）  对语法分析还没有可运行的完整程序和子函数，正在编写与完善中 | | |
| **程序界面及运行截图** | | |
| 词法分析-小数点错误  词法分析-超界错误 | | |
| **已完成部分的源程序代码** | | |
| #include <iostream>  #include <fstream>  #include <string>  #include <math.h>  #include <ctype.h>  #include <cstdlib>  using namespace std;  #define Max 655 //最大代码长度  #define WordMaxNum 256 //变量最大个数  #define DigitNum 256 //常量最大个数  #define MaxKeyWord 33 //关键字数量  #define MaxOptANum 8 //运算符最大个数  #define MaxOptBNum 4 //运算符最大个数  #define MaxEndNum 11 //界符最大个数  /\*关键字\*/const char\* const KeyWord[MaxKeyWord] = {  "and",  "array",  "begin",  "case",  "char",  "constant",  "do",  "else",  "end",  "false",  "for",  "if",  "input",  "integer",  "not",  "of",  "or",  "output",  "packed",  "procedure",  "program",  "read",  "real",  "repeat",  "set",  "then",  "to",  "type",  "until",  "var",  "while",  "with",  "prn" };  /\*单目运算\*/const char OptA[] =  { '+',  '-',  '\*',  '/',  '=',  '#',  '<',  '>' };  /\*双目运算符\*/const char\* OptB[] =  { "<=",  ">=",  ":=",  "<>" };  /\*界符\*/const char End[] =  { '(',  ')' ,  ',' ,  ';' ,  '.' ,  '[' ,  ']' ,  ':' ,  '{' ,  '}' ,  '"'  };  typedef struct DisplayTable  {  int Index; //标识符所在表的下标  int type; //标识符的类型  int line; //标识符所在表的行数  char symbol[20]; //标识符所在表的名称  }Table;  int TableNum = 0; //display表的表项总数  char Word[WordMaxNum][20]; //标识符表  char Digit[WordMaxNum][20]; //数字表  int WordNum = 0; //变量表的下标  int DigNum = 0; //常量表的下标  bool errorFlag = 0; //错误标志  int TableIndex = -1; //display 表的下标索引  int beginCount = 0;//遇到begin加1，遇到end减1  int ifCount = 0; //遇到if加1  Table\* table = new Table[Max];  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*输出错误类型\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void error(char str[20], int nLine, int errorType)  {  errorFlag = 1;  cout << " \nError : ";  switch (errorType)  {  case 1:  cout << "第" << nLine - 1 << "行" << str << " 变量的长度超过限制！\n";  break;  case 2:  cout << "第" << nLine - 1 << "行" << str << " 小数点错误！\n";  break;  case 3:  cout << "第" << nLine - 1 << "行" << str << " 常量的长度超过限制！\n";  break;  }//switch  }//error  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*字符扫描\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Scanner(char ch[], int chLen, int nLine)  {  int chIndex = 0;  while (chIndex < chLen) //对输入的字符扫描  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*处理空格和tab\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //忽略空格和tab  while (ch[chIndex] == ' ' || ch[chIndex] == 9) { chIndex++; }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*处理换行符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //遇到换行符，行数加1  while (ch[chIndex] == 10)  {  nLine++; chIndex++;  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*标识符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    if (isalpha(ch[chIndex])) //以字母、下划线开头  {  char str[256];  int strLen = 0;  //是字母、下划线  while (isalpha(ch[chIndex]) || ch[chIndex] == '\_') {  str[strLen++] = ch[chIndex];  chIndex++;  while (isdigit(ch[chIndex]))//不是第一位，可以为数字  {  str[strLen++] = ch[chIndex];  chIndex++;  }  }  str[strLen] = 0; //字符串结束符  if (strlen(str) > 20) //标识符超过规定长度，报错处理  {  error(str, nLine, 1);  }  else  {  int i;  for (i = 0; i < MaxKeyWord; i++) //与关键字匹配  //是关键字，写入table表中  if (strcmp(str, KeyWord[i]) == 0)  {  strcpy(table[TableNum].symbol, str);  table[TableNum].type = 1; //关键字  table[TableNum].line = nLine;  table[TableNum].Index = i;  TableNum++;  break;  }  if (i >= MaxKeyWord) //不是关键字  {  table[TableNum].Index = WordNum;  strcpy(Word[WordNum++], str);  table[TableNum].type = 2; //变量标识符  strcpy(table[TableNum].symbol, str);  table[TableNum].line = nLine;  TableNum++;  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*常数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  else if (isdigit(ch[chIndex])) //遇到数字  {  int flag = 0;  char str[256];  int strLen = 0;  //数字和小数点  while (isdigit(ch[chIndex]) || ch[chIndex] == '.') {  //flag表记小数点的个数，0时为整数，1时为小数，2时出错  if (ch[chIndex] == '.')  flag++;  str[strLen++] = ch[chIndex];  chIndex++;  }  str[strLen] = 0;  if (strlen(str) > 20) //常量标识符超过规定长度20，报错处理  {  error(str, nLine, 3);  }  if (flag == 0)  {  table[TableNum].type = 3; //整数  }  if (flag == 1)  {  table[TableNum].type = 4; //小数  }  if (flag > 1)  {  error(str, nLine, 2);  }  table[TableNum].Index = DigNum;  strcpy(Digit[DigNum++], str);  strcpy(table[TableNum].symbol, str);  table[TableNum].line = nLine;  TableNum++;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*运算符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  else  {  //用来区分是不是无法识别的标识符，0为运算符，1为界符  int errorFlag;  char str[3];  str[0] = ch[chIndex];  str[1] = ch[chIndex + 1];  str[2] = '\0';  int i;  for (i = 0; i < MaxOptBNum; i++)//MaxOptBNum)  if (strcmp(str, OptB[i]) == 0)  {  errorFlag = 0;  table[TableNum].type = 6;  strcpy(table[TableNum].symbol, str);  table[TableNum].line = nLine;  table[TableNum].Index = i;  TableNum++;  chIndex = chIndex + 2;  break;  }  if (i >= MaxOptBNum)  {  for (int k = 0; k < MaxOptANum; k++)  if (OptA[k] == ch[chIndex])  {  errorFlag = 0;  table[TableNum].type = 5;  table[TableNum].symbol[0] = ch[chIndex];  table[TableNum].symbol[1] = 0;  table[TableNum].line = nLine;  table[TableNum].Index = k;  TableNum++;  chIndex++;  break;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*界符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  for (int j = 0; j < MaxEndNum; j++)  if (End[j] == ch[chIndex])  {  errorFlag = 1;  table[TableNum].line = nLine;  table[TableNum].symbol[0] = ch[chIndex];  table[TableNum].symbol[1] = 0;  table[TableNum].Index = j;  table[TableNum].type = 7;  TableNum++;  chIndex++;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*其他无法识别字符\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //开头的不是字母、数字、运算符、界符  if (errorFlag != 0 && errorFlag != 1) {  char str[256];  int strLen = -1;  str[strLen++] = ch[chIndex];  chIndex++;  while (\*ch != ' ' || \*ch != 9 || ch[chIndex] != 10)  {  str[strLen++] = ch[chIndex];  chIndex++;  }  str[strLen] = 0;  table[TableNum].type = 8;  strcpy(table[TableNum].symbol, str);  table[TableNum].line = nLine;  table[TableNum].Index = -2;  TableNum++;  }  }  }  }  }  void Gerror(int errorType, int nIndex)  {  errorFlag = 1;  switch (errorType)  {  case 1:  case 2:  case 3:  case 4:  case 5:  case 6:  case 7:  case 8:  case 8:  case 9:  case 10:  //intmain（）  int main()  {  ifstream in;  ofstream out;  char in\_file\_name[26], out\_file\_name[26];  char ch[Max];  cin.getline(ch, Max, '#');  int nLine = 1;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*调用词法分析\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Scanner(ch, strlen(ch), nLine);  return 0;  } | | |