电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221302006

姓 名 周玉川

（实验） 课程名称 编译技术

理论教师 邓伏虎

实验教师 邓伏虎

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：周玉川 学号：2017221302006 指导教师：邓伏虎**

**组长姓名：周玉川 组长学号:2017221302006 组内排序：第一**

**组员姓名：无 组员学号：无 组内排序：无**

**实验地点：信软楼305 实验时间：2018.12.07**

1. **实验名称：编译技术实验二语法分析**
2. **实验学时：课堂4学时 课后4学时**
3. **实验目的：**
4. 学习语法分析相关概念。
5. 识别文法的回溯，包括公共左因子，左递归以及间接左递归，学习对应消除回溯的文法改写。
6. 学习如何计算Follow集，解决由于产生式引起的回溯。
7. 理解递归下降分析法原理，学习用C语言代码编程实现。
8. 理解语法分析在编译程序中的作用。
9. 理解并掌握递归下降分析法的主要原理。
10. 理解该分析方法对文法的要求。
11. 熟练掌握文法改造算法
12. 消除左递归，
13. 消除公共左因子。
14. 掌握递归下降分析程序的构造方法。
15. **实验原理：**
16. 在递归下降分析法中，文法不能含有左递归以及公共左因子，这样每一个非终结符左面的终结符是不同的，通过构造一个不含有回溯的自上而下的分析程序，称为递归下降分析器。
17. 在进行原始文法的改写之后，得到不含回溯的文法，计算Follow集，定义各个非终结符的函数，按照文法进行函数间的相互调用，可以得到递归下降分析程序。
18. 本实验采用递归下降的思想，每一个终结符对应于一个子程序，文法的开始符号对应于子程序的入口。每一个产生式的右侧对应于各子程序的函数体，每一次识别到终结符号时，即表示语法分析器识别到某一个输入字符串。当所有的输入字符都识别到时，即表示当前输入字符串语法正确。反之则表示存在语法错误。
19. **实验内容：**
20. 检查所处理语言的文法是否满足递归下降算法要求如不满足，对其进行改造。
21. 根据递归下降算法编写所处理语言的递归下降程序。
22. 对测试程序进行分析、检查输出分析结果：接受或拒绝 将语法错误输出到文件error.txt。
23. 将详细的分析步骤输出到detail.txt 每一步使用的产生式、已分析过的串 当前分析字符。
24. **实验器材（设备、元器件）：**

Pc一台。

1. **实验步骤：**
2. 步骤一：检查文法是否有回溯，包括左递归以及公共左因子，如果右改写文法，得到不含有回溯的文法。
3. 步骤二：计算非终结符的Follow集，准确来说是计算First集中含有的非终结符的Follow集。
4. 根据递归下降算法，编写C语言的递归下降程序。
5. 输出结果：接受或拒接，错误输出到error.txt文件。
6. 将详细步骤输出到detail.txt，包括每一步使用产生式，已经分析的串，正在分析的串，以及未分析的串。
7. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

8.1程序代码（C语言编写,包含详细注释）

|  |
| --- |
| /\*  Name:syntaxAnalysis.cpp  Time:2018.11.30  Auther：周玉川  \*/    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  原始文法： \*  Expr->Expr+Term|Term|Expr-Term \*  Term->Term\*Factor|Factor|Term/Factor \*  Factor->(Expr)|i \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  消除回溯后的语法,其中空记为 $ \*  Expr->Term Expr\_ \*  Expr\_->+Term Expr\_|-Term Expr\_|$ \*  Term->Factor Term\_ \*  Term\_ ->\*Factor Term\_|/Facter Term\_|$ \*  Factor->(Expr)|i \*  输入字符串为source.txt、source1.txt、source2.txt等三个文档 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*计算Follow集,#为终结符 \*  \*Follow(E)={ # , ) } \*  \*Follow(E\_)={ #, ) } \*  \*Follow(T)={ +, - , # , ) } \*  \*Follow(T\_) ={ +, - , #, ) } \*  \*Follow(F) = { \*, / } \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <ctype.h>  #define MAX\_SIZE 100  char token[12] = {'\0'}; //记录当前分析的字符串  int token\_len = 0; //token长度  char character; //当前字符  int place = -1; //记录字符在字符串的位置  char str[MAX\_SIZE]; //储存字符串  FILE\* fp\_src, \*fp\_out, \*fp\_error;//源文件，详细步骤文件，错误输出文件  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*getchr()：功能从str[]字符串中读入字符 \*  \*error()：功能：打印错误信息 \*  \*print(): 功能：打印串，包括已经分析的串，当前字串，以及剩余串 \*  \*getToken(): 功能：得到当前要分析的字符串 \*  \*concat(char character): 功能：将getchr得到的字符添加到token中 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void getchr();  void getToken();  void error();  void print();  void concat(char character);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*文法对应函数，取首字母大写 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void E();  void E\_();  void F();  void T();  void T\_();  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 主函数 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int main(){  char\* source = "source1.txt"; //源文件  char\* output = "detail.txt"; //分析步骤文件  char\* error = "error.txt"; //错误输出文件  fp\_src = fopen(source,"r"); //打开源文件  if ( fp\_src == NULL ){  printf("Error:打开源文件失败。");  exit(-1); //失败策略  }  fp\_out = fopen(output, "w"); //打开分析步骤文件  if ( fp\_out == NULL ){  printf("Error:打开步骤分析输出文件失败。");  exit(-1); //失败策略  }  fp\_error = fopen(error, "w"); // 打开错误输出文件  if ( fp\_error == NULL ){  printf("Error:打开错误输出文件失败。");  exit(-1); //失败策略  }    fprintf(fp\_out,"学号：2017221302006\t姓名：周玉川\n");  fprintf(fp\_error,"学号：2017221302006\t姓名：周玉川\n");//输出学号信息到detail.txt,error.txt以及屏幕上  printf("学号：2017221302006\t姓名：周玉川\n");    fgets(str,MAX\_SIZE,fp\_src); //从文件中读入表达式串到字符串str中  printf("The experssion: %s\n",str); //输出表达式  int str\_len = strlen(str); //表达式串的长度  str[str\_len] = '#';  str[str\_len+1] = '\0'; //在表达式串中添加终结符    getToken(); //读入第一个单词符号  E(); //开始递归分析    fprintf(fp\_out, "\n=========================================\n");  fprintf(fp\_out, "分析表达式串结束\n");//递归结束后，打印终结符串到detai.txt文件中  print();  printf("分析结果：接受"); //递归成功，输出接受信息到屏幕    fclose(fp\_src); //关闭文件  fclose(fp\_out);  fclose(fp\_error);    return 0; //程序结束  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 产生式：Expr->Term Expr\_ \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void E(){  T();  E\_();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*产生式：Expr\_->+Term Expr\_|-Term Expr\_|$ \*  \*包含空串，要判断 Follow(Expr\_) = { #, ) } \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void E\_(){  if ( character == '+' ){//当前符号是加号  fprintf(fp\_out,"\n=====================================\n");  fprintf(fp\_out,"使用的产生式：Expr\_ ----> +Term Expr\_\n");//输出信息到detail.txt文件中  print(); //打印当前串，已经分析串，未分析串  getToken(); //读取下一个单词符号  T();  E\_(); //继续递归  }  else if ( character == '-' ){ // 当前符号是减号  fprintf(fp\_out,"\n======================================\n");//输出信息到detail.txt文档中  fprintf(fp\_out,"使用的产生式：Expr\_ ----> -Term Expr\_\n");  print();  getToken(); //读取下一个单词符号  T();  E\_(); //继续递归  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*添加follow集,Follow(E\_)={ #, ) } \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  else if ( character == ')'|| character == '#' ) //如果下一个字符输入Expr\_的Follow集，就继续递归  return ;  else{ //其余的作为错误，输出错误信息到error.txt文件中  fprintf(fp\_error, "\n=====================================\n");  fprintf(fp\_error, "使用的产生式：Expr\_ ----> +Term Expr\_ | -Term Expr\_ | $\n");  error();  }    }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*产生式： Term->Factor Term\_ \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void T(){  F();  T\_();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*产生式：Term\_ ->\*Factor Term\_|/Facter Term\_|$ \*  \*包含空串，要使用Follow(Term\_) = { +，-,),# } \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void T\_(){ //  if ( character == '\*' ){  fprintf(fp\_out, "\n======================================\n");//输出信息到detail.txt文档中  fprintf(fp\_out, "使用的产生式：Term\_ ----> \*Factor Term\_\n");  print();  getToken(); //读入下一个单词符号  F();  T\_(); //继续递归  }  else if ( character == '/' ){ //判断token串为除号  fprintf(fp\_out, "\n=======================================\n");//输出信息到detail.txt文档中  fprintf(fp\_out, "使用的产生式：Term\_ ----> /Facter Term\_\n");  print();  getToken(); //读入下一个单词符号  F();  T\_(); //继续递归    }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*添加follow集,Follow(E\_)={ +，-,),# } \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  else if ( character == '+' || character == '-'  || character == ')' || character == '#' )//在Follow集的字符串，就继续递归  return ;  else{ //否则作为错误，输出错误信息到error.txt文件中  fprintf(fp\_error, "\n=========================================\n");  fprintf(fp\_error, "使用的产生式：Term\_ ----> \*Factor Term\_ | /Facter Term\_ | $\n");  error();  }    }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*产生式： Factor->(Expr)|i \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void F(){  if ( character == '(' ){ //匹配左括号  fprintf(fp\_out, "\n=======================================\n");  fprintf(fp\_out, "使用的产生式：Factor ----> (Expr) \n");  print(); //输出信息  getToken();  E(); //继续递归  if ( character == ')'){ //匹配右括号，输出信息到detail.txt文件中  fprintf(fp\_out, "\n=======================================\n");  fprintf(fp\_out, "使用的产生式：Factor ----> (Expr) \n");  print();  getToken();  }else{ //没有匹配到右括号，视为错误  fprintf(fp\_error, "\n=======================================\n");  fprintf(fp\_error, "使用的产生式：Factor ----> (Expr) \n");  error();  }  }  else if ( isdigit( token[0] ) || token\_len > 1){ //匹配到数字，或者标识符，输出信息到detail.txt文件  fprintf(fp\_out, "\n=======================================\n");  fprintf(fp\_out, "使用的产生式：Factor ----> i\n");  print();  getToken();  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*name:getchr \*  \*function:省略空格方式读入字符 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void getchr(){  while( ( character = str[++place] ) == ' ');  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*name:concat \*  \*function:把单个字符连接到token末尾，插入'\0',并修改长度 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void concat(char character){  token[token\_len] = character;  token\_len++;  token[token\_len] = '\0';  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*name:getToken \*  \*function:从表达式串中得到单词符号，标识符，数字，或者运算符，结束符 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void getToken(){  int i;  for ( i = 0 ; i < token\_len ; i++)  token[i]='\0';  token\_len = 0; //清空token，开始读入新的单词符号  getchr();  if (isdigit( character )){ //判断字符为数字，最终在token中得到数字串  while (isdigit( character)){  concat(character);  character = str[++place];  }  place--; //多读入一次，退回一个字符  }  else if (isalpha( character) || character == '\_') //判断首字母未字母或者下划线，最终在token中得到一个标识符  {  while ( isalnum( character )|| character == '\_' ){ //当前字符是字母，数字，下划线  concat(character);  character = str[++place]; //读入下一个字符  }  place--; //因为多读入一次，回退一个字符  }  else{ //处理其他情况  switch ( character ){  case '+': case '-': case '\*': case '/': case '#': case '(': case ')': //若是规定字符，存入token中  concat(character);  break;  default:  concat(character); //将未知字符添加到token中，方便输出  error(); //否则视为错误  }  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*name：print \*  \*function: 打印已经分析的串，当前串，已经剩余串到detail.txt文件中 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void print(){  fprintf(fp\_out, "已经分析的串: ");  int i;  for ( i = 0 ; i < place-token\_len+1 ; i++)  fprintf(fp\_out, "%c", str[i]); //输出分析过的字串  fprintf(fp\_out, "\n当前字符: %s", token ); //输出当前字串  fprintf(fp\_out, "\n剩余串: %s\n", str+place+1); //输出剩余表达式串    }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*name：error \*  \*function：打印错误信息到error.txt文件中 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void error(){  fprintf(fp\_error, "\nError: %s\n", token); //输出当前错误字符串  printf("分析结果：拒绝\n"); //出现错误，输出拒绝信息到屏幕  fclose(fp\_src); //关闭文件  fclose(fp\_out);  fclose(fp\_error);  exit(-1); //退出程序  } |

8.2代码流程分析

1. 先把文件中的一串表达式串读到一个字符串数组里面，从字符串中读入字符。
2. 若是字符是数字，存入token中，继续读入字符直到不是数字，它们组成的字符串为数字，类型为终结符i。

核心代码

|  |
| --- |
| if (isdigit( character )){ //判断字符为数字，最终在token中得到数字串  while (isdigit( character)){  concat(character);  character = str[++place];  }  place--; //多读入一次，退回一个字符  } |

1. 若是字母或下划线，继续读入字符直到读入的不是数字或者字母或者下划线，它们组成的字符串存在token[]中，记为标识符类型为终结符i。

核心代码

|  |
| --- |
| else if (isalpha( character) || character == '\_') //判断首字母未字母或者下划线，最终在token中得到一个标识符  {  while ( isalnum( character )|| character == '\_' ){ //当前字符是字母，数字，下划线  concat(character);  character = str[++place]; //读入下一个字符  }  place--; //因为多读入一次，回退一个字符  } |

1. 先在main函数中读入一个单词串，调用文法开始符号开始递归，根据当前字符或者字符串判断程序下一步进行哪一步，直到作为符合要求字符串进行匹配。
2. 若当前字符匹配不到串，那就判断字符是否属于当前非终结符的Follow集，若在其中，程序继续进行，否则作为错误输出。
3. 程序结束，当且仅当无错误输出而且把表达式串全部匹配完成。

8.3实验结果（程序输出结果）

选取source2.txt: abc+(bc-123)\*(ab-456)

|  |
| --- |
| 屏幕输出  学号：2017221302006 姓名：周玉川  the experssion: abc+(bc-123)\*(ab-456)  分析结果：接受 |
| dtail.txt文件  学号：2017221302006 姓名：周玉川  =======================================  使用的产生式：Factor ----> i  已经分析的串:  当前字符: abc  剩余串: +(bc-123)\*(ab-456)#  =====================================  使用的产生式：Expr\_ ----> +Term Expr\_  已经分析的串: abc  当前字符: +  剩余串: (bc-123)\*(ab-456)#  =======================================  使用的产生式：Factor ----> (Expr)  已经分析的串: abc+  当前字符: (  剩余串: bc-123)\*(ab-456)#  =======================================  使用的产生式：Factor ----> i  已经分析的串: abc+(  当前字符: bc  剩余串: -123)\*(ab-456)#  ======================================  使用的产生式：Expr\_ ----> -Term Expr\_  已经分析的串: abc+(bc  当前字符: -  剩余串: 123)\*(ab-456)#  =======================================  使用的产生式：Factor ----> i  已经分析的串: abc+(bc-  当前字符: 123  剩余串: )\*(ab-456)#  =======================================  使用的产生式：Factor ----> (Expr)  已经分析的串: abc+(bc-123  当前字符: )  剩余串: \*(ab-456)#  ======================================  使用的产生式：Term\_ ----> \*Factor Term\_  已经分析的串: abc+(bc-123)  当前字符: \*  剩余串: (ab-456)#  =======================================  使用的产生式：Factor ----> (Expr)  已经分析的串: abc+(bc-123)\*  当前字符: (  剩余串: ab-456)#  =======================================  使用的产生式：Factor ----> i  已经分析的串: abc+(bc-123)\*(  当前字符: ab  剩余串: -456)#  ======================================  使用的产生式：Expr\_ ----> -Term Expr\_  已经分析的串: abc+(bc-123)\*(ab  当前字符: -  剩余串: 456)#  =======================================  使用的产生式：Factor ----> i  已经分析的串: abc+(bc-123)\*(ab-  当前字符: 456  剩余串: )#  =======================================  使用的产生式：Factor ----> (Expr)  已经分析的串: abc+(bc-123)\*(ab-456  当前字符: )  剩余串: #  =========================================  分析表达式串结束  已经分析的串: abc+(bc-123)\*(ab-456)  当前字符: #  剩余串: |
| error.txt文件  学号：2017221302006 姓名：周玉川 |

使用source3.txt文件

|  |
| --- |
| 屏幕上：  学号：2017221302006 姓名：周玉川  The experssion: if abc+32/64 = 43 else abc=54  分析结果：拒绝 |
| dtail.txt文件里： |
| error.txt里面：  学号：2017221302006 姓名：周玉川  Error: = |

8.4分析结束

1. **总结及心得体会：**
2. 解决了多字符识别的难题，比如识别长数字，长标识符。
3. 实验完成后，再回想一下理论知识，发现理解加深了。
4. 完成程序要求，达到实验目的，很有成就感。
5. C语言的功底很重要。
6. 学习书本后，再写代码会容易一些。
7. 带着兴趣会事半功倍。
8. **对本实验过程及方法、手段的改进建议：**
9. 希望实验能多一些，能更有趣一点，最好有选做题让基础好一点的同学挑战自我。
10. 希望老师和同学之间能多交流，答疑课多一点点。
11. 虽然代码写得很顺畅，但是应试考试还是有点担心，希望老师能划一下考点。

**报告评分：**

**指导教师签字：**