电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221105003

姓 名 黄文杰

（实验） 课程名称 编译技术

理论教师 邓伏虎

实验教师 陆庆

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：黄文杰 学号：2017221105003 指导教师：邓伏虎**

**实验地点：信软学院楼西304 实验时间：2018.11.02**

1. **实验名称：语法分析-递归下降分析方法**
2. **实验学时：4学时**
3. **实验目的：**

1.理解语法分析在编译程序中的作用

2.理解并掌握递归下降分析法的主要原理

3.理解该分析方法对文法的要求

4.熟练掌握文法改造算法

①消除左递归

②消除公共左因子

5.掌握递归下降分析程序的构造方法

1. **实验原理：**

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。

1. **实验内容：**

1)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

2)根据递归下降算法

编写所处理语言的递归下降程序

3)调用词法分析器

获取测试程序的词法记号

4)将词法分析后的符号表

输出到文件symtab1.txt

5)对测试程序的词法记号进行分析、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

6)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、剩余串

1. **实验器材（设备、元器件）：**

PC机一台

1. **实验步骤：**

1.对文法进行处理，消除回溯；

2.利用递归下降算法编写语法分析器代码；

3.利用gcc编译器对代码进行编译；

4.编译成功后利用样本对代码进行测试。

1. **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）**

|  |
| --- |
| 核心代码① |
| void OutPutCur() {  printf("已分析过的串：");  fprintf(outFile, "已分析过的串：");    for(int i=0;i<head-1;i++) {  printf("%c ", Queue[i]);  fprintf(outFile,"%c ", Queue[i]);  }    printf("\n当前字符：%c\n", sym);  fprintf(outFile,"\n当前字符：%c\n", sym);    printf("剩余串：");  fprintf(outFile, "剩余串：");  for(int i=head;i<end;i++) {  printf("%c ", Queue[i]);  fprintf(outFile, "%c ", Queue[i]);  }  printf("\n");  fprintf(outFile,"\n");  } |
| 该函数主要用于语法分析的结果显示，printf()用于屏幕上的格式化输出，fprintf()用于文件中的格式化输出；函数主要输出实验要求的三种信息：已分析过的串、当前分析的字符、未分析的串。 |

|  |
| --- |
| 核心代码② |
| char Queue[MAXQUEUE];  int head = 0, end = 0;  void InitQueue();  void AddQueue();  void OutQueue();  void InitQueue() {  advance();  while (sym!=EOF) {  AddQueue();  advance();  }  }  void AddQueue() {  Queue[end++]=sym;  }  void OutQueue() {  sym=Queue[head++];  } |
| 此代码是核心代码①的基础，它简单地利用并实现了队列，以及队列初始化、入队、出队的功能函数，用于存储待分析串，并记录串的分析进度。 |

|  |
| --- |
| 核心代码③ |
| void E();  void E1();  void T();  void T1();  void F();  void error();  void E() {  printf("=================================================\n");  fprintf(outFile, "=================================================\n");  printf("使用的产生式：E --> TE\'\n");  fprintf(outFile, "使用的产生式：E --> TE\'\n");  OutPutCur();  printf("=================================================\n");  fprintf(outFile,"=================================================\n");  T();  E1();  }  …… |
| 这部分为语法分析最为核心的代码，利用递归的方式生成了语法分析树的结构，以类似深度优先算法的方式层层执行文法的推导式，每个非终结符都有对应的函数，通过特定终结符来判断不同的推导方向。每当成功执行一次推导过程，就输出相应的提示信息，直到推导至最后的终结符。 |

|  |
| --- |
| 测试文件①:sample1.txt |
| a+(b- 123)\* (456) |
| 该测试的理想情况是得到正确的推导过程，并写入到指定文档detail.txt中。并且验证程序能否识别和跳过空白符。 |

|  |
| --- |
| 测试文件②:sample2.txt |
| a+(b- 123)\* -456 |
| 该测试的理想情况是得到部分推导过程后程序发现错误并发出错误信息，最后将结果输出到detail.txt和error.txt中。用于验证程序对错误的识别和处理情况 |

|  |
| --- |
| 测试结果① |
|  |
| 程序正常执行，在屏幕和文件中均输出了分析过程，没有错误信息出现，符合实验预期结果。 |

|  |
| --- |
| 测试结果② |
|  |
| 程序正常运行，程序在屏幕和文件中均输出了推导过程和错误信息，符合实验的预期结果。 |

1. **总结及心得体会：**

①递归下降的语法分析虽然在编码上比词法分析器更为简单明了，但其算法意义上更为巧妙，尤其是递归的运用。递归虽然在性能上并没有突出优势，但更易于直观地理解算法的执行过程。

②对于串的分析进度处理，处理利用队列，其实还有许多思路可以选择，比如利用栈也可以实现已分析串和未分析串的存储。这些都可以锻炼我们对于数据结构的实现和应用。

③队列的另一个作用是对多字符的常量、单词的处理，队列能够化整为零，通过判断语句确定常量和单词。

1. **对本实验过程及方法、手段的改进建议：**

①错误信息输出不够详细，在错误类型、错误发生位置上有改进之处；

②单词、常量和运算符等分析完成后可以提供相关分类信息。

**报告评分：**

**指导教师签字：**