

《编译原理》课程实验报告



(2021年)

专 业: 计算机科学与技术

学 号:\_\_\_\_\_\_\_19063140\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

班 级:\_\_\_\_\_\_\_19185312\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

姓 名:\_\_\_\_\_\_\_郑凯心\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

上课时间: 周四第10,11,12节{第1-16周}

目录

[1. 课程实验概述 1](#_Toc74826438)

[2. 实验专题一、词法分析 2](#_Toc74826439)

[2.1 实验目的与内容 2](#_Toc74826440)

[2.2 程序总体设计思路和框架 3](#_Toc74826441)

[2.3 主要的数据结构和流程描述 3](#_Toc74826442)

[2.4 测试结果与说明 4](#_Toc74826443)

[2.5 实验收获与反思 6](#_Toc74826444)

[3. 实验专题三：递归下降⼦程序的语法分析技术 7](#_Toc74826445)

[3.1 实验目的与内容 7](#_Toc74826446)

[3.2 程序总体设计思路和框架 7](#_Toc74826447)

[3.3 主要的数据结构和流程描述 8](#_Toc74826448)

[3.4 测试结果与说明 8](#_Toc74826449)

[3.5 实验收获与反思 8](#_Toc74826450)

[4. 实验专题四 LL(1)语法分析实验 9](#_Toc74826451)

[4.1 实验目的与内容 9](#_Toc74826452)

[4.2 程序总体设计思路和框架 9](#_Toc74826453)

[4.3 主要的数据结构和流程描述 10](#_Toc74826454)

[4.4 测试结果与说明 10](#_Toc74826455)

[4.5 实验收获与反思 10](#_Toc74826456)

# 课程实验概述

实验概述部分说明自己的课程实验完成情况，介绍完成了哪些实验，每个实验之间关联度

报告正文用小四字号、宋体，每个专题写一章（如果多个实验专题在一个程序中实现的，可以写在一章中，并在标题中注明包括了哪几个实验专题）

实验材料提交要求:

1. 材料整理方式: 以“学号\_姓名”新建一个文件夹，将本文档命名为”学号\_姓名\_实验报告.docx或pdf“，放入该文件夹。每个程序源码一个子文件夹

提交以”学号\_姓名”为压缩文件名

1. 截止时间：6月21日晚上12点
2. 提交地址: [huangxx@hdu.edu.cn](mailto:huangxx@hdu.edu.cn)
3. 邮件主题: 编译原理实验报告-学号-姓名，邮件内容注明上课时间

# 实验专题一、词法分析

## 2.1 实验目的与内容

分析SysY语⾔的词法规范，列出词法单元

构建单词分类编码体系（关键词、标识符、整型常数、运算符、分隔符等）

实现词法分析器

功能

采⽤的实现语⾔不限

能够查出SysY源代码中可能包含的词法错误

错误1：出现词法中未定义的字符

错误2：任何不符合SysY语⾔词法单元定义

可额外完成以下要求

识别⼋进制、⼗六进制的整型常数

识别"//“和"/\*...\*/"两种形式的注释

如果输⼊的⽂件中包含这种形式的注释，需要能够过滤这部分内容

如果输⼊的⽂件中包含不符合要求的注释（如/\*...\*/中缺少其中的/\*或者\*/），需要给出错误的提示信息

输⼊

SysY语⾔的源代码⽂件.sy或者从控制台输入

输出

打印出程序分析结果

出错信息

格式: Error: [错误类型] at line [⾏号]： [错误说明]

错误说明可⾃定义，⿎励给出⾏号

⽆出错情况

输出单词符号序列

单词类别只包含⼀个单词的，只输出类别码即可 如每个关键词单独⼀类，则整型输出INT

单词类别包含多个单词的，输出⼆元组形式

如标识符ID，输出<ID, "abc">

如果识别的是整型常数，要求以⼗进制形式输出对应的数值，如<NUM, 123>

## 2.2 程序总体设计思路和框架

先处理注释，再进行词法分析

词法分析时分成两大类，一类是关键词，标识符，数值常量，另一类是运算符，分隔符

因为前者只包含字母数组和下划线，后者的字符集与前者无相交

## 2.3 主要的数据结构和流程描述

数据结构：列表，用于存储关键词，运算符，分隔符集合等

key\_words存储关键词集合

operator存储运算符集合

delimiter存储分隔符集合

判断分类编码体系的方法

判断是否在编码体系的集合里

用正则表达式判断

流程：

读入数据

使用python打开指定文件用readlines()函数保存为字符串列表

识别注释

为方便处理将字符串列表形式的代码合并成字符串

从前往后寻找”//”, “/\*”符号

如果找到”//”则将该行之后的删除(直到下一个’\n’位置)

如果找到”/\*”则找之后的”\*/”将之间的字符用空格替换，这样能防止行号改变，方便错误定位

词法分析

一行一行处理

寻找非字母数字下划线的字符，依次作为间隔字符将代码分成一个个词

对两个间隔字符之间的字符串进行关键词，标识符，数值常量的判断

对于间隔字符进行运算符，分隔符的判断

## 2.4 测试结果与说明

测试用例1

int a;

int main()

{

  a = 10;

                b = 0x11ff;

                c = 027;

                // d = 20;

               /\* e = 30;

              \*/

  if ( a><0 )

               {

    return 1a;

  }

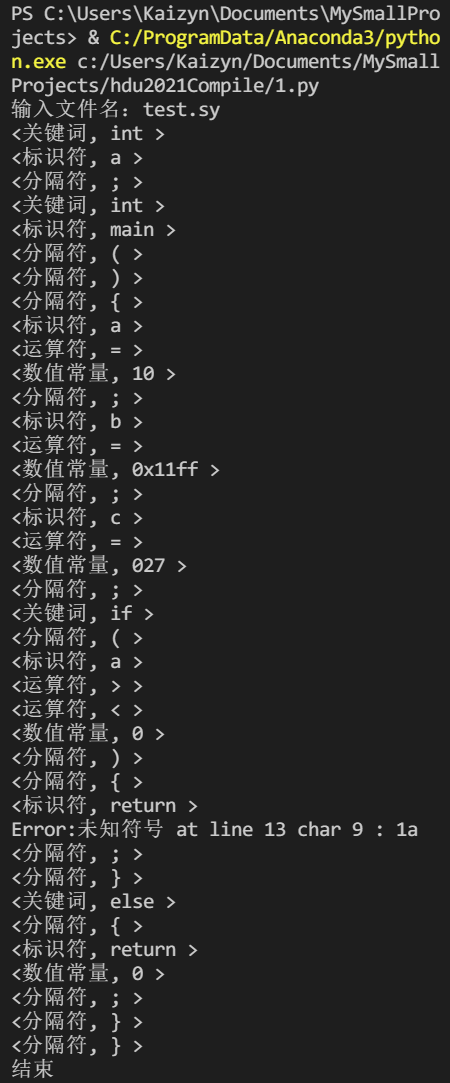
  else{

    return 0;

  }

}

运行结果



测试用例2

int main(){

    int a=10;

    while(a>0){

        if(a>5){

            a=a-1;

            continue;

        }

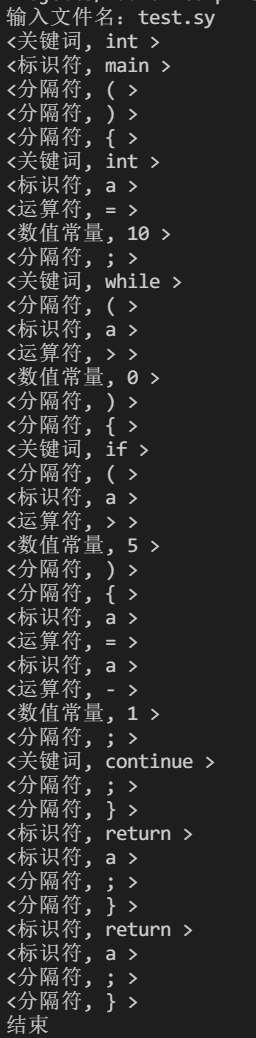
        return a;

    }

    return a;

}

运行结果



测试用例3

// look here /\*

say hello

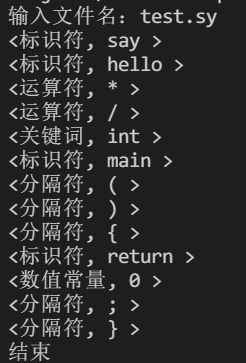
\*/

int main(){

  return 0;

}

运行结果



测试用例4

/\*

look here

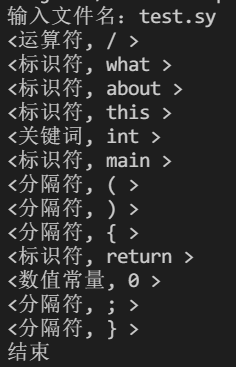
\*// what about this

int main(){

  return 0;

}

运行结果



测试用例5

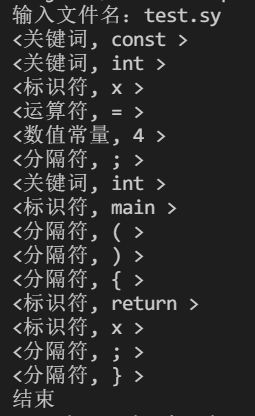
const int x=4;

int main(){

    return x;

}

运行结果



## 实验收获与反思

词法分析的基本任务是从字符串表示的源程序中识别出具有独立意义的单词符号,其基本思想是根据扫描到单词符号的第一个字符的种类，拼出相应的单词符号。通过这次实验让我对词法分析的理解更深了一步，这也是做实验和学习这门课程的目的所在。一步步完善实验的过程也是在增强自己的知识。

# 3. 实验专题三：递归下降子程序的语法分析技术

## 3.1 实验目的与内容

目的

掌握最基本的自顶向下分析方法，即递归下降子程序方法，理解其特点和适用范围（回溯，左递归等现象），锻炼递归调用程序的构造方法。

内容

给定SysY语言中简单算术表达式文法G[E]:

E→TE’

E’→ATE’|ε

T→FT’

T’→MFT’ |ε

F→(E) | i

A → + | -

M → \* | /

根据该文法，编写递归下降分析子程序。

【说明】

终结符号i为用户定义的简单变量，即专题1中标识符的定义

输入：是词法分析输出的二元组序列，即任意简单算术表达式经过专题1程序输出后得到的结果。【上述文法中i即对应词法分析的标识符， +-\*/分别对应词法分析得到的运算符】

输出：判定输入串是否为该文法定义的合法算术表达式

处理：程序应能发现输入串的错误

设计5个以上的测试用例(尽可能完全，包括正确和出错情况)，给出测试结果。

1. 输入：是词法分析输出的二元组序列，即任意简单算术表达式经过专题1程序输出后得到的结果。【上述文法中i即对应词法分析的标识符， +-\*/分别对应词法分析得到的运算符】

2. 处理：基于分析表进行 LL(1)语法分析，判断其是否符合文法。

3. 输出：串是否合法。

## 3.2 程序总体设计思路和框架

根据递归下降子程序要求，为每个非终结符号写一个函数，然后根据文法右部写函数

## 3.3 主要的数据结构和流程描述

数据结构：无

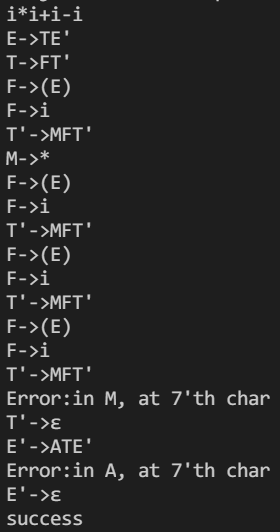
流程：根据递归下降子程序要求，为每个非终结符号写一个函数，然后根据文法右部写函数

## 3.4 测试结果与说明

测试用例1

i\*i+i-i

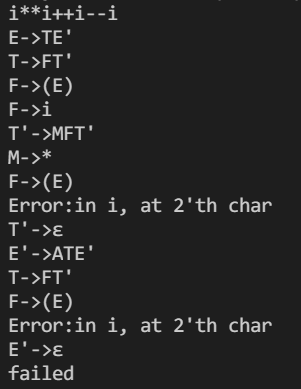
运行结果



测试用例2

i\*\*i++i--i

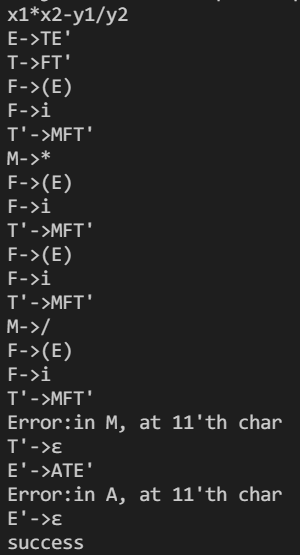
运行结果



测试用例3

x1\*x2-y1/y2

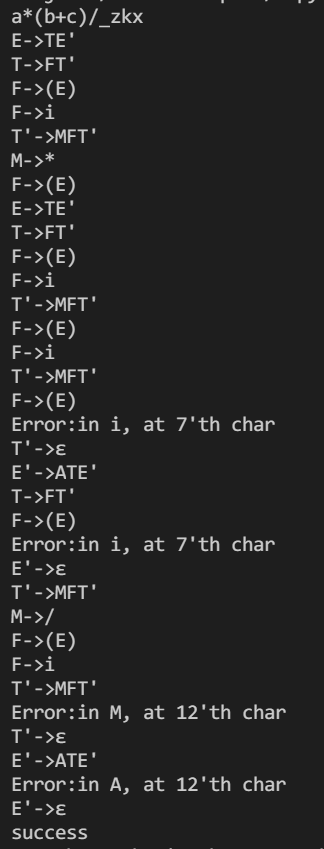
运行结果



测试用例4

a\*(b+c)/\_zkx

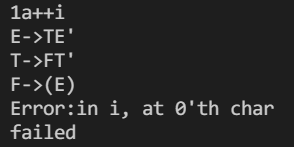
运行结果



测试用例5

1a++i

运行结果



## 3.5 实验收获与反思

通过实验专题三递归下降子程序的语法分析技术，我掌握最基本的自顶向下分析方法，即递归下降子程序方法，理解了他的特点和适用范围（回溯，左递归等现象），通过这次实验我害锻炼了递归调用程序的构造方法。

在编程实现递归下降子程序的语法分析时我发现，整个技术的代码实现难度并不是很大，调试难度也不大，代码功能非常直观，逻辑清晰，但相应的也造成了递归下降子程序的语法分析的效率有限，对于每条规则都要递归遍历所有的选项。期待在之后学到的语法分析技术能够实现更加高效的语法分析。

# 4. 实验专题四 LL(1)语法分析实验

## 4.1 实验目的与内容

目的

1. 了解 LL(1)语法分析是如何根据语法规则逐一分析词法分析所得到的单词，检查语法错误，即掌握语法分析过程。

2. 掌握LL(1)语法分析器的设计与调试。

内容

针对SysY语言中简单算术表达式文法G[E]:

E→TE’

E’→ATE’|ε

T→FT’

T’→MFT’ |ε

F→(E) | i

A → + | -

M → \* | /

求解相应的FIRST、FOLLOW集，构造预测分析表，并编写LL(1)语法分析程序，并给出测试句子的分析过程。（注：如果有选做专题6关于LL(1)文法判断的同学，可以将专题6的部分整合到这个实验的前面，自动产生预测分析表，相当于把这个程序做成一个通用的LL(1)分析器）

1. 输入：是词法分析输出的二元组序列，即任意简单算术表达式经过专题1程序输出后得到的结果。【上述文法中i即对应词法分析的标识符， +-\*/分别对应词法分析得到的运算符】

2. 处理：基于分析表进行 LL(1)语法分析，判断其是否符合文法。

3. 输出：串是否合法。

## 4.2 程序总体设计思路和框架

求解相应的FIRST、FOLLOW集，构造预测分析表。

再根据分析表通过栈的辅助完成对句子的分析

## 4.3 主要的数据结构和流程描述

数据结构：

terminals一维列表，表示终结符集合

non\_terminals一维列表，表示非终结符集合

parsing\_table二维列表，存储预测分析表

s 字符串，存储输入串

stk 列表，用于模拟栈

流程：

1. 输入句子
2. 将符号E入栈
3. 进行预测分析
   1. 根据栈顶元素以及串首元素在预测表中找到对应的动作
   2. 如果有对应的动作则输出并执行，否则报错
   3. 如果栈顶是终结符，比较栈顶元素和串首元素，如果匹配则输出，否则报错

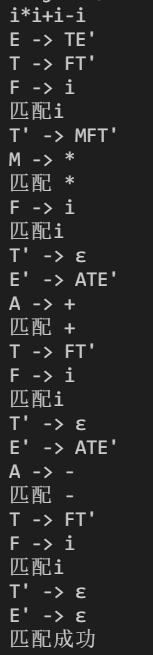
4. 判断句子是否匹配成功

## 4.4 测试结果与说明

测试用例1

i\*i+i-i

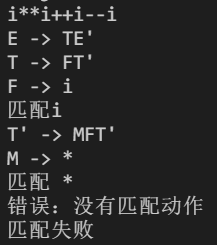
运行结果



测试用例2

i\*\*i++i--i

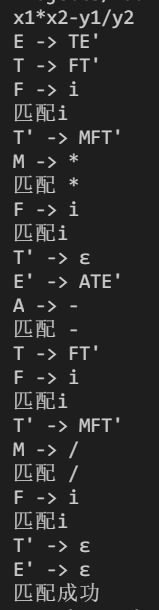
运行结果



测试用例3

x1\*x2-y1/y2

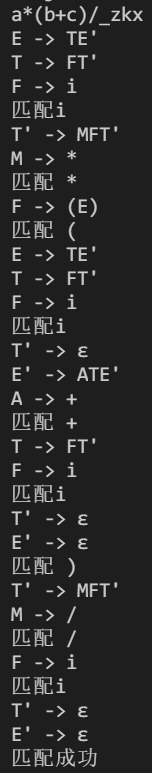
运行结果



测试用例4

a\*(b+c)/\_zkx

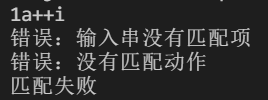
运行结果



测试用例5

1a++i

运行结果



## 4.5 实验收获与反思

通过实验专题四LL(1)语法分析实验，我了解到了 LL(1)语法分析是如何根据语法规则逐一分析词法分析所得到的单词，检查语法错误，掌握了语法分析过程。通过这次的实验，我掌握了LL(1)语法分析器的设计与调试。

比起上一个实验，实验专题三递归下降子程序的语法分析技术，很明显的一点就是这个比起来这个实验要稍微复杂一点，你需要求解相应的FIRST、FOLLOW集，构造预测分析表，并编写LL(1)语法分析程序，并给出测试句子的分析过程。

通过本次实验，我觉得难点并非在于代码编程，此实验的编程条理清洗，只需按照书中给出的分析过程一步一步写下来即可完成。但此实验的关键却在于FIRST、FOLLOW集，预测分析表的构造，这就需要你对课程的学习，对LL(1)文法足够了解才能得到。也许代码上仅仅只是一个数据结构表示的小小分析表，背后确实LL(1)文法的精髓，手工的推导。

与递归下降子程序的语法分析技术相比，LL(1)语法分析省去了很多没必要的分支，因此整个分析过程更加快速，精简。