



题 目:	编译原理上机报告
姓 名:	刘宇鹏
院 系:	计算机学院
班 级:	191181
学 号:	20181003174
指导老师:	刘远兴

2020 年 12 月 6 日

# 一、 作者简介

刘宇鹏，计算机学院计算机科学与技术专业 18 级学生，班号 191181，学号 20181003174.

# 二、 报告摘要

本报告为对编译原理课程第二次上机实验的分析总结，包括实验内容及要求、实验代码、实验小组分工实验具体设计思路以及实验结果。

# 三、 报告目录

一、	作者简介.....	2
二、	报告摘要.....	2
三、	报告目录.....	2
四、	报告正文.....	3
1.	小组分工： .....	3
2.	需求分析： .....	3
3.	设计： .....	3
3.1	总体设计： .....	3
3.2	自己负责的模块设计——算符优先表构造模块 .....	5
3.3	调试分析： .....	8
3.4	使用手册： .....	8
3.5	测试结果.....	8
五、	总结.....	10
六、	致谢 .....	11

## 四、 报告正文

题目：构造算符优先分析程序问题

实习时间：2020.11.27

### 【问题描述】

构造算符优先分析程序问题的一种描述是：设计实现一个可以进行算符优先文法识别的算符优先分析程序。

### 【基本要求】

- ① 根据给定文法，先求出 **FirstVt** 和 **LastVt** 集合。
- ② 构造算符优先关系表（要求算符优先关系表可以输出到屏幕或者输出到文件）；
- ③ 根据算法和优先关系表分析给定表达式是否是该文法识别的正确的算术表达式
- ④ ③中的移进归约过程要输出到控制台

## 1. 小组分工：

191181 第三组 组长：吴勇

转换文法：解天宇

求 **FIRSTVT**、**LASTVT** 集：杨彤、吴勇

求算符优先表并输出：陶叶、刘宇鹏

移进归约过程：黄李波

## 2. 需求分析：

- 1) 求出 **FIRSTVT** 和 **LASTVT** 集合
- 2) 构造该算符优先文法的优先关系矩阵
- 3) 根据算法和优先关系表分析给定表达式是否是该文法识别的正确的算术表达式，以分析表的形式来逐步分析并输出移进和归约过程
- 4) 算符优先分析程序应能发现按输入串出错

## 3. 设计：

### 3.1 总体设计：

#### 1) 设计思想：

根据实验需求，需要写出以下几个模块：

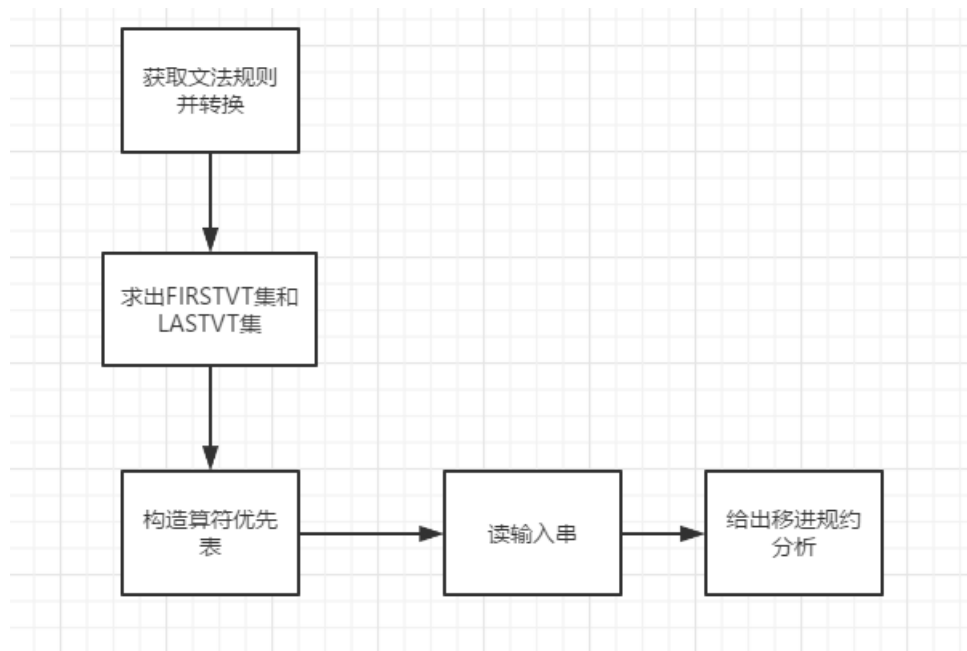
文法转换模块

求 **FIRSTVT** 和 **LASTVT** 集

求算符优先表模块

移进归约分析表构造模块

#### 2) 流程图：



### 3) 文件结构:

所有的函数都在“源.cpp”中

### 4) 数据结构:

<code>map&lt;char, char&gt; mymap;</code>	//用于存储 firstvt 和 lastvt 集,方便输出
<code>char Data[20][20];</code>	//算符优先关系
<code>char s[100];</code>	//模拟符号栈 s
<code>char lable[20];</code>	//文法终极符集
<code>char input[100];</code>	//文法输入符号串
<code>char In_str[20][10];</code>	//用于输入串的分析
<code>int r;</code>	//文法规则个数
<code>int r1;</code>	//转化后文法规则个数
<code>char st[10][30];</code>	//用来存储文法规则
<code>char first[10][10];</code>	//文法非终结符 FIRSTVT 集
<code>char last[10][10];</code>	//文法非终结符 LASTVT 集
<code>int fflag[10] = { 0 };</code>	//标志第 i 个非终结符的 FIRSTVT 集是否
已求出	
<code>int lflag[10] = { 0 };</code>	//标志第 i 个非终结符的 LASTVT 集是否
已求出	
<code>int deal();</code>	//对输入串的分析
<code>int Terminator(char c);</code>	//判断字符 c 是否是终极符
<code>int getIndex(char c);</code>	//求字符 c 在算符优先关系表中的下标

数据处理的流程如下:

- 先使用 st 数组来存储文法规则
- 然后把文法的终结符集识别出来放到 lable 数组中
- 将文法转为单候选式文法存放在 text 数组中
- 将产生式右部存放在 In\_str 数组中
- 算符优先关系存放在 Data 数组

f) 求 firstvt、lastvt 集放在 mymap 容器中

### 5) 基本思想

通过模拟算符优先算法的分析处理过程，对文法进行处理后求出 FIRSTVT 和 LASTVT 集，然后得出算符优先关系表，再由此表来模拟构造分析栈中的移进-归约过程，从而达到算符优先分析的目的。

## 3.2 自己负责的模块设计——算符优先表构造模块

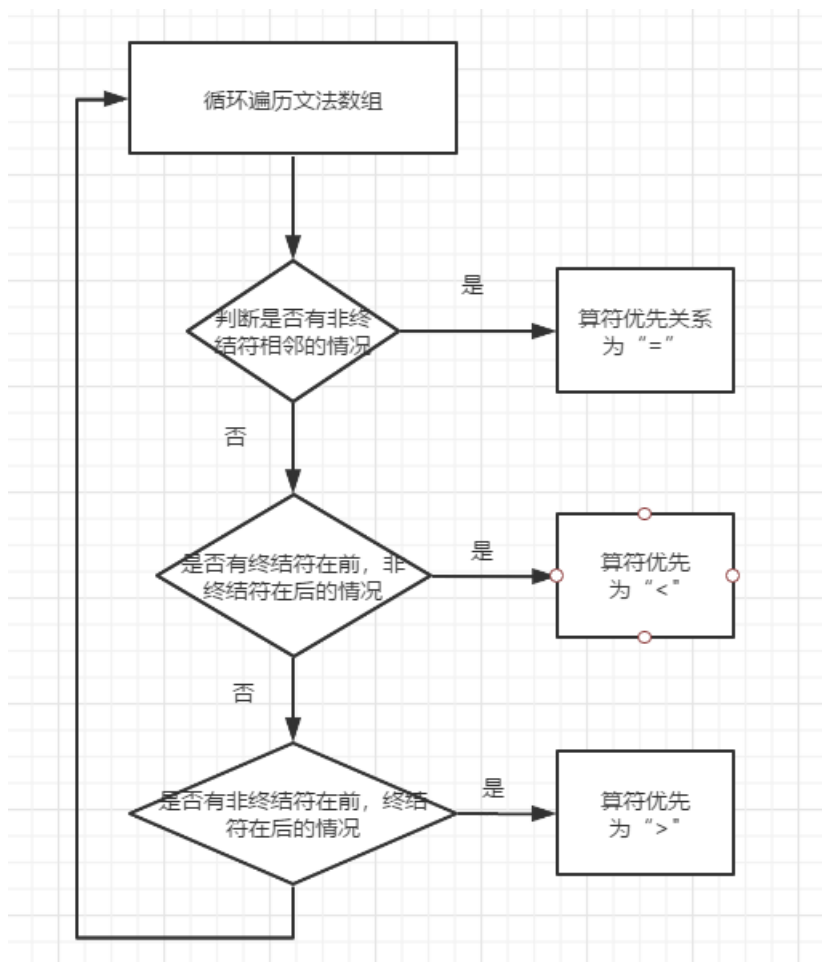
算符优先表构造：需要先调用组员写的 firstvt 函数和 lastvt 函数求出 FIRSTVT 集合 LASTVT 集，然后对每两个终结符进行比较，其主要算法为书上的算符优先分析表构造算法：

对前后相邻的两个终结符优先关系为 “=”；

对非终结符在终结符前的，该非终结符的 LASTVT 集中元素和非终结符的优先关系为 “>” ；

对终结符在非终结符前的，该终结符与该非终结符的 FIRSTVT 集中元素的优先关系为 “<”。

### 1) 思想流程图：



### 2) 算法设计与代码如下：

使用 for 循环对转换后的文法逐条进行遍历：

```
for (i = 0; i < x; i++)
```

```
{
    for (j = 1; text[i][j + 1] != '\0'; j++)
    {
```

给出两个相邻的终结符优先关系：

```
        if (Terminator(text[i][j]) && Terminator(text[i][j + 1]))
        {
            m = getIndex(text[i][j]);
            n = getIndex(text[i][j + 1]);
            Data[m][n] = '=';
        }
        if (text[i][j + 2] != '\0' && Terminator(text[i][j]) && Terminator(text[i][j + 2])
            && !Terminator(text[i][j + 1]))
        {
            m = getIndex(text[i][j]);
            n = getIndex(text[i][j + 2]);
            Data[m][n] = '=';
        }
    }
```

给出非终结符在终结符后时终结符与非终结符的 **FIRSTVT** 集元素优先关系：

```
        if (Terminator(text[i][j]) && !Terminator(text[i][j + 1]))
        {
            for (k = 0; k < r; k++)
            {
                if (st[k][0] == text[i][j + 1])
                    break;
            }
            m = getIndex(text[i][j]);
            for (t = 0; t < first[k][0]; t++)
            {
                n = getIndex(first[k][t + 1]);
                Data[m][n] = '<';
            }
        }
    }
```

给出终结符在非终结符后时非终结符的 **LASTVT** 集元素与终结符优先关系：

```
        if (!Terminator(text[i][j]) && Terminator(text[i][j + 1]))
        {
            for (k = 0; k < r; k++)
            {
                if (st[k][0] == text[i][j])
                    break;
            }
            n = getIndex(text[i][j + 1]);
            for (t = 0; t < last[k][0]; t++)
```

```

        {
            m = getIndex(last[k][t + 1]);
            Data[m][n] = '>';
        }
    }
}
}
//将#E#开始文法也加上去求出其对应的 firstvt 和 lastvt 集。

```

```

m = getIndex('#');
for (t = 0; t < first[0][0]; t++)
{
    n = getIndex(first[0][t + 1]);
    Data[m][n] = '<';
}
n = getIndex('#');
for (t = 0; t < last[0][0]; t++)
{
    m = getIndex(last[0][t + 1]);
    Data[m][n] = '>';
}
Data[n][n] = '=';

```

3) 函数声明原型与功能实现:

**模块中的 Terminator 函数原型与功能实现为**

```

int Terminator(char c) //判断字符 c 是否是终极符
{
    int i;
    for (i = 0; lable[i] != '\0'; i++)
    {
        if (c == lable[i])
            return 1;
    }
    return 0;
}

```

**模块中的 getIndex 函数原型与功能实现为:**

```

int getIndex(char c) //求字符 c 在文法终极符表中的下标
{
    int i;
    for (i = 0; lable[i] != '\0'; i++)
    {
        if (c == lable[i])
            return i;
    }
    return -1;
}

```

### 3.3 调试分析：

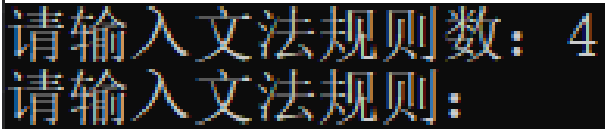
- (1) 优点分析：可以输出出错信息，操作简单，功能全面
- (2) 缺点分析：没有从文件中读取文法的能力，还需要手动输入
- (3) 改进的方法：增加从文件中读取文法的功能直接读取文法

### 3.4 使用手册：

- (1) 先输入文法产生式的个数
- (2) 逐个输入产生式
- (3) 输入要识别的输入串
- (4) 输出过程与结果

### 3.5 测试结果

输入文法产生式个数：



```
请输入文法规则数： 4
请输入文法规则：
```

输入文法产生式：



转化后的文法为:

```
E -> E
E -> E+T
E -> T
T -> T*F
T -> F
T -> (E)
T -> i
```

每个非终结符的FIRSTVT集为:

```
E: + * ( i
E: + * ( i
T: * ( i
T: ( i
```

每个非终结符的LASTVT集为:

```
E: + * ) i
E: + * ) i
T: * ) i
T: ) i
```

算符优先分析表如下:

	+	*	(	)	#
+	>	<	<	>	>
*	>	>	<	>	>
(	<	<	<	=	>
)	>	>	>	>	>
#	<	<	<	=	=

请输入文法输入符号串以#结束:

输入要识别的输入串:

```

请输入文法输入符号串以#结束:(i+i)*i#
#           (i+i)*i#           移进
#(          i+i)*i#           移进
#(i         +i)*i#           规约
#(F         +i)*i#           移进
#(F+        i)*i#           移进
#(F+i       )*i#           规约
#(F+F       )*i#           规约
#(E         )*i#           移进
#(E)        *i#           规约
#F          *i#           移进
#F*         i#           移进
#F*i        #           规约
#F*F        #           规约
#T          #           结束

```

输入串符合文法的定义！  
 请输入文法输入符号串以#结束：

输入一个会出错的输入串：i+i)\*i#

```

请输入文法输入符号串以#结束:i+i)*i#
#           i+i)*i#           移进
#i          +i)*i#           规约
#F          +i)*i#           移进
#F+         i)*i#           移进
#F+i        )*i#           规约
#F+F        )*i#           规约

```

输入串不符合文法的定义！  
 请输入文法输入符号串以#结束：\_

## 五、 总结

通过本次实验，我掌握了算符优先文法分析的过程，包括文法产生式的处理、FIRSTVT 集与 LASTVT 集的构造、算符优先表的构造以及对输入串的分析过程，并可以把算符优先表的构造过程用代码实现出来。之前对算符优先算法中算符优先关系表的理解不透彻，经过这次实验，我对算符优先文法的分析处理过程有了深刻的理解。从这次实验中我更加深刻地

体会到了对计算机人来说编译原理的重要性，它对我们的编程能力提升是其他学科所不能比拟的。

## 六、 致谢

感谢刘老师几个月以来的辛勤付出与谆谆教导，编译原理这门课让我受益匪浅，懂得了很多底层编译知识，让我能在以后的学习中得到很大帮助，希望以后有机会能与您合作。