天津理工大学 实验报告

学院名称：计算机科学与工程学院

姓名	王帆	学号	20152180	专业	计算机科学与技术
班级	2015 级 1 班	实验项目	实验二：语法分析		
课程名称		编译原理		课程代码	0668056
实验时间		2018 年 5 月 30 日　第 1、2 节 2018 年 6 月　4　日　第 5、6 节		实验地点	软件实验室　7-219 软件实验室　7-212

实验成绩考核评定分析

实验过程综合评价 30 分	实验目标结果评价 20 分	程序设计规范性评价 20 分	实验报告完整性评价 30 分	实验报告雷同分析分类标注	实验成绩
■实验过程认真专注，能独立完成设计与调试任务　30 分 ■实验过程认真，能较好完成设计与编成调试任务　25 分 ■实验过程较认真，能完成设计与编成调试任务　20 分 ■实验过程态度较好，基本完成设计与编成调试任务　15 分 ■实验过程态度欠端正，未完成设计与编成调试任务　10 分	■　功能完善，且人机交互界面友好 20 分 ■　满足功能要求，但人机交互界面一般 15 分 ■基本满足功能需求，人机交互界面欠缺　10 分 ■功能缺失　5 分	■程序易读性好 20 分 ■程序易读性较好 15 分 ■程序易读性欠缺 10 分 ■程序易读性较差　5 分 **注：易读性要求标识符命名见名知意，程序编制采用嵌套方式，层次结构清晰可读，关键部分具有简明注释。	■报告完整 30 分 ■报告较完整 25 分 ■报告内容一般　20 分 ■报告内容极少　10 分	凡雷同报告将不再重复评价前四项考核内容，实验成绩将按低学号雷同学生成绩除雷同人数计算而定。 标记为： **S 组号-人数(组分)**	前四项评价分数之总和 (** 雷同报告按第五项标准核算**)

实验内容：

可选择 LL1 分析法、算符优先分析法、LR 分析法之一，实现如下表达式文法的语法分析器：

（1）E→E+T | E-T | T

（2）T→T*F | T/F | F

（3）F→P^F | P

（4）P→(E) | i

实验目的：

1．掌握语法分析的基本概念和基本方法；

2．正确理解 LL1 分析法、算符优先分析法、LR 分析法的设计与使用方法。

实验要求：

1．按要求设计实现能识别上述文法所表示语言的语法分析器，并要求输出全部分析过程；

2．要求详细描述所选分析方法针对上述文法的分析表构造过程；

3．完成对所设计语法分析器的功能测试，并给出测试数据和实验结果；

4．为增加程序可读性，请在程序中进行适当注释说明；

5．按软件工程管理模式完成实验报告撰写工作，最后需要针对实验过程进行经验总结；

6．认真完成并按时提交实验报告。

****** 实验报告内容详述 ******

一、需求分析

概要介绍：

编译程序在对源程序完成了词法分析之后的下一个步骤是语法分析。语法分析是编译过程的一个逻辑阶段，是在词法分析的基础上将单词序列组合成各类语法短语，如“程序”，“语句”,“表达式”等等。语法分析程序判断源程序在结构上是否正确，源程序的结构由上下文无关文法描述。

语法分析主要可以通过自顶向下分析与自底向上分析两种方式完成。自顶向下分析是根据形式语法规则，在语法分析树的自顶向下展开中搜索输入符号串可能的最左推导。单词按从左到右的顺序依次使用。自底向上分析是从现有的输入符号串开始，尝试将其根据给定的形式语法规则进行改写，最终改写为语法的起始符号。

实现意义：

语法分析是编译器的重要组成部分，它承上启下，接续词法分析器得到的字符串，通过语法检测后，交由语义分析部分进行翻译生成中间代码，再进行优化处理形成目标代码，从而实现从高级程序设计语言到汇编语言/机器语言的过程。它是

功能概述：

在词法分析输出的基础上，语法分析器可以根据文法进行确定是否能够以语法的起始符号推导出输入，并分析其是否正确组成了语法组分（短语、子句、程序段、程序）。根据语言的语法规则，逐一扫描源程序的 ASCII 码序列或词法分析后的 TOKEN 序列，从而确定其如何形成语句与程序。

二、可行性分析

技术可行性：

通过学习语法分析器的原理和实现方法，通过高级程序设计语言对其进行编程实现较为可行，且可以通过图形化界面设计优化同用户的交互体验，实现对语法分析过程的展示，技术可行性高。

经济可行性：

语法分析器实现成本一般，但其学习价值高。通过对编译原理综合实验的封装与打包，适合在基本的 X86/AMD64 兼容机，Win 平台上执行演示，基于计算机系统的成本低，经济可行性高。

操作可行性：

由于交互友好，用户使用本语法分析器作为演示软件的可能性较高，开发时间进度方面也是可行的，故操作可行性高。

社会可行性：

作为一个教学学习软件，本软件不涉及共享软件与其他付费软件的知识产权侵权责任，故社会可行性高。

综合以上几点，本实验实现的可行性高。

三、概要设计

系统模块结构设计：

本系统包括以下几个模块：字符串读取模块、分析结果模块、交互提示模块

代码设计

根据给定文法，消除左递归，并使用 LL(1)分析法进行判断。

算法实现流程如下：

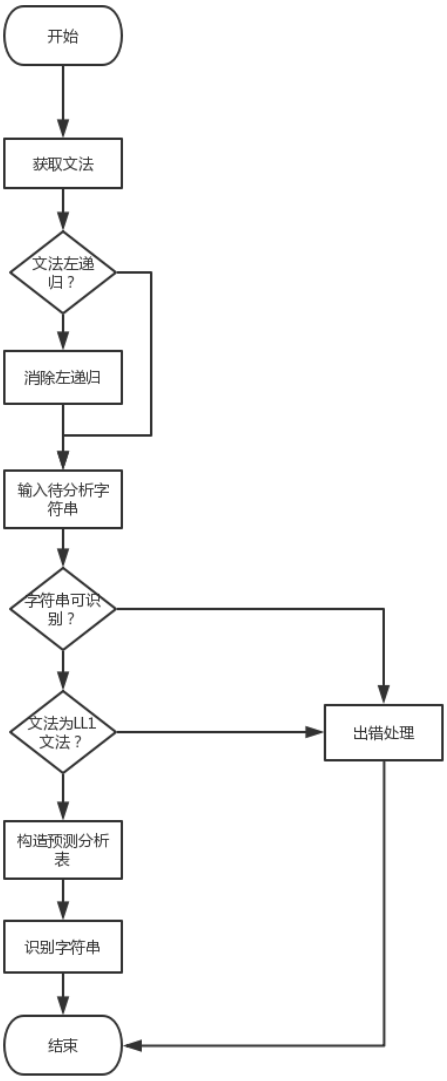

图 1 概要设计算法流程图

系统可靠性与内部控制设计：

由于用户可能存在误操作与其他将导致系统崩溃的因素，对本系统可靠性设计主要包括对交互的提示，对代码的异常捕获，以及对边界条件的细化与提示等。

四、详细设计

交互界面设计：

顶部菜单为操作按键，初始化包括对文法左递归消除处理，以及对带分析字符串的输入；输入字符串后，点击开始按钮，开始对字符串进行分析。点击步进进行下一步分析，直到提示“分析结束”，可在左侧参数列最下方的“分析结果”一栏看到该串的分析结果，包括“分析成功”与“分析失败”；点击“重置”按钮，对当前字符串以及分析结果进行重置。

右侧分析结果部分，能够以表格的形式显示语法分析过程中“符号栈”、“输入串”、“产

生式”的变化过程。

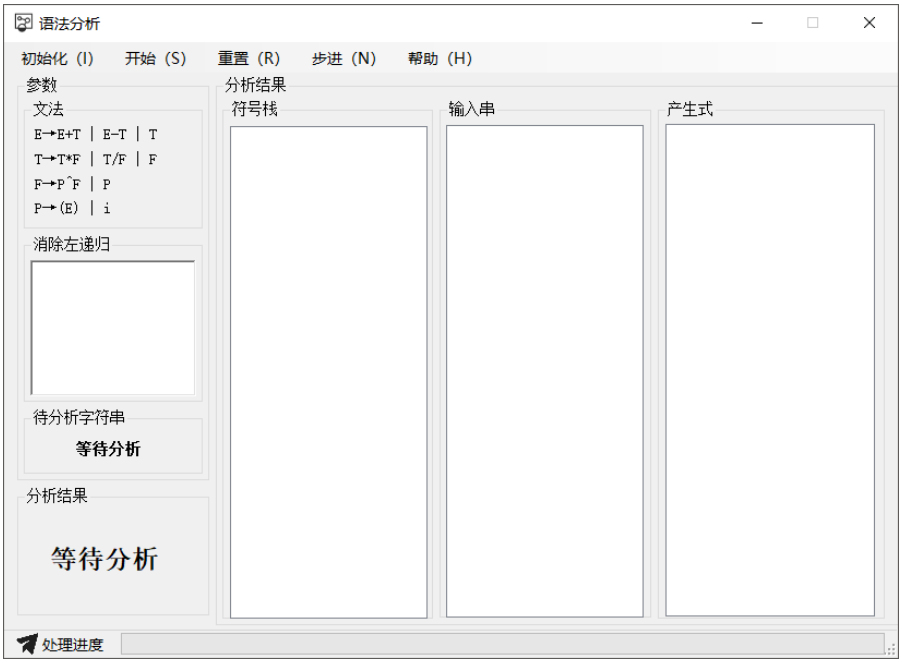

图 2 语法分析主界面

核心功能实现：

消除左递归

1. 消除直接左递归：

将所有产生式写成候选式形式，如：T -> Tα1 | Tα2 | ... | Mαn | β1 | β2 | ... |βm。其中，α 非 ε，且 β 不以 T 开头。则非终结符 T 可改写为

T -> β1T’ | β2T’ | ... | βmT’

T’ -> α1T’ | α2T’ | ... | αnT’ | ε

2. 消除间接左递归：

(1) 以某种顺序排列非终结符 A1，A2，……，An；

(2) 使用以下算法进行迭代处理

for i ＝ 1 to n do{

for j ＝ 1 to i - l do{

用产生式 Ai→a1b｜a2b｜……｜akb 代替每个形如 Ai→Ajb 的产生式

其中，Aj→a1｜a2｜……｜ak 是所有的当前 Aj 产生式；

}

消除关于 Ai 产生式中的直接左递归性；

}

}

(3) 化简由步骤 2 所得到的文法。

构造 FIRST 集与 FOLLOW 集

(1) 对于 G 中的任意非终结符 A，其 FIRST（α）= {a|α 经过 0 或多步推导为 a...的形式，其中 a∈VT}

(2) 对于 G 中的任意非终结符 A，其 FOLLOW（A）= {a|S 经过 0 或多步推导会出现 ...Aa... 的形式，其中 a∈VT 或#号}

构造 LL(1)分析表

对于 G 中的每一个产生式， A -> α ,执行以下 2 步：

for ∀ a ∈ FIRST(α)， 将 A -> α 填入 M [A, a];

if（ε ∈ FIRST(α)）∀ a ∈ FOLLOW (A) ， 将 A -> ε 填入 M [A, a];

分析过程

定义相关全局变量，使用二位数组定义 LL1 分析表

定义开始按钮点击事件：点击后开始读入输入的字符串，将#压入栈，开始符号 E 压入栈

定义步进按钮点击事件，点击按钮从当前符号串中读取符号，“#”则分析成功，否则将栈顶元素和读取的符号进行匹配，然后读取下一符号。若当前栈顶符为非终结符，则查询分析表：

1. 如果表中含有相应的产生式，则弹出栈顶元素，并将产生式右部符号串按逆序逐一压入栈；
2. 如果产生式右部是 ε ，则只弹出栈顶元素；
3. 如果表中对应产生式为空，则语法错误，调用出错处理。点击按钮直到显示出错或分析成功。

定义重置按钮点击事件，初始化分析结果与字符串。

五、测试用例与测试结果

正确测试用例：i*(i+i)/(i^i)

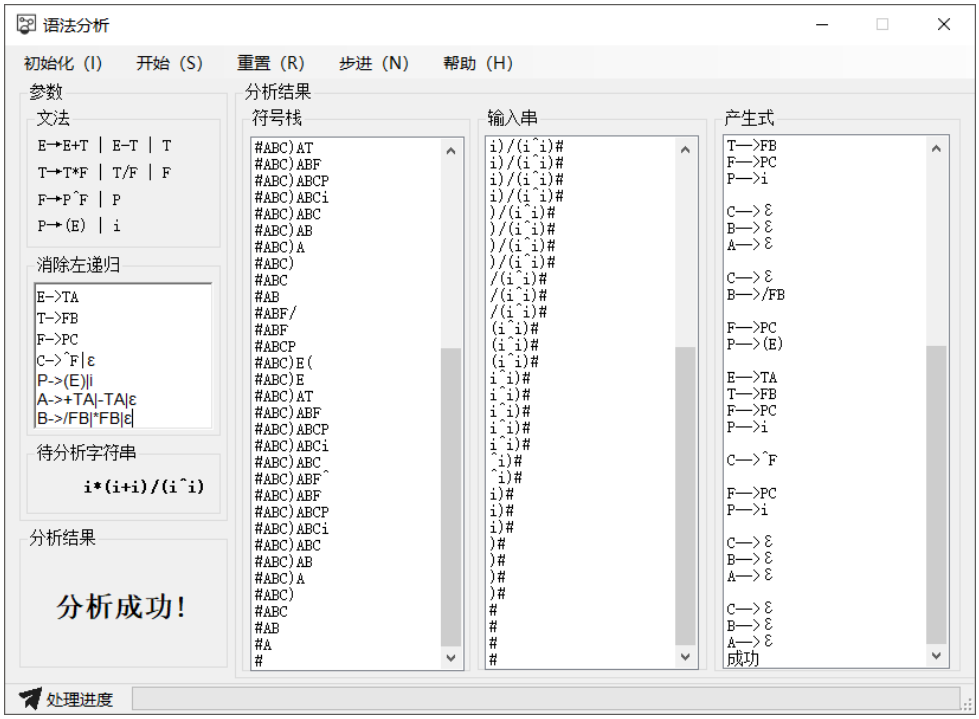

图 3 正确字符串测试结果

错误测试用例：(-i)*)i(

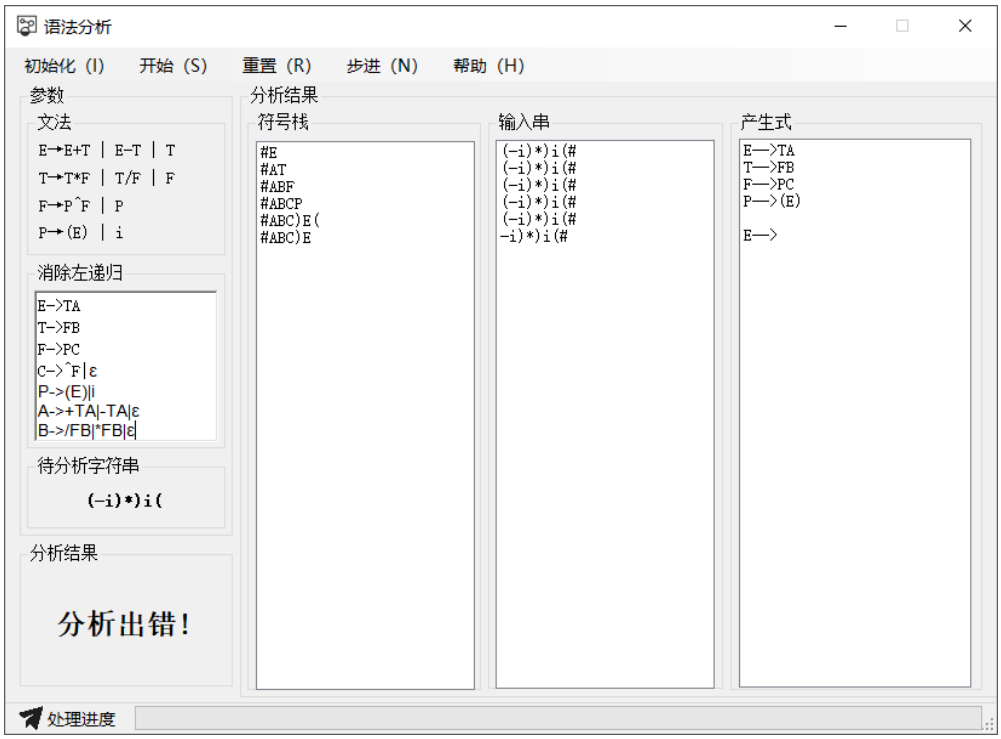

图 4 错误字符串测试结果

六、实验代码

```
//窗体设计器代码（由 VS 自动生成）
namespace CompilerLab
{
    partial class mainForm
    {
        /// <summary>
        /// 必需的设计器变量。
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 清理所有正在使用的资源。
        /// </summary>
        /// <param name="disposing">如果应释放托管资源，为 true；否则为 false。</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Windows 窗体设计器生成的代码

        /// <summary>
        /// 设计器支持所需的方法 - 不要修改
        /// 使用代码编辑器修改此方法的内容。
        /// </summary>
         //初始化界面
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(mainForm));
            this.menuStrip = new System.Windows.Forms.MenuStrip();
            this.文件ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.打开ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.退出ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.词法分析WToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.语法分析GToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.语义分析MToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.帮助ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.帮助文档ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.关于ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.statusStrip = new System.Windows.Forms.StatusStrip();
            this.toolStripStatusLabel = new System.Windows.Forms.ToolStripStatusLabel();
            this.toolStripProgressBar = new System.Windows.Forms.ToolStripProgressBar();
            this.richTextBox = new System.Windows.Forms.RichTextBox();
            this.测试ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.processbarToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.menuStrip.SuspendLayout();
            this.statusStrip.SuspendLayout();
            this.SuspendLayout();
            //
            // menuStrip
            //
            this.menuStrip.ImageScalingSize = new System.Drawing.Size(18, 18);
            this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.文件ToolStripMenuItem,
            this.词法分析WToolStripMenuItem,
            this.语法分析GToolStripMenuItem,
            this.语义分析MToolStripMenuItem,
            this.帮助ToolStripMenuItem,
            this.测试ToolStripMenuItem});
            this.menuStrip.Location = new System.Drawing.Point(0, 0);
            this.menuStrip.Name = "menuStrip";
            this.menuStrip.Padding = new System.Windows.Forms.Padding(7, 2, 0, 2);
            this.menuStrip.Size = new System.Drawing.Size(728, 28);
```

```
            this.menuStrip.TabIndex = 0;
            this.menuStrip.Text = "menuStrip1";
            //
            // 文件ToolStripMenuItem
            //
            this.文件ToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.打开ToolStripMenuItem,
            this.退出ToolStripMenuItem});
            this.文件ToolStripMenuItem.Name = "文件ToolStripMenuItem";
            this.文件ToolStripMenuItem.Size = new System.Drawing.Size(84, 24);
            this.文件ToolStripMenuItem.Text = "文件（&F）";
            //
            // 打开ToolStripMenuItem
            //
            this.打开ToolStripMenuItem.Name = "打开ToolStripMenuItem";
            this.打开ToolStripMenuItem.Size = new System.Drawing.Size(110, 24);
            this.打开ToolStripMenuItem.Text = "打开";
            this.打开ToolStripMenuItem.Click += new System.EventHandler(this.打开
ToolStripMenuItem_Click);
            //
            // 退出ToolStripMenuItem
            //
            this.退出ToolStripMenuItem.Name = "退出ToolStripMenuItem";
            this.退出ToolStripMenuItem.Size = new System.Drawing.Size(110, 24);
            this.退出ToolStripMenuItem.Text = "退出";
            this.退出ToolStripMenuItem.Click += new System.EventHandler(this.退出
ToolStripMenuItem_Click);
            //
            // 词法分析WToolStripMenuItem
            //
            this.词法分析WToolStripMenuItem.Name = "词法分析WToolStripMenuItem";
            this.词法分析WToolStripMenuItem.Size = new System.Drawing.Size(119, 24);
            this.词法分析WToolStripMenuItem.Text = "词法分析（&W）";
            this.词法分析WToolStripMenuItem.Click += new System.EventHandler(this.词法分析
WToolStripMenuItem_Click);
            //
            // 语法分析GToolStripMenuItem
            //
            this.语法分析GToolStripMenuItem.Name = "语法分析GToolStripMenuItem";
            this.语法分析GToolStripMenuItem.Size = new System.Drawing.Size(115, 24);
            this.语法分析GToolStripMenuItem.Text = "语法分析（&G）";
            this.语法分析GToolStripMenuItem.Click += new System.EventHandler(this.语法分析
GToolStripMenuItem_Click);
            //
```

```
            // 语义分析MToolStripMenuItem
            //
            this.语义分析MToolStripMenuItem.Name = "语义分析MToolStripMenuItem";
            this.语义分析MToolStripMenuItem.Size = new System.Drawing.Size(119, 24);
            this.语义分析MToolStripMenuItem.Text = "语义分析（&M）";
            this.语义分析MToolStripMenuItem.Click += new System.EventHandler(this.语义分析
MToolStripMenuItem_Click);
            //
            // 帮助ToolStripMenuItem
            //
            this.帮助ToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.帮助文档ToolStripMenuItem,
            this.关于ToolStripMenuItem});
            this.帮助ToolStripMenuItem.Name = "帮助ToolStripMenuItem";
            this.帮助ToolStripMenuItem.Size = new System.Drawing.Size(88, 24);
            this.帮助ToolStripMenuItem.Text = "帮助（H）";
            //
            // 帮助文档ToolStripMenuItem
            //
            this.帮助文档ToolStripMenuItem.Name = "帮助文档ToolStripMenuItem";
            this.帮助文档ToolStripMenuItem.Size = new System.Drawing.Size(202, 24);
            this.帮助文档ToolStripMenuItem.Text = "帮助文档";
            this.帮助文档ToolStripMenuItem.Click += new System.EventHandler(this.帮助文档
ToolStripMenuItem_Click);
            //
            // 关于ToolStripMenuItem
            //
            this.关于ToolStripMenuItem.Name = "关于ToolStripMenuItem";
            this.关于ToolStripMenuItem.Size = new System.Drawing.Size(138, 24);
            this.关于ToolStripMenuItem.Text = "关于";
            this.关于ToolStripMenuItem.Click += new System.EventHandler(this.关于
ToolStripMenuItem_Click);
            //
            // statusStrip
            //
            this.statusStrip.ImageScalingSize = new System.Drawing.Size(18, 18);
            this.statusStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.toolStripStatusLabel,
            this.toolStripProgressBar});
            this.statusStrip.Location = new System.Drawing.Point(0, 488);
            this.statusStrip.Name = "statusStrip";
            this.statusStrip.Padding = new System.Windows.Forms.Padding(1, 0, 16, 0);
            this.statusStrip.Size = new System.Drawing.Size(728, 26);
            this.statusStrip.TabIndex = 1;
```

```
            this.statusStrip.Text = "statusStrip1";
            //
            // toolStripStatusLabel
            //
            this.toolStripStatusLabel.Image =
((System.Drawing.Image)(resources.GetObject("toolStripStatusLabel.Image")));
            this.toolStripStatusLabel.Margin = new System.Windows.Forms.Padding(10, 3, 10, 3);
            this.toolStripStatusLabel.Name = "toolStripStatusLabel";
            this.toolStripStatusLabel.Size = new System.Drawing.Size(83, 20);
            this.toolStripStatusLabel.Text = "处理进度";
            //
            // toolStripProgressBar
            //
            this.toolStripProgressBar.Name = "toolStripProgressBar";
            this.toolStripProgressBar.Size = new System.Drawing.Size(595, 20);
            //
            // richTextBox
            //
            this.richTextBox.Location = new System.Drawing.Point(14, 33);
            this.richTextBox.Name = "richTextBox";
            this.richTextBox.Size = new System.Drawing.Size(699, 451);
            this.richTextBox.TabIndex = 2;
            this.richTextBox.Text = "";
            this.richTextBox.WordWrap = false;
            //
            // 测试ToolStripMenuItem
            //
            this.测试ToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.processbarToolStripMenuItem});
            this.测试ToolStripMenuItem.Enabled = false;
            this.测试ToolStripMenuItem.Name = "测试ToolStripMenuItem";
            this.测试ToolStripMenuItem.Size = new System.Drawing.Size(49, 24);
            this.测试ToolStripMenuItem.Text = "测试";
            //
            // processbarToolStripMenuItem
            //
            this.processbarToolStripMenuItem.Name = "processbarToolStripMenuItem";
            this.processbarToolStripMenuItem.Size = new System.Drawing.Size(202, 24);
            this.processbarToolStripMenuItem.Text = "Processbar";
            this.processbarToolStripMenuItem.Click += new
System.EventHandler(this.processbarToolStripMenuItem_Click);
            //
            // mainForm
            //
```

```
            this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 14F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(728, 514);
            this.Controls.Add(this.richTextBox);
            this.Controls.Add(this.statusStrip);
            this.Controls.Add(this.menuStrip);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MainMenuStrip = this.menuStrip;
            this.MaximizeBox = false;
            this.Name = "mainForm";
            this.Text = "编译原理综合实验";
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.mainForm_FormClosing);
            this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(this.mainForm_FormClosed);
            this.menuStrip.ResumeLayout(false);
            this.menuStrip.PerformLayout();
            this.statusStrip.ResumeLayout(false);
            this.statusStrip.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.MenuStrip menuStrip;
        private System.Windows.Forms.ToolStripMenuItem 文件ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 词法分析WToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 语法分析GToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 语义分析MToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 帮助ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 帮助文档ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 关于ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 打开ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem 退出ToolStripMenuItem;
        private System.Windows.Forms.StatusStrip statusStrip;
        private System.Windows.Forms.ToolStripStatusLabel toolStripStatusLabel;
        protected internal System.Windows.Forms.ToolStripProgressBar toolStripProgressBar;
        private System.Windows.Forms.RichTextBox richTextBox;
        private System.Windows.Forms.ToolStripMenuItem 测试ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem processbarToolStripMenuItem;
    }
}
```

```
//业务逻辑代码
using CompilerLab.COMUtil;
using System;
using System.Collections;
using System.Windows.Forms;
namespace CompilerLab.ChildForm.GrammarAnalysis
{
    public partial class GrammarAnalysis : Form
    {
        /// <summary>
        /// 初始化预测分析表
        /// </summary>
         //终结符表
        public string[] term = { "+", "-", "*", "/", "^", ")", "#", "(", "i" };
         //非终结符表
        public string[] unterm = { "E", "A", "T", "B", "F", "C", "P" };
        //产生式表
        public string[,] grammar = { { "", "", "", "", "", "", "", "TA" ,"TA"},
                                     { "+TA", "-TA", "", "", "", "ε", "ε", "", ""},
                                     { "", "", "", "", "", "", "", "FB", "FB"},
                                     { "ε", "ε", "*FB", "/FB", "", "ε", "ε", "" ,""},
                                     { "", "", "", "", "", "", "", "PC" ,"PC"},
                                     { "ε", "ε", "ε", "ε", "^F", "ε", "ε", "" ,""},
                                     { "", "", "", "", "", "", "", "(E)", "i"} };
        //符号栈
        public Stack Symbo = new Stack();
        //输入字符栈
        public Stack InputStack = new Stack();
        //状态标识符
        public bool con = true;
        //字符串
        public string strNow;
        //行列标识符
        public int column, row;

        public GrammarAnalysis()
        {
            InitializeComponent();
        }
        private void 步进NToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //如果状态为假
            if (con == false)
            {
```

```
                MessageBox.Show("分析结束", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }
            //如果待分析字符串为空
            else if (analysestr.Text == "" || analysestr.Text == "等待分析")
            {
                MessageBox.Show("请输入要分析的字符串", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                return;
            }
            int loc;
            string SymboTop, SymboInputTop;
            SymboTop = Symbo.Peek().ToString();//取符号栈顶字符
            SymboInputTop = InputStack.Peek().ToString();//取字符栈顶字符
            SymboTopOne(SymboTop);
            if (SymboTopOne(SymboTop)) //若栈顶第一字符为符号栈栈顶字符
            {
                SymboTopTwo(SymboTop, SymboInputTop); //进入第二位判断
            }
            else if (SymboTopThree(SymboTop) &&
                InputStackTopOne(SymboInputTop)) //进入第三位判断
            {
                loc = column * 6 + row;
                strNow = grammar[column, row];
                this.outputListBox.Items.Add(Symbo.Peek().ToString() + "-->" + grammar[column,
row]);
                Analyse();
            }
            else
            {
                analyseResult.Text = "分析出错!";
                con = false;
            }
            if (con)
            {
                ShowSymbo();
                ShowInpuStack();
            }
        }
        private void 开始SToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (analysestr.Text == "" || analysestr.Text == "等待分析")
            {
```

```
                MessageBox.Show("请输入要分析的字符串", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else if (con==false)
            {
                MessageBox.Show("分析结束", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else
            {
                Symbo.Push("#");
                Symbo.Push("E");
                symboListBox.Items.Add("#E");
                inputListBox.Items.Add(analysestr.Text + "#");
                InputStack.Push("#");
                for (int i = analysestr.Text.Length - 1; i >= 0; i--)
                {
                    InputStack.Push(analysestr.Text[i]);
                }
            }
        }
        public bool SymboTopOne(string stackstring)
        {
            bool symbol = false;
            for (int i = 0; i < term.Length; i++)
            {
                if (stackstring == term[i])
                {
                    symbol = true;
                    break;
                }
            }
            return symbol;
        }
        private void SymboTopTwo(string stack, string input)
        {
            if (stack == input)
            {
                if (stack == "#")
                {
                    analyseResult.Text = "分析成功!";
                    outputListBox.Items.Add("成功");
                    con = false;
                }
                else
```

```
        {
            Symbo.Pop();
            InputStack.Pop();
            this.outputListBox.Items.Add("");
        }
    }
}
public bool SymboTopThree(string stack)
{
    bool symbol = false;
    for (int i = 0; i < unterm.Length; i++)
    {
        if (unterm[i] == stack)
        {
            column = i;
            symbol = true;
            break;
        }
    }
    return symbol;
}
private bool InputStackTopOne(string myinput)
{
    bool symbol = false;
    for (int i = 0; i < term.Length; i++)
    {
        if (term[i] == myinput)
        {
            row = i;
            symbol = true;
            break;
        }
    }
    return symbol;
}
private void Analyse()
{
    if (strNow == "")
    {
        analyseResult.Text = "分析出错!";
        con = false;
    }
    else if (strNow == "ε")
    {
        Symbo.Pop();
```

```
    }
    else{
        Symbo.Pop();
        for (int i = strNow.Length - 1; i >= 0; i--)
        {
            Symbo.Push(strNow[i]);
        }
    }
}
private void ShowSymbo()
{
    string ch = "";
    int len = Symbo.Count;
    string display = "";
    for (int i = 0; i < len; i++){
        string t;
        t = Symbo.Pop().ToString();
        display = display + t;
    }
    for (int i = len - 1; i >= 0; i--){
        ch = ch + display[i];
        Symbo.Push(display[i]);
    }
    this.symboListBox.Items.Add(ch);
}
private void 重置RToolStripMenuItem_Click(object sender, EventArgs e)
{
    analysestr.Text = "等待分析";
    analyseResult.Text="等待分析";
    this.symboListBox.Items.Clear();
    this.inputListBox.Items.Clear();
    this.outputListBox.Items.Clear();
    Symbo.Clear();
    InputStack.Clear();
    con = true;
}
private void 输入待分析字符串ToolStripMenuItem_Click(object sender, EventArgs e)
{
    try{
        Dialog dialog = new Dialog();
        dialog.ShowDialog();
        if (dialog.flag) {
            analysestr.Text = dialog.str;
        }
    }
```

```
            catch (Exception ex) {
                //错误提示
                MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
            }
        }
        private void 消除文法左递归ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox.Text =
"E->TA\nT->FB\nF->PC\nC->^F|ε\nP->(E)|i\nA->+TA|-TA|ε\nB->/FB|*FB|ε";
        }
        private void 关于ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("版本号：1.0", "关于", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
        private void ShowInpuStack()
        {
            int len = InputStack.Count;
            string display = "";
            for (int i = 0; i < len; i++)
            {
                string t;
                t = InputStack.Pop().ToString();
                display = display + t;
            }
            for (int i = len - 1; i >= 0; i--)
            {
                InputStack.Push(display[i]);
            }
            this.inputListBox.Items.Add(display);
        }
    }
}
```

七、心得体会

通过本次实验，我加深了对语法分析器的理解，并通过对 LL(1)分析法进行实现，加深了对 LL(1)分析法分析流程的理解。LL(1)分析法又称预测分析法，是一种不带回溯的非递归自顶向下分析方法。LL(1)的含义是：第一个 L 表明自顶向下分析是从左向右扫描输入串，第 2 个 L 表明分析过程中将使用最左推导，“1”表明只需向右看一个符号便可决定如何推导，即选择哪个产生式(规则)进行推导。