**武汉大学计算机学院**

**本科生课程设计报告**

**如：XXX系统总体设计与实现**

专 业 名 称 ：软件工程

课 程 名 称 ：《Windows编程实验》

指 导 教 师 ：胡继承 职称：教授

学 生 学 号 ：2017302580198

学 生 姓 名 ：邹鑫

二○一九年十二月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名： 邹鑫 日期： 2019.12.20

摘□□要

（黑体小2）

XXXX实验的实验目的是XXXX。

实验设计主要遵循XXXX。

实验内容主要包括：

实验结论为XXXX

……

……

……

**关键词：**关键词1；关键词2；关键词3

（黑体小4） （宋体小4）

**目□□录**（黑体小2）

**1**□**实验目的和意义**

1.1□ 实验目的 ………………………………………………………………………1

1.2□ 实验意义………………………………………………1

(各章的名称黑体4号，其余宋体小4)

**……**

**……**

**……**

**2**□**实验设计**

2.1□概述……………………………………………………………………………… 35

2.2□实验原理………………………………………………………………………… 37

2.3□实验方案………………………………………………………………………… 39

**……**

**……**

**……**

**结论** …………………………………………………………………………………… 57

**参考文献 ………………………………………………………………………………** 59

**附录 ……………**………………………………………………………………………… 72

(结论、参考文献、致谢及附录黑体4号)

# 实验要求与背景

## 实验背景

为了进一步了解Windows程序设计在.Net平台上的各种应用，加强对理论

知识的掌握以及实践的能力，本文选择制作一个有比较多的功能的计算器。

## 1.2 实验要求

●实现加、减、乘、除四种基本运算功能

●上述四种运算均要求使用dll实现，其中dll使用c++或c#编程语言来创建

●上述四种运算输入、操作和运算结果均要求直接在界面中进行显示，其中界面可以选用MFC、Winform、WPF技术实现

●上述四种运算的实现均要求输入数据兼容数据类型int、double，输出结果也兼容数据类型int、double

●\*对于输入的非法数据或无效数据要进行相关提示；

●\*对于输入的数据有回退删除功能

## 1.3 实现的功能

为了更好地考察自己的能力，本文除了完成要求中的6个功能之外，还进行了一定的扩展，为计算器添加了更多的功能，具体如下：

●除了加、减、乘、除外，本文还为计算器添加了一些新的运算符，比如：乘方，阶乘等。

●本计算器不是那种很简单的一次只能执行一个运算符的计算器，它可以执行表达式，比如(3+2)×5÷2^3+sin(20-3÷2)。

●为了进一步增强计算器的功能，本文还在初始的传统界面的基础上，实现了扩展界面，扩展界面占空间更大、有更多功能。

●扩展界面中，增加了一些常用得函数界面，比如sin、cos、tan、exp、log、ln等，方便用户进行复杂一点的函数运算。

●扩展界面中，增加了进制转换的功能，支持2进制、8进制、10进制以及16进制的相互转换。

●扩展界面中，增加了函数自定义功能，用户可以自定义函数f，比如f(x) = x^2 + 2x + 1，然后在计算器中输入f(5)就能得到结果5^2 + 2\*5 + 1 = 36

。

●扩展界面中，可以实现函数的嵌套，如sin(f(cos(5)+3)\*6)。

# 2 项目架构简要分析

## 2.1 项目结构图

此项目的总体结构图如图2.1所示。

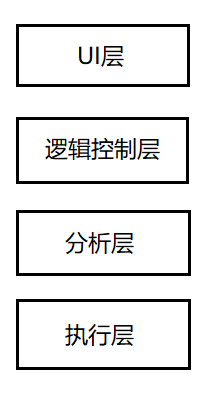


图2.1 项目总体结构图

其中，UI层负责显示UI界面，捕捉用户与系统的交互动作，并将之传递给逻辑控制层进行处理。

逻辑控制层负责整个系统的逻辑控制，它从UI层接收交互时间，并进行相应的处理。在需要进行算术表达式求解时，它调用分析层和执行层的功能得出结果。

分析层主要负责对传入的算术表达式的语法规范性进行分析，如果表达式的语法不规范，比如：1+2+-3，则会弹出警告框告知用户输入不规范。如果输入符合语法，则生成相应的语法执行树，该执行树中包含了运算符的优先级和结合性信息，用于执行层进行执行。图2.2展示了表达式3^4 × sin(5+7)的语法执行树。

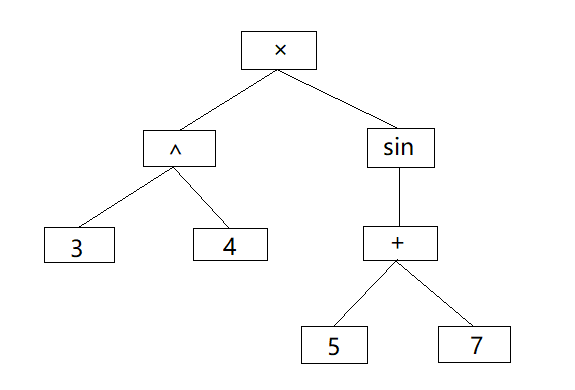


图2.2 表达式3^4 × sin(5+7)的语法执行树

执行层则是接收分析层传入的语法执行树，从树的根节点开始，递归地执行，获得每一个节点的值，最终可以获得根节点的值，也就是表达式的值。

## 2.2 项目进程视图

本项目的进程视图如图2.3所示。

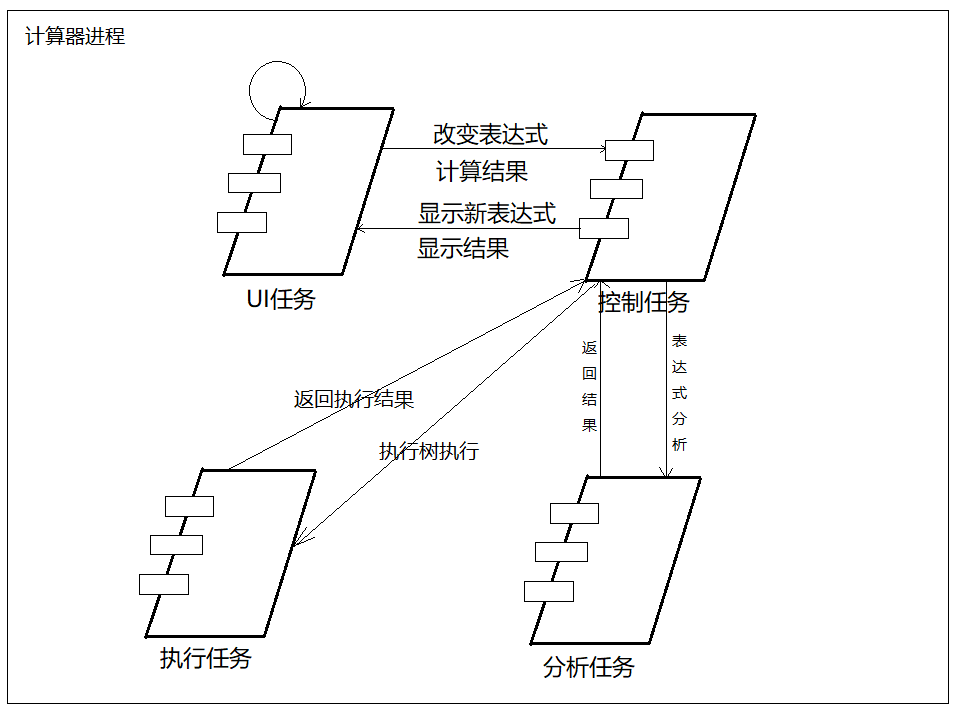


图2.3 项目进程视图

从图中可以看到本项目是单进程的，在计算器进程中，UI任务显示UI界面，当用户想要改变表达式，则将信息传递给控制任务，控制任务得到相应的信息之后，更改表达式字符串，并显示在UI界面上。当UI界面传递要计算结果的信息之后，控制任务则调用分析任务，传入表达式的字符串，分析任务分析获得计算执行树或者得到语法错误的结论，返回给控制任务，如果语法错误，控制任务直接将信息传递给UI任务，显示出来提示用户。如果语法正确，控制任务则根据正确的语法执行树调用执行任务，获得执行结果，并将其发送给UI任务进行显示。

# 3 UI界面

本文为了将功能更好地展现，尽可能不占用过多的桌面空间，将项目界面分

为了3个部分——基础功能部分、扩展功能部分以及进制转换部分。

基础功能部分主要实现基本计算功能——加减乘除、乘方运算，基础功能界面占用空间比较小，适合进行简单的运算，如图3.1所示。



图3.1 基础界面图

扩展功能界面实现了更多的功能，比如常见函数（sin、cos、tan等），阶乘，以及自定义函数功能。在遇到基础功能界面解决不了或者解决起来非常麻烦的问题时，就可以考虑使用扩展功能界面。扩展功能界面图3.2所示。

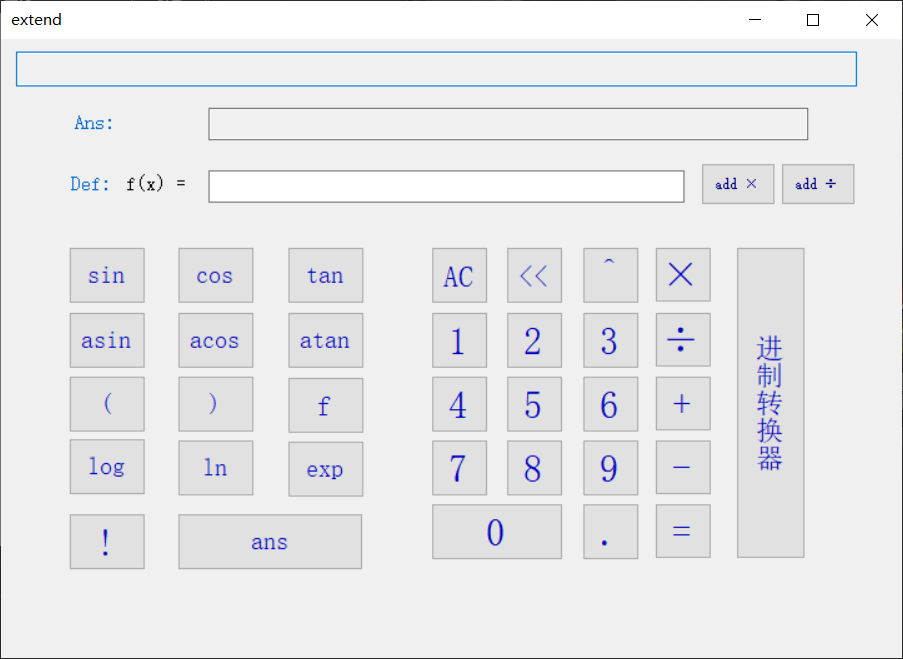


图3.2 扩展功能界面图

进制转换界面主要用于处理进制转换的问题，可以在2进制、8进制、10进制以及16进制之间互相转换，如图3.3所示。

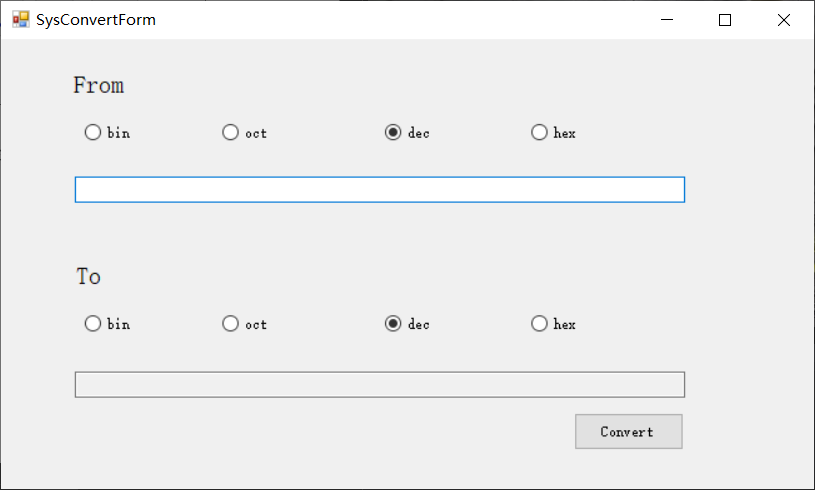


图3.3 进制转换界面图

其中，bin表示二进制、oct表示8进制、dec表示10进制、hex表示16进制。

# 4 功能实现

为了实现上述功能，下面我将一一介绍编码过程中的一些比较重要的过程。

## 4.1 加、减、乘、除基本运算的动态链接库创建

根据要求，为了考察我们是否能创建动态链接库并进行调用，我们必须为基本的加、减、乘、除运算创建相应的动态链接库，以供执行时执行器Excutor调用。

首先，创建一个新项目，选择“类库”项目，如图3.1.1所示。

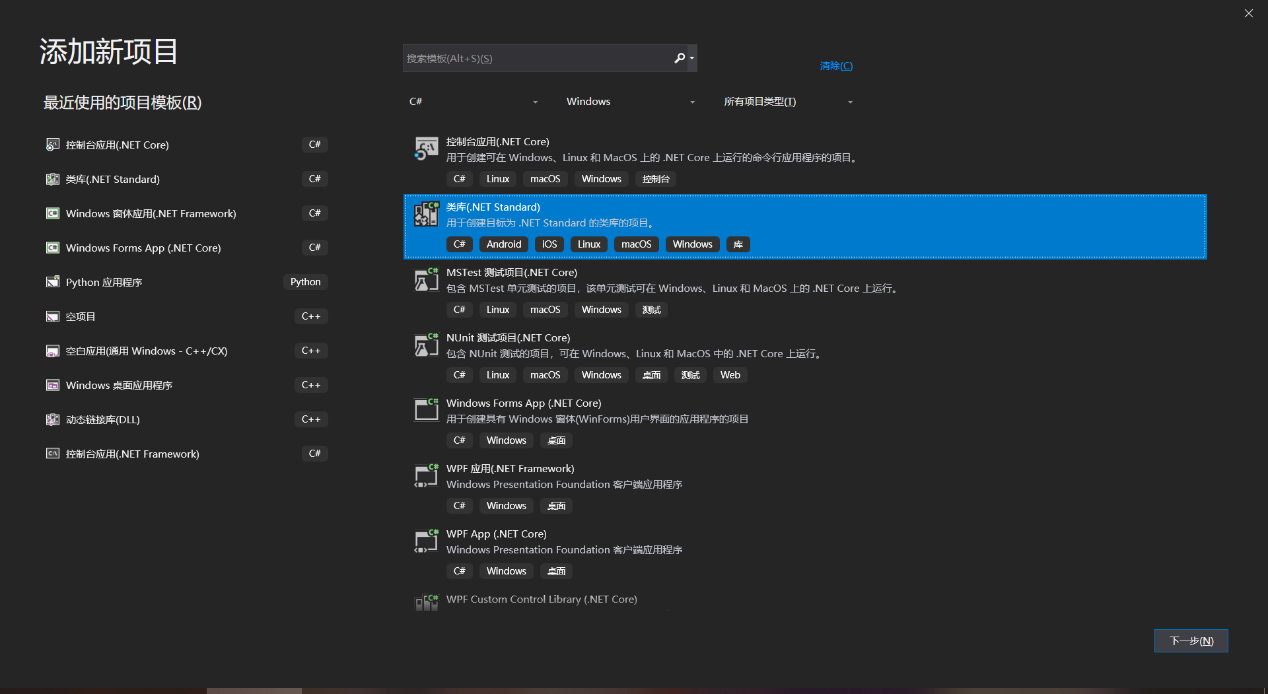


图3.1.1 新建类库项目

然后，为项目命名，选择存储路径，我为项目命名为“OPS”。

然后，新建一个静态类，我将之命名为Operations，并且在其中建了许多静态函数用于实现基本运算功能，该类的代码如下：

public static class Operations

{

public static double add(double a,double b)

{

return a + b;

}

public static double minus(double a, double b)

{

return a - b;

}

public static double multiply(double a, double b)

{

return a \* b;

}

public static double divide(double a, double b)

{

return a / b;

}

}

我针对加、减、乘、除各写了一个函数，传入的参数是double类型，返回值也是double类型。我为什么不根据用户输入的数字决定到底是传int还是double，然后为每个运算符写4个重载函数呢？因为我个人觉得这不是非常有必要，绝大多数运算都是带有浮点数的，运算中全是整数的情况很少，既然几乎所有运算都含有浮点数，那又何必到时候再将整数转为浮点数呢？于是，我选择将所有输入的数字都当作浮点数使用。

然后，重新生成该项目，生成完毕后，进入该项目的Debug文件夹下，就会发现有一个名为OPS.dll的文件，这就是我们要使用的动态链接库。

使用时，只需要在项目的引用中添加该动态链接库，然后就可以将其当作正常的静态类、静态函数使用了。

## 4.2 回退的实现

针对回退，本文对不同功能的界面采取了不同的措施。

### 4.2.1 基本功能界面的回退实现

对于基本功能界面，由于用户输入的只有加减乘除运算符、乘方运算符、左右括号以及数字，所以针对此界面，本文决定直接采用去掉输入字符串的最后一个字符的方法来进行回退。过程如图4.2.1所示。

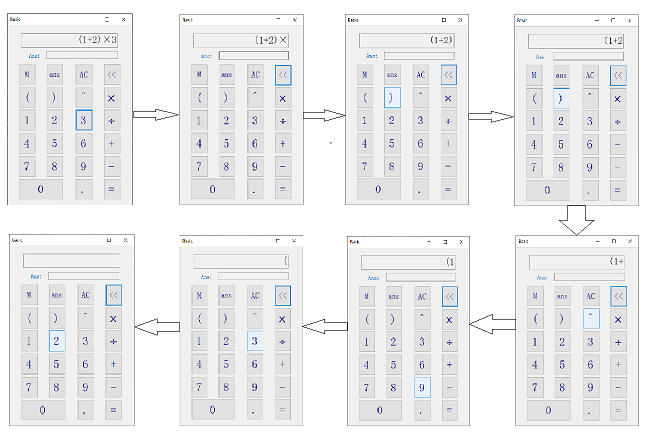


图4.2.1 基本功能界面连续回退图

从图中可以清晰地看到，随着回退按钮不断被点击，输入框每次少一个字符。

### 4.2.2 扩展功能界面的回退功能的实现

针对扩展功能界面，由于扩展界面实现了许多函数功能，故不能再像基础功能界面那样一个字符一个字符地删除，而应该以每次的输入为单位进行删除（每次的输入不一定是一个字符，比如sin就是3个字符）。为此，我建了一个栈，用于存放字符串，每次添加内容，则将新内容入栈，每次需要回退，则将字符串出栈，并且从输入框的尾部删掉该字符串。

过程如图4.2.2所示。

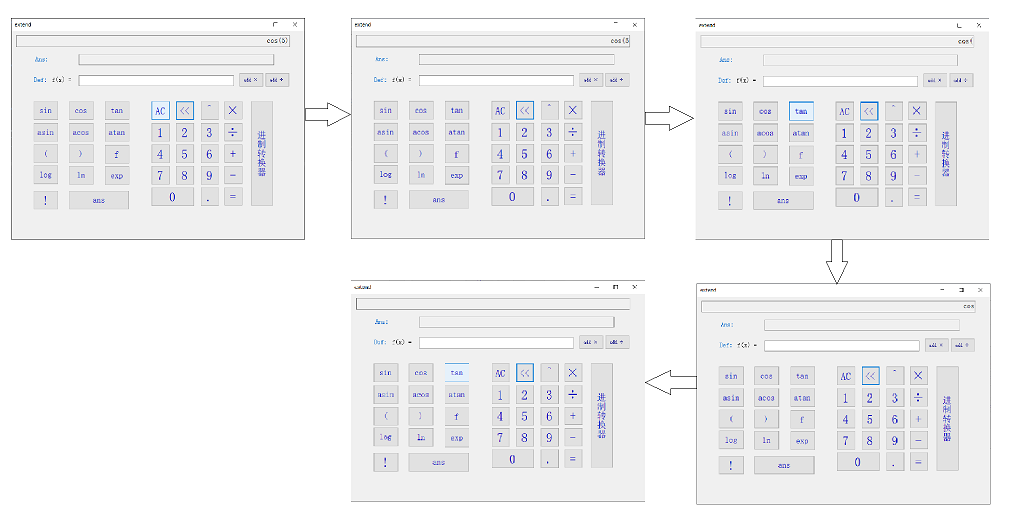


图 4.2.2 扩展功能界面连续回退图

从图中可以看到，前几步是和基本功能界面相同，都是回退了一个字符，这是因为输入的时候就只输入了那个字符。但在最后一步，整个“cos”都消失了，因为cos被看作整体，一次回退就会将其整体删除。

## 4.3 单词元素获取

（item获取）

# 5 反思与改进

F实现时的思路转变