**中国矿业大学计算机学院**

**2018 级本科生课程设计报告**

课程名称 系统软件开发实践

实验名称 实验3-4可视化计算器

报告时间 2021年3月27日

学生姓名 丁旭行

学 号 14184501

专 业 计算机科学与技术

任课教师 徐东红

**目 录**

[**1 实验目的 1**](#_Toc68253019)

[**2 实验内容 1**](#_Toc68253020)

[**3 高级计算器功能 1**](#_Toc68253021)

[**4 实现工具 1**](#_Toc68253022)

[**5 整体思路 2**](#_Toc68253023)

[**6 设计原理 2**](#_Toc68253024)

[**6.1 基于Flex&Bison编译器整体原理 2**](#_Toc68253025)

[**6.2 C#调用后端实现与前端的交互 4**](#_Toc68253026)

[**7 实现过程 5**](#_Toc68253027)

[**7.1 利用Flex和Bison实现词法分析和语法分析 5**](#_Toc68253028)

[**7.2 编写并修改前端代码 7**](#_Toc68253029)

[**8 设计成果展示 10**](#_Toc68253030)

[**9 实验心得与总结 12**](#_Toc68253031)

[**10 实验补充—python语言的dll文件调用 12**](#_Toc68253032)

# 1 实验目的

使用Flex和Bison开发一个具有全部功能的桌面计算器并借助其他编程语言实现可视化界面。

# 2 实验内容

使用Flex和Bison开发一个具有全部功能的桌面计算器，能够支持变量，过程，循环和条件表达式，使它成为一个虽然短小，但是具有现实意义的编译器。重点学习抽象语法树的用法，它具有强大而简单的数据结构来表示分析结果。

# 3 高级计算器功能

* 变量命名；
* 实现赋值功能；
* 实现比较表达式(大于、小于、等于等等)；
* 实现if/then/else和do/while的流程控制；
* 用户可以自定义函数；
* 简单的错误恢复机制。

(注：此处实现功能均是在基本计算器功能基础之上追加的功能)

# 4 实现工具

使用C#Windows窗体制作计算器的可视化界面，后端使用Flex和Bison对前端接受的表达式进行词法和语法分析处理，返回前端结果值，开发一个可视化的可编程计算器。

实验环境：Visual Studio 2019

编程语言：C#

# 5 整体思路

首先利用Flex和Bison构造一个可以识别一定语法和功能的编译器，再利用Visual Studio 2019生成其编译函数的dll文件，用C#调用dll文件中的主函数，使其读取in.txt的内容进行分析，这里in.txt中的内容也是由前端接受用户输入保存到in.txt中的。分析完之后，主函数将结果输出到out.txt文件中，然后由前端读取out.txt中的内容，进行结果展示。

但在实验途中发现，在Visual Studio 2019中生成dll文件始终报错，在经过用百度查阅各种资料，并多次尝试无果后放弃原来的思路，改换直接用C#调用编译生成的可执行的fb3-3.tab.exe文件，使其读取in.txt的内容进行分析，分析完之后将结果输出到out.txt文件中。

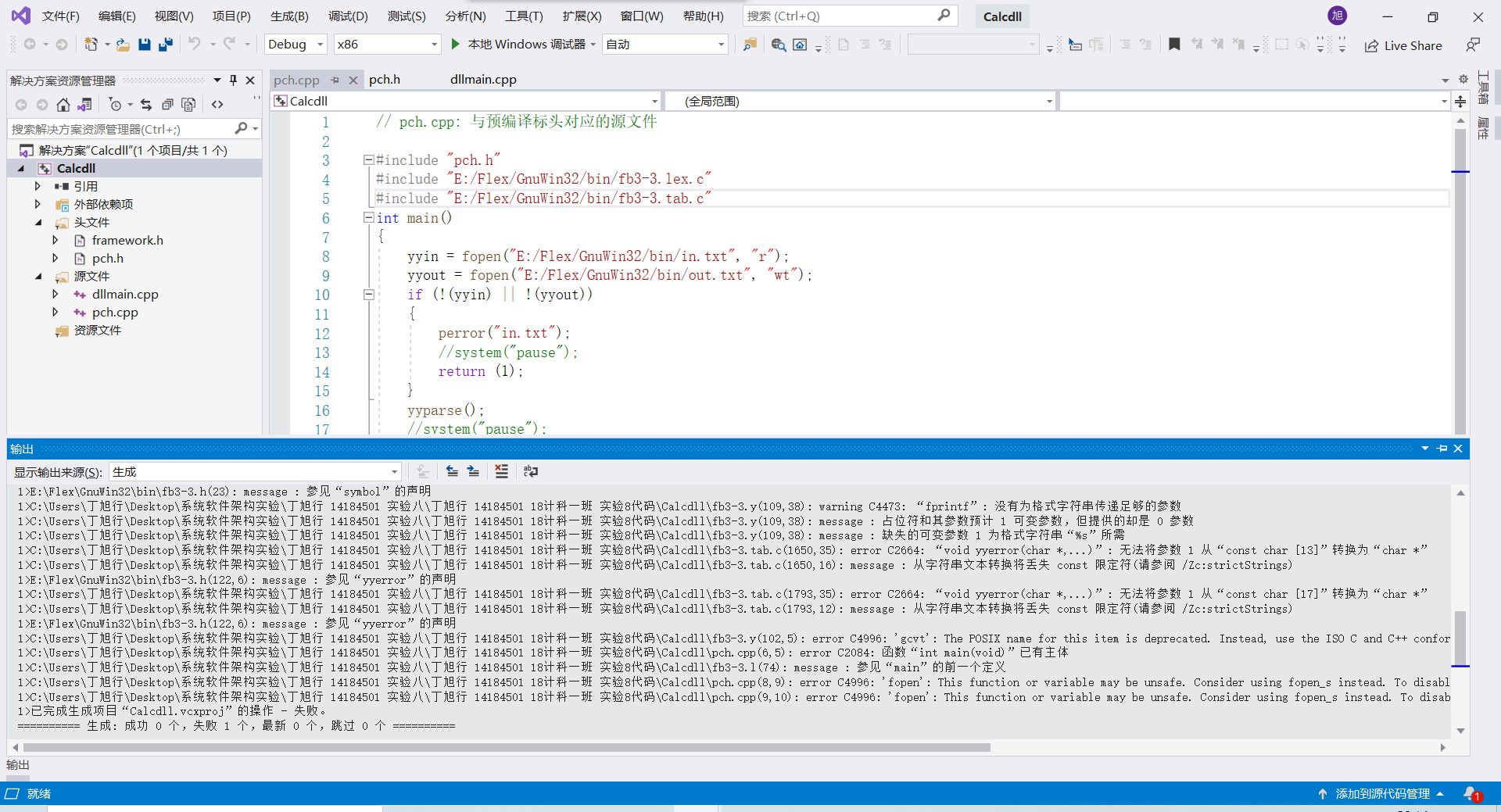


图1 封装dll文件失败

# 6 设计原理

## 6.1 基于Flex&Bison编译器整体原理

Flex是一个自动化工具，可以按照定义好的规则自动生成一个C函数yylex()，称为扫描器(Scanner)。这个函数可以将输入按照定义好的规则分析文本串中的字符，找到符合规则的一些字符序列后，就执行在规则中定义好的动作(Action)。例如可以在规则中定义：如果遇到一个换行字符’\n’，那么就把行计数器的值加一。

Bison是一种通用目的的分析器生成器，它可以将LALR(1)上下文无关文法的描述转化成分析该文法的C程序。整体过程图如下：

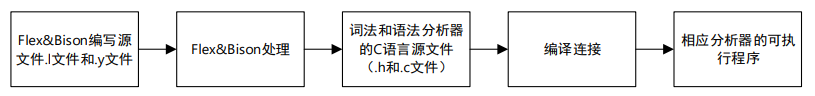


图2 生成过程

进行词法分析和语法分析的整体过程：

1. flex把输入流分解为若干片段记号，也就是所谓的终结符，即语法树的叶子节点，而bison则分析这些记号并基于逻辑进行组合。利用移进规约原理以及深度优先构造语法树，树的建立顺序是自底向上的，flex中先给词法单元建立了对应的结点，在语法树中属于叶子结点，在bison中，由词法单元对应的叶子结点规约出新的父节点，又由新的父节点和其他结点规约到新的父节点，若匹配则调用相应的抽象语法树构造函数自底向上建立语法树，如果规约到起始语法单元就表示整棵语法树都被建立好了。
2. 计算树由多个节点构成，每个节点都有一个nodetype，遍历树的代码使用这个变量来判断当前访问的节点，譬如内置函数使用fncall节点，用户自定义函数使用ufcall节点，流程控制表达式if/then/else和while/do使用flow节点，它包含控制表达式、then分支或者do语句体以及可选的else分支。
3. 每个计算树都有相应的值，计算器的核心是函数eval，它计算语法分析器中构造的计算树，对于if/then/else而言，它的值就是所选的分支的值；while/do的值则是do语句列表的最后一条语句的值。
4. 用户自定义函数：函数定义包括函数名、虚拟参数列表和代表函数体的计算树。当函数被定义时，参数列表和计算树的根节点地址将被简单地保存到符号表中函数名对应的条目中。在调用自定义函数时，就可以通过这两个地址，将虚拟参数改为实际传入的参数，然后再用eval()函数计算计算树的结果，计算完毕，再将虚拟参数改回原来的值。

## 6.2 C#调用后端实现与前端的交互

DLL(Dynamic Link Library)文件为动态链接库文件，又称“应用程序拓展”，是软件文件类型。在Windows中，许多应用程序并不是一个完整的可执行文件，它们被分割成一些相对独立的动态链接库，即DLL文件，放置于系统中。当我们执行某一个程序时，相应的DLL文件就会被调用。按照最开始的思路使用C#调用dll文件，我查询了关于生成dll文件以及C#调用dll文件的很多资料，其中最有用的一篇为：<https://blog.csdn.net/weixin_34976988/article/details/99625533>

但由于封装dll文件始终出问题，并且没有找到解决方式，所以改用C#直接调用flex和bison的编译器执行文件，为此我也查询了很多资料，这是其中一篇较为有用的文章：<https://blog.csdn.net/dark_tone/article/details/100190740>

本系统的总体结构图如图所示，首先C#前端将获取待分析的语句以及操作，送到int.txt文件中，之后调用fb3-3.tab.exe文件读入内容并计算返回结果到out.txt文件，最后前端读取返回内容并显示：

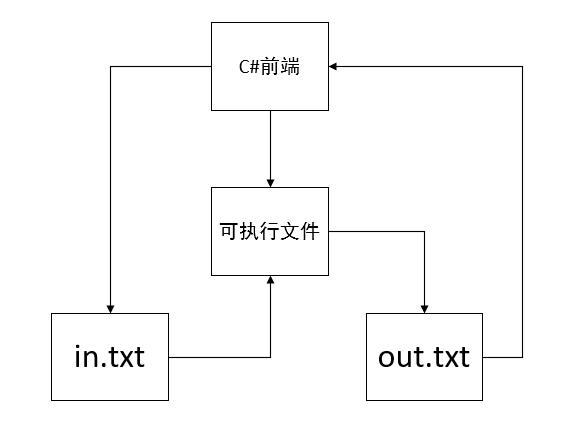


图3 总体结构图

# 7 实现过程

## 7.1 利用Flex和Bison实现词法分析和语法分析

这里可以直接使用上次实验完成的flex和bison代码，但是需要注意：前几次实验命令的输入为键盘输入，这次我们需要修改输入输出流来完成文件的输入输出。这里需要明白编译器工作的几个原理：

1. .l文件中的第三段即用户代码段，是会被原样拷贝到.lex.c的文件中的，可以定义一些辅助函数或代码供扫描器yylex()调用，或者调用扫描器，一般来说就是 main()函数，这一部分是可选的，如果没有的话，Flex会在.lex.c里默认生成。
2. Flex的输出是.lex.c文件，它包含扫描例程yylex()、一些用来存放匹配的token的table和一些附加的例程和宏。yylex()被调用后，它从输入文件yyin扫描tokens，直到遇到EOF时返回。其中yyin是一个文件指针，指向的是作为输入的文件，它在缺省条件下是stdin，为标准输入，可以将其改为到指定的文件来获取输入，yyout同理。
3. yyin的原理在YY\_INPUT宏中有定义：YY\_INPUT(buf,result,max\_size)是最多将max\_size个字符放在字符数据缓冲区，返回一个整型变量，其值要么是读入的字符数，要么是常量YY\_NULL(Unix下是0)指示EOF。扫描器通过yyin一次读一个数据块而不是用getc()读一个个字符，从而提高了效率。当扫描器从 YY\_INPUT收到EOF，它会检查yywrap()函数。如果该函数返回false即0，则假定该函数已经执行并设置了yyin指向另一个输入文件，扫描继续进行；如果该函数返回true非0，扫描器终止，返回0给它的调用者。.l文件中使用了%option noyywrap(相当于返回1)。

基于如上认知，将yyin,yyout修改为指定的文件来输入输出，而不再依赖于用户在键盘的输入和显示器输出。

1. 在.l文件中在用户代码段添加自己定义的主函数如下：

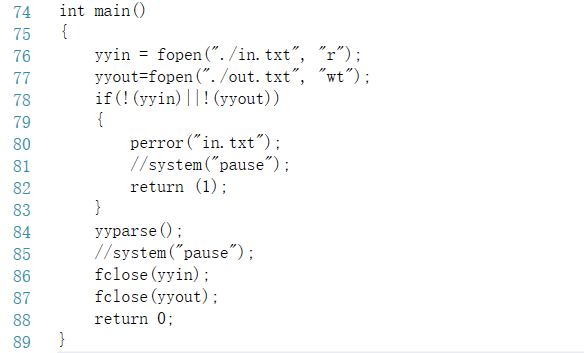


图4 .l文件修改

1. 这里只是定了输入的来源和最终结果输出的位置，需要修改.y文件将函数的输出写入到yyout里，在.y 文件里对输出进行修改如下：

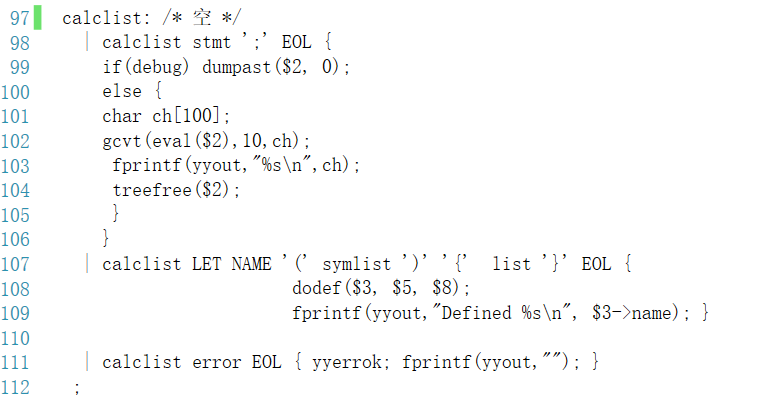


图5 .y文件修改

1. 为了执行.l文件里的主函数需要将fb3-3funcs.c中定义的主函数注释掉，防止冲突：

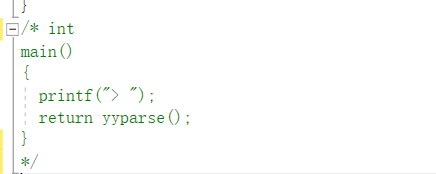


图6 fb3-3funcs.c文件修改

1. 修改后重新编译，生成可执行文件fb3-3.tab.exe：

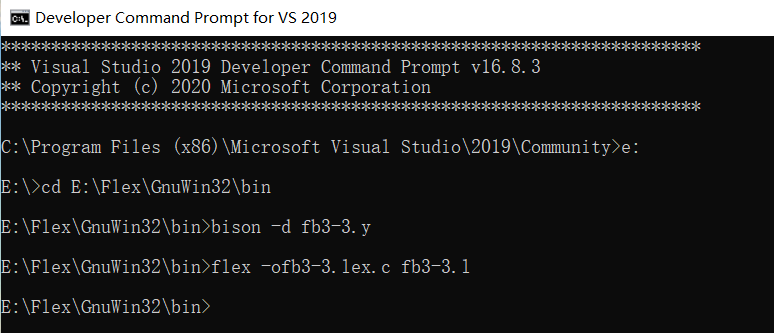


图7 输入命令1

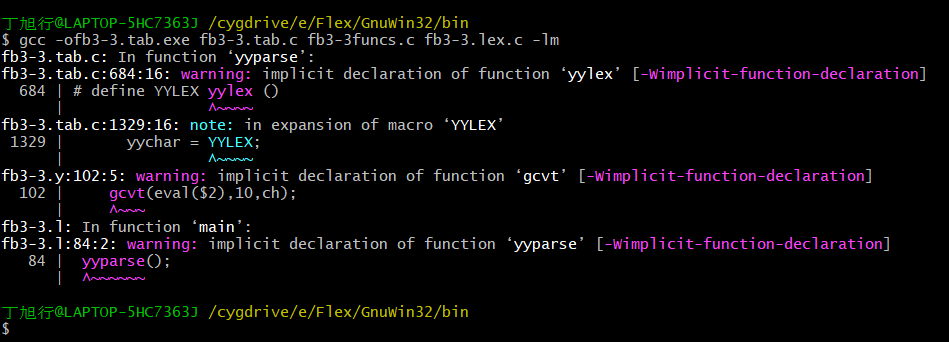


图8 输入命令2

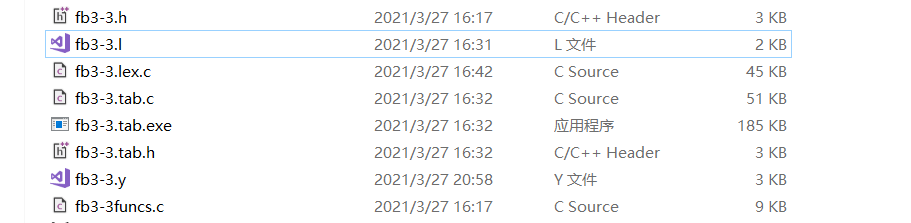


图9 生成文件

1. 在in.txt文件中输入计算表达式，之后运行生成的fb3-3.tab.exe文件，可在out.txt文件中显示计算结果，说明编译测试成功：

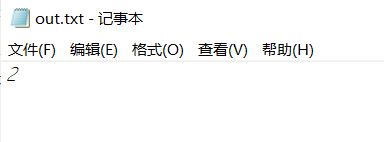
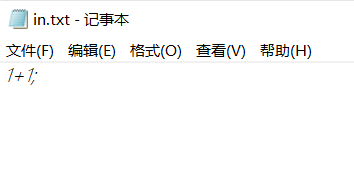


图10 测试成功

## 7.2 编写并修改前端代码

(1) 对于前端，我在之前做过的C#实现的基本计算器基础上进行二次开发，做了相应的修改，添加新的功能按钮以及新的窗体。

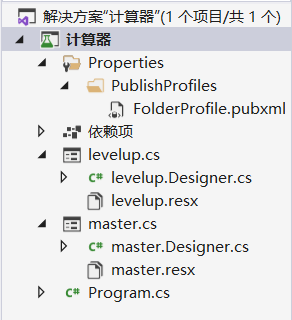


图11 项目目录结构

(2) Visual Studio 2019的C#中的窗体类提供了友好的UI制作界面，可以通过拖动工具箱里的工具进行UI绘制并自动生成代码，同时可以设置点击事件。

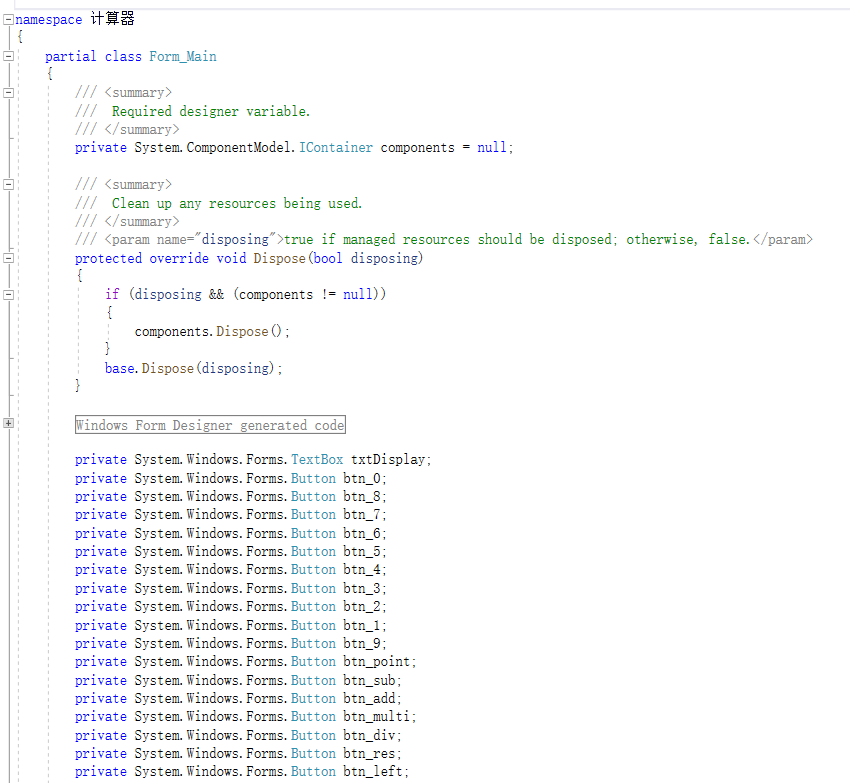


图12 主窗体部分代码

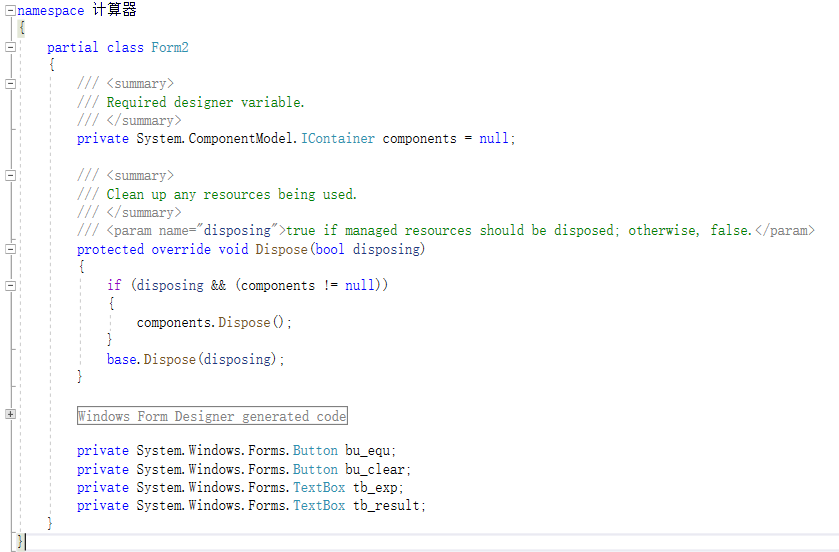


图13 副窗体代码

(3) 用户可以在基本计算器界面进行点击操作，也可点击高级版按钮进入高级计算器界面，在下方区域输入指令，点击”=”按钮即可在上方区域返回结果。

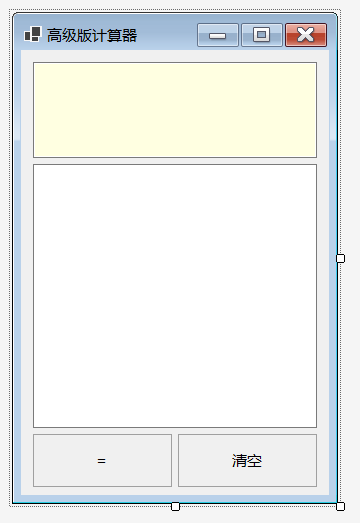
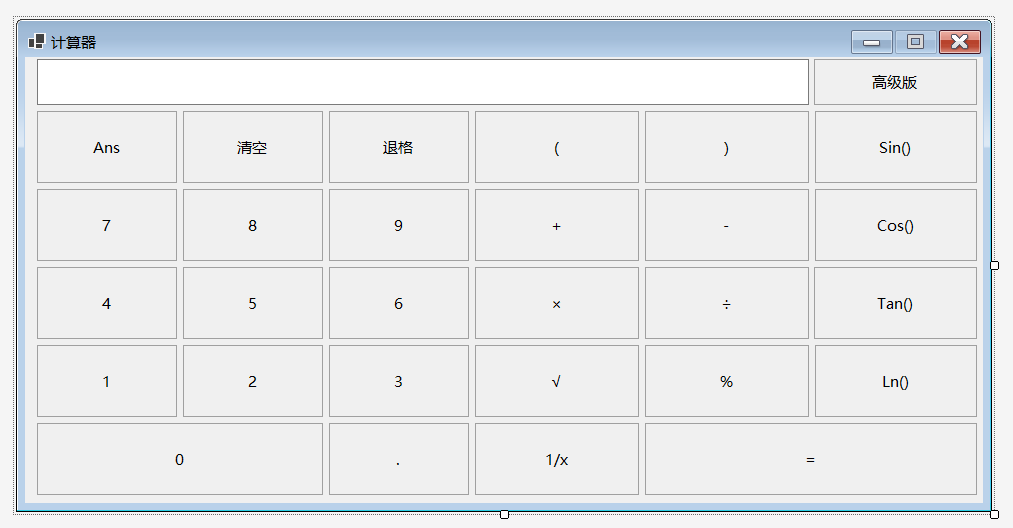


图14基本计算器 图15高级版计算器

(4) 对于新增窗体的按钮以及基本计算器的按钮进行代码编写与更改，使其能够完成相应功能，并可以把用户输入信息输去到in.txt文件中，并能够调用fb3-3.tab.exe文件对in.txt文件进行解析，然后从out.txt文件中读取最终结果返回到窗体显示区域。

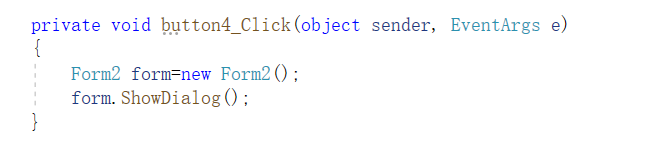


图16 “高级版”按钮

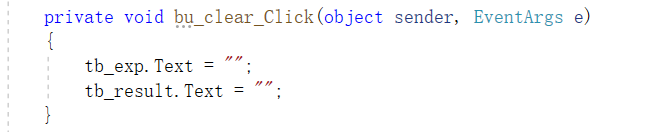


图17 “清空”按钮

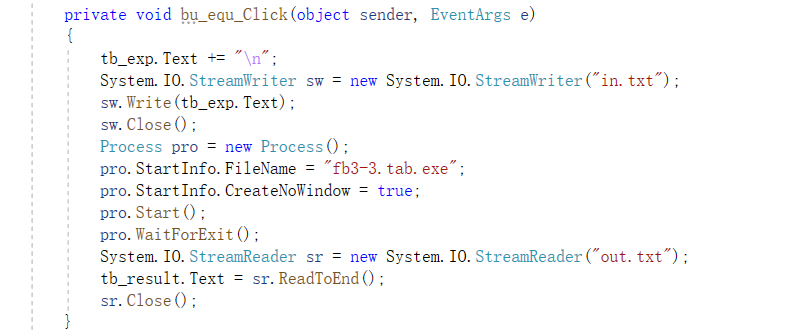


图18 “=”按钮

其中Process类进行调用exe文件，StreamWriter与StreamReader分别用来完成对in.txt文件的写入和对out.txt文件的读取。

# 8 设计成果展示

计算器前端运行界面如图：

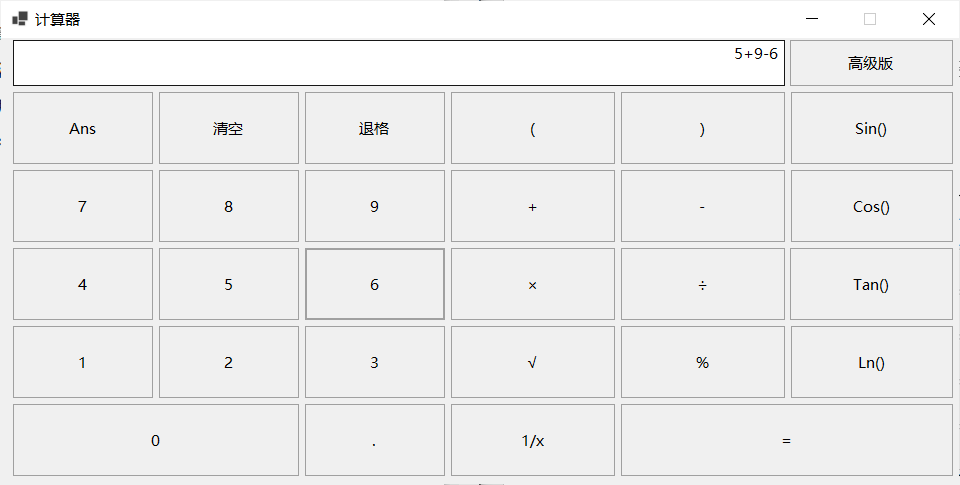


图19 基本计算器

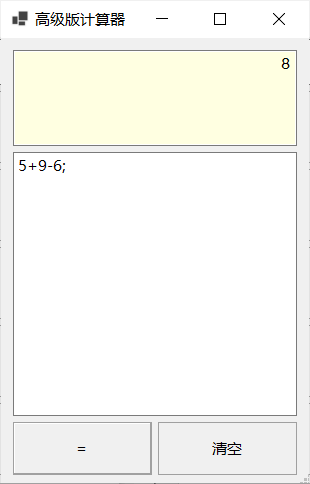


图20 高级版计算器

其中基本计算器与普通计算器无异，可以实现简单的四则运算、部分函数、以及部分非常规运算，通过点击操作在显示框内显示输入的计算公式，点击”=”按钮即可返回结果到显示框。高级版计算器可满足高级计算器的功能包括控制语句、变量命名、自定义函数等，用户在下方输入语句，点击”=”按钮后返回信息显示在上方区域。”清空”按钮可以清除上方和下方的内容。

高级版计算器功能展示：

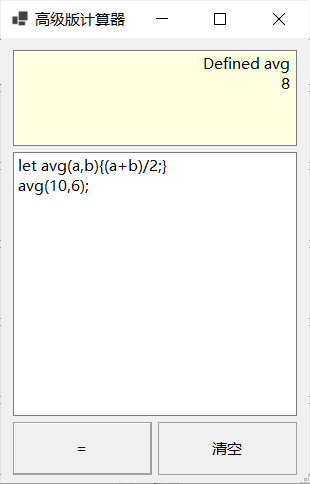
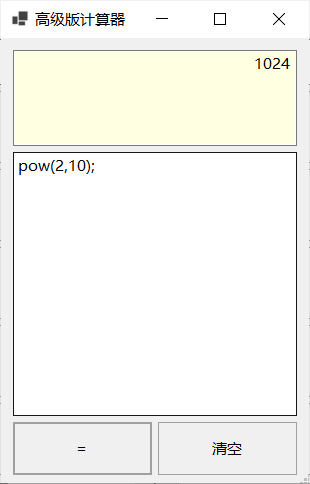


图21 内置函数 图22 自定义函数

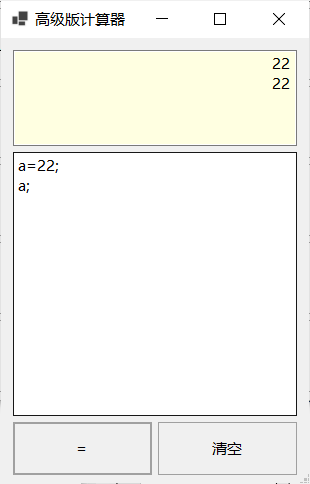
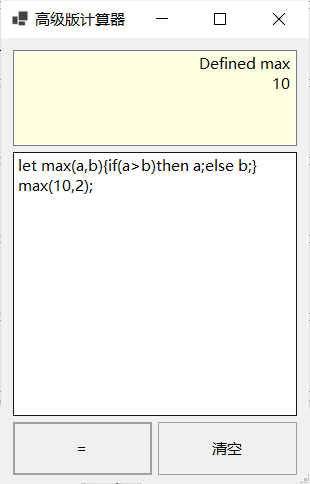


图23 控制语句 图24 变量命名

# 9 实验心得与总结

最终实现了这个高级计算器的可视化工作，开发了一个能够支持变量、过程、循环和条件表达式、提供标准函数、用户可自定义函数的多功能桌面计算器。通过本次的实验，加深了我对词法分析、语法分析及语义分析的理解，更加熟悉掌握了flex和bison结合构建编译器的原理，巩固了软件开发相关知识，同时增强了我的查错能力、代码能力以及查阅资料的能力，我可以说已经能够成功地运用一些工具来做一些实际的东西了，可谓是收获颇多！

但是仍旧有些许的不足，比如对编译器的封装dll文件未能实现，程序的一些错误处理不够完善，在所使用的开发工具及语言上没有太大创新仍旧使用的是以前接触过的语言，这些都有待于去提高和改进。希望未来自己的能力可以变得更强！

# 10 实验补充—python语言的dll文件调用

通过与同学间的讨论，得知可以通过gcc命令生成dll文件，命令如图：

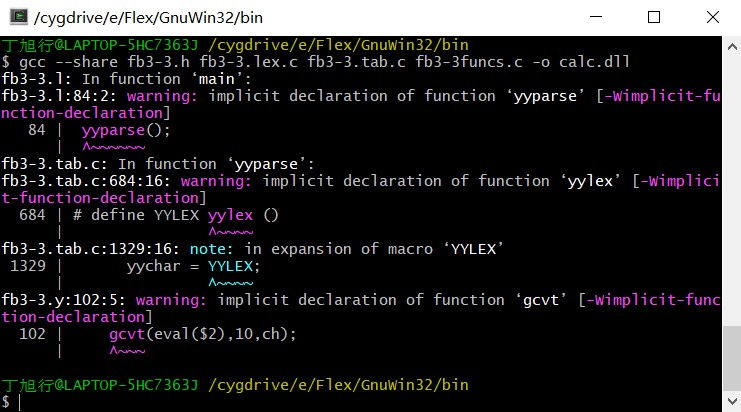


图25 gcc命令生成dll文件

但是因为没有匹配的lib文件，导致dll文件无法在C#环境下使用，所以我更换了编程语言使用python调用dll文件内的函数。我在码云上找到了使用pyqt5制作的科学计算器<https://gitee.com/ngc13009/scientific_calculator>，并在此基础上进行二次开发。

我在原本的科学计算器上添加高级版模块，点击弹出高级计算器子窗口，用户可以在此窗口输入表达式，点击“=“后返回结果值。

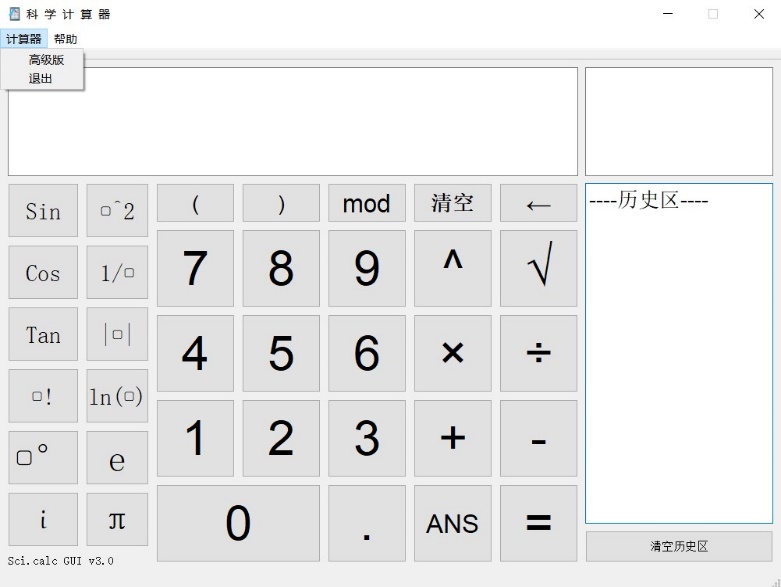


图26 高级版模块按钮 图27 高级计算器子窗口

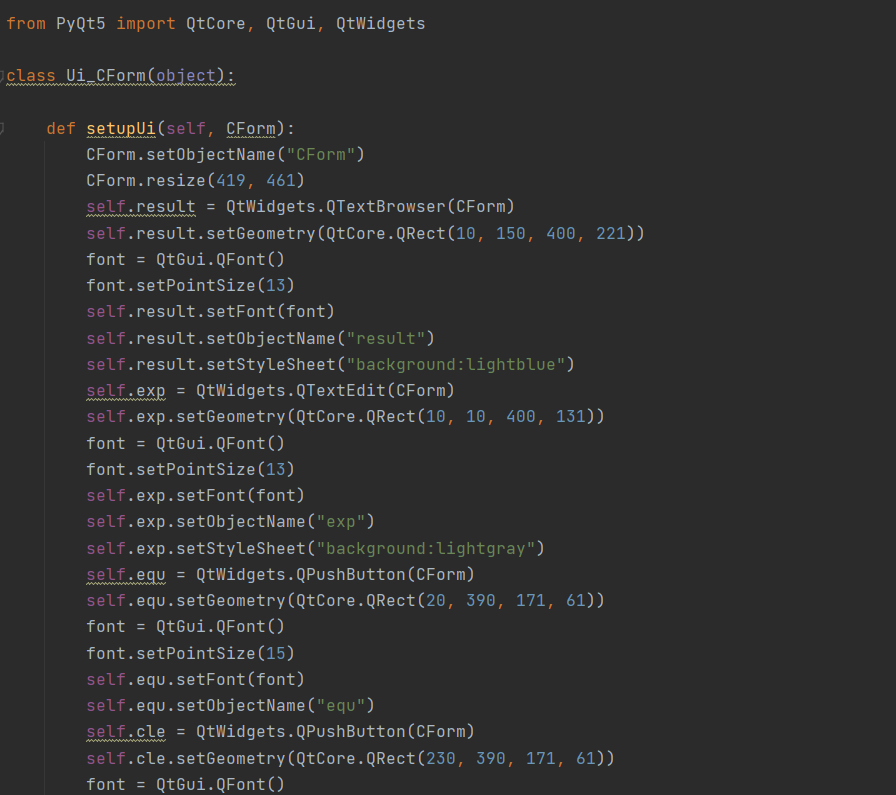


图28 子窗体部分代码

通过查找资料了解到了python调用dll文件的方法，其中最有价值的一篇文章是<https://www.php.cn/python-tutorials-422498.html，> 通过引入ctypes库，生成dll对象，进而调用对象内提供的公共函数：



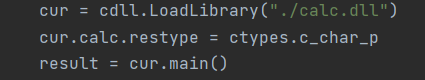
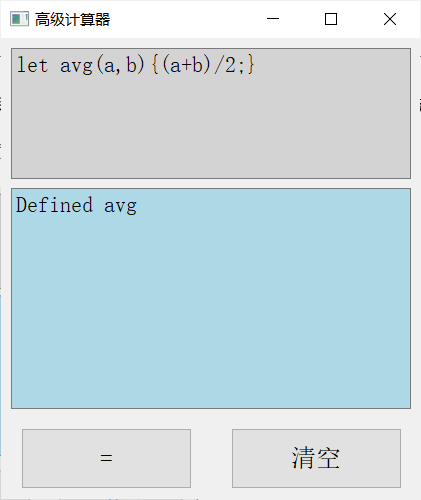
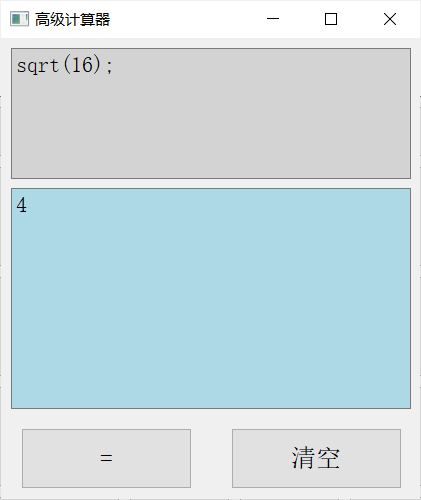


图29 关键代码

此计算器系统的总体架构与C#环境下的计算器架构一致，工作流程也相仿，在此不再赘述。最后运行效果如图：



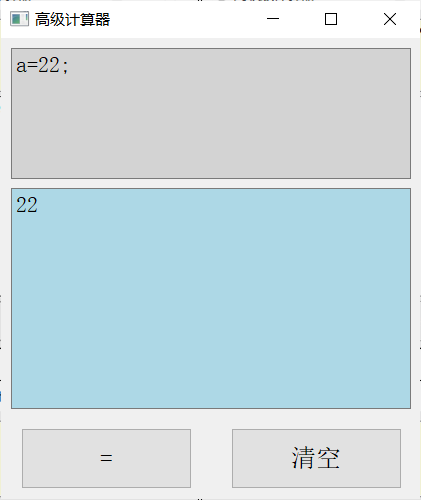
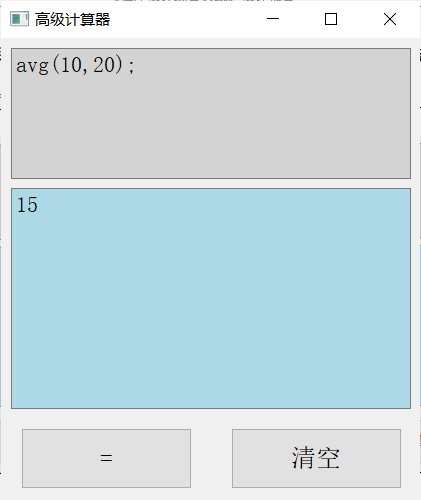


图30 运行效果图