## 实验八 使用 bison/flex编写高级计算器

### 一 实验要求

使用flex和bison开发了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。

计算器具体需要实现的功能：

变量命名；

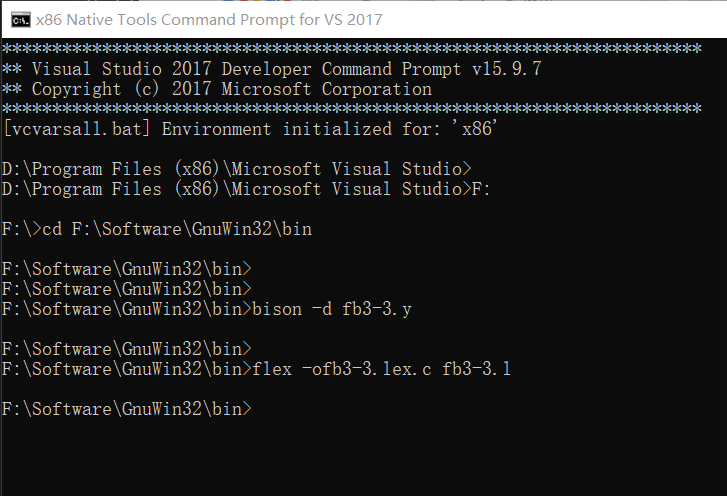
实现赋值功能；

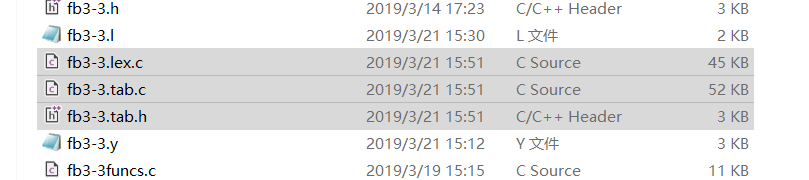
实现比较表达式（大于、小于、等于等等）

实现if/then/else和do/while的流程控制；

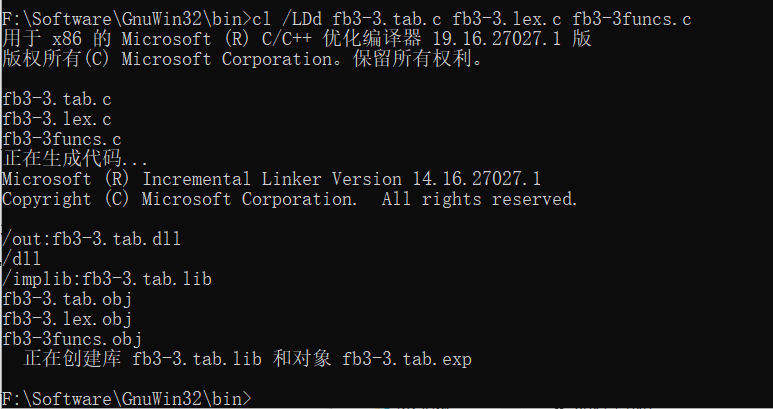
用户可以自定义函数；

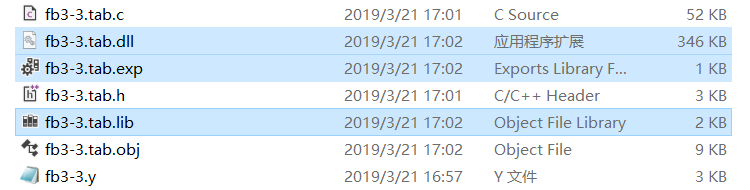
简单的错误恢复机制。

进行编译

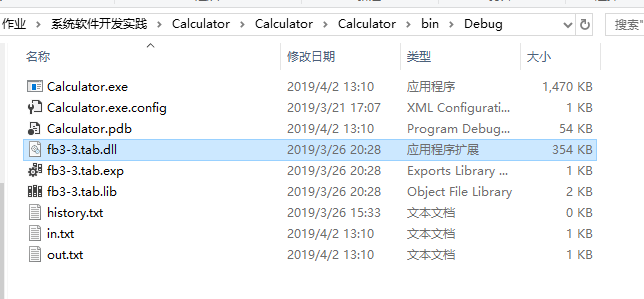
生成C源文件

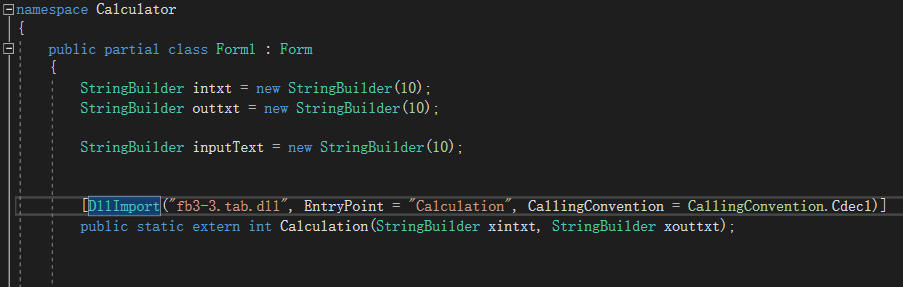
使用 cl /LDd fb3-3.tab.c fb3-3.lex.c fb3-3funcs.c 命令

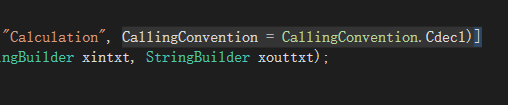
生成dll文件



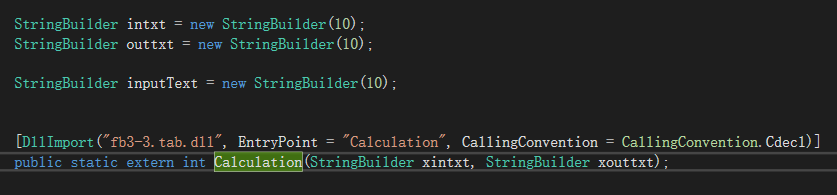
#### C#可视化

将生成的dll文件，拷贝到项目的Debug文件夹

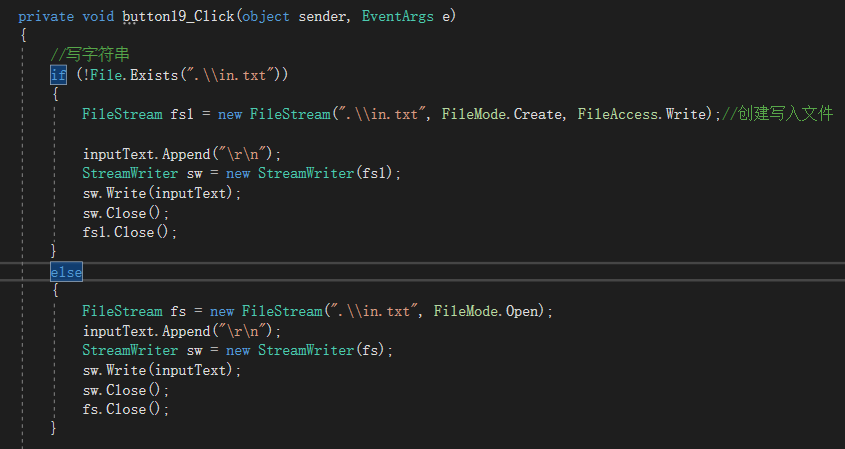
在窗体代码中，使用DllImport导入dll文件，并声明Calculation()函数。

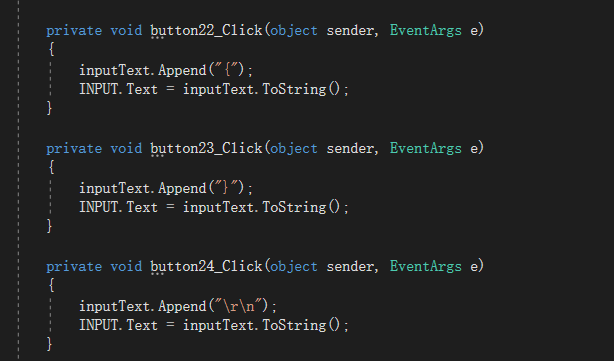
注意需要添加CallingConvention = CallingConvention.Cdecl，否则调用错误。

声明StringBuilder类型，因为C# Calculation()函数参数无法使用在C文件中定义的char\*类型，需要使用StringBuilder类型替代转换。



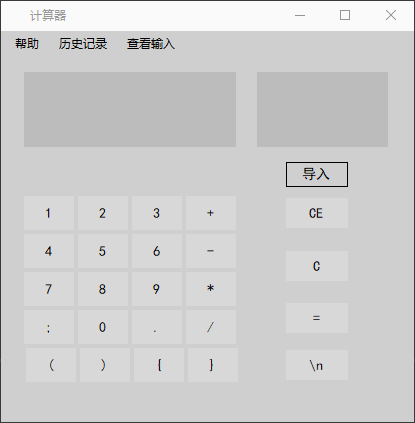
创建文件流，写入信息，并且调用Calculation(intxt, outtxt)，进行计算，并最终输出结果。



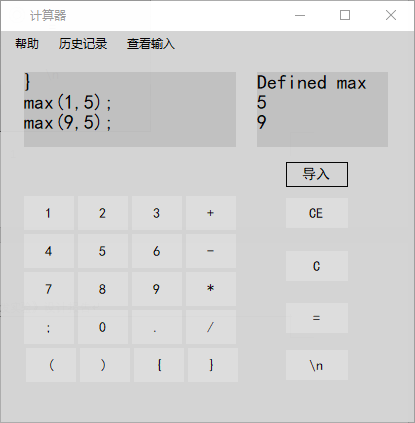
依次定义各个符号button的输入。

对可视化界面进行优化



实际运行显示效果

## 测试运行结果

直接在文本框输入的话，需要先点击导入按钮****

### 解决的问题

1. 语法的移进规约冲突
2. Dll中函数的声明，如何生成dll
3. 使用gcvt转化浮点数后会保留小数点的问题，通过使用sprintf解决。
4. C#与C文件中函数char\*参数类型的转换问题，通过使用StringBuilder解决。

## 实验心得：

1. 遇到的难题：

如何生成dll动态链接库，以及遇到了一些语法的移进规约冲突问题，但最后成功解决。C#可视化的一些细节实现。

1. 程序评价：

通过前几次实验的积累，以及这次实验的最终实践，实现了一个可视化的，并且具备完整功能的高级计算器。其各项功能完善，具备自定义函数的功能。可以通过点击按钮进行运算，也可以通过在文本框中输入算式进行运算。

1. 实验收获：

了解了dll动态链接库的作用，并且能够成功生成使用，对于以后自己如何C或C++的程序有了更多的经验。解决的一些移进规约冲突，对于语法分析有了更多的认识。最后自己编写开发出一个可视化的计算器，对于自己也是一个较大的鼓励。