# Plotting Tool

---Windows Programming Midterm Project

By: 1352958 金敏 软件工程 4 班
My Blog : www.cnblogs.com/guguli
My Github: www.github.com/yue9944882

注：源代码在提交文件中，源代及可运行 IDE 配置因规模过大已托管至本人 github 中：www.github.com/yue9944882/win32plot

## 项目要求：

**Requirements**:

1. No MFC or other third party GUI libraries allowed, all source code must be using Win32 API directly, or use your own encapsulation.
2. It is recommended to use C++ to encapsulate your own data structure. However, standard C++ library elements, such as vector, list are also allowed.
3. No third party controls, such as MSGrid, Crystal Report are allowed.

## 项目细节：（红色★ 标记为"完成实现"）

**基础要求：**

Basic functionality [80%]

1. ★ Plot display: X-axis, Y-axis, grid, tick marks and numbers,
2. ★ Plotting of the simplest math equation. E.g., f(x) = 2*x, f(x) = -5*x^2
3. ★ Plotting of the simplest function. E.g., f(x) = sin(x), f(x) = log(x)
4. ★ Display multiple lines with different colors.
5. ★ Adjustable background color, line color, tick distance, grid spacing, etc.
6. ★ Save plot as bitmap (can be opened by other image viewers).

**进阶要求：**

Advanced functionality [20%]

1. Zoom in / Zoom out and page scrolling
2. ★ Data source from file (in .csv format)

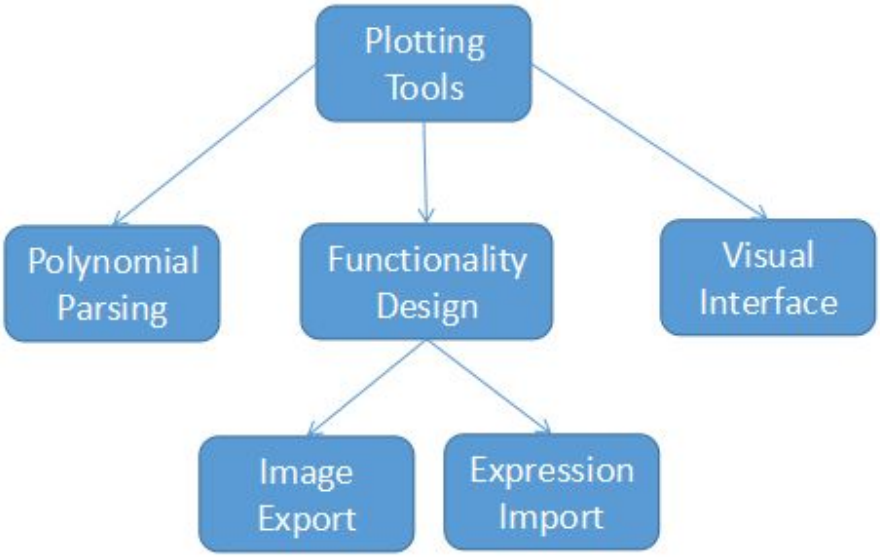
**额外要求：**

Extra functionality:

1. Save as other file formats (png, pdf, svg…)
2. Complicated math equation: f(x) = x^3 + 3*x^2/(5x+4) + sin(x)/(2*x^2)
3. Bar chart, pie chart…

# 项目设计思路：

首先整个项目我们分为如下图所示的几个模块进行分离地设计：

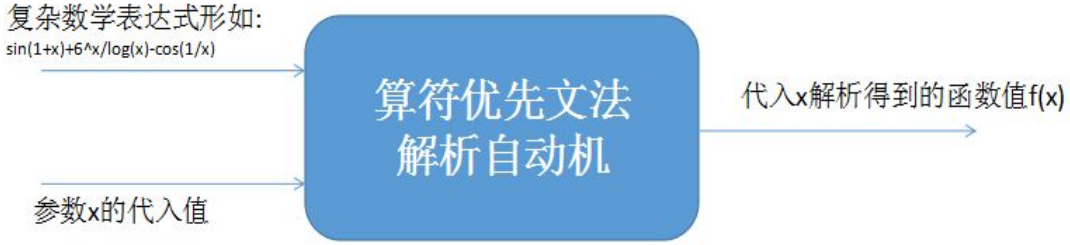



**第一部分：多项式解析**

在本次项目中，多项式的解析使用的是算法优先文法解析自动机进行解析：

在代码阶段我们将这一个模块作为一个单独的 branch 进行分离，作为一个黑匣子进行使用，使代码更易于维护且数据层次更加清晰，便于单元测试。

关于算法优先文法解析器的数学原理在这里就不多赘述，所谓黑匣子大致如下：



根据算符优先文法解析器的设计原理，我们总共有 11 种非终结符，其优先级表如下：

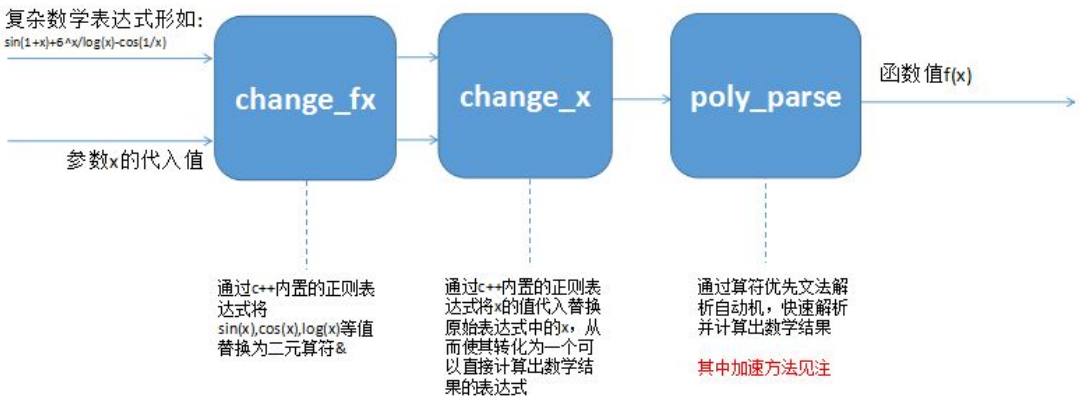
```
#define INF 99999

// VT-camparision GRAPH
// -1 for < , +1 for > , 0 for equal
//
//      \d   +   -   *    /   .   (    )   #    ^   &
// \d   -1   +1   +1   +1   +1   -1       +1   +1   +1   +1
// +    -1   +1   +1   -1   -1        -1   +1   +1   -1   -1
// -    -1   +1   +1   -1   -1        -1   +1   +1   -1   -1
// *    -1   +1   +1   +1   +1        -1   +1   +1   -1   -1
// /    -1   +1   +1   +1   +1        -1   +1   +1   -1   -1
// .    -1
// (    -1   -1   -1   -1   -1        -1   0         -1   -1
// )    +1   +1   +1   +1   +1             +1   +1   +1   +1
// #    -1   -1   -1   -1   -1        -1        0    -1   -1
// ^    -1   +1   +1   +1   +1        -1   +1   +1   -1   -1
// &    -1   +1   +1   +1   +1        -1   +1   +1   +1   -1

extern int cmp_prio[11][11];
```

整个解析算符优先文法的过程就是将表达式逐字符压入栈再弹出的过程，下面分别介绍多项式解析模块中每个文件中的函数和全局量的含义：

更细节地看，这个模块的结果如下所示：




parse.cpp: 整个多项式解析模块”黑匣子”的总入口 parse_poly 函数即从表达式得数学结果

Inter_change.cpp: 定义上图中的 change_fx 和 change_x 函数，功能如上所述

regex_trans.cpp: 利用 C++内置的 regex 类对+,-,*,/,^等基本运算进行计算并得到结果

VT_func.cpp: 初始化算法优先文法优先级表和相关结构

global_val.cpp: 全局变量的定义，如包括一个优先级表和一个速度优化的栈

其他各个头文件大部分是对各个函数和全局变量的前向声明和全局 extern 声明
包括： global_val.h inter_change.h VT_func.h regex_trans.h parse.h

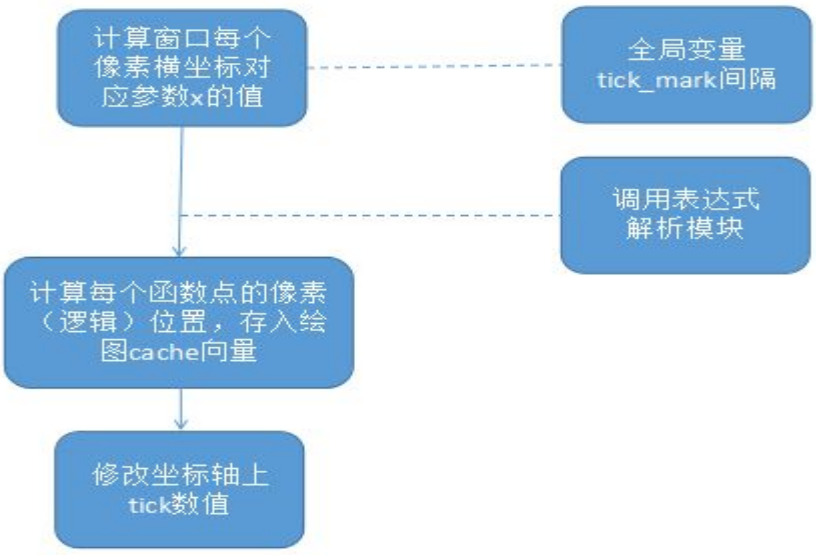
注：算符优先文法速度优化方法

在使用自动机进行算符优先文法的解析的时候，我们会不断地将一个非终结符 N 压入表达式栈中，直到整个符号栈中只剩下#N#为止，在这个时候我们一般的做法是维护一个符号表，并且用一个 allocate 符号的符号池来实现，这种做法一定程度上会减慢程序的运行速度，而我们的做法是维护一个非终结符的栈 N_stack 来通过弹出压入操作，自动机和 N_stack 的逻辑是完全隔离的，在不影响自动机运行原理的前提下，我们加快非终结符的记录操作。（具体实现在 inter_change.cpp 中）

## 第二部分：内部功能设计

首先来看内部功能之中的最重要的绘图操作背后的数据计算：

不妨先默认整个绘图的空间是 500*500 的像素空间，为了实现坐标轴间距，跳间隔数的修改，我们将绘图逻辑和函数计算逻辑分离，并且设计为异步式逻辑。



从数学值坐标到窗口显示的逻辑值坐标的转换是一个完全线性的工作，即从全局变量限制的数学空间映射数据到绘图空间，与函数像素点 cache 的维护类似的是，我们创建一个向量用来保存坐标轴显示的上 tick 的间隔值。

**绘制 BMP 格式图像**

绘制 BMP 格式的图像我们使用的是 GetDIBits 函数，这个函数完成的是一个”截图”的工作，在调用一些其他的 API 进行修改之后，我们就得到了绘图空间的图像，由一个 HBITMAP 指向。再使用 windows.h 中声明的 BITMAPFILEHEADER 结构体直接进行二进制写入文件，文件的命名就用截图的时间格式化串。

**绘制 PNG 格式图像**

绘制 PNG 格式的图像相比 BMP 格式图像绕了一些弯，因为 win32 的 api 中并不像 bmp 提供了那么多可用的 api 给我们直接调用，也没有结构体以便我们直接赋值就可以进行二进制输出。我们需要加入一个头文件<gdiplus.h>，根据官方文档，从 windows2000 版本之后的 windows 版本都内置了 gdi+的支持和 win32-api 已经合为了一体，我们调用个 gdi+的办法来实现输出 png 格式的图像文件。

其次是支持外部文件的导入操作，由于 csv 格式比较简单，这里就不多赘述，只需要使用 c 语言的<string.h>头文件中声明的函数就可以完美处理，唯一的一点小问题是宽字符串处理的函数和我们经常使用的 strxxx 函数名称不一样，需要查阅官方文档。

## 第三部分：交互窗口设计

窗口的设计我们逐控件进行分析：

**绘图按钮**

绘图按钮的逻辑：

灰化按钮->从全局向量绘制函数（阻塞）->亮化按钮

**绘图平面**

绘图平面由 500*500 的像素空间构成，其中函数点之间的连线通过 winapi 中的：

```
BOOL MoveToEx( HDC hdc, int x, int y, LPPOINT lpPoint );
BOOL LineTo( HDC hdc, int xEnd, int yEnd );
```

**菜单--调整画笔颜色，背景颜色**

程序维护一个静态全局变量 penStyle 来实时调整画笔的颜色
在每次绘图计算的时候，压入全局向量之前都会以全局画笔颜色作为参数

**菜单--调整距离标记的值间隔**

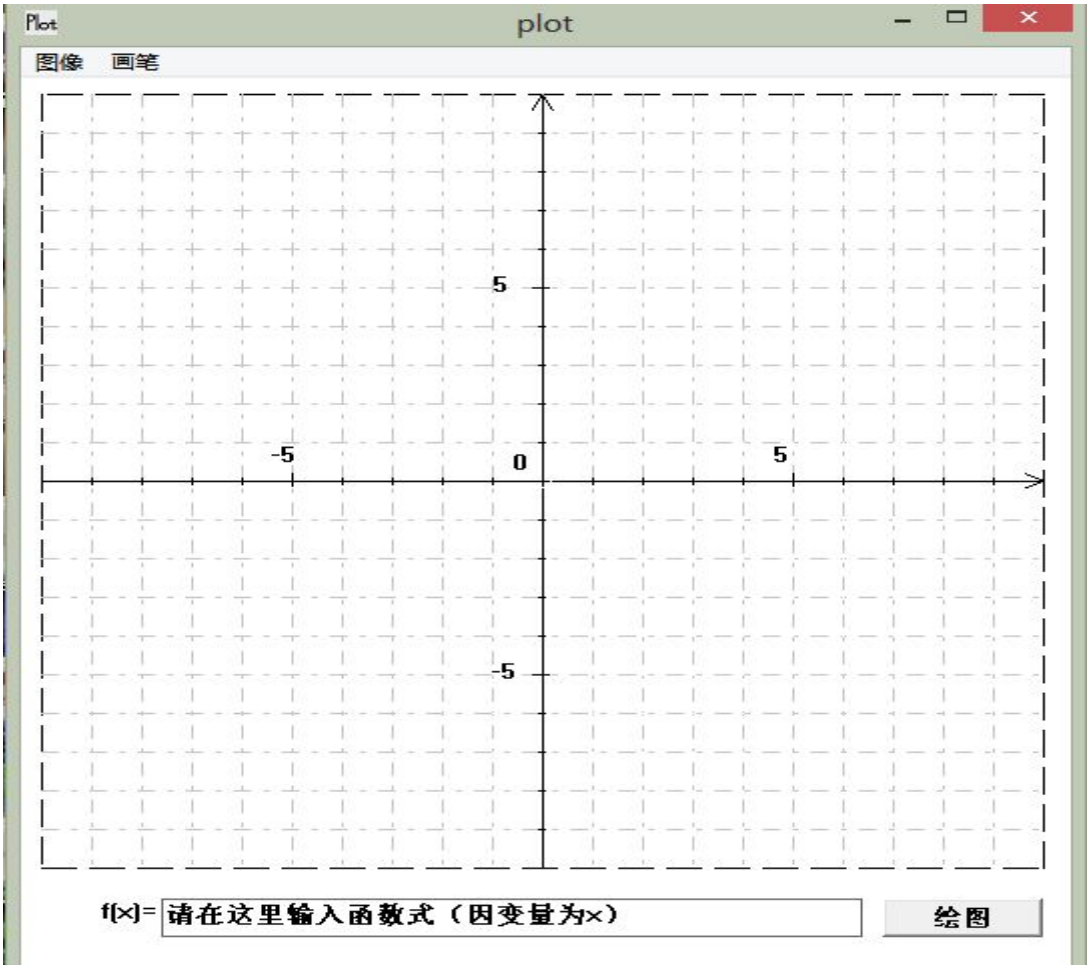
程序同样维护一个静态全局变量 tick_dist 来实时调整数值间隔的大小
绘图函数同样以此作为参数，并且每次调整距离都会附带一次清空画板的操作

**菜单--导出图像(BMP,PNG..)，导入文件**

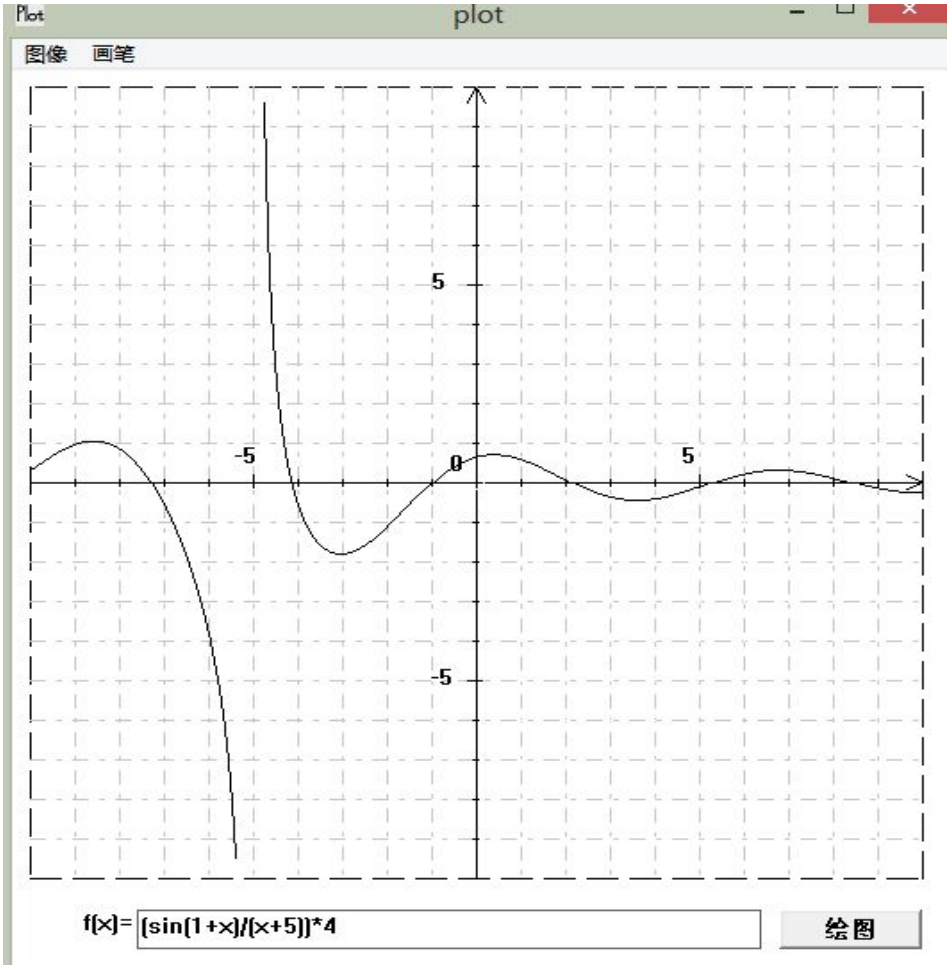
导出 BMP 和 PNG 图像文件和导入 CSV 格式文件的具体原理已经在上文中解释
都是比较直观的流式逻辑这里暂不赘述。

# 项目实现结果：

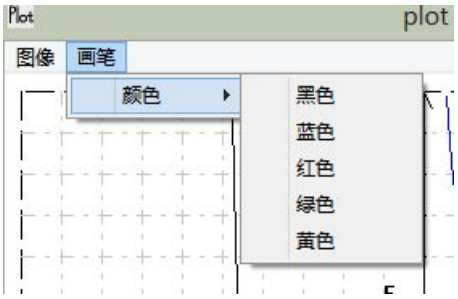
程序运行的初始画面如下所示：



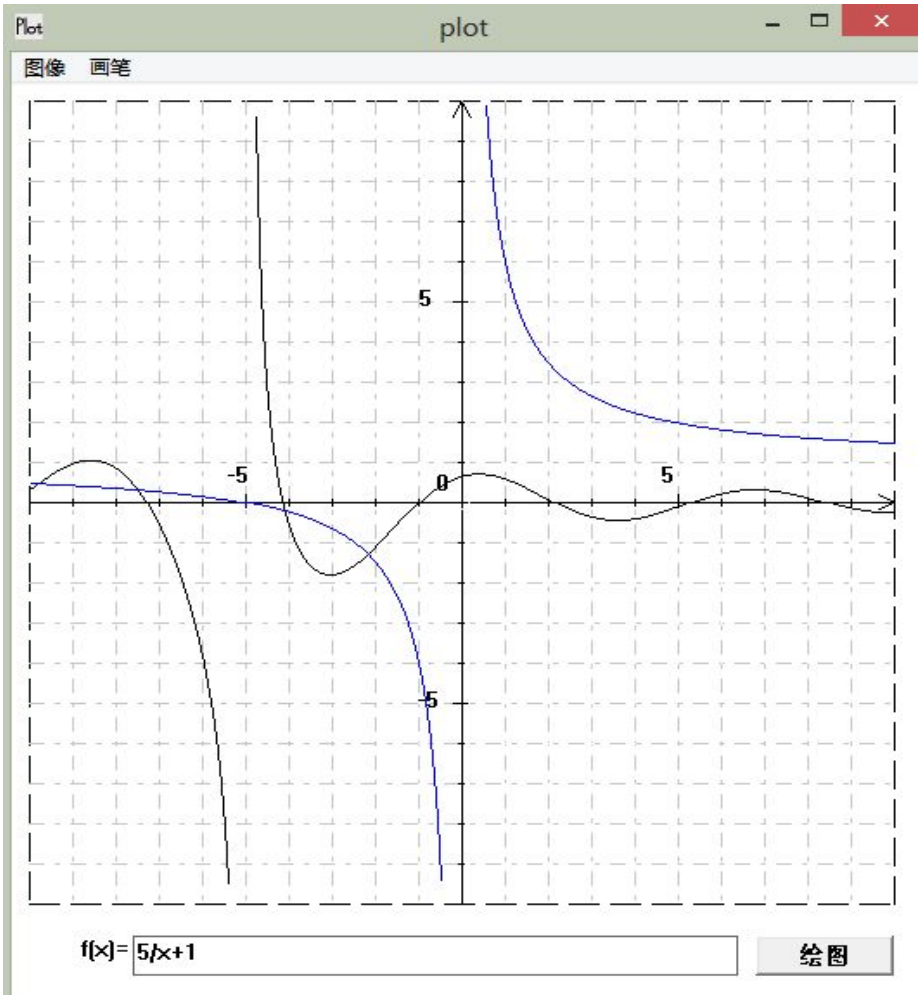
输入一个较复杂的表达式进行绘图 f(x)=(sin(1+x)/(x+5))*4




窗口中每个菜单的及其子菜单如下所示：

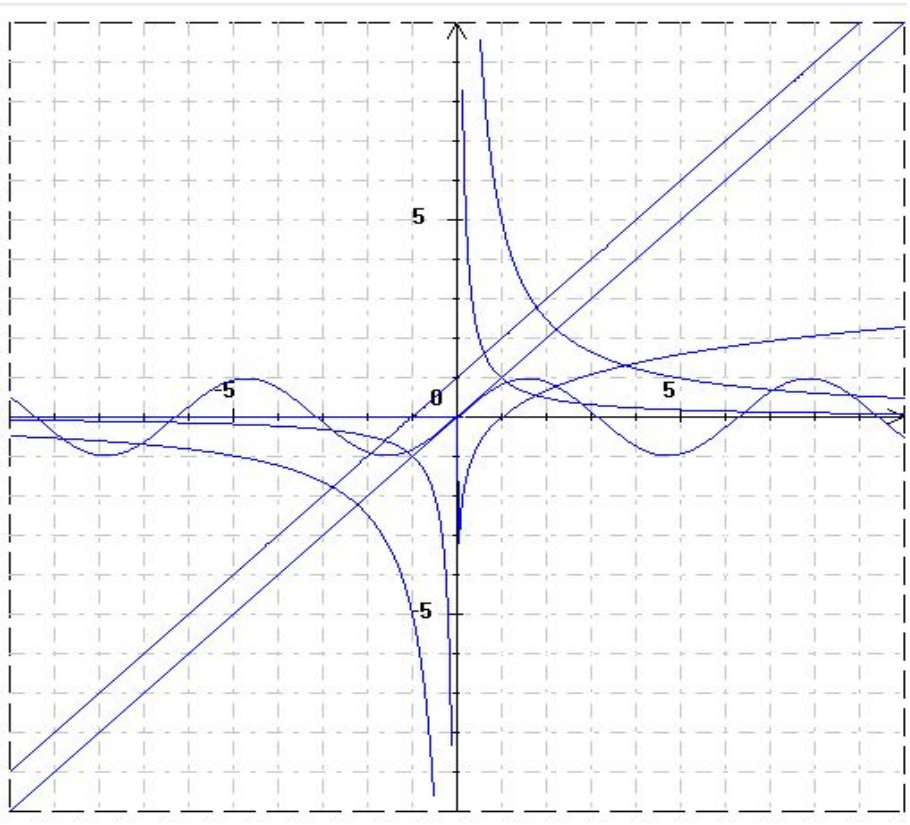
将画笔调整为蓝色，再绘制另一个函数的图像（图中以 5/x+1 为例）



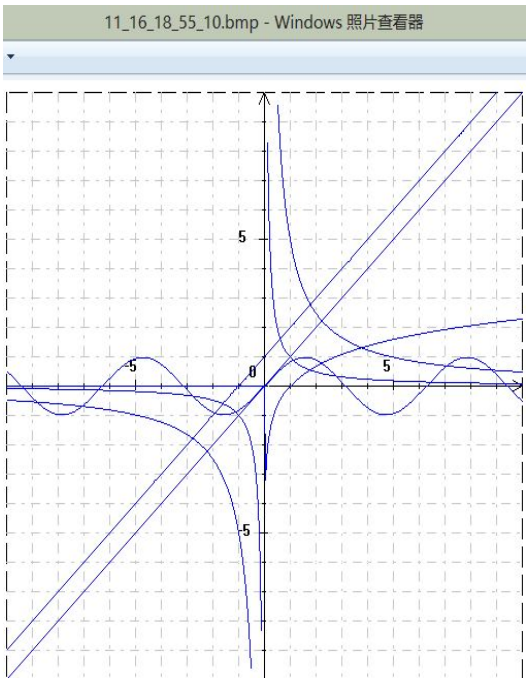

导入 CSV 格式文件示例 （ 使用 fx.csv 文件内容如下所示 ）

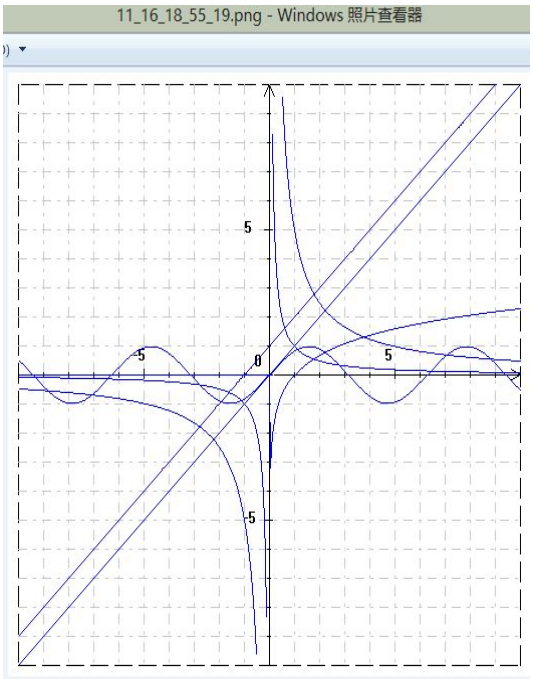
```
1+x
x,log(x)
5/x,1/x
sin(x)
```

导入时会不断更新画板中的函数图像，最终结果如下图所示：




导出得到的 bmp 格式图像及 png 格式图像。



# 项目结束反思：

函数绘制有抖动现象

相邻点之间插值过大会导致函数绘制”断层”现象

函数绘制的速度还可以改善

没有实现动态”滚轮”缩放