

**程 序 设 计 专 题**

**大 程 序 报 告**



1. **学生姓名 ： 学号：**
2. **学生姓名 ： 学号：**
3. **学生姓名 ： 学号：**

**指导老师： 陈建海**

**2018~2019春夏学期 2019 年 5 月 31 日**

**报告撰写注意事项**

1. 图文并茂。文字通顺，语言流畅，无错别字。
2. 书写格式规范，排版良好，内容完整。
3. 存在拼凑、剽窃等现象一律认定为抄袭；0分

# 题目描述和题目要求

单变量多项式的化简与绘图

基于libgraphics，实现鼠标点击输入单变量多项式，化简输出到文件中，并绘制函数的图像。

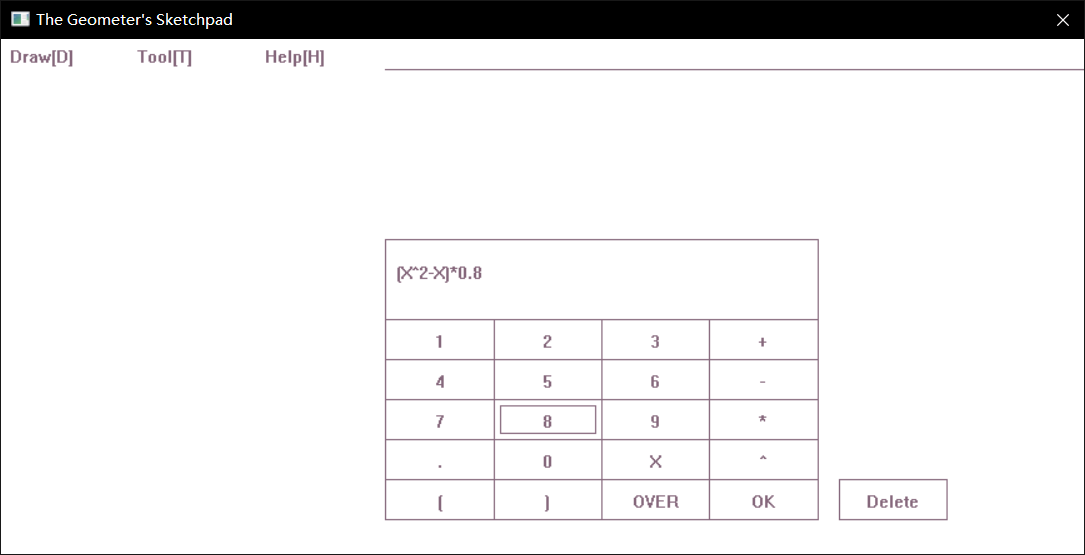
|  |  |  |  |
| --- | --- | --- | --- |
| 考评项 | | 分数 | 说明 |
| 菜单系统 | | 10 | Draw：输入新函数、清除当前界面、退出  Tool：求导、求积分、求交点  Help：开发者信息、功能介绍、 |
| 图标工具栏 | | 5 | 在输入新函数时，有OVER、OK、Delete三个功能来控制返回，输入完毕，清除字符。 |
| 快捷键 | | 5 | 与菜单功能对应 |
| 状态信息栏 | | 5 | 在窗口底部，即时显示操作的中间状态/结果 |
| 功能（35）  （分项自行添加） | 开始界面图形绘制 | 5 | 绘制美观、菜单栏功能完善 |
| 输入数据 | 5 | 点击输入/清除数据，显示当前已输入的结果 |
| 处理化简  数据 | 10 | 判断数据合法性，化简为升幂最简多项式 |
| 绘制图像 | 10 | 建立直角坐标，系绘制图像 |
| 保存为文件 | 5 | 将化简后的多项式保存为文本文件 |
| 链表 | | 5 | 利用链表存放化简后函数，以便绘制函数图像 |
| 文件 | 文件写入 | 10 | 保存化简后的表达式到文本文件 |
| 多文件组织 | 5 | 系统程序必须采用多文件组织的方式 |
| 大程序报告 | 分析和设计 | 5 | 软件简介，功能结构，全局、函数及重要算法说明，源程序中功能、函数、文件的组织关系 |
| 部署和运行 | 5 | 编译安装、运行测试、用户使用手册 |
| 分工 | 5 | 成员各自承担任务、挑战点和感言 |
| 合作 | 5 | 合作记录、系统开发亮点和应用知识点总结 |
| 总分 | 100 | | | |

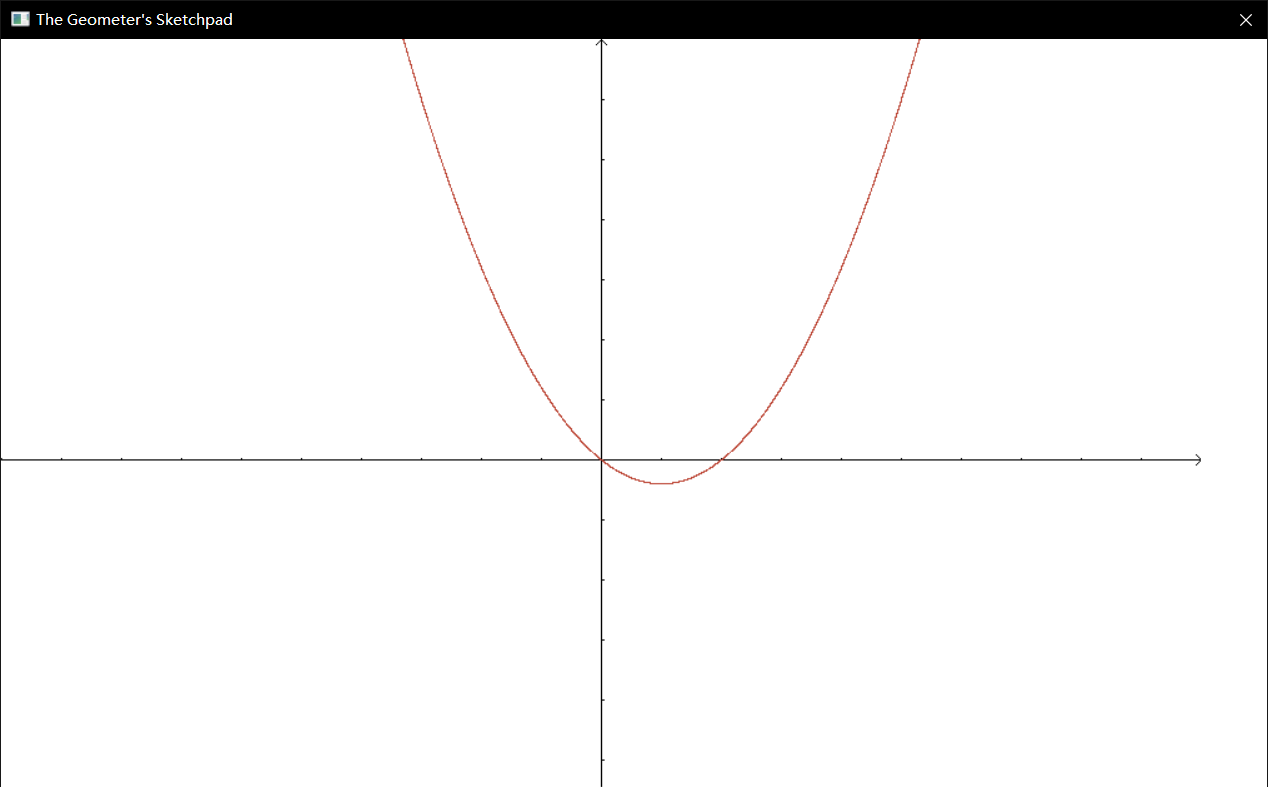
）

# 需求分析

目的：通过点击box输入一串包含0123456789X^\*-+()的字符，判断是否合法，如果不合法，在屏幕上绘制错误提示；如果合法，在屏幕上绘制该函数的图像，并把该函数升幂排列的化简结果输出到文件中。

例如：





# 3、总体设计

## 3.1 功能模块设计

分界面、输入、处理、输出文件、绘图五部分

界面：菜单功能，介绍功能

输入：点击输入，点击删除，当前内容显示

处理：化简表达式

输出文件：输出已化简表达式的文本文件

绘图：绘制函数图像、移动函数图像

## 3.2数据结构设计

核心数据流：以字符串形式输入，存入自定义结构体，利用栈进行处理，输出成为文件或存入链表再打印成为图像。

其中自定义数据结构有：

struct List{/\*输出的链表\*/

double coefficient;

double power;

struct List \*next;

};

typedef struct List List;

typedef struct Polynomial{/\*polynomial，多项式\*/

int cnt;/\*共cnt项\*/

double coef[MAX];/\*coef，系数\*/

double pow[MAX];/\*pow，幂次\*/

}PN; /\*用PN结构体存储多项式的项数、系数、幂指数\*/

struct Calculus\_data{

double up\_limit;

double down\_limit;

int function\_serial;

};//计算积分的结构体

typedef struct Calculus\_data Calculus\_data;

struct Derivation\_data{

double local;

int function\_serial;

};//计算导数的结构体

typedef struct Derivation\_data Derivation\_data;

struct Coordinate{

double x;

double y;

};

typedef struct Coordinate Coordinate;

## 3.3 函数功能描述

函数原型：void drawMenu(void)；  
功能描述：绘制菜单并实现相应功能的选择，以便之后对菜单栏功能的操作  
参数描述：void  
返回值描述：void  
重要局部变量定义：static char \* menuListDraw[] = {"Draw[D]",    
"New Function | Ctrl-O",   
"Clear All | Ctrl-E",  
"Exit | Ctrl-P"};  
static char \* menuListTool[] = {"Tool[T]",  
"Derivative | Ctrl-F",  
"Accumulate Points | Ctrl-L",  
"Crossover Points | Ctrl-H"};  
static char \* menuListHelp[] = {"Help[H]",  
"Developer Information  | Ctrl-M",  
"Function Introduction | Ctrl-N",  
"Return GUI | Ctrl-B",  
"Exit Help | Ctrl-K"};  
  
double fH = GetFontHeight();  
double x = 0; //fH/8;  
double y = winheight;  
double h = fH\*1.5;   
double w = TextStringWidth(menuListDraw[0])\*2;   
double wlist = TextStringWidth(menuListHelp[1])\*1.2;  
double xindent = winheight/20;   
int selection;  
重要局部变量用途描述：static char \* menuListDraw[] = {"Draw[D]",    
"New Function | Ctrl-O", // 快捷键必须采用[Ctrl-X]格式，放在字符串的结尾  
"Clear All | Ctrl-E",  
"Exit | Ctrl-P"};  
static char \* menuListTool[] = {"Tool[T]",  
"Derivative | Ctrl-F",  
"Accumulate Points | Ctrl-L",  
"Crossover Points | Ctrl-H"};  
static char \* menuListHelp[] = {"Help[H]",  
"Developer Information  | Ctrl-M",  
"Function Introduction | Ctrl-N",  
"Exit Help | Ctrl-K"};  
上述数组用于储存菜单栏中的细分项并且含有对快捷键的设置  
double fH = GetFontHeight();  
double h = fH\*1.5; // 控件高度  
double w = TextStringWidth(menuListDraw[0])\*2; // 控件宽度  
上述变量设置了控件的高度和宽度  
double wlist = TextStringWidth(menuListHelp[1])\*1.2;  
该变量设置了list的宽度  
int selection;  
该变量用于描述被选中的功能  
double x = 0; //fH/8;  
double y = winheight;  
上述变量用于确定菜单栏的位置  
函数算法描述：选择算法  
  
  
函数原型：void MouseEventProcess(int x, int y, int mybutton, int event)；  
功能描述：判断选择区域实现输入数据的储存与显示  
参数描述：int x, int y, int mybutton, int event；  
返回值描述：void；  
重要局部变量定义：   
double X,Y;  
static int Inputk=0;  
double fh=GetFontHeight();//字体高度   
double butw=10\*TEXTWIDTH,buth=2\*fh;  
int i,j,m=0;  
  
X=ScaleXInches(x);  
Y=ScaleYInches(y);  
重要局部变量用途描述：  
double X,Y;  
X=ScaleXInches(x);  
Y=ScaleYInches(y);  
上述变量用于定义inch单位下的坐标X,Y以用于后面相应的函数功能  
double fh=GetFontHeight();//字体高度  
确定字体高度  
double butw=10\*TEXTWIDTH,buth=2\*fh;  
按键的宽度及高度  
static int Inputk=0;  
用于标志字符储存的位置  
函数算法描述：使用循环功能判别选择区域，用switch语句分类  
  
  
函数原型：void DrawInput(void);  
功能描述：用于绘制输入界面以及显示输入数据  
参数描述：void  
返回值描述：void  
重要局部变量定义：   
double lx,ly;  
double fh=GetFontHeight();   
int i,j,k=0;   
double butw=10\*TEXTWIDTH,buth=2\*fh;  
重要局部变量功能描述：  
double lx,ly;  
上述变量为输入函数界面的左上角的坐标  
double fh=GetFontHeight();  
上述变量为字体高度   
int i,j,k=0;  
上述变量为循环变量   
double butw=10\*TEXTWIDTH,buth=2\*fh;  
上述变量为按键的宽度以及按键的高度设置  
函数算法描述：  
  用了循环语句来实现按键的绘制  
  
  
函数原型：void DevelopInformation(void)  
功能描述：用于显示开发者相关信息  
参数描述：void  
返回值描述：void  
重要局部变量定义：   
double fh=GetFontHeight();   
double fw1=2\*TEXTWIDTH, fw2=TEXTWIDTH;   
double center=winwidth/2,left=center-13\*fw2;   
double inheight=winheight-7\*fh;   
double interval=fh,fh1=2\*fh;   
double x,y;  
int i,j,k=1;  
重要局部变量用途描述：  
double fh=GetFontHeight();  
该变量为英文字体高度   
double fw1=2\*TEXTWIDTH, fw2=TEXTWIDTH;  
上述变量分别为汉字宽度和英文字母宽度   
double center=winwidth/2,left=center-13\*fw2;  
上述变量分别用于确定中间对齐位置和开头对齐位置   
double inheight=winheight-7\*fh;  
该变量为初始文本显示高度   
double interval=fh,fh1=2\*fh;  
上述变量分别为行间隔和汉字高度   
double x,y;  
    该变量用于确定显示位置  
int i,j,k=1;   
    前两个变量用于循环结构，后一个变量用于选择对应的字符  
函数算法描述：循环语句  
  
  
函数原型：void FunctionIntroduction(void);  
功能描述：用于显示函数介绍  
参数描述：void;  
返回值描述：void;  
重要局部变量定义：   
double fh=GetFontHeight();//中文字体高度  
double fw= 2\*TEXTWIDTH;//汉字宽度  
double lx=winwidth/2-23\*TEXTWIDTH;  
double ly=winheight-7\*fh;  
重要局部变量用途描述：  
double fh=GetFontHeight();  
该变量用于中文字体高度  
double fw= 2\*TEXTWIDTH;  
该变量为汉字宽度  
double lx=winwidth/2-23\*TEXTWIDTH;  
double ly=winheight-7\*fh;  
   上述变量用于确定显示的位置  
函数算法描述：无  
  
  
  
函数原型：void drawpicture(void);  
功能描述：用于显示界面装饰图案  
参数描述：void  
返回值描述：void  
重要局部变量定义：   
double fh=GetFontHeight();  
double butw=10\*TEXTWIDTH,buth=2\*fh;  
double starlength[3]={2\*TEXTWIDTH,3\*TEXTWIDTH,2\*TEXTWIDTH};  
char \*StarColor[]={"Galaxy Blue","Forest Biome","Niagara"};  
int i,j;  
重要局部变量用途描述：  
double fh=GetFontHeight();  
    该变量用于储存字体高度  
double butw=10\*TEXTWIDTH,buth=2\*fh;  
    上述变量分别用于储存按键宽度及按键长度（用于图案位置的确定）  
double starlength[3]={2\*TEXTWIDTH,3\*TEXTWIDTH,2\*TEXTWIDTH};  
    上述变量用于保存star一边的长度  
char \*StarColor[]={"Galaxy Blue","Forest Biome","Niagara"};  
    上述字符指针用于储存相关的颜色  
int i,j;  
    上述变量为循环变量  
函数算法描述：用了循环语句来实现相应功能

函数原型：void PrintPN(PN a);

功能描述；将一个PN结构体输出成为文本文件

参数描述：PN a，自定义结构体

返回值描述：void

重要局部变量定义：FILE \*fp;

重要局部变量用途描述：指向文件输出地址的指针

函数算法描述：循环遍历

函数原型：void AddTwoPN(PN\* a,PN\* b)

功能描述；将两个多项式相加，存入第一个多项式。

参数描述： PN\* a：第一个参与加法的多项式的指针

PN\* b：第二个参与加法的多项式的指针

返回值描述：void

重要局部变量定义：int flag[MAX]={0};

重要局部变量用途描述：flag[i]记录b多项式中第i项是否参与了第一次运算。

函数算法描述：先遍历b多项式，发现由于a中某项幂次相同的，就把系数加到a中对应项上，并记录参与了第一次运算。再遍历b多项式，把b中未参与运算的项全部添加到a多项式后面，最后调用SortPN函数进行整理，使之保持升序且最简。

函数原型：void MultiTwoPN(PN\* a,PN\* b);

功能描述；将两个多项式相乘，存入第一个多项式

参数描述： PN\* a：第一个参与乘法的多项式的指针

PN\* b：第二个参与乘法的多项式的指针

返回值描述：void

重要局部变量定义：PN\* ptmp1=&temp1, \*ptmp2=&temp2; int ptr=0;

重要局部变量用途描述：两个ptmp指向两个PN类型的temp全局变量，temp1用于暂存a中某项与b相乘的中间结果，temp2用于存储中间结果的累加和。

函数算法描述：遍历b，使每项都与a中某项乘，存入temp1；再把temp1与temp2加存入temp2。在该运算的外层遍历a，使得每一项都相乘相加。最后让temp2中的值存入a所指内存空间。

函数原型：void SortPN(PN\* a)

功能描述；将某多项式整理，使得重复项合并，系数为零项剔除，升幂排列。

参数描述： 待整理表达式的指针a

返回值描述：void

重要局部变量定义：double min=-2147483647,tmp =2147483647; PN\* b=&temp1;

int ptr=0;

重要局部变量用途描述：

min：上次流程中的最小值

tmp：1.作为找到当前“极小值”的中间变量

2.作为循环跳出的标识条件

b:存放整理过程中的中间结果

ptr：记录b尾项的类似“指针”的整型。

函数算法描述：遍历a，每次初始化tmp的值为最大值，找出凡是幂次比第min项大且比第tmp项小的，赋给第tmp项，即找出当前大于第min项最小的幂次并把该项的序数记为tmp。随后遍历b，如果b中无与该项的幂次相同的项，就在b中第ptr项存入a中原tmp项的数值。如果b中有与该项幂次相同的项，就在b中相同幂次项的系数上加上该项系数。如果b中第ptr项的系数是不是零，ptr自增，可以在尾部写入，反之则覆盖写入。进行以上过程直到某次a遍历结束后tmp值未改变，把ptr赋给b的cnt（即总计项数），把b赋给a所指的结构体。

函数原型：PN ReadPN(int len,int p,int\* q,char\* s)

功能描述；读入一个elmt（“元素”），即一个单项的多项式。

参数描述：

len，总长度；

p，从第p位开始读；

q返回读取停止的位置是q；

s，待读取字符串

返回值描述：返回一个PN多项式结构体,即读取好的单项的多项式

重要局部变量定义：PN\* a; double coef=0,pow=0;

重要局部变量用途描述：

a：将读入值存入a所指结构体

coef：暂存系数变量

pow：暂存幂指数变量

函数算法描述：调用自己实现的ReadDight函数判断字符串待读入位是否是数字（含小数点），如果不是数字，令coef为1，p++ ，判断下一位是否为^，如果不是，pow为1，q所指值变为p。如果是，pow就用ReadDight读入一个浮点数。另一方面，如果读入位是数字，那coef就用ReadDight读入的浮点数，p改变为返回的q的值，接着判断第p位是否是X，否则pow为零，是则p++，判断下一位是否为^，如果不是，pow为1，q所指值变为p。如果是，pow就用ReadDight读入一个浮点数。最后把他们pow，coef存入a所指结构体，令a的cnt为1，返回a的值。

函数原型：int isStrLegal(char\* str)

功能描述；判断读入字符串是否合法

参数描述： 字符串指针str，读入的字符串

返回值描述：整型，实际上是0/1的布尔值。

重要局部变量定义：int：bracket=0/\*括号对应性\*/,legal=1/\*合法布尔值\*/,dot=0/\*单点性\*/

重要局部变量用途描述：

bracket=0/\*括号对应性\*/

legal=1/\*判断合法与否的布尔值\*/,

dot=0/\*单点性\*/

函数算法描述：

列举合法字符串的字符接续原则，判断之。

函数原型：PN Rpn(char\* str)

功能描述；进行多项式之间有优先级的+-\*运算，结果为升幂最简多项式。

参数描述：char\* str传入的字符串

返回值描述：PN结构体：结果多项式

重要局部变量定义： char op[MAX];int oTop,eTop;PN elmt[MAX];

重要局部变量用途描述：

char op[MAX]; operator栈，操作符栈。

int oTop;操作符栈顶指针。

int eTop;element栈，多项式结构体栈，操作数栈。

PN elmt[MAX];操作数栈顶指针。

函数算法描述：逆波兰式算法，重定义了操作数栈（变为结构体数组），运算（变为Add，Minus，Multi函数）

函数原型：List\* Change(PN a)

功能描述； 把多项式结构体转成链表

参数描述： a，待转换结构体

返回值描述： List\*，结果链表的指针

重要局部变量定义：struct List \*head=NULL,\*pre,\*newNode;

重要局部变量用途描述：创建链表的头，下节点，新节点。

函数算法描述：生成链表的一般算法，生成的同时赋初值。

函数原型：

功能描述；

参数描述：

返回值描述：

重要局部变量定义：

重要局部变量用途描述：

函数算法描述：

## 3.4 程序文件结构

1）文件函数结构

分为七个文件：

main.c standard.h standard.c functiondisplay.c functiondisplay.h

其中每个c文件中函数都在h中得到了声明

standard文件的函数有：

int Dequal(double a,double b);/\*double 相等\*/

int isDight(char a);/\*某字符是不是数字或者小数点\*/

void PrintPN(PN a);/\*将一个PN结构体输出成为文本文件\*/

void PrintList(struct List\* ptr); /\*打印一个链表（—调试用—）\*/

int isStrLegal(char\* a);/\*判断字符串合法性\*/

/\*IO及运算重定义\*/

int Read(char\* str,int pstr);/\*读取一项，存入结构体\*/ /\*一项的定义是：从数字或X开始，到op结束\*/

void AddTwoPN(PN\* a,PN\* b);/\*将两多项式相加,存入第一项\*/

void MultiTwoPN(PN\* a,PN\* b);/\*相乘\*/

void MinusTwoPN(PN\* a,PN\* b);/\*相减\*/

void SortPN(PN\* a);/\*将某多项式整理，使得重复项合并，系数为零项剔除，升幂排列\*/

/\*IO及运算重定义\*/

/\*RPN相关\*/

PN ReadPN(int len,int p,int\* q,char\* s);/\*读入一个多项式 （len，总长度；p，从第p位开始读；q返回读取停止的位置是q；s，待读取字符串)\*/

double ReadDight(int len,int p,int\* q,char\* s);/\*读入一整个数字 （len，总长度；p，从第p位开始读；q返回读取停止的位置是q；s，待读取字符串)\*/

void Calc();/\*计算，批量出栈\*/

int OpLevel(char op);/\*给定操作符op，返回其优先级\*/

/\*RPN相关\*/

PN Rpn(char\* str);/\*逆波兰式主函数\*/

struct List\* Change(PN a);/\*结构体转链表\*/

functiondisplay文件的函数有：

void CoordinateAxis(Coordinate OriginalPoint);

通过原点位置画函数

//画坐标轴

void Calculate\_Display(List \*ploynomial,Coordinate OriginalPoint,char judge,int f\_num);

通过以直代区的方法疯狂打点画图。

Char judge判断是不是重画的

//画图像

void printploynomial(List \*ploynomial);

//在console中显示多项式 （测试用）

Calculus\_data Input\_Tip\_Calculus(void);

//console中的提示（测试用）

double Calculus(Calculus\_data Data,List \*ploynomial);

//计算积分 （新版本中会出现）

double price(List \*ploynomial,double x);（计算函数值）

//Derivation\_data Input\_Tip\_Derivation(void);

//console测试用

double Derivation(Derivation\_data Data,List \*ploynomial);

//计算导数 （新版本会有）

void KeyboardEventProcessFunctionMove(int key,int event);

List \*ConsoleGetPloynomial(void);

//从Console中读，测试用。

2）多文件构成机制

（说明分文件构成程序的实现机制，即如何具体采用文件包含、#define保护、外部变量或外部函数）

# 4.部署与运行

* 1. 编译安装运行说明

打开.dev文件，在项目属性-包含文件处设置文件的路径。设置好后，编译生成.exe文件。

.exe文件的菜单栏有Draw，Tool，Help三栏，点击Draw 栏的 New Fuction，弹出输入框，鼠标点击输入。Over是返回上级界面，OK是开始处理，Delete是删除末尾字符。点击OK后，就会弹出绘制好的图像，按上下左右键移动函数图像，按F5返回输入界面。

* 1. 典型测试情况

例子一：（丙同学）

起初，在遇到1+（2）\*（3+1）之类的字符串时，由于（优先级低于\*，\*需要弹出栈，而\*的第二操作数尚未压入操作数栈，就会出现问题。

问题的本质是：（应该无条件入栈，而在优先级系统中，则不能保证这一点。

解决方法：在Rpn函数（即逆波兰式算法函数）中，判断操作符类型处加一个特殊判断：判断是否为（字符，如果是，则无条件入栈。这个判断应该放在）的特判之后，运算操作符的特判之前。这样，问题就得到了解决。

# 5.组内分工

5.1 组内分工情况

甲：汇报演示文稿，MYGUI.h,MYGUI.c

乙：主程序的连接，functiondisplay.c，functiondisplay.h，cipher.c, cipher.h, break\_continue.c, break\_continue.h, MYGUI.c, MYGUI.h

丙：撰写实验报告，standard.h， standard.c

5.2个人实践过程中遇到的难点及解决方案

（分人员，多个点加（1）（3）…的标号）

乙：回调函数掌握不熟，废了些劲。

整合的时候很多细节问题，函数不够优雅。（主要是GUI部分）

算法复杂度比较高，还需要优化。

甲：GUI涉及到很多windows的东西，读不懂很困难，很多时候只能靠照葫芦画瓢。在写界面的过程中，有一个相应选项的功能实现的问题，一开始这个问题我以为在我所能接受的范围之外，比如显示用户输入界面，一开始界面总是只能在点击相关按键的时候才能显示，一放开就会消失，最后我想到了将该显示界面的函数加入到display当中利用判断解决了问题

丙：逆波兰式算法实现过程遇到困难

查阅资料（CSDN），我了解了逆波兰式算法的核心思路，但实现仍有困难。我一边自己分析，一边询问室友，总算大体将其拆分为若干函数，分别予以实现。然而，在运行时频繁遇到错误。因为思路不清，我常常漏掉了许多情况，也常常写错一些逻辑。利用单步调试和室友教我的“输出调试”，我慢慢确定了每个问题的所在，并予以修正解决。最后实现了（不怎么优雅）的整个模块。

# 6.合作纪要

（体现团队合作方面的证据，如小组会议记录等体现良好的沟通能力，如邮件关键内容屏幕截图）

//受匿名限制，截图无法放出。

2019年 5 月6 日：

东一麦斯威。拟定项目类型为几何画板类函数绘图软件。

确定了以下要求：先做多项式，后做其他函数；甲负责界面绘制，乙负责图像绘制，丙负责数据处理。提出了方向性的要求。

2019年5 月 18日：

水墨云峰。三人一同完成了各自部分的初稿。

2019年5 月28 日：

水墨云峰，修复若干bug，开始连接。

# 7.总结

7.1程序亮点或创新之处

利用了栈的数据结构和逆波兰式算法实现了单元多项式有优先级的加减乘法运算。

利用图形库的知识，绘制了函数的图像。

采用伦敦时尚周配色，美观优雅。

用文件系统做到了可以保存已画图像的数据。

有一定（很傻）的软件保护机制。

7.2应用知识点

图形库：回调函数，绘图，菜单，box。

数据结构：结构，链表，数组，栈。

算法：排序，逆波兰式。

文件：多文件组织，写入文件。

软件工程：模块化编程，耦合度，保护定义，

7.3 心得体会

（总结团队在项目过程中的体会和心得，经验和教训，以及团队所有成员的自我评价。每位成员自我评价字数在200-400字之间。）

总体描述

甲：我提出的设想，最后成为了一个大概的现实，蛮高兴的。

还有好多功能没实现，最后答辩的时候会show给你们看的。现在身体感觉被掏空，做项目真不容易（暴风哭泣）

实现的时候感觉很棒。

Dev的项目构成总算是有所理解了，这门课没白学（www…）

为了编代码第一次熬夜，夜里的紫金港好看的。

代码的注释要学1，是明白了为什么劣质代码想骂人。

真的累了，感想就在我答辩时说吧，夏季地的晚风中不缺故事，先晚安了。

乙：做大作业的过程是将理论转为实践的过程，在这个过程中会遇到许许多多在课堂上不会遇到的问题，这些问题的解决需要细心地去探究，有的时候靠个人的力量无法完成，但是当几个人一起

去面对的时候，遇到问题互相讨论有些问题就可以引刃而解，做大作业的过程就是认证团体力量的有力的证明。在这个过程中想象中的程序的功能要想变成现实，必须经过合理的设计，然后是构造相关的函数来实现功能，从而实现由全体到局部，再通过局部功能的实现来实现全体的功能，在这个过程中函数这个小的单元起了很大的作用，当实践与理论完美结合，所收获的是更深入的认识，所收获的是更深层次的体验！

丙：个人认为这次项目效果不是很理想，教训如下。1.开始时各人应尽快写好负责模块的核心，然后连接，尽快做出alpha版，然后在其框架上添加新功能。而不是写好各模块所有功能，再凑成alpha版。2.应当有时间紧迫意识，不能看轻编程的难度，bug往往出现在意想不到的地方，调试又极需要时间。3.就我个人而言，应该加强代码风格的调整锻炼，争取写出美观、可读性高、耦合度低的代码。应培养加注释的习惯，以及限定变量作用域的习惯。学习更多的算法与数据结构知识，让功能实现更加优雅。我认为我在团队中尽到了自己的责任，所实现的部分也有一定难度，然而实现不够优雅，可读性和代码风格都比较差。综合来看，我认为我在这次项目中的表现勉强及格。

# 8、参考文献和资料

（列出参考的书籍、论文、网站的信息和地址等。）