浙江大学实验报告

姓名：赵泓珏　　**学号**：3190104515　　**日期**：2020.4.7　　**成绩**：_____

课程名称：C 程序设计专题　　**指导老师**：翁恺　　**实验名称**：函数绘制

一、　实验题目要求

实现一个 GUI 程序，从标准输入读入一个 f(x)的函数表示，如 2x^2-6x+20。其中幂次用^表示，系数和 x 之间不需要乘号，最高的幂次不限。然后读入 x 的区间的起点和终点，如-1.4 1.4。然后你的程序要计算合理的 y 轴的范围，在窗口中画出这个图形和坐标轴。坐标轴要求画出原点和两条轴线，轴线上标出合理的坐标点。是否画出网格线可以自己决定。

二、　实验思路和过程解释

在整个实验中，最核心的四个步骤为：读入函数，对函数进行处理，绘制坐标轴和网格线，绘制函数图像。

1. 读入函数

在这个步骤中我受到了之前某次作业里实现多项式运算的启发，在这里采用链表对函数中的核心参数进行储存。与此同时也不会限制出入字符串的长短，也能保存尽可能多的信息。

2. 对函数进行处理

这里涉及到的核心处理有：获得函数 f(x)在 x 处的函数值；找出最大最小值；进行坐标变换。具体有关内容见实验代码解释。

3. 绘制坐标轴和网格线

在这个步骤中开始调用 acllib.h，根据输入区间和最值实现网格线和坐标轴的绘制。

4. 绘制函数图像

在这个步骤中调用 acllib.h，设置足够小的精度，采用绘制函数最古老的思想描点画图法实现对函数图像绘制。

三、 实验代码解释

在此次大程序中除去 acllib.c 和 acllib.h 外还涉及到了三个文件：main.c, function.c，function.h。接下来对三个文件中的有关代码进行解释

1) main.c

在 main 中见到的有关函数详见在 function.h 和 function.c 中的有关代码解释。虽然原则上不在实验报告复制粘贴大段代码但为了使阅读方便对代码进行了复制粘贴。还望理解。

i. 有关宏的定义

```
#define WinWidth 1000
#define WinHeight 750
#define epsilon 0.00001
```

WinWidth：定义窗口宽度为 1000 像素；

WinHeight：定义窗口高度为 750 像素；

epsilon：定义遍历函数的最小精度为 0.00001

ii. 具体代码

```
    initConsole();
    printf("this project can help you get the image of power function.
 please follow the guidance.\n\
please make sure that what you enter is right, because this project ca
n't find your wrong.\n\
hope this project can help you :)\n");

    printf("Please enter the function expression(e.g:2x^3-
x^2+x+1): ");
    func* f=getFunction();//get the function expression
```

首先调用 initConsole()以确保能够使用标准库的有关函数。之后采用 printf()打印有关提示内容，使用者输入函数表达式。让为了防止在传给别人时出现乱码在这里采用了英文进行提示。之后采用 getFunction()读入函数表达式。

```
    double left, right;
hr:
    printf("Please enter the interval(e.g: -1.4 1.4):");
    scanf("%lf %lf",&left,&right);

    if(left>=right){
        printf("wrong!");
        goto hr;
    }
```

之后提示使用者输入输入区间左右端点值，同时进行判断，如果出现左端点的值大于等于右端点的值，则告诉使用者出现错误并且使用 goto 函数返回之前提示输入区间的地方。虽然程序中最好不使用 goto 以免破坏程序结构，但是这这个大程序中只使用了一次 goto，并无大碍。

```
    double max=sup(f,left,right);
    double min=inf(f,left,right);

    double x0=changex(0,left,right);
    double y0=changey(0,max,min);
```

之后获得函数的最大最小值，采用 changex()和 changey()函数对（0，0）点进行坐标变换，使其对应到在画布中的坐标。

```
    initWindow("function image",DEFAULT,DEFAULT,WinWidth,WinHeight);

    beginPaint();

    putCell(x0,y0,left,right,max,min);
    putcoordinate(x0,y0,left,right,max,min);
    DrawImage(f,left,right,max,min);

    endPaint();
    return 0;
```

之后创建窗口，命名为“function image”，并根据一开始定义的宏去定义窗口大小。在 beginPaint()和 endPaint()之间分别调用 putCell()，putcoordinate()，DrawImage()实现对网格线、坐标轴、函数图像的绘制。最后以 return 0；结束 main.c 的编写。整个 main.c 的代码行数将近在 25 行左右，也符合了老师上课所讲的原则。

2) function.h

一开始先写入标准头文件结构防止重复引用，考虑到在 main.c 中不涉及链表具体结点的操作，同时减少用户知道的不需要知道的信息，对链表结点 struct _func 进行前向声明（有关内容见 4.1 的课程）。之后写入需要使用的函数，用#endif 结束头文件编写。具体函数功能如下。

func* getFunction();读入函数表达式，获得函数表达式中的核心参数；

double ValueofFunc(func* f, double x);得到函数 f 在 x 处的函数值；

double sup(func* f,double left,double right);得到函数 f 在区间[left, right]上的最大值；

double inf(func* f,double left,double right)；得到函数 f 在区间[left, right]上的最小值；

double changex(double x,double left,double right);将原本坐标系中的横坐标

转化为画布坐标系中的横坐标；

double changey(double y,double max,double min)；将原本坐标系中的纵坐标转化为画布坐标系中的纵坐标；

void putcoordinate(double x0,double y0,double left,double right,double max,double min);绘制坐标系；

void putCell(double x0,double y0,double left,double right,double max,double min);绘制网格；

void DrawImage(func* f,double left,double right,double max,double min);绘制函数图像。

3) function.c

i. 有关宏定义

```
#define WinWidth 1000
#define WinHeight 750
#define epsilon 0.00001
#define delta 20
#define maxn 20
```

WinWidth，WinHeight，epsilon 与在 main.c 中宏的定义一致，此处不再赘述。delta 为一个设定值，之后会用于确定图像位置以及文字放置位置。maxn 在之后用于定义字符串数组大小。

ii. 链表结点声明

```
typedef struct _func{
    struct _func* prev;
    double coef;
    double exp;
    struct _func* next;
} func;
```

分析函数，实际上可以由若干个结点组成，每一个结点里存储幂函数一项

的信息（系数、指数），因此在结点中定义 double 类型的 coef 和 exp，分别用来存储函数中一项的系数和指数，同时添加指针 prev 和 next 进行结点之间的连接构成双向链表。这里采用双向链表没有特殊原因，只是单纯地想练习，之后也没用到，不必过分在意。

iii. func* getFunction()

这个函数意在将函数有关的核心信息读入到一个链表中。在这里采用返回指针的函数，先定义 head 为空指针，完成连接后返回头指针。定义指针 tail 便于之后连接。同时定义两项之间的操作符'+'。

```
func* p=(func*)malloc(sizeof(func));
if(op=='+')p->coef=1.0;
else p->coef=-1.0;
p->exp=1.0;
p->prev=NULL;
p->next=NULL;
```

之后进入 while 循环来获取每项的信息并存入结点。在每次循环里，先采用 malloc 函数给 func 的指针 p 分配内存。考虑到系数和指数为 1 时不需要打出，所以一开始直接令 p->exp 为 1.0，同时根据 op 为'+'还是'-'决定 p->coef 为 1.0 还是-1.0（op 的具体判断见后）。同时令结点中的两个指针同时为 NULL。

```
double value;
int flag;
flag=scanf("%lf",&value);
if(flag==1) p->coef*=value;
```

首先读入系数 coef。考虑到系数为 1 时不会打出并且只有 x 的几次方，在这里利用 scanf 函数的返回值。采用 scanf 读入数值，如果 x 前没有数字即系数为 1，则返回值为 0，因为在之前已经定义好 p->coef 在这里不进行操作；如果读入数字即返回值为 1，则令 p->coef 乘以对应数值。具体操作：定义

double 类型变量 value 和 int 类型变量 flag。令 flag=scanf("%lf",&value)，应用如果 flag==1 则令 p->coef*=value。完成系数的读入。

```
        char ch;
        if((ch=getchar())=='x'){
            if((ch=getchar())=='^'){
                flag=scanf("%lf",&value);
                if(flag==1) p->exp*=value;
                ch=getchar();
            }
        }
        else p->exp=0.0;
```

在读入系数后会有三种情况，在这里以系数为 2 举例。情况 1：2x^3 op；情况 2：2x op；情况 3：2 op。op 为'+'，'-'，或'\n'。具体判断方法如下：定义 char 类型变量 ch，用 getchar()读入 ch 并判断是否为 x。如果不是 x 说明函数表达式中该项为常数，直接令 p->exp=0.0，此时 ch 内存储着 op 的信息。如果是 x，再次用 getchar()读入 ch 并判断是否为^，如果不是则说明该项的指数为 1，不需做任何处理，并且此时 ch 内存储着 op 的信息。如果是，同样利用 scanf 返回值读入 p->exp。之后再令 ch=getchar()使 ch 中存储 op 的信息。此时结点内存储的核心数据全部读入完成。

```
        if(head==NULL)head=tail=p;
        else{
            p->prev=tail;
            tail->next=p;
            tail=p;
```

之后完成链表内结点的连接，具体内容与上次大程作业存在很大的的相似之处，唯一区别在于双向链表的连接，此处不再赘述。

最后令 op=ch，完成信息传递，循环退出的标志就是 op=='\n'。最后返回 head，func* getFunction()的代码编写全部结束。

iv. double ValueofFunc(func* f, double x);

```
    func* p;
    double sum=0.0;

    for(p=f;p;p=p->next){
        sum+=p->coef*pow(x,p->exp);
    }

    return sum;
```

这个函数用于实现函数值的计算。考虑到是幂函数以及采用链表存信息，先定义 func 的指针用于之后遍历，定义 double 类型变量 sum 用来计算最后结果（注意对 sum 赋初值 0.0）。对链表进行遍历，采用 pow()函数计算每项的值并加到 sum 上。最后返回 sum 的值，double ValueofFunc(func* f, double x)的代码编写结束。

v. double sup(func* f,double left,double right);

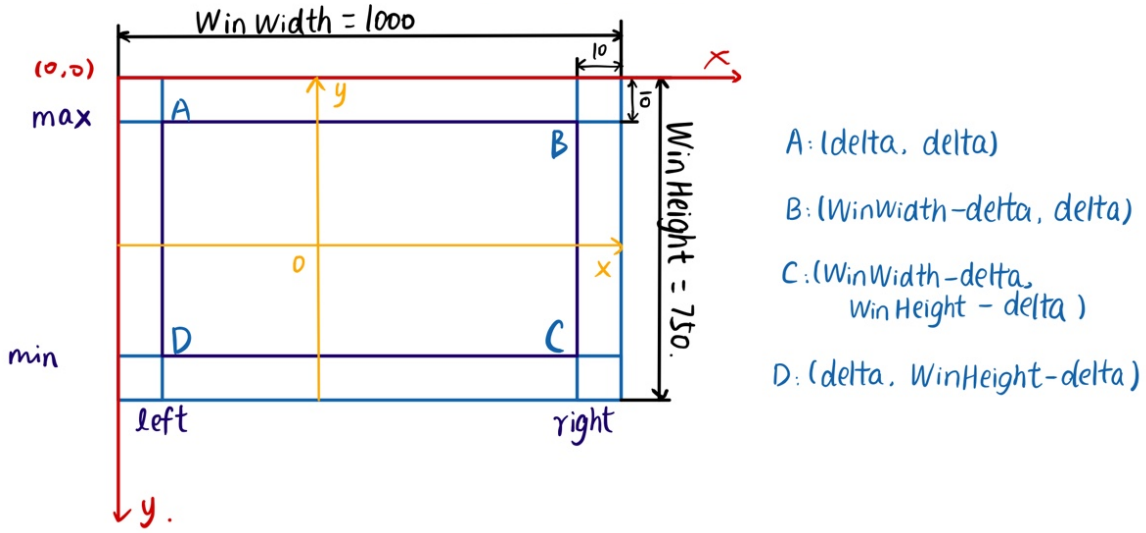
```
double sup(func* f,double left,double right){
    double i, max=ValueofFunc(f,left);
    for(i=left;i<=right;i+=epsilon){
        if(ValueofFunc(f,i)>max)max=ValueofFunc(f,i);
    }
    return max;
}
```

这个函数用于找出函数的最大值（取名为 sup 来自于数学分析里常用 sup 表示上确界）。这个函数与寻找数组中最大值的思路基本一致。根据一开始定义的常量定义 double 类型变量 i 用于遍历定义域区间（从数学角度严格来说无法遍历整个函数区间因为区间为无限集，但是绘制函数图像的核心思路是描点作图，选取足够多的点可以认为对函数区间进行了遍历）。同时先定义 max 为左端点的值。之后利用 for 循环，每次 i 的值加 epsilon，如果当前函数的值大于 max 则令 max 为当前点函数的值，最后返回 max 的值。double sup(func*

f,double left,double right)的代码编写结束。double inf(func* f,double left,double right)的思路以及代码内容基本类似，此处不再赘述。至此函数的基本信息基本全部获得。

vi. double changex(double x,double left,double right);

在解释接下来的代码之前先解释整个画布的布局以及坐标轴的确定以及坐标变换的原理。



首先解释画布布局。为了能够完整显示区间[left, right]和[max, min]以及便于之后的在坐标轴上标注数值，在这里我选在不再整个画布内绘制函数，只在矩形 ABCD 内对函数图像进行绘制。ABCD 每条边距画布边的距离都是 delta。同时 AB 边代表函数最大值所在，CD 边表示函数最小值所在，AD 边表示区间左端点所在，BC 边表示区间右端点所在。

接下来解释坐标变换。在整个画布中有两个坐标系：一个是以画布左上角为坐标原点的坐标系（红色坐标系），一个是需要画出的函数坐标系。（黄色坐标系）。我们可以根据之前获取函数信息的函数轻松得到函数坐标系下某点的坐标，但实际绘图需要红色坐标系下的坐标。因此在这里我们需要进行坐标变换。在线性代数里我们有专门进行坐标变换的公式，在这里我们采取一种相对

而言便于运算的方法。对横坐标变换进行分析。首先假设区间左右端点异号，容易看出实际上两个坐标系中的坐标有线性关系。以横坐标的坐标变换为例，设在红色坐标系中的坐标为 x，橙色坐标系下的坐标为 x'。设二者满足关系：

$$x = Ax' + B$$

则应有：

$$\text{delta} = A \cdot \text{left} + B$$
$$\text{WinWidth} - \text{delta} = A \cdot \text{right} + B$$

于是有：

$$A = \frac{\text{WinWidth} - 2 \cdot \text{delta}}{\text{right} - \text{left}}, B = \text{delta} - A \cdot \text{left}$$

如果左右端点数值同号，为了能够画出两条坐标轴，我们可以进行适当变换。此时函数图像全在 y 轴同一侧，因此如果原来的区间全部在 y 轴的左侧，我们可以令右端点的值变为 0，如果原来的区间全部在 y 轴右侧，我们可以令左端点的值为 0。这样的话我们就可以顺利画出函数在坐标轴一侧的图像。对纵坐标 y 的变换同理，不再赘述。

```
    double x1;

    if(left>=0)left=0;
    if(right<=0)right=0;

    double A=(WinWidth-2*delta)/(right-left);
    double B=delta-A*left;

    x1=A*x+B;

    return x1;
```

vii. void DrawImage(func* f,double left,double right,double max,double min);

```
    double x;
    for(x=left;x<=right;x+=epsilon){
        double x1=changex(x,left,right);
        double y1=changey(ValueofFunc(f,x),max,min);
        putPixel(x1,y1,RED);
    }
```

从这个函数起进入画图部分。因为这个函数相对而言比较好解释先解释此函数。这个函数要求对函数图像进行绘制。同样采用一种类似遍历的方法，定义 double 类型变量 x，用 for 循环近似地对区间内每个点进行遍历，同时对(x, f(x))进行坐标变换。再采用 putPixel()函数绘制出该点。为了突出函数图像将其颜色设置为红色。至此 void DrawImage(func* f,double left,double right,double max,double min)的代码编写结束。

viii. void putcoordinate(double x0,double y0,double left,double right,double max,double min);

这个函数用于绘制坐标轴。坐标轴的绘制核心在于寻找到坐标原点，在 main.c 中已经用 changex()和 changey()实现。接下来只需用 line()函数画出坐标轴（包括箭头）即可。之后利用 paintText()对坐标轴上必要的 x，y，O 进行标注。接下来用 line 函数分别标出左右端点和最大最小值所在的地方。利用 sprintf()函数将浮点数转化为字符串并将左右端点和最大最小值标注到坐标轴上，void putcoordinate(double x0,double y0,double left,double right,double max,double min)的编写结束。

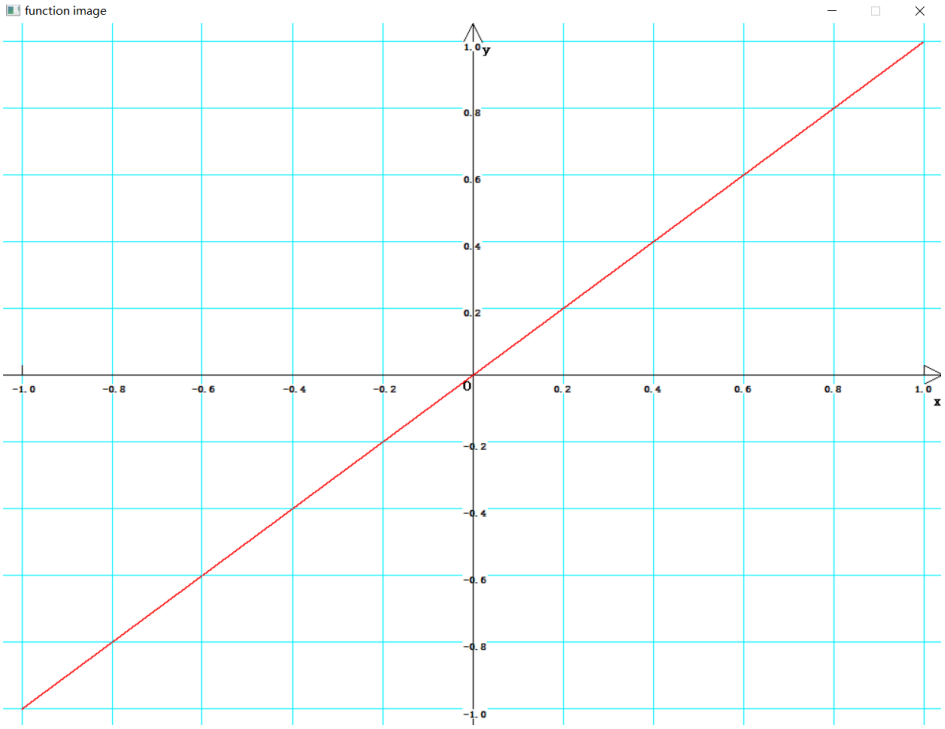
ix. void putCell(double x0,double y0,double left,double right,double max,double min)

这个函数用于画出网格。同样在一开始判断左右端点是否同号，最大最小值是否同号，如果同号做和在 changex()，changey()中同样处理。之后计算合

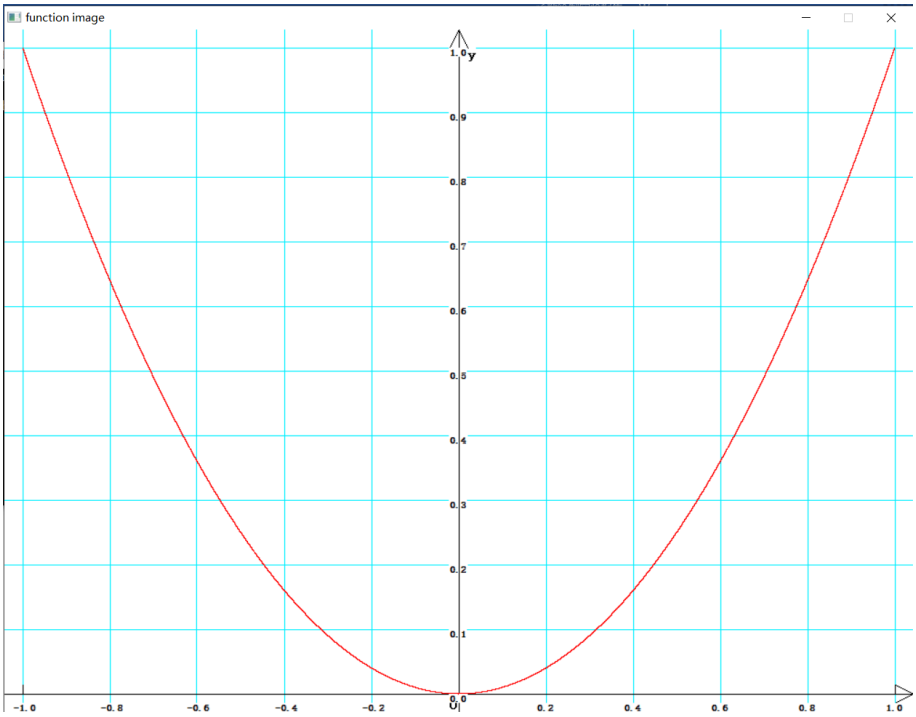
理间隔长度，从坐标原点分别向上下、左右进行画线，并用 paintText()进行标注（同样利用 sprintf 将浮点数转化为字符串类型），至此全部代码编写结束。

四、　样例展示

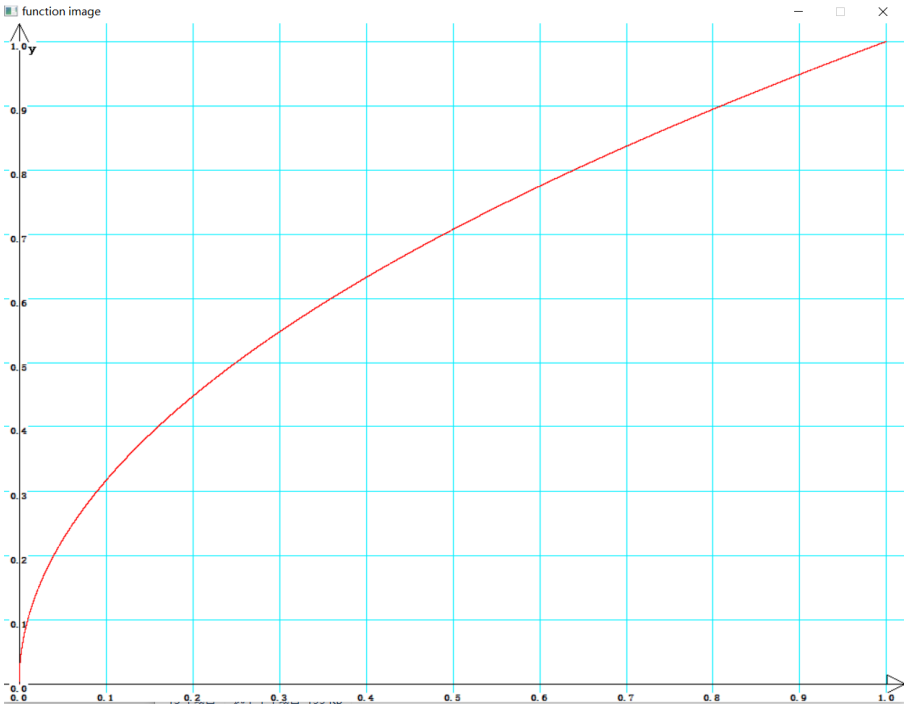
```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x
Please enter the interval(e.g: -1.4 1.4):-1 1
```



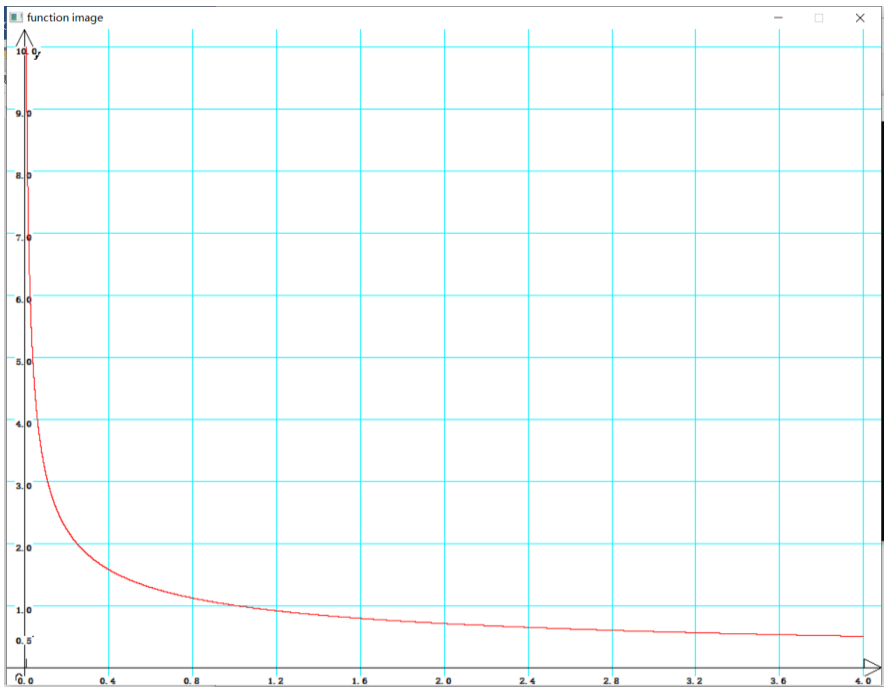

```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x^2
Please enter the interval(e.g: -1.4 1.4):-1 1
```



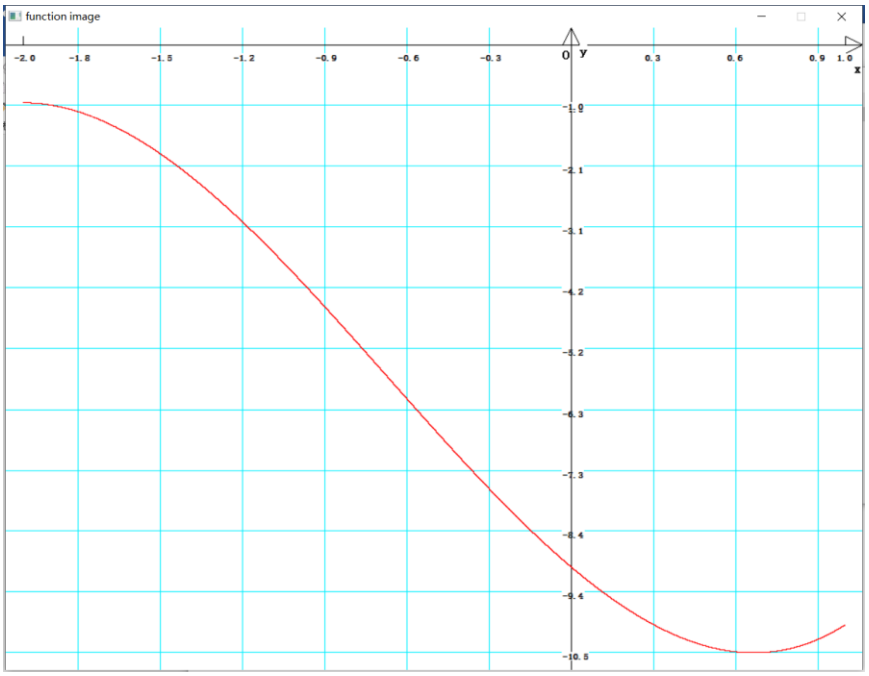

```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x^0.5
Please enter the interval(e.g: -1.4 1.4):0 1
```




```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x^-0.5
Please enter the interval(e.g: -1.4 1.4):0.01 4
```




```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): -x^3+2x^2-4x-9
Please enter the interval(e.g: -1.4 1.4):-2 1
```



五、 实验心得体会

1. 大程序初体验

学 C 程已半年有余，虽说之前也写过一些多文件的程序，但是这是第一次从头到尾全部自己设计，总结原理最后全部实现。在这次大程实验中我也进一步学

习了编程中的有关思想。可以看到我在这次大程序里尽可能地使用了函数，同时使用多文件尽可能减少每个.c 文件的代码行数，也尽可能对有关内容进行封装减少用户知道不需要知道的信息。在我看来这次大程实验是对我之前学到的编程思想很好的实践和练习，进一步增加了我自己写大程序的经验。

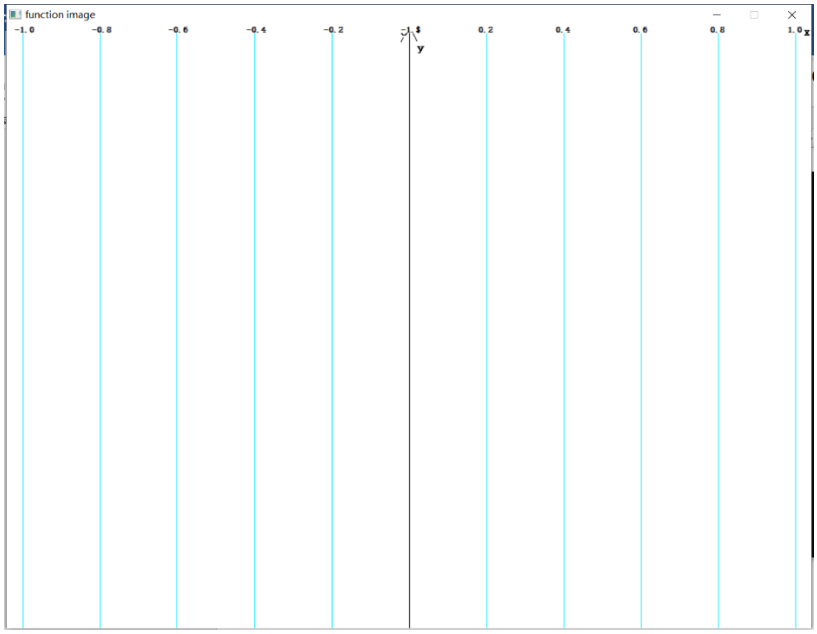
2. 对有关函数绘制的思考

在这个程序中实现了对幂函数图像的绘制。实际上如果能完成幂函数图像的绘制，在某种程度上我们也能完成对其他函数图像的绘制，因为我们可以应用泰勒展开将其他函数展开为幂级数。但是这种情况下会对函数的输入和读取带来困难。根据我平时使用的一些软件的经验，这种情况下最好采用交互式界面——即在交互面板内点击组成函数的元素，这样会让读取更加简单。

3. 此程序的不足之处

在我对程序进行测试时也发现了这个程序的一些不足。首先，该程序不具备查错能力，如给定函数在给定区间上没有定义，则进行报错等等。此时程序运行则会出现极其诡异的结果。如：

```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x^0.5
Please enter the interval(e.g: -1.4 1.4):-1 1
```



其次，绘制函数的区间不能过大，因为幂次较高时，如果区间过大一方面可能造成溢出导致图像无法正常表示，同时由于函数是通过描点作图的方法画出，这种情况下可能会导致运行时间过长而无法响应。

此外，该函数绘制程序还存在绘制出的图像失真的情况。如：

```
this project can help you get the image of power function. please follow the guidance.
please make sure that what you enter is right, because this project can't find your wrong.
hope this project can help you :)
Please enter the function expression(e.g:2x^3-x^2+x+1): x^-0.4
Please enter the interval(e.g: -1.4 1.4):3 4
```

显然该函数图像应为曲线但是在这种情况下却画出近似直线的图像，显然发生了失真。

由于我水平有限，时间精力有限，不能在有限时间内做出更大改进。如果您有好的建议，还望不吝赐教！