浙江大学 实验报告

课程名称：C 程序设计专题　实验类型：大程序作业

实验项目名称：函数绘图

学生姓名：潘致远　专业　计算机科学与技术　学号：3190101093

实验平台：Windows　实验时间：2020.4　指导教师：翁恺

目录

1 实验要求

用户从终端输入一行函数和定义域，程序计算出合理的坐标轴和原点的位置，并作出函数图像。

2 实验过程和思路描述

我处理的是多项式函数，我将程序分为若干个模块。

1. 多项式函数的识别

   这是用户完成输入后程序第一个执行的部分，核心就是字符串的处理，不过我在实际写程序的时候把这个放在了最后，一开始以输入项数、系数、指数的形式实现。

   首先需要通过正负号将大的字符串分解成字串，然后再对每一项进行提取系数指数的操作。关键点在于自然输入的多项式并不都是 ax^b 的形式。比如 $a=0$ 时这一项就是 0，$a=1$ 时系数一般被用户省略，$b=0$ 时 x^b 被省略,$b=1$ 时 ^{b} 被省略。这就需要我们对这些分支进行讨论。

2. 多项式函数的保存

   我使用链表进行保存.

```
typedef struct _polyNode{
    double coef;
    double exp;
    struct _polyNode *next;
}polynomial;
```

   与此前需要输出多项式的题目不同，这里我们只关心函数值，因此不需要考虑对各项进行排序。

3. 坐标轴和原点的确定

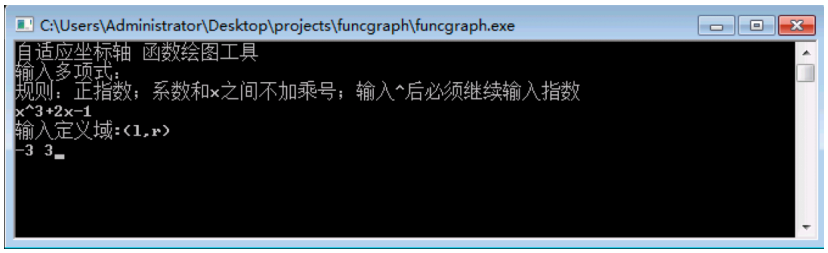
   我们要根据用户输入的函数和定义域判断函数图像所在的象限，然后根据不同的情况确定如何绘制坐标轴（实际上就是确定原点的坐标）。我为了让程序所画的函数图像都是“占整个窗口的大小”的，显得比较美观而且能呈现函数的细节，要做到：两坐标轴的位置会根据定义域、值域而移动，而不是标准的对称的

“十字架形”；x, y 轴不一定是等比例的，而是根据实际情况进行缩放；实际原点 $(0, 0)$ 和函数图像相距较远时，平移坐标系。

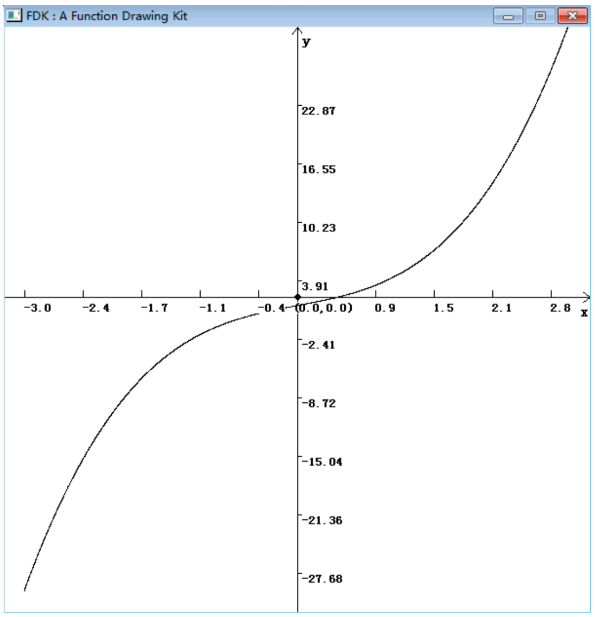
4. 图像、坐标轴刻度的绘制

   核心在于从点的实际坐标到窗口坐标（像素）的变换。根据自己设定的窗口大小，可以通过比例关系建立映射。然后设定间隔，通过绘图函数（描点）连线即可。

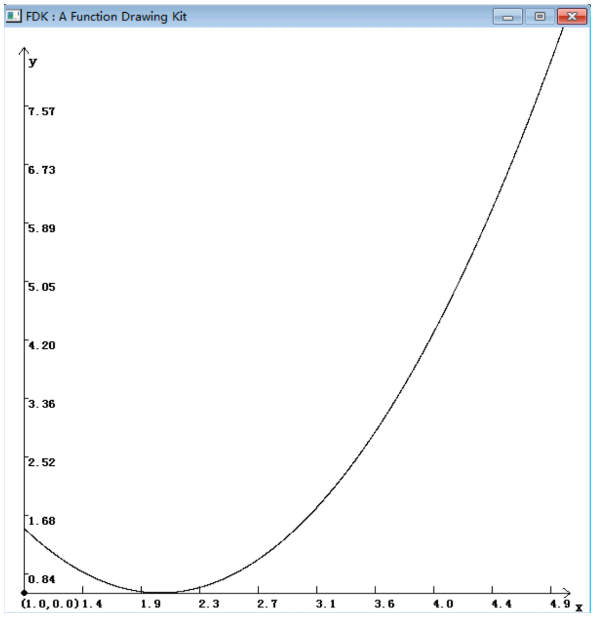
3 实验代码解释



(a) 终端输入



(b) $x^3 + 2x - 1$



(c) $x^2 - 4x + 4$

图 1: 效果呈现

我的函数写在`polynomial.c`中，函数原型和结构体声明在对应的`.h`中，下面选取一些重要函数进行解释。

```
/* 用到的结构体和函数 */
typedef struct polyNode{
    double coef;
    double exp;
    struct polyNode *next;
}polyFunc;
typedef struct{
    double xl;
    double xr;
}funcDomain;
typedef struct{
    double max;
    double min;
}funcBoundary;
typedef struct{
    int x;
    int y;
    double xval;
    double yval;
}pixel;

polyFunc inputPolyFunc(void);
void drawPolyFunc(polyFunc *p,funcDomain *D,int windowSize);
funcDomain* getDomain(funcDomain *p); //获取函数的定义域
funcBoundary* getBoundary(polyFunc *p,funcDomain *D); //返回函数的值域
double calcValue(polyFunc *p,double x);//返回在 x 处的函数值
pixel* setAxis(funcDomain *D,funcBoundary *B,int windowSize);
void setPoint(double value,double x,pixel *zeroPoint,funcDomain *D,\
    funcBoundary *B,int windowSize);
```

函数inputPolyFunc：

```
/* 读入多项式 */
polyFunc inputPolyFunc(void){
    polyFunc *head=NULL,*tail=NULL;
```

```
4       //用正号和负号作为分割字符
5       const char delim[2]="+-";
6       char a[1000],*b[1000]={"\0"};
7       scanf("%s",a);
8       int pos=0,l=strlen(a);
9       /* 对负号做特殊处理 */
10      for(int i=0;i<l;i++)
11          if(a[i]=='-'){
12              for(int j=l;j>i;j--)
13                  a[j+1]=a[j];
14              a[i+1]='m';
15          }
16      /* 以下使用 strtok 函数分割字符串并对子串进行处理 */
17      b[0]=strtok(a,delim);
18      while(b[pos]!=NULL){
19          int i=0,flag=1,cpos=0,epos=0;
20          char scoef[50]={'\0'},sexp[50]={'\0'};
21          //以下是对各种情况的讨论，具体代码可见源文件
22          if(b[pos][i]=='m'){ //系数为负数
23          }
24          if(b[pos][i]=='x'){ //系数为 1
25          }
26          while(b[pos][i]!='x' &&  b[pos][i]!='\0'){
27              //向后移动到 x 所在位置或者末尾处
28          }
29          if( b[pos][i]=='\0'){   //指数为 0
30          }else if( b[pos][i+1]=='\0'){   //指数为 1
31          }else{
32              while( b[pos][i]!='\0'){
33              }
34          }
35          //使用 atof 函数将字符串转化为 double 类型
36          //之后对链表执行插入操作，代码略
37          b[++pos]=strtok(NULL,delim);
38      }
```

```
39      return *head;
40  }
```

函数setAxis:

```
1   pixel* setAxis(funcDomain *D,funcBoundary *B,int windowSize){
2       setTextSize(16);
3       pixel* p=(pixel*)malloc(sizeof(pixel));
4       double xPos=0.0,yPos=0.0;
5       switch(sgn(B->max*B->min)){
6           case -1:
7           //画 x 轴
8           //函数值有正有负
9               //画 y 轴
10              switch(sgn(D->xl*D->xr)){
11                  case -1:
12                      //x 值有正有负
13                      break;
14                  case 1:
15                      if(D->xl>=0){
16                          //x ≥ 0
17                      }else if(D->xr<=0){
18                          //x ≤ 0
19                      }
20                      break;
21              }
22              break;
23          default:
24          //函数值全大于等于 0 或全小于等于 0
25              //画 x 轴
26              if(B->min>=0){
27                  //y ≥ 0
28              }else if(B->max<=0){
29                  //y ≤ 0
30              }
31              //画 y 轴
```

```
32                  //与上面一致，代码略
33                  break;
34          }
35          /* 描原点，标原点坐标 */
36          char cord[100];
37          sprintf(cord,"(%.1f,%.1f)",p->xval,p->yval);
38          beginPaint();
39          setTextSize(12);
40          //根据象限位置改变标签显示位置
41          if(!flag%2){
42              paintText(p->x-40,p->y-5,cord);
43          }else{
44              paintText(p->x-5,p->y+5,cord);
45          }
46          setBrushColor(BLACK);
47          ellipse(p->x-3,p->y+3,p->x+4,p->y-3);
48          endPaint();
49          return p;
50      }
```

其他函数根据思路可以较容易地写出，因此不再赘述，可以查看polynomial.c文件。

4 实验体会和心得

首先我体会到版本控制的重要性。每次做好一个小功能，验证通过后就应该 commit，这样的话以后如果误操作或者产生了问题，也可以查看此前保存的结点。否则，一次性写了一大堆代码，然后发现有问题，要想找到错误之处就会比较累，而如果选择撤销重写的话，更是浪费时间。

就这个程序而言，处理用户输入的字符串和绘制坐标轴都需要非常谨慎，要考虑存在的各种情况，这样才能保证程序的正确性。