测试文件以及测试**demo**使用指南

demo均放在0exe内，demo主文件均放在test内

1.实现牛顿迭代法

- 实现迭代公式为多项式的牛顿迭代

  demo主文件test_ndd.c，demo为test_ndd.exe

  根据提示输入多项式

  然后输入迭代初值和精度w1,w2

```c
#include "ndd.h"

//通过输入的函数func,和初始值st,以函数果误差限w1，和导数误差限w2，使用牛顿迭代法求func=0方程的解
//通过ifok来记录是否成功
double ndd_count(Func func,double st,double w1,double w2,int* ifok){
    do{
        if(fabs(ndd_dao(func,st))<w2){
            *ifok=0;
            break;
        }
        if(fabs(ndd_yuan(func,st)<w1)){
            *ifok=1;
            break;
        }
        st=st-ndd_yuan(func,st)/ndd_dao(func,st);
    }while(1);
    return st;
}

//获得func在x处的函数值
double ndd_yuan(Func func ,double x){
    return func_get_value(func,x);
}

//获得func导数在x处的函数值
double ndd_dao(Func func,double x){
    Func fdao=func_dao(func);
    return func_get_value(fdao,x);
}
```

```c
> C test_ndd.c > main()

    do{
        printf("输入用来牛顿迭代的多项式型迭代式的最高次:");
        int max;
        scanf("%d",&max);
        double coe[max+1];
        for(int i=0;i<max+1;i++){
            printf("%d次项系数为",i);
            scanf("%lf",&coe[i]);
        }
        Func func=func_get(coe,max+1);
        char x='x';
        printf("使用牛顿迭代法求方程");
        func_show(func,x);
        printf("=0 的解\n");
        double w1=0.0001;
        double w2=0.00000001;
        printf("\n输入迭代初值:");
        double st;
        scanf("%lf",&st);
        printf("\n输入函数限差w1:");
        scanf("%lf",&w1);
        printf("\n输入导数限差w2:");
        scanf("%lf",&w2);
        printf("\n函数值限差为:%.10f导数值限差为:%.10f迭代初值为:%.3f\n",w1,w2,st);
        int ifok;
        double v=ndd_count(func,st,w1,w2,&ifok);
        if(ifok==0){
            printf("\n计算失败\n");

        }else{
            printf("\n计算成功\n");
            printf("结果:%.4Lf",v);
            printf("输入e退出,输入其他任意键继续");
            char choice=getch();
            if(choice=='e'){
                break;
            }
        }
    }while(1);
    getch();
    return 0;
}
```

- 实现迭代公式为任意可导函数式的牛顿迭代

  demo为test_ndd_b.exe,demo主文件为test_ndd_b.c

  输入函数和导函数：在test_ndd_b.c源文件的y_yuan函数中输入符合c语言语法的函数，

  在y_dao函数中输入导函数

  然后输入迭代初值和精度w1,w2

```c
int main(){
    do{
        double w1=0.0001;
        double w2=0.00000001;
        printf("\n输入迭代初值:");
        double st;
        scanf("%lf",&st);
        printf("\n输入函数限差w1:");
        scanf("%lf",&w1);
        printf("\n输入导数限差w2:");
        scanf("%lf",&w2);
        printf("\n函数值限差为:%.10f导数值限差为：%.10f迭代初值为:%.3f\n",w1,w2,st);
        int ifok;
        double v=ndd_count_b(st,w1,w2,&ifok);
        if(ifok==0){
            printf("\n计算失败\n");

        }else{
            printf("\n计算成功\n");
            printf("结果：%.4Lf",v);
            printf("输入e退出,输入其他任意键继续");
            char choice=getch();
            if(choice=='e'){
                break;
            }
        }
    }while(1);
    getch();
    return 0;
}
```

```c
//根据输入的自变量x的值返回原函数的值
double f_yuan(double x){
    return sin(x)*pow(x,3)-log(x+1)/log(10)+pow(x,2)-x;
}

//根据输入的自变量x的值求原函数导数的值
double f_dao(double x){
    return sin(x)*3*pow(x,2)+cos(x)*pow(x,3)-1.0/(x+1)/log(10)+2*x-1;
}

double ndd_count_b(double st,double w1,double w2,int* ifok){
    do{
        if(fabs(f_dao(st))<w2){
            //牛顿迭代计算失败
            *ifok=0;
            break;
        }
        if(fabs(f_yuan(st))<w1){
            //牛顿迭代计算成功
            *ifok=1;
            break;
        }
        st=st-f_yuan(st)/f_dao(st);
    }while(1);
    return st;
}
```

2.实现拉格朗日插值法

demo代码主文件，test_lglrc.c

根据提示输入用来插值的原函数的一系列坐标点。

然后可以返回插值结果，并且可以新增点

注意！为了美观，实际上打印显示多项式时，系数只打印两位小数

```c
#include "..\all.h"

int main(){
    do{
        int n;
        printf("\n输入插值的点的个数为: ");
        scanf("%d",&n);
        Pos points[n];
        printf("以x,y的形式输入%d个点为:\n",n);
        for(int i=0;i<n;i++){
            printf("输入点[%d]:",i);
            double x;
            double y;
            scanf("%lf,%lf",&x,&y);
            points[i]=pos_get(x,y);
        }
        printf("\n生成的拉格朗日插值公式为: ");
        Lglrc lglrc=lglrc_get(points,n);
        char x='x';
        lglrc_show(lglrc,x);
        do{
            printf("\n是否加入新的点,生成新的拉格朗日插值公式");
            printf("\n输入e退出,其他键加入新的点");
            char c=getch();
            if(c=='e'){
                break;
            }else{
                printf("\n输入新的点(输入格式如上):");
                double tx;
                double ty;
                scanf("%lf,%lf",&tx,&ty);
                Pos add=pos_get(tx,ty);
                lglrc=lglrc_insert(lglrc,&add,1);
                printf("新的拉格朗日插值函数为: ");
                lglrc_show(lglrc,x);
            }
        }while(1);
        printf("\n输入e退出,其他键继续");
        char c=getch();
        if(c=='e'){
            break;
        }
    }while(1);
```

```c
#include "lglrc.h"


//获得拉格朗日插值多项式

//通过输入的点的数组得到一个拉格朗日插值多项式
Lglrc lglrc_get(Posp points,int n){
    //进行边界判断
    if(n>=LGLR_MAX_N){
        printf("输入的点的个数超过限制,请进入lglrc.h文件中修改");
    }
    Lglrc out;
    out.n=n;
    for(int i=0;i<out.n;i++){
        out.points[i]=points[i];
    }
    for(int i=0;i<out.n;i++){
        double a=out.points[i].y;
        Mult add=mult_one();
        for(int j=0;j<out.n;j++){
            if(i==j){
                continue;
            }
            a/=(out.points[i].x-out.points[j].x);
            double coe_t[2];
            coe_t[0]=-out.points[j].x;
            coe_t[1]=1;
            Mult term=mult_get(coe_t,2);
            add=mult_mult(add,term);
        }
        out.mults[i]=mult_num_mult(add,a);
    }
    return out;
}
```

```
    //拉格朗日插值多项式操作


    //往拉格朗日插值多项式中多插一系列点获得新的拉格朗日多项式
    Lglrc lglrc_insert(Lglrc lglrc,Posp adds,int numOfAdds){
        if(lglrc.n+numOfAdds>LGLR_MAX_N){
            printf("输入的点的个数超过限制,请进入lglrc.h文件中修改");
        }
        //增加新的点列
        //以及为更新旧的项准备的参数
        Mult times=mult_one();
        for(int i=0;i<numOfAdds;i++){
            lglrc.points[lglrc.n+i]=adds[i];
            double ta[2];
            ta[0]=-adds[i].x;
            ta[1]=1;
            Mult term=mult_get(ta,2);
            times=mult_mult(times,term);
        }
        //修改旧的项
        for(int i=0;i<lglrc.n;i++){
            Mult term=mult_num_mult(times,1.0/mult_get_value(times,lglrc.points[i].x));
            lglrc.mults[i]=mult_mult(lglrc.mults[i],term);
        }
        //增加新的项
        for(int i=lglrc.n;i<lglrc.n+numOfAdds;i++){
            double a=lglrc.points[i].y;
            Mult add=mult_one();
            for(int j=0;j<lglrc.n+numOfAdds;j++){
                if(i==j){
                    continue;
                }
                a/=(lglrc.points[i].x-lglrc.points[j].x);
                double coe_t[2];
                coe_t[0]=-lglrc.points[j].x;
                coe_t[1]=1;
                Mult term=mult_get(coe_t,2);
                add=mult_mult(add,term);
            }
            lglrc.mults[i]=mult_num_mult(add,a);
        }
        //更新长度
        lglrc.n+=numOfAdds;
        return lglrc;
    }

```

```
//获取拉格朗日多项式信息

//打印拉格朗日多项式
void lglrc_show(Lglrc lglrc,char x){
    Mult show=mult_zero();
    for(int i=0;i<lglrc.n;i++){
        show=mult_add(show,lglrc.mults[i]);
    }
    mult_show(show,x);
}

//根据自变量x的值，获得拉格朗日多项式的函数值
double lglrc_getvalue(Lglrc lglrc,double x){
    double out=0;
    for(int i=0;i<lglrc.n;i++){
        out+=mult_get_value(lglrc.mults[i],x);
    }
    return out;
}
```

具体多项式的实现可以看文件mult.c

3.实现龙贝格积分法

demo：test_lbg.exe.文件：test_lbg.c

在test_lbg.c里面的f_yuan函数里面修改用来求积分的原函数。

然后用来运行。

默认是使用x^2+x^3当做原函数来使用的。

```c
int main(){
    do{
        printf("输入求积分的范围: ");
        double a,b;
        printf("(a应该小于b)a,b=");
        scanf("%lf,%lf",&a,&b);
        printf("输入求积分的精度");
        double w;
        scanf("%lf",&w);
        double c=lbg_count(a,b,w);
        printf("\n积分结果为: %.6Lf\n",c);
        printf("输入e退出,其他键继续");
        char ord=getch();
        if(ord=='e'){
            break;
        }else{
            system("cls");
        }
    }while(1);
    return 0;
}
```

```c
double f_yuan(double x);

//输入w为精度,a,b为积分区域，a应该小于b
double lbg_count(double a,double b,double w);

//实现梯形法的递推
double tui(double xs,double h,double a,double b);


int main(){
    do{
        printf("输入求积分的范围：");
        double a,b;
        printf("(a应该小于b)a,b=");
        scanf("%lf,%lf",&a,&b);
        printf("输入求积分的精度");
        double w;
        scanf("%lf",&w);
        double c=lbg_count(a,b,w);
        printf("\n积分结果为：%.6Lf\n",c);
        printf("输入e退出，其他键继续");
        char ord=getch();
        if(ord=='e'){
            break;
        }else{
            system("cls");
        }
    }while(1);
    return 0;
}
```

```c
double f_yuan(double x){
    return x;
    // return pow(x,2)+pow(x,3);
}

//输入w为精度,a,b为积分区域，a应该小于b
double lbg_count(double a,double b,double w){
    double h=(b-a)/2;
    int n=3;
    double jieguo[170][170];
    jieguo[0][0]=h/4*(f_yuan(a)+2*f_yuan(a+h)+f_yuan(b));
    printf("\nt数表如下:\n");
    printf("%.6f\n",jieguo[0][0]);
    for(int i=1;i<170;i++){
        jieguo[i][0]=tui(jieguo[i-1][0],h,a,b);
        printf("%.2Lf\t",jieguo[i][0]);
        h/=2;
        for(int j=1;j<=j;j++){
            double term=pow(4,j);
            jieguo[i][j]=term/(term-1)*jieguo[i][j-1]-1/(term-1)*jieguo[i-1][j-1];
            printf("%.2Lf\t",jieguo[i][j]);
        }
        printf("\n");
        //判断是否满足精度
        if(fabs(jieguo[i][i]-jieguo[i-1][i-1])<w){
            return jieguo[i][i];
        }
    }
    return 0;
}

double tui(double xs,double h,double a,double b){
    double th=h/2;
    double out=xs/2;
    for(int i=0;a+th+i*h<b;i++){
        out+=th*f_yuan(a+th+h*i);
    }
    return out;
}
```

4.实现四阶龙格贝塔方法

具体实现的文件结构

1.pos模块

表示二维坐标点的结构

2.item表示单项的结构

记录基本初等函数(没有一个是和式）的乘积。

有效的初等函数类型有，指数函数，幂函数,三角函数等

这个结构存储基本初等函数的乘积。

3.mult

表示一元幂函数多项式的结构

能够进行多项式的加减，乘除运算

4.lglrc拉格朗日插值多项式

能够根据输入的点的序列，得到拉格朗日多项式结果

5.ndc牛顿插值多项式

6.func模拟函数多项式

实现item的单项的加减组合。

也就是func用来来存储多个item单项组合成的多项式

func只能是简单基本初等函数的乘积通过加减组合起来的形式。

当前实现到：

- 过渡实现一：
  用自变量幂函数组合的多项式代替func

7.ndd进行牛顿迭代法计算