西安交通大学实验报告

课程名称：算法设计与问题求解 实验名称：牛顿插值、二分法、牛顿迭代

学 院：机械工程学院 实验日期 2020 年 9 月 26 日

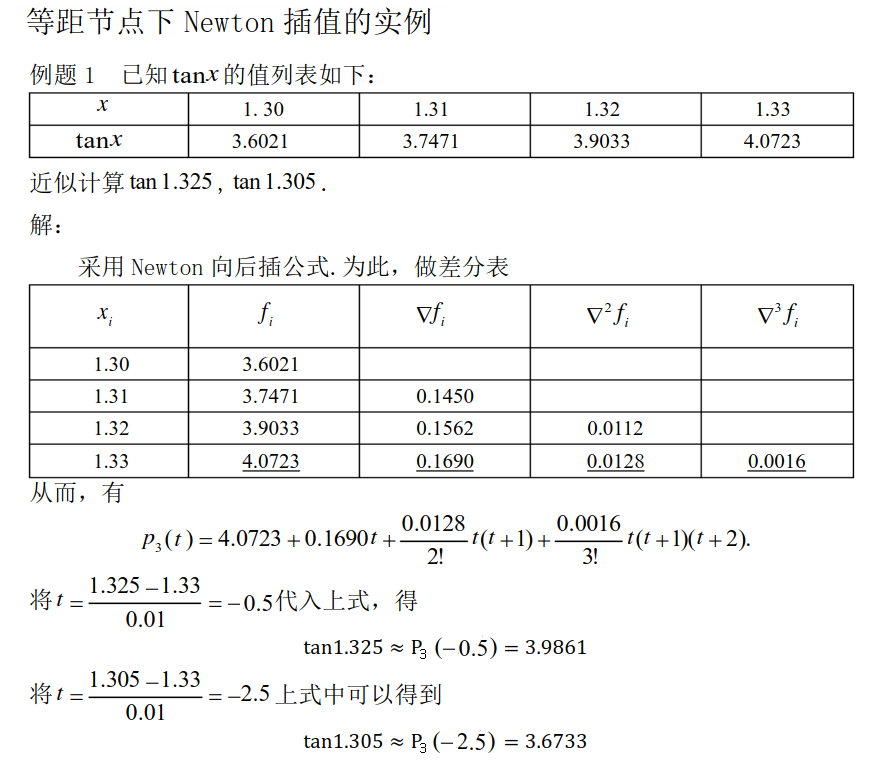
班 级：机械97班 姓 名：杨逢诜 学号：2193712613

#### 一、实验内容和结果

* **题目1**

编程实现牛顿插值法，并用后面的例子中的数据验证。

已知为插值点，求牛顿插值法在*xx*位置的近似值。



【源程序】

#include<stdio.h>

int main() //主函数开始

{

float x[4],y[4][4],x0,y0=0; //声明数组，用于存放输入的点的xy坐标以及全部差商；声明变量，用于存储待计算的点的x坐标和计算出来的y坐标

int i,j; //声明辅助计数变量

for(i=0;i<4;i++) //按照提示输入四个点的坐标

{

printf("Please input the points' x lable and y lable:");

scanf("%f %f",&x[i],&y[i][0]);

}

for(i=1;i<=3;i++) //按照差分计算原则计算三阶以内的全部差分

{

for(j=i;j<=3;j++)

{

y[j][i]=(y[j][i-1]-y[j-1][i-1]);

}

}

printf("PLEASE INPUT THE NUMBER WHICH WOULD BE CALCULATED:"); //输入待计算的点的x坐标

scanf("%f",&x0);

y0=y[0][0]; //按照牛顿插值原则，根据已经计算出的差分计算y坐标

for(i=1;i<=3;i++)

{

float t=(x0-x[0])/(x[1]-x[0]),x1=1;

for(j=1;j<=i;j++)

{

x1=x1\*(t-j+1)/j;

}

y0=y0+y[i][i]\*x1;

//这一句是用来检验程序运行的指标：printf("这一阶差商数值为%f\n",y[i][i]);

}

printf("The approximation of the tan%f is:%f",x0,y0); //输出以牛顿插值法计算出的点的坐标，主函数结束

return 0;

}

这下面还有一个相对通用的版本的程序：

#include<stdio.h>

double DifferenceQuotient(double \*x,double \*y,int k) //声明函数：用于计算差商

{

double df=0; //声明变量：差商值，计数辅助变量

int i,j;

for(i=0;i<=k;i++) //计算k阶差商的和式总共有k+1项，因此循环需要执行k+1次

{

double dfj=1.0; //声明并计算和式中特定项的分母：注意累乘过程中需要剔除“自己减自己”的情形

for(j=0;j<=k;j++)

{

if(i!=j)

{

dfj=dfj\*(x[i]-x[j]);

}

}

df=df+y[i]/dfj; //计算差商的特定一项，并将其加到差商值中

}

return df; //返回差商结果

}

double NewtonDifference(double \*x,double \*y,double xx, int n)//声明函数：用于计算特定点处的牛顿插值结果

{

double result=y[0]; //声明变量，用于储存运算的中间结果，并预先储存插值起点的函数值

int i; //这是辅助计数变量

double diff=1.0; //声明与对应差商相乘的因子

for(i=1;i<n;i++) //含有差商的项总共有n-1项，因此总共需要有n-1项相加

{

diff=diff\*(xx-x[i-1]); //计算与对应差商相乘的因子

result=result+DifferenceQuotient(x,y,i)\*diff; //将因子与差商相乘并加到最终的结果中

}

return result; //返回插值结果

}

int main() //主函数开始

{

int n; //声明并输入变量，这一变量用于存储插值过程中涉及到的点的个数

printf("Please tell us the number of the dots:");

scanf("%d",&n);

double x[n],y[n]; //声明数组，用于存储插值涉及的点的坐标

int i; //声明辅助计数变量

for(i=0;i<n;i++) //按照提示输入n个点的坐标

{

printf("Please input the points' x lable and y lable:");

scanf("%lf %lf",&x[i],&y[i]);

}

double x0; //声明并输入变量，这一变量用于存储待计算的点的横坐标

printf("Please input the point's x lable which would be calculated:");

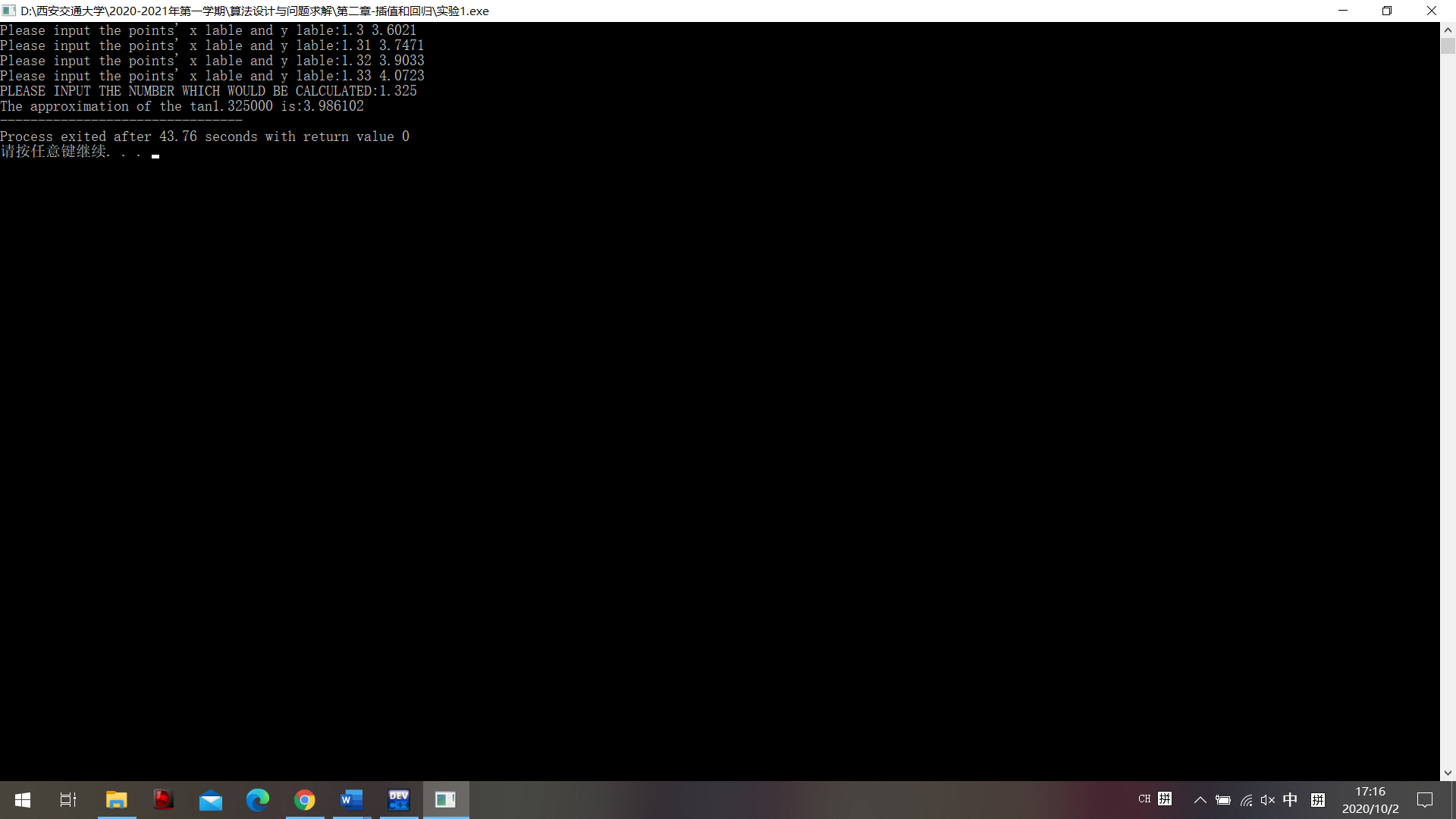
scanf("%lf",&x0);

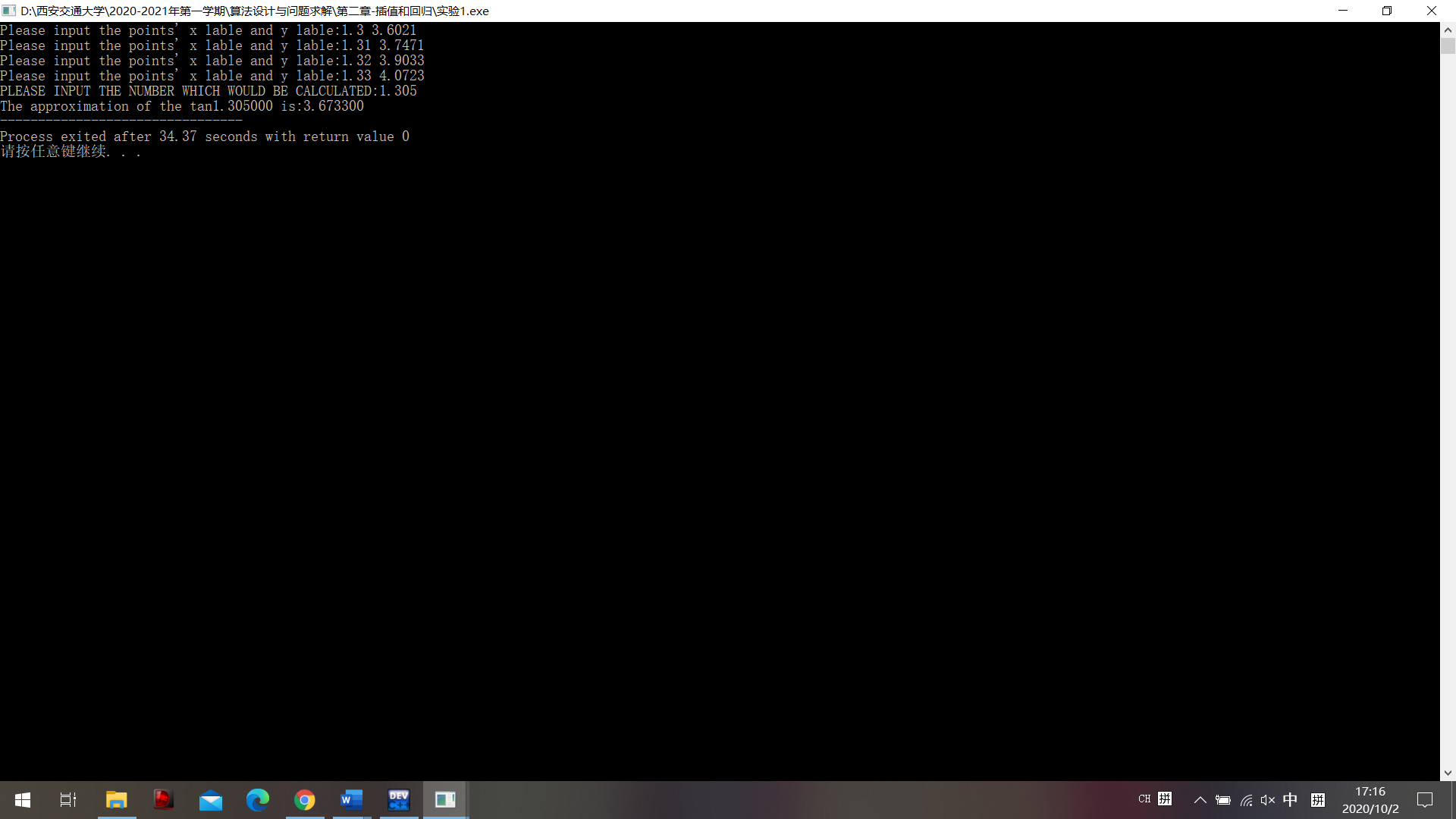
printf("The result is:%f",NewtonDifference(x,y,x0,n)); //计算该点经由插值原理算出的纵坐标，主函数结束。

return 0;

}

【运行结果】





* **题目2**

用二分法求解5sin(x)-x/2-1=0

【源程序】

#include<stdio.h>

#include<math.h>

float Function(float x) //声明函数，这一函数用于计算方程左侧表达式的真值

{

//（程序浮标，用于检测程序的运行情况：printf("f(%f)=%f\n",x,5\*sin(x)-x/2-1); ）

return 5\*sin(x)-x/2-1;

}

int main() //主函数开始

{

float fa,fb,mid,epsilon,solution=0; //声明变量：区间两端，区间终点，求根精度，解的结论（我们给解赋了一个初值0，它并不是一个实际的可行解，但是这个0以后会用到）

printf("Please give us the left and right range of the root solving:"); //按照提示输入区间端点；

scanf("%f %f",&fa,&fb);

if(fa>fb) //若输入的区间端点并不是左小右大，程序会自动给他调个个，以防后面出错

{

int k=fb;

fb=fa;

fa=k;

}

if(Function(fa)\*Function(fb)>0) //若方程左侧表达式在区间端点处的真值同号，则未必能解除数值解，此时程序自动终止；

{

printf("ERROR RANGE,PROGRAM ENDING");

}

else if((Function(fa)==0)||(Function(fb)==0)) //若方程左侧表达式在区间某个端点处取到了0，则将那个区间端点的位置赋给解变量

{

solution=Function(fa)==0?fa:fb; //采用了相对简洁的问号表达式；

}

else

{

printf("Please give us accuracy of the root solving:"); //在一般情况下，则需要先按照提示输入精度要求；

scanf("%f",&epsilon);

do{

mid=(fa+fb)/2; //计算区间中点处函数表达式真值；

if(Function(mid)\*Function(fa)<0) //如果区间中点处和区间左端点处函数表达式值异号，则将区间中点值赋给区间右端点变量；

{

fb=mid;

}

else if(Function(mid)==0) //如果函数表达式恰好在中点处取到0，循环停止，并将中点位置赋给解变量；

{

break;

}

else //如果区间中点处和区间右端点处函数表达式值异号，则将区间中点值赋给区间左端点变量；

{

fa=mid;

}

//（程序浮标，用于检测程序的运行情况：printf("%f\n",fabs(fa-mid));）

}while(fabs(fa-fb)>=epsilon); //在区间长度小于精度要求时，循环终止，并将区间中点赋给解变量

solution=mid;

}

if(solution!=0) //只要方程区间内有解（也就是说，解变量不是初始的0），那么就输出这个解；

{

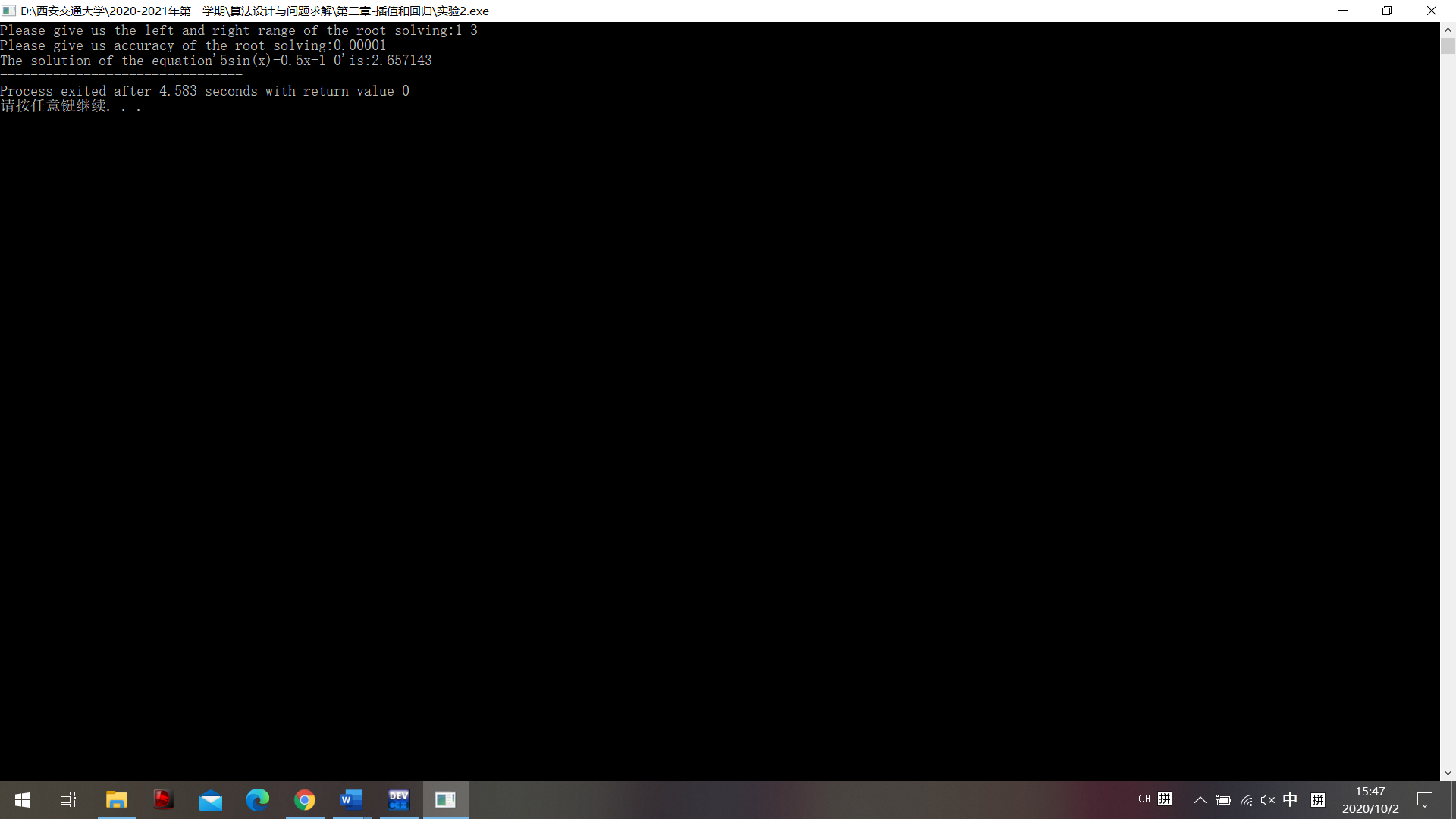
printf("The solution of the equation'5sin(x)-0.5x-1=0'is:%f",solution);

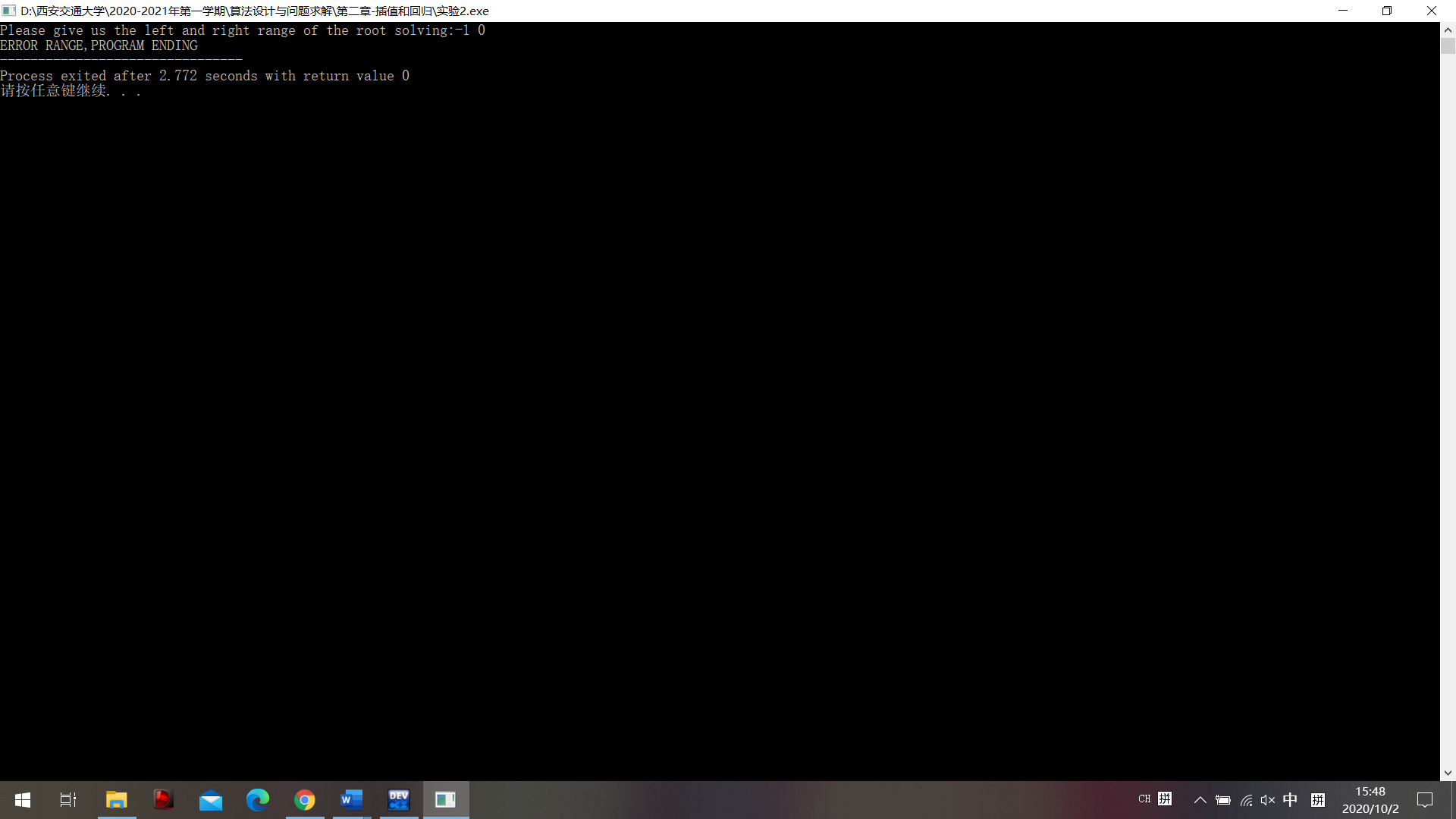
}

return 0; //主函数结束

}

【运行结果】





* **题目3**

用牛顿迭代法求e*x* + *x* = 0等于0.5附近的根。

【源程序】

#include<stdio.h>

#include<math.h>

float Function(float x) //声明函数，用于计算给定方程左端表达式的值

{

return exp(x)+x;

}

float Derivative(float x) //声明函数，用于计算给定方程左端表达式的导数值；

{

return exp(x)+1;

}

int main() //主函数开始；

{

float x0=0.5,x,epsilon; //声明变量：起始迭代位置，迭代结果，精度值

printf("Please give us the accuracy of the solution:"); //按照提示输入解的精度值

scanf("%f",&epsilon);

do

{

x=x0; //将上一次的迭代结果赋给迭代起始位置变量；

x0=x0-Function(x)/Derivative(x); //计算迭代起始位置处曲线切线与数轴的交点，作为迭代的结果；

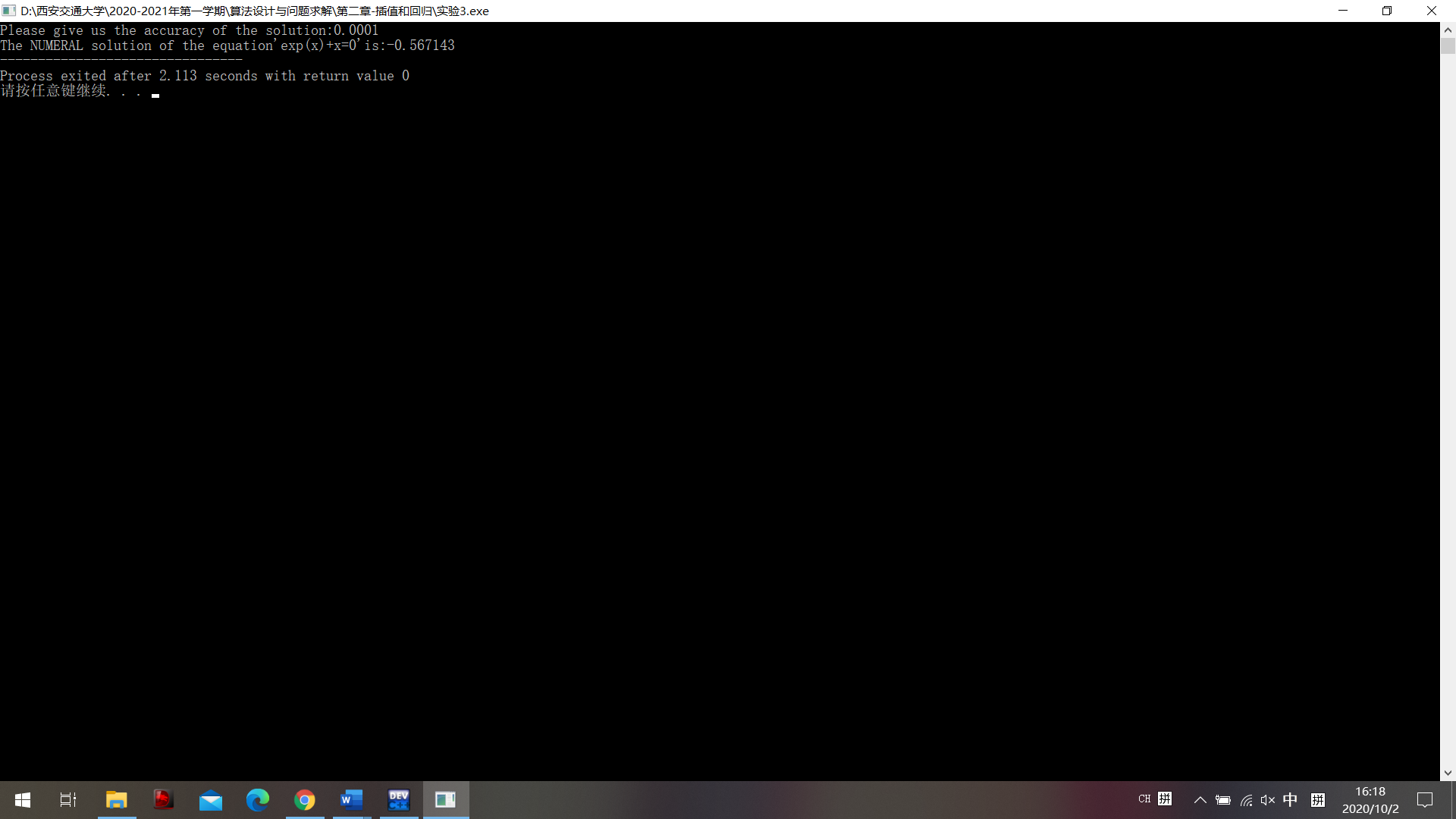
}while(fabs(x0-x)>=epsilon); //在相邻两次迭代结果的差值小于精度时，终止迭代；

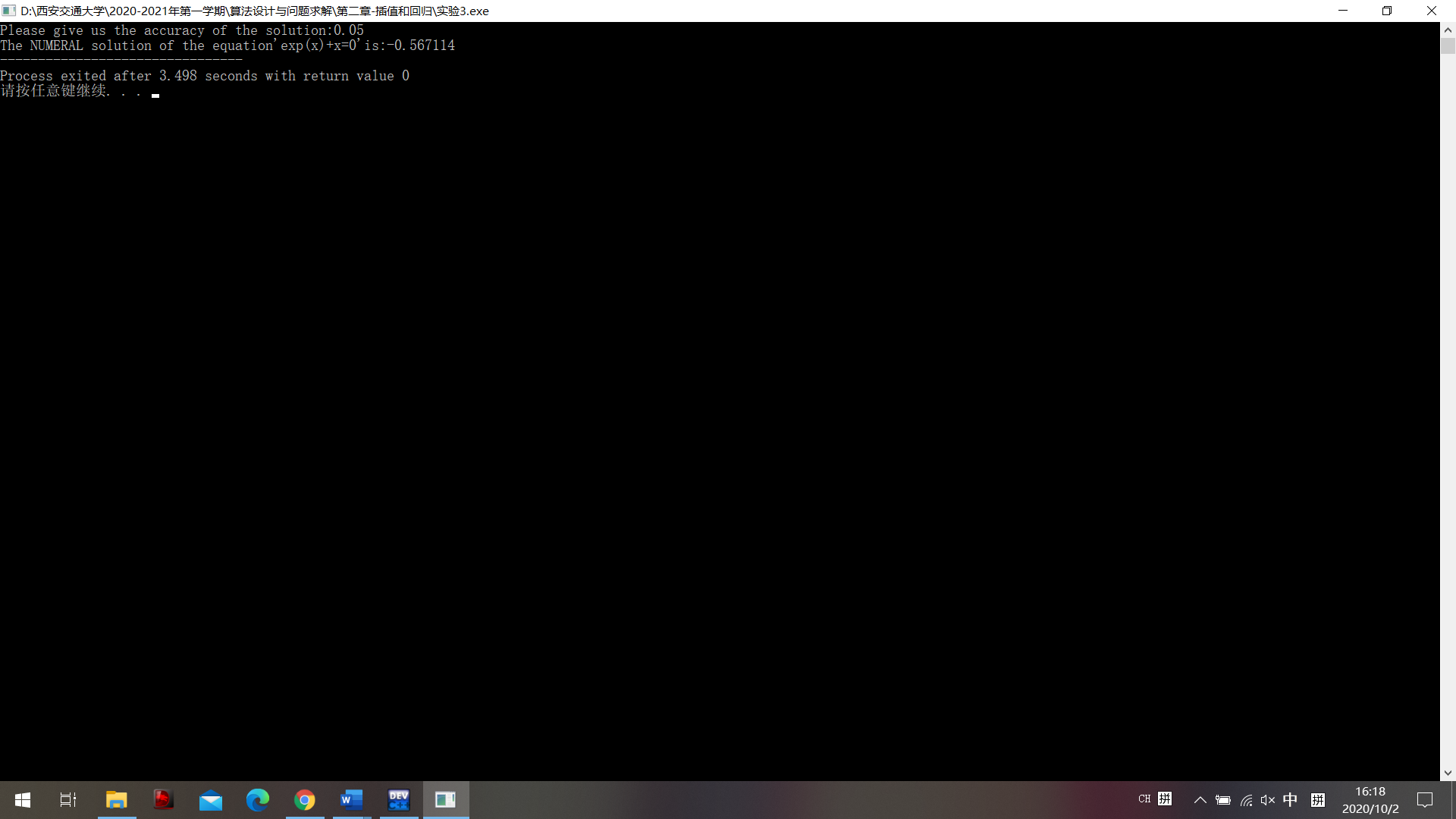
printf("The NUMERAL solution of the equation'exp(x)+x=0'is:%5f",x0); //输出方程的解，主函数结束；

return 0;

}

【运行结果】





* **题目4**

将书上一元线性回归的程序输入电脑运行。对程序中for循环的作用加注释。

【源程序】

#include<stdio.h>

#include<math.h>

#define max\_size 100

int main() //主函数开始；

{

float x[max\_size]={165,165,157,170,175,165,155,170},y[max\_size]={48,57,50,54,64,61,43,59};

int count=8; //声明数据组：第六行分别是身高和体重数据，第七行声明的是数据数量；

float x\_sum=0,y\_sum=0; //第八行声明的是身高、体重的数据和；第九行声明的是身高、体重的均值

float x\_avg,y\_avg;

float Lxy=0,Lxx=0,Lyy=0; //第十行声明的是三种残差和，第十一行声明的是拟合曲线的截距和斜率

float a,b;

int i; //声明辅助计数变量

for(i=0;i<count;i++) //对身高和体重两个数组，从第一个数据开始直至最后一个数据，计算所有数据之和

{

x\_sum=x[i]+x\_sum;

y\_sum=y[i]+y\_sum;

}

x\_avg=x\_sum/count; //计算身高体重的均值

y\_avg=y\_sum/count;

for(i=0;i<count;i++) //对身高和体重两个数组，从第一个数据开始直至最后一个数据，计算所有数据的三种残差和

{

Lxy=(x[i]-x\_avg)\*(y[i]-y\_avg)+Lxy; //这一行计算的是身高体重的残差乘积

Lxx=(x[i]-x\_avg)\*(x[i]-x\_avg)+Lxx; //这一行计算的是身高的残差平方和

Lyy=(y[i]-y\_avg)\*(y[i]-y\_avg)+Lyy; //这一行计算的是体重的残差平方和

}

b=Lxy/Lxx; //依据最小二乘原则计算拟合曲线的斜率和截距

a=y\_avg-b\*x\_avg;

printf("线性拟合的结果为："); //以人能识别的方式输出拟合结果

if(fabs(a)==0)

printf("y=%5.3fx\n",b);

else if(a>0)

printf("y=%5.3fx+%5.3f\n",b,a);

else if(a<0)

printf("y=%5.3fx%5.3f\n",b,a);

float y1=a+b\*172; //计算身高172cm的学生的身高预测值并输出，主函数结束

printf("预测172cm的学生体重为：%4.2fkg\n",y1);

return 0;

}

【运行结果】

