**Mathematics for Embedded Systems, MTTH-1444\_1**

**Lab #1–Part 2**

**Truncation Error**

**Submitted:**

**Navjot Saini**

**Question:**

Write a C++ program proving that to approximate using Taylor series with an error less than :

a) We will need at least 17 terms around a=0.

b) We will need only 17terms around 10.

What do you conclude from these calculations?

**Program:**

1. **C++ program to calculate truncation error of less than , at least 17 terms around a=0 are needed**:

**#include** <iostream> //header file to include input & output stream

**#include** <math.h> //math header file from library is included

**#include** <iomanip> ////manipulators header file include

**using** **namespace** std; ////using standard library namespace

**int** **main**() //main function

{

**float** x= 10.5; //float type variable x is declared and initialized

**float** factorial=1; //float type variable factorial is declared

**float** trunc\_error,rf; //

**float** a=0; // a is initialized to 10

**int** j=0, i=0; //i and j is initialized to zero

**for**(j=0;j<=16;j++) //for loop to find factorial of number

{

factorial=factorial\*(1+j); //factorial is calculated

}

**while**(i<=16) //while loop will execute for i<=16 to calculate rf

{

i++; // increment i by 1

rf=(**exp**(10.5)\*pow((x-a),(i+1))); //rf is calculated used further to calculate truncation error

}

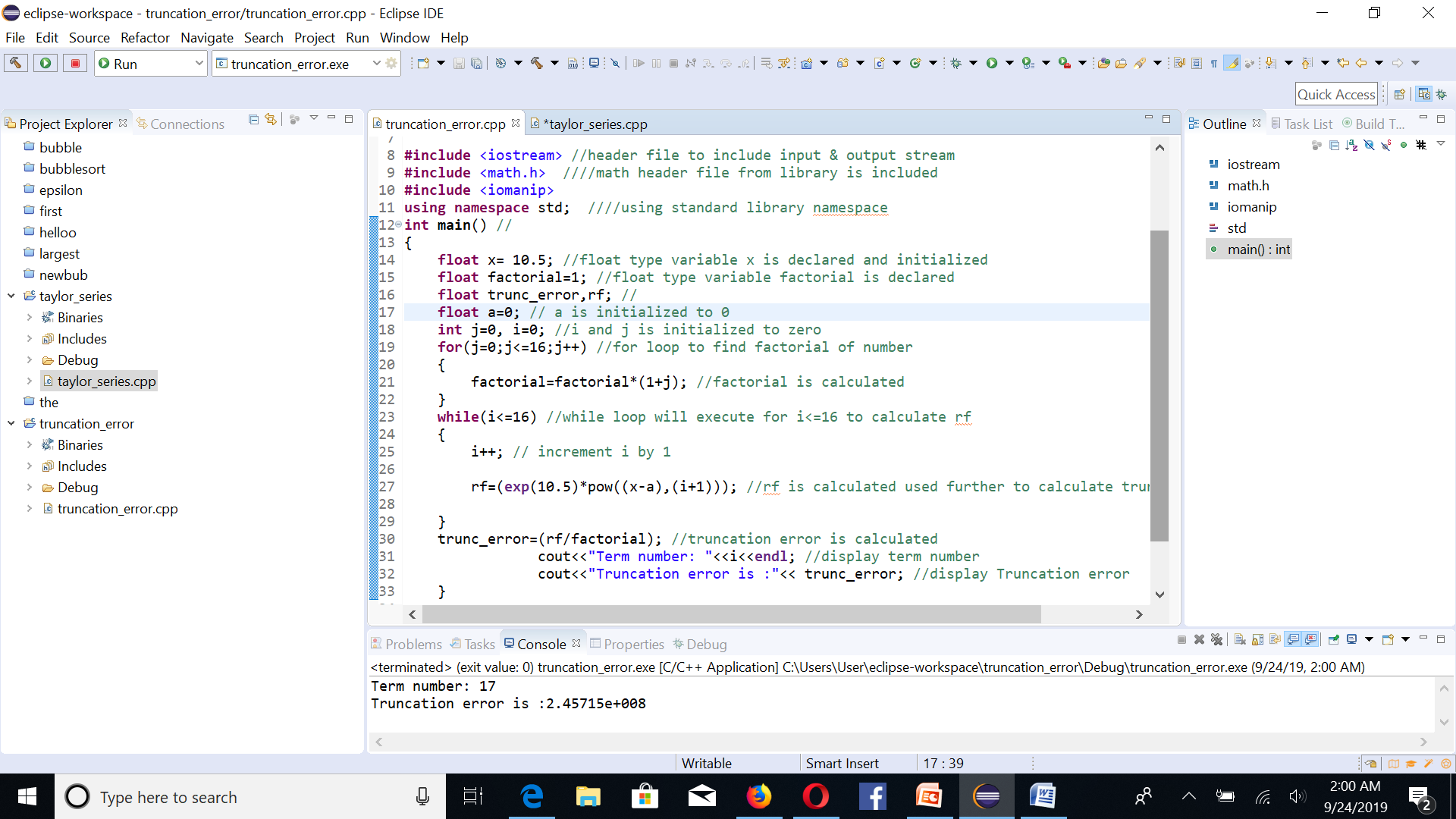
trunc\_error=(rf/factorial); //truncation error is calculated

cout<<"Term number: "<<i<<**endl**; //display term number

cout<<"Truncation error is :"<< trunc\_error; //display Truncation error

}

**Output:**



**Result:**

Term number: 17

Truncation error is: 2.45715e+008

With 17 terms, truncation error is quite high, so greater than 17 terms are required to reduce truncation error to less than.

1. **C++ program to calculate truncation error of less than around a=10, 17 terms are only needed :**

**#include** <iostream> //header file to include input & output stream

**#include** <math.h> //math header file from library is included

**#include** <iomanip> //manipulators header file include

**using** **namespace** std; ////using standard library namespace

**int** **main**() //

{

**float** x= 10.5; //float type variable x is declared and initialized

**float** factorial=1; //float type variable factorial is declared

**float** trunc\_error,rf; //

**float** a=10; // a is initialized to 10

**int** j=0, i=0; //i and j is initialized to zero

**for**(j=0;j<=16;j++) //for loop to find factorial of number

{

factorial=factorial\*(1+j); //factorial is calculated

}

**while**(i<=16) //while loop will execute for i<=16 to calculate rf

{

i++; // increment i by 1

rf=(**exp**(10.5)\*pow((x-a),(i+1))); //rf is calculated used further to calculate truncation error

}

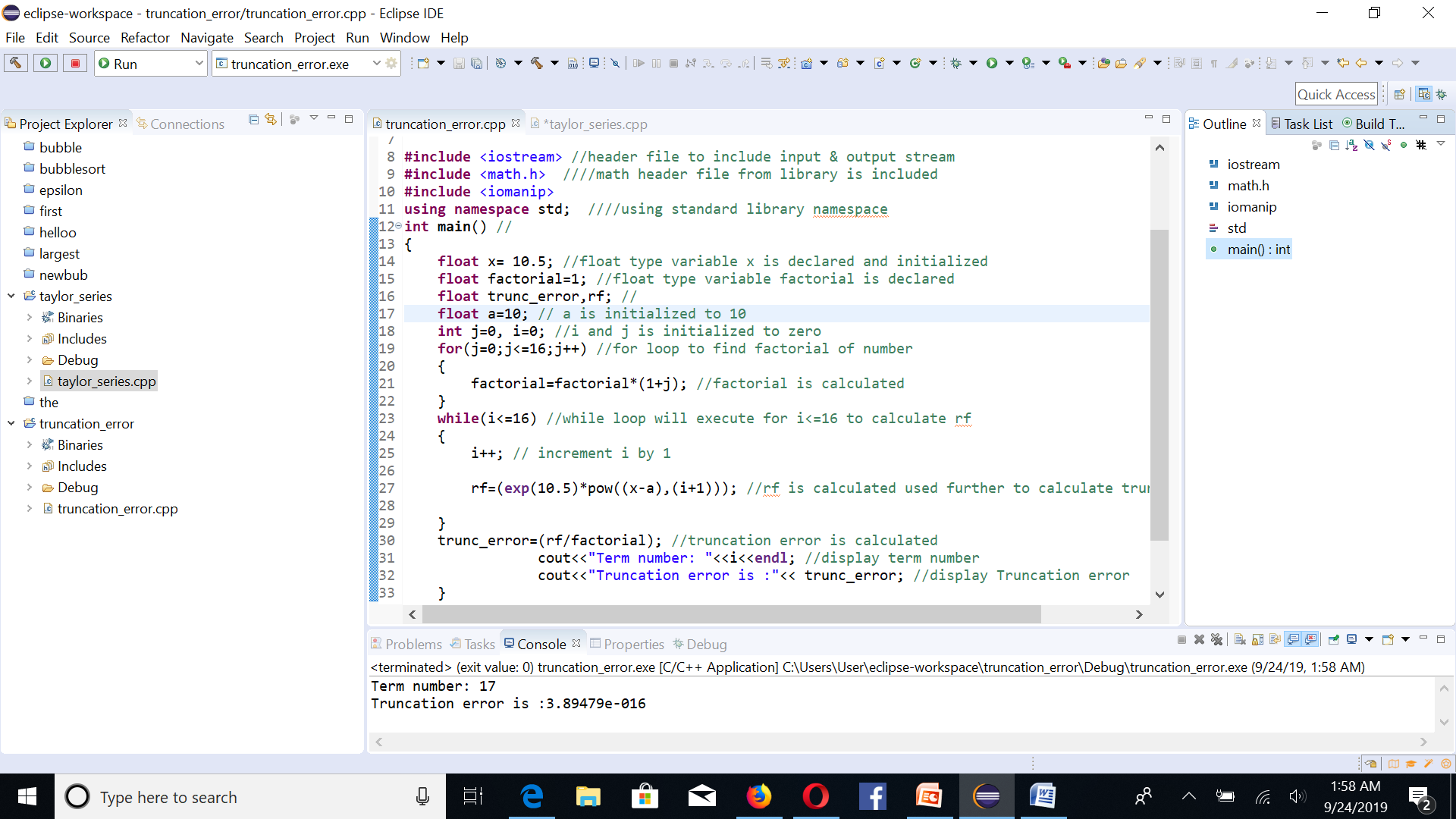
trunc\_error=(rf/factorial); //truncation error is calculated

cout<<"Term number: "<<i<<**endl**; //display term number

cout<<"Truncation error is :"<< trunc\_error; //display Truncation error

}

**Output:**



**Result:**

Term number: 17

Truncation error is :3.89479e-016

At term number 17, truncation error is less than . So, 17 terms are sufficient to calculate error of less than .

**Conclusion:**

* As value of a is increased from 0 to 10, truncation error with 17 number of terms decrease to less than .
* With a=0 and 17 number of terms, truncation error is very high. So, more terms are required to reduce error to less than .
* With a=10 and 17 number of terms, truncation error is less than . So, this much terms are needed only to calculate error

**Lab #1–Part 3**

**Taylor Series**

**Question:**

Write a C++ program equivalent to the following code to calculate the exact value for using Taylor series around a=0.

**Program:**

/\*

\* taylor\_series.cpp

\*

\* Created on: Sep 23, 2019

\* Author: User

\*/

**#include** <iostream> //header file to include input & output stream

**#include** <math.h> //math header file from library is included

**using** **namespace** std; //using standard library namespace

**int** **main**() // main function

{

**float** x = 10, sum = 1, term = 1, temp = 0; //float type variables are declared and initialized

**int** i = 0; // int type i is declared and initialized to 0

**while** (temp != sum) // while loop to check temp is not equal to sum and if true then execute loop

{

i++; // increment i by 1

term = term \* x / i; // calculate sum terms of taylor's series

temp = sum; //temp is initialized with sum value

sum = sum + term; // all terms in taylor series are added

cout<< "Term number:"<<i <<**endl**; // display number of terms in series is i

cout<< "value of "<<i<<" sum term:"<<term<<**endl**; //display terms of taylor series

cout<< "Sum of taylor series with "<<i<<" terms is:"<<sum<<**endl**; //display sum of series

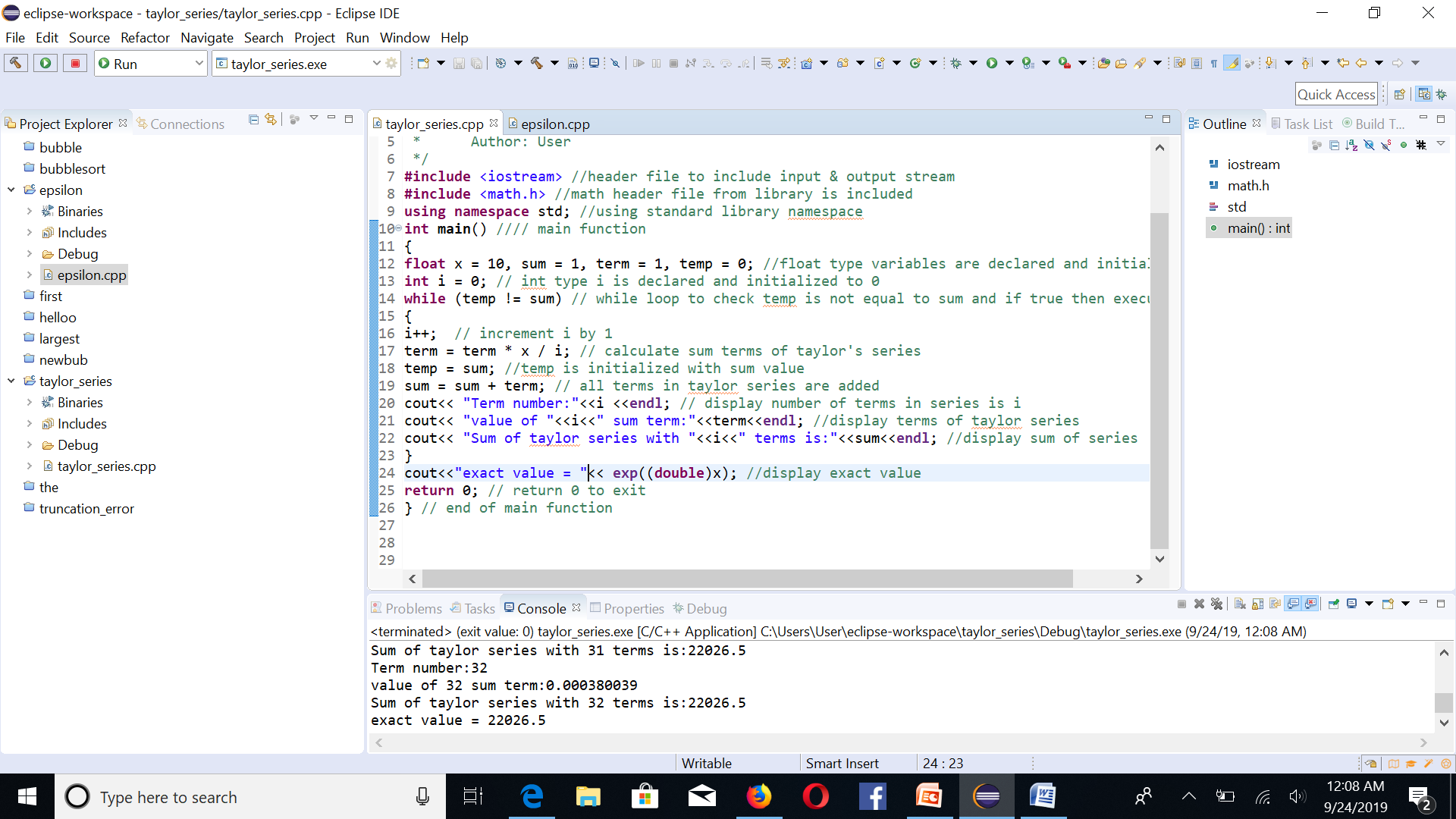
}

cout<<"exact value =\n"<< **exp**((**double**)x); //display exact value

**return** 0; // return 0 to exit

} // end of main function

**Output:**



**Output:**

Sum of taylor series with 32 terms is: 22026.5

exact value of = 22026.5