REPORT OF PROJECT 1

The following document contains:

• Comments and explanations that I think are necessary for understanding my program.

• The output of my program according to the tasks.

• Program listing.

Alejandra Velasco Sánchez-Villares

Explanations

EXCERCISE 1

//STEP 0 : I prepare all I need.

//STEP 0.1 : I include the files I will need to execute my program.

#include <iostream>

#include <cmath>

//STEP 0.2 : I declare the features of the C++ Standard Library.

using namespace std;

//STEP 1 : I define the "sinTaylor" function that approximates the sine function.

double sinTaylor(double e, int i)

{

double s = 1;

//STEP 1.1 : I want to calculate the expression

//e – [ e^2 / ( 4 + 2) ]{ e – [e^2 / ( 16 + 4)] (…( e – [e^2 / ( 4(i-1)(i-1) + 2(i-1) ]){ e – [e^2 / ( 4ii + 2i )]}…)

while (i > 0)

{

s = e - (e\*e / ( 4\*i\*i + 2\*i )) \* s;

--i;

}

return s;

}

//STEP 2 : I define the "lastTermSin" function that calculates the N+1 term of the sine series.

double lastTermSin(double e, int i)

{

double s = 1;

double t = 1;

int j;

int u = 1;

//STEP 2.1 : I multiply the terms by themselves the required times.

for (j = 0; j<= 2\*i+1; j++)

{

s = s\*e;

t = t\*j;

u = u\*(-1);

}

return u\*s/t;

}

//STEP 3 : I define the "lastTermCos" function that calculates the N+1 term of the cosine series.

double lastTermCos(double e, int i)

{

double s = 1;

double t = 1;

int j;

int u = 1;

//STEP 3.1 : I multiply the terms by themselves the required times.

for (j = 1; j<= 2\*i; j++)

{

s = s\*e;

t = t\*j;

u = u\*(-1);

}

return u\*s/t;

}

//STEP 4 : I define the "cosTaylor" function that approximates the cosine function.

double cosTaylor(double e, int i)

{

//STEP 4.1 : I want to calculate the expression

//1 – [ e^2 / ( 4 + 2) ]{ 1 – [e^2 / ( 16 + 4)] (…( 1 – [e^2 / ( 4(i-1)(i-1) + 2(i-1) ]){ 1 – [e^2 / ( 4ii + 2i )]}…)

double s = 1;

while (i > 0)

{

s = 1 - (e\*e / (4\*i\*i - 2\*i)) \* s;

--i;

}

return s;

}

//STEP 5 : I implement the compulsory main function.

int main()

{

//STEP 5.1 : I ask for the input.

cout << " Enter a number of iterations and a number to evaluate the functions " << endl;

//STEP 5.2 : I introduce the type of the input initialising the variables.

double x;

int N;

double sine;

double cosine;

double Sterm;

double Cterm;

//STEP 5.3 : I tell the program the initialised variables are the input.

cin >> N >> x;

//STEP 5.4 : I call the functions.

sine = sinTaylor(x,N);

cosine = cosTaylor(x,N);

Sterm = lastTermSin(x,N);

Cterm = lastTermCos(x,N);

//STEP 5.5 : I give the solutions and I compare them as required.

cout << " The aproximation of the sine is " << sine << " and the aproximation of the cosine is " << cosine << endl;

double a;

a = abs(sin(x)-sine);

cout << " The error of the sine is " << a << endl;

double b;

b = abs(cos(x)-cosine);

cout << " The error of the cosine is " << b << endl;

if (a < Sterm)

{

cout << " The error of the sine is bounded by the N+1 term " << endl;

}

else

{

cout << " The error of the sine is not bounded by the N+1 term " << endl;

}

if (b < Cterm)

{

cout << " The error of the cosine is bounded by the N+1 term " << endl;

}

else

{

cout << " The error of the cosine is not bounded by the N+1 term " << endl;

}

return 0;

}

EXCERCISE 2

//STEP 0 : I prepare all I need.

//STEP 0.1 : I include the files I will need to execute my program.

#include <iostream>

#include <cmath>

//STEP 0.2 : I declare the features of the C++ Standard Library.

using namespace std;

//STEP 1 : I define the function "f" that I want to integrate.

typedef double (\*FunctionPointer)(double);

double f(double x){return (1 + sin(exp(x\*x\*x)));}

//STEP 2 : I implement the algorithm "ASI".

double ASI ( FunctionPointer f, double a, double b, double tol)

{

//STEP 2.1 : I calculate all the values I need.

double I1 = ((b - a)/6)\*(f(a) + 4\*f((a + b)/2) + f(b));

double c = (a + b)/2;

double I2 = ((c - a)/6)\*(f(a) + 4\*f((a + c)/2) + f(c)) + ((b - c)/6)\*(f(c) + 4\*f((c + b)/2) + f(b));

//STEP 2.2 : I calculate the error between these values and check if it is smaller than 15\*tol.

// If it is smaller I obtain the I2

double errest = abs(I1-I2);

if (errest < 15\*tol)

{

return I2;

}

//STEP 2.3 : I obtain the solution of the algorithm.

return ASI(f,a,(a+b)/2,tol/2) + ASI(f,(a+b)/2,b,tol/2);

}

//STEP 3 : I implement the compulsory main function.

int main()

{

//STEP 3.1 : I ask for the input.

cout << "Introduce two numbers and a tolerance " << endl;

//STEP 3.2 : I introduce the type of the input initialising the variables.

double a;

double b;

double tol;

//STEP 3.3 : I tell the program the initialised variables are the input.

cin >> a >> b >> tol;

//STEP 3.4 : I call the function of the algorithm.

double A = ASI(&f, a, b, tol);

//STEP 3.5 : I give the solution.

cout << A << endl;

//STEP 3.6 : Every main function must accomplish the following.

return 0;

}

Output

EXCERCISE 1

For x=-1

N=10

Enter a number of iterations and a number to evaluate the functions

10

-1

The aproximation of the sine is -0.841471 and the aproximation of the cosine is 0.540302

The error of the sine is 0

The error of the cosine is 0

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 36.671 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

-1

The aproximation of the sine is -0.841471 and the aproximation of the cosine is 0.540302

The error of the sine is 0

The error of the cosine is 0

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 7.881 s

Press any key to continue.

For x=1

N=10

Enter a number of iterations and a number to evaluate the functions

10

1

The aproximation of the sine is 0.841471 and the aproximation of the cosine is 0.540302

The error of the sine is 0

The error of the cosine is 0

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 11.761 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

1

The aproximation of the sine is 0.841471 and the aproximation of the cosine is 0.540302

The error of the sine is 0

The error of the cosine is 0

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 7.983 s

Press any key to continue.

For x=2

N=10

Enter a number of iterations and a number to evaluate the functions

10

2

The aproximation of the sine is 0.909297 and the aproximation of the cosine is -0.416147

The error of the sine is 2.02061e-014

The error of the cosine is 3.83027e-015

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 3.655 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

2

The aproximation of the sine is 0.909297 and the aproximation of the cosine is -0.416147

The error of the sine is 1.11022e-016

The error of the cosine is 5.55112e-017

The error of the sine is bounded by the N+1 term

The error of the cosine is not bounded by the N+1 term

Process returned 0 (0x0) execution time : 4.015 s

Press any key to continue.

For x=3

N=10

Enter a number of iterations and a number to evaluate the functions

10

3

The aproximation of the sine is 0.14112 and the aproximation of the cosine is -0.989992

The error of the sine is 1.32906e-010

The error of the cosine is 2.747e-011

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 5.608 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

3

The aproximation of the sine is 0.14112 and the aproximation of the cosine is -0.989992

The error of the sine is 1.94289e-016

The error of the cosine is 0

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 3.903 s

Press any key to continue.

For x=5

N=10

Enter a number of iterations and a number to evaluate the functions

10

5

The aproximation of the sine is -0.958931 and the aproximation of the cosine is 0.283664

The error of the sine is 7.02391e-006

The error of the cosine is 2.02867e-006

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 4.810 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

5

The aproximation of the sine is -0.958924 and the aproximation of the cosine is 0.283662

The error of the sine is 6.66134e-016

The error of the cosine is 4.44089e-016

The error of the sine is bounded by the N+1 term

The error of the cosine is not bounded by the N+1 term

Process returned 0 (0x0) execution time : 9.061 s

Press any key to continue.

For x=10

N=10

Enter a number of iterations and a number to evaluate the functions

10

10

The aproximation of the sine is -14.8546 and the aproximation of the cosine is 6.66456

The error of the sine is 14.3105

The error of the cosine is 7.50364

The error of the sine is bounded by the N+1 term

The error of the cosine is bounded by the N+1 term

Process returned 0 (0x0) execution time : 12.705 s

Press any key to continue.

N=50

Enter a number of iterations and a number to evaluate the functions

50

10

The aproximation of the sine is -0.544021 and the aproximation of the cosine is -0.839072

The error of the sine is 9.04832e-014

The error of the cosine is 1.36446e-013

The error of the sine is bounded by the N+1 term

The error of the cosine is not bounded by the N+1 term

Process returned 0 (0x0) execution time : 5.993 s

Press any key to continue.

We can conclude that the approximation is worse the bigger the input is, in general.

EXCERCISE 2

For tol=10^(-2)

Introduce two numbers and a tolerance

-1

1

10^(-2)

The approximation of the integral is 3.52761

Process returned 0 (0x0) execution time : 9.180 s

Press any key to continue.

For tol=10^(-3)

Introduce two numbers and a tolerance

-1

1

10^(-3)

The approximation of the integral is 3.52761

Process returned 0 (0x0) execution time : 11.280 s

Press any key to continue.

For tol=10^(-4)

Introduce two numbers and a tolerance

-1

1

10^(-4)

The approximation of the integral is 3.52761

Process returned 0 (0x0) execution time : 16.949 s

Press any key to continue.

For tol=10^(-8)

Introduce two numbers and a tolerance

-1

1

10^(-8)

The approximation of the integral is 3.52761

Process returned 0 (0x0) execution time : 10.332 s

Press any key to continue.

The value provided by MATLAB is 2.500809110336167

We can see the result doesn’t depend on the given tolerances and difeers with the MATLAB value in around one.

Program Listing

EXCERCISE 1

#include <iostream>

#include <cmath>

using namespace std;

double sinTaylor(double e, int i)

{

double s = 1;

while (i > 0)

{

s = e - (e\*e / ( 4\*i\*i + 2\*i )) \* s;

--i;

}

return s;

}

double lastTermSin(double e, int i)

{

double s = 1;

double t = 1;

int j;

int u = 1;

for (j = 0; j<= 2\*i+1; j++)

{

s = s\*e;

t = t\*j;

u = u\*(-1);

}

return u\*s/t;

}

double lastTermCos(double e, int i)

{

double s = 1;

double t = 1;

int j;

int u = 1;

for (j = 1; j<= 2\*i; j++)

{

s = s\*e;

t = t\*j;

u = u\*(-1);

}

return u\*s/t;

}

double cosTaylor(double e, int i)

{

double s = 1;

while (i > 0)

{

s = 1 - (e\*e / (4\*i\*i - 2\*i)) \* s;

--i;

}

return s;

}

int main()

{

cout << " Enter a number of iterations and a number to evaluate the functions " << endl;

double x;

int N;

double sine;

double cosine;

double Sterm;

double Cterm;

cin >> N >> x;

sine = sinTaylor(x,N);

cosine = cosTaylor(x,N);

Sterm = lastTermSin(x,N);

Cterm = lastTermCos(x,N);

cout << " The aproximation of the sine is " << sine << " and the aproximation of the cosine is " << cosine << endl;

double a;

a = abs(sin(x)-sine);

cout << " The error of the sine is " << a << endl;

double b;

b = abs(cos(x)-cosine);

cout << " The error of the cosine is " << b << endl;

if (a < Sterm)

{

cout << " The error of the sine is bounded by the N+1 term " << endl;

}

else

{

cout << " The error of the sine is not bounded by the N+1 term " << endl;

}

if (b < Cterm)

{

cout << " The error of the cosine is bounded by the N+1 term " << endl;

}

else

{

cout << " The error of the cosine is not bounded by the N+1 term " << endl;

}

return 0;

}

EXCERCISE 2

#include <iostream>

#include <cmath>

using namespace std;

typedef double (\*FunctionPointer)(double);

double f(double x){return (1 + sin(exp(x\*x\*x)));}

double ASI( FunctionPointer f, double a, double b, double tol)

{

double I1 = ((b - a)/6)\*(f(a) + 4\*f((a + b)/2) + f(b));

double c = (a + b)/2;

double I2 = ((c - a)/6)\*(f(a) + 4\*f((a + c)/2) + f(c)) + ((b - c)/6)\*(f(c) + 4\*f((c + b)/2) + f(b));

double errest = abs(I1-I2);

if (errest < 15\*tol)

{

return I2;

}

return ASI(f,a,(a+b)/2,tol/2) + ASI(f,(a+b)/2,b,tol/2);

}

int main()

{

cout << "Introduce two numbers and a tolerance " << endl;

double a;

double b;

double tol;

cin >> a >> b >> tol;

cout << A << endl;

return 0;

}