Basic Math Operations

C++ can be used to perform basic mathematical operations. The following program can be used to illustrate this.

les_02_code_01.cpp

```
#include<iostream>

using namespace std;

int main()

{

    int number1;
    int number2;
    int result;

    cout<<"Please enter number1 & number2";
    cin>>number1>>number2;

    result = number1 + number2;
    result = number1 - number2;
    result = number1 * number2;
    result = number1 / number2;
    result = number1 % number2;

    cout<<result;
    return 0;
}
```

S.No	number1	number2	operation	result
1	12	8	+	20
2	12	8	-	4
3	12	8	*	96
4	12	8	/	1
5	12	8	%	4

In the above program the variable `number1` and `number2` are called operands and they are connected via different operators in expressions given on line numbers 16, 17, 18, 19 20. Response of each operation is stored in the variable `result.`

Keep in mind that modulus (`%`) operator is only defined for the data type `integers`

It is important to emphasis that result of the division is not as we expect in general. This is because the data type of `number1` and `number2` is integer, an integer divided by an integer will give an integer response, while truncating the decimal part of the value. This makes the order of precedence of arithmetic operators very significant. Consider the following example.

```
y = 5 / 2 * 5 + 3 * 5 + 7;
cout<<y;
```

```
y =  5 * 5 / 2 + 3 * 5 + 7;
cout<<y;
```

What will be the value of y in above cases?

Case – I y = 32

Case – II y = 34

Precedence of Arithmetic Operators

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does not specify the order in which these parenthesized sub expressions will be evaluated.]
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+, -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

les_02_code_02.cpp

```
1.      #include<iostream>
2.
3.      using namespace std;
4.
5.      int main()
6.      {
7.
8.          int number1 = 74, number2 = 82, number3 = 88;
9.          double average;
10.         average = number1 + number2 + number3 / 3;
11.         cout<<average;
12.         return 0;
13.     }
```

les_02_code_03.cpp

```
1.      #include<iostream>
2.
3.      using namespace std;
4.
5.      int main()
6.      {
7.
8.          int number1 = 74, number2 = 82, number3 = 88;
9.          double average;
10.         average = (number1 + number2 + number3) / 3;
11.         cout<<average;
12.         return 0;
13.     }
```

Using Constant Variable

les_02_code_04.cpp

```
1.      #include<iostream>
2.
3.      using namespace std;
4.
5.      int main()
6.      {
7.
8.          int number1 = 74, number2 = 82, number3 = 88;
9.          double average;
10.         const int numGrades = 3;
11.         average = (number1 + number2 + number3) / numGrades;
12.         cout<<average;
13.         return 0;
14.     }
```

If the value of constant is tried to be changed during the program the compiler will throw an error

```
Error: assignment of read only variable
```

Constants are much helpful in situations when a certain constant is required again and again. For example value of pi used in a program for different calculations.

Advanced Mathematical Functions

Some very useful and advanced mathematical functions are present in `<cmath>` library. Which can be included in a program through preprocessor directive `#include<cmath>.` Following are the few functions which are available in this library.

Category	Function	Description
Trigonometry	cos	Returns the cosine of an angle of *x* radians.
	sin	Returns the sine of an angle of *x* radians.
	tan	Returns the tangent of an angle of *x* radians.
	acos	**The acos** function computes the principal value of the arc cosine of **x.** A domain error occurs for arguments not in the range [-I. +I].
	asin	The **asin** function computes the principal value of the arc sine of **x.** A domain error occurs for arguments not in the range [-I, +I].
	atan	The atan function returns the arc tangent in the range [[-pi/2, +pi/2]]
Exponential and logarithmic function	exp	Returns the base-e exponential function of *x*, which is e raised to the power *x*: ex.
	log	Returns the natural logarithm of *x*. If the argument is negative, a domain error occurs.
	log10	Returns the common (base-10) logarithm of *x*. If the argument is negative, a domain error occurs.
Power Functions	pow	Returns base raised to the power exponent: e.g. pow(7.0, 3.0); will find 7 ^ 3
	sqrt	Returns the square root of *x*. If *x* is negative, a domain error occurs:
	cbrt	Returns the cubic root of *x*.
Rounding and Remainder Functions	ceil	Rounds *x* upward, returning the smallest integral value that is not less than *x*.
	floor	Rounds *x* downward, returning the largest integral value that is not greater than *x*.
	fmod	Returns the floating-point remainder of *numer*/*denom*
	trunc	Rounds *x* toward zero, returning the nearest integral value that is not larger in magnitude than *x*.
	round	Returns the integral value that is nearest to *x*, with halfway cases rounded away from zero.
Other Functions	fabs	Returns the absolute value of *x*: |*x*|.
	abs	Returns the absolute value of *x*: |*x*|.

How to use trigonometric functions

les_02_code_05.cpp

```
1.       #include<iostream>
2.       #include<cmath>
3.
4.       using namespace std;
5.
6.       int main()
7.       {
8.
9.          const double pi = 3.141592;
10.         double angle = pi/6;
11.         cout<<endl<<"******** Calculating Trigonometric Ratios
            ********"<<endl;
12.         cout<<endl<<"All calculations on Angle "<<angle<<"
            Radians"<<endl;
13.         cout<<endl<<"cos("<<angle<<") "<<"= "<<cos(angle)<<endl;
14.         cout<<endl<<"sin("<<angle<<") "<<"= "<<sin(angle)<<endl;
15.         cout<<endl<<"tan("<<angle<<") "<<"= "<<tan(angle)<<endl;
16.         cout<<endl<<"******** Calculations Terminated ********"<<endl;
17.
18.         return 0;
19.      }
```

Output

```
******** Calculating Trigonometric Ratios ********

All calculations on Angle 0.523599 Radians

cos(0.523599) = 0.866025

sin(0.523599) = 0.5

tan(0.523599) = 0.57735

******** Calculations Terminated ********
```

Task: Write a program that takes the angle input from the user in degrees and find its cos, sin and tan.

How to use exponential and logarithmic functions

les_02_code_06.cpp

```
1.       #include<iostream>
2.       #include<cmath>
3.
4.       using namespace std;
5.
6.       int main()
7.      {
8.          double num = 10.3;
9.          cout<<endl<<"exp("<<num<<") "<<"= "<<exp(num)<<endl;
10.         cout<<endl<<"log("<<num<<") "<<"= "<<log(num)<<endl;
11.         cout<<endl<<"log10("<<num<<") "<<"= "<<log10(num)<<endl;
12.
13.         return 0;
14.      }
```

Output

```
exp(10.3) = 29732.6

log(10.3) = 2.33214

log10(10.3) = 1.01284
```

How to use power functions

```
1.       #include<iostream>
2.       #include<cmath>
3.
4.       using namespace std;
5.
6.       int main()
7.       {
8.          double num1 = 10.3, num2 = 2.0;
9.          cout<<endl<<"pow("<<num1<<","<<num2<<") "<<"=
            "<<pow(num1,num2)<<endl;
10.         cout<<endl<<"sqrt("<<num1<<") "<<"= "<<sqrt(num1)<<endl;
11.         cout<<endl<<"cbrt("<<num1<<") "<<"= "<<cbrt(num1)<<endl;
12.
13.         return 0;
14.      }
```

Output

```
pow(10.3,2) = 106.09

sqrt(10.3) = 3.20936

cbrt(10.3) = 2.17577
```

How to use rounding and remainder functions

les_02_code_08.cpp

```
1. #include<iostream>
2. #include<cmath>
3.
4. using namespace std;
5.
6. int main()
7. {
8. double num1 = 2.3,num2 = 3.8,num3 = 5.5,num4 = -2.3,num5 = -3.8,num6 =
   -5.5;
9. cout<<"value\tround\tfloor\tceil\ttrunc\n";
10.      cout<<"-----\t-----\t-----\t----\t-----\n";
11.      cout<<num1<<"\t"<<round(num1)<<"\t"<<floor(num1)<<"\t"<<ceil(num1
   )<<"\t"<<trunc(num1)<<"\n";
12.      cout<<num2<<"\t"<<round(num2)<<"\t"<<floor(num2)<<"\t"<<ceil(num2
   )<<"\t"<<trunc(num2)<<"\n";
13.      cout<<num3<<"\t"<<round(num3)<<"\t"<<floor(num3)<<"\t"<<ceil(num3
   )<<"\t"<<trunc(num3)<<"\n";
14.      cout<<num4<<"\t"<<round(num4)<<"\t"<<floor(num4)<<"\t"<<ceil(num4
   )<<"\t"<<trunc(num4)<<"\n";
15.      cout<<num5<<"\t"<<round(num5)<<"\t"<<floor(num5)<<"\t"<<ceil(num5
   )<<"\t"<<trunc(num5)<<"\n";
16.      cout<<num6<<"\t"<<round(num6)<<"\t"<<floor(num6)<<"\t"<<ceil(num6
   )<<"\t"<<trunc(num6)<<"\n";
17.
18.      return 0;
19.      }
```

Output

```
value   round   floor   ceil    trunc
-----   -----   -----   ----    -----
2.3     2       2       3       2
3.8     4       3       4       3
5.5     6       5       6       5
-2.3    -2      -3      -2      -2
-3.8    -4      -4      -3      -3
-5.5    -6      -6      -5      -5
```

les_02_code_09.cpp

```
1.      #include<iostream>
2.      #include<cmath>
3.
4.      using namespace std;
5.
6.      int main ()
7.      {
8.          cout<<"fmod(5.3, 2.0) = "<<fmod(5.3,2)<<endl;
9.          cout<<"fmod(18.5, 4.2) = "<<fmod (18.5,4.2)<<endl;
10.         return 0;
11.     }
```

Output

```
fmod(5.3, 2.0) = 1.3
fmod(18.5, 4.2) = 1.7
```

les_02_code_10.cpp

```
1.      #include<iostream>
2.      #include<cmath>
3.
4.      using namespace std;
5.
6.      int main ()
7.      {
8.          float num1 = -9.5;
9.          cout<<fabs(num1);
10.
11.         return 0;
12.     }
```

Write a program that takes a floating point input from the user, finds its square, cube, square root, floor, truncate, ceiling and round off, natural log, base 10 log.

Type Casting

les_02_code_11.cpp

```
1.      #include<iostream>
2.      #include<cmath>
3.
4.      using namespace std;
5.
6.      int main ()
7.      {
8.         float num1 = -9.5;
9.         int num2 = 101;
10.        cout<<(int)num1;
11.        cout<<endl<<(float)num2/10;
12.
13.        return 0;
14.     }
```

Output

```
-9
10.1
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

Formatting with <iomanip>

A header file contains function to control output stream

les_02_code_12.cpp

```
1.      #include<iostream>
2.      #include<cmath>
3.      #include<iomanip>
4.
5.      using namespace std;
6.
7.      int main()
8.      {
9.
10.         double planck = 6.62607004e-34;
11.         double gravitation = 6.67408e-11;
```

```
12.          double charge = 1.60217662e-19;
13.          cout<<"********Output Without Precision********"<<endl;
14.          cout<<endl<<endl;
15.          cout<<"Planck's Constant = "<<planck<<endl;
16.          cout<<"Gravitational Constant = "<<gravitation<<endl;
17.          cout<<"Charge on electron = "<<charge<<endl;
18.          cout<<endl<<endl;
19.          cout<<"********Output with Precision of 10********"<<endl;
20.          cout<<endl<<endl;
21.          cout<<setprecision(10);
22.          cout<<"Planck's Constant = "<<planck<<endl;
23.          cout<<"Gravitational Constant = "<<gravitation<<endl;
24.          cout<<"Charge on electron = "<<charge<<endl;
25.
26.          return 0;
27.      }
```

Output

```
********Output Without Precision********


Planck's Constant = 6.62607e-034
Gravitational Constant = 6.67408e-011
Charge on electron = 1.60218e-019


********Output with Precision of 10********


Planck's Constant = 6.62607004e-034
Gravitational Constant = 6.67408e-011
Charge on electron = 1.60217662e-019
```

les_02_code_13.cpp

```
1.      #include<iostream>
2.      #include<cmath>
3.      #include<iomanip>
4.
5.      using namespace std;
6.
7.      int main()
8.      {
9.
10.         cout<<0.1234567809<<endl;
11.         cout<<setprecision(10)<<0.1234567809<<endl;
12.         cout<<setprecision(12)<<0.123412341234<<endl;
13.         cout<<setprecision(8)<<0.12341234<<endl;
14.         return 0;
15.     }
```

Output

```
0.123457
0.1234567809
0.123412341234
0.12341234
```

les_02_code_14.cpp

```
1.    #include<iostream>
2.    #include<iomanip>
3.    #include<cmath>

4.    using namespace std;

5.    int main()
6.    {
7.       double num1 = 2.3,num2 = 3.8,num3 = 5.5,num4 = -2.3,num5 = -
         3.8,num6 = -5.5;
8.       cout<<setw(8)<<"value"<<setw(8)<<"round"<<setw(8)<<"floor"<<setw(
         8)<<"ceil"<<setw(8)<<"trunc\n";
9.       cout<<setw(8)<<"-----"<<setw(8)<<"-----"<<setw(8)<<"-----
         "<<setw(8)<<"-----"<<setw(8)<<"-----\n";
10.      cout<<setw(8)<<num1<<setw(8)<<round(num1)<<setw(8)<<floor(num1)<<
         setw(8)<<ceil(num1)<<setw(8)<<trunc(num1)<<"\n";
11.      cout<<setw(8)<<num2<<setw(8)<<round(num2)<<setw(8)<<floor(num2)<<
         setw(8)<<ceil(num2)<<setw(8)<<trunc(num2)<<"\n";
12.      cout<<setw(8)<<num3<<setw(8)<<round(num3)<<setw(8)<<floor(num3)<<
         setw(8)<<ceil(num3)<<setw(8)<<trunc(num3)<<"\n";
```

```
13.      cout<<setw(8)<<num4<<setw(8)<<round(num4)<<setw(8)<<floor(num4)<<
         setw(8)<<ceil(num4)<<setw(8)<<trunc(num4)<<"\n";
14.      cout<<setw(8)<<num5<<setw(8)<<round(num5)<<setw(8)<<floor(num5)<<
         setw(8)<<ceil(num5)<<setw(8)<<trunc(num5)<<"\n";
15.      cout<<setw(8)<<num6<<setw(8)<<round(num6)<<setw(8)<<floor(num6)<<
         setw(8)<<ceil(num6)<<setw(8)<<trunc(num6)<<"\n";

16.      return 0;
17.   }
```

Output

```
   value   round   floor    ceil  trunc
   -----   -----   -----   -----  -----
     2.3       2       2       3      2
     3.8       4       3       4      3
     5.5       6       5       6      5
    -2.3      -2      -3      -2     -2
    -3.8      -4      -4      -3     -3
    -5.5      -6      -6      -5     -5
```

Exercise

Write a note on the importance of operator precedence, also emphasis why your portable calculator don't cause that problem.

Try to use cmath library functions with integer data types and find out the possible problems.

Explore the usage of functions `scientific` and `fixed` in C++

Implement the following expressions in C++

i. $|x + y|$

ii. $|x| + |y|$

iii. $\frac{x^3}{y} + y$

iv. $\sqrt{x^6 + y^5}$

v. $\left(x + \sqrt{y}\right)^7$

Try to give wrong inputs to the following functions

i. sqrt, ii. acos, iii. atan, iv. asin