ceil(double x) returns the smallest integer value greater than or equal to **x**.

Declaration

Following is the declaration for ceil() function.

```
double ceil(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the smallest integral value not less than **x**.

Example

The following example shows the usage of ceil() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   float val1, val2, val3, val4;

   val1 = 1.6;
   val2 = 1.2;
   val3 = 2.8;
   val4 = 2.3;

   printf ("value1 = %.1lf\n", ceil(val1));
   printf ("value2 = %.1lf\n", ceil(val2));
   printf ("value3 = %.1lf\n", ceil(val3));
   printf ("value4 = %.1lf\n", ceil(val4));

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
value1 = 2.0
value2 = 2.0
value3 = 3.0
```

value4 = 3.0

fabs(double x) returns the absolute value of **x**.

Declaration

Following is the declaration for fabs() function.

```
double fabs(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the absolute value of x.

Example

The following example shows the usage of fabs() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   int a, b;
   a = 1234;
   b = -344;

   printf("The absolute value of %d is %lf\n", a, fabs(a));
   printf("The absolute value of %d is %lf\n", b, fabs(b));

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The absolute value of 1234 is 1234.000000
The absolute value of -344 is 344.000000
```

----

pow(double x, double y) returns **x** raised to the power of **y** i.e. x^y.

Declaration

Following is the declaration for pow() function.

```
double pow(double x, double y)
```

Parameters

- **x** − This is the floating point base value.
- **y** − This is the floating point power value.

Return Value

This function returns the result of raising **x** to the power **y**.

Example

The following example shows the usage of pow() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   printf("Value 8.0 ^ 3 = %lf\n", pow(8.0, 3));

   printf("Value 3.05 ^ 1.98 = %lf", pow(3.05, 1.98));

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Value 8.0 ^ 3 = 512.000000
Value 3.05 ^ 1.98 = 9.097324
```

---

sqrt(double x) returns the square root of **x**.

Declaration

Following is the declaration for sqrt() function.

```
double sqrt(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the square root of x.

Example

The following example shows the usage of sqrt() function.

```
#include <stdio.h>
#include <math.h>

int main () {

   printf("Square root of %lf is %lf\n", 4.0, sqrt(4.0) );
   printf("Square root of %lf is %lf\n", 5.0, sqrt(5.0) );

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Square root of 4.000000 is 2.000000
Square root of 5.000000 is 2.236068
```

----

double floor(double x) returns the largest integer value less than or equal to **x**.

Declaration

Following is the declaration for floor() function.

```
double floor(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the largest integral value not greater than x.

Example

The following example shows the usage of floor() function.

```
#include <stdio.h>
#include <math.h>
```

```
int main () {
   float val1, val2, val3, val4;

   val1 = 1.6;
   val2 = 1.2;
   val3 = 2.8;
   val4 = 2.3;

   printf("Value1 = %.1lf\n", floor(val1));
   printf("Value2 = %.1lf\n", floor(val2));
   printf("Value3 = %.1lf\n", floor(val3));
   printf("Value4 = %.1lf\n", floor(val4));

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Value1 = 1.0
Value2 = 1.0
Value3 = 2.0
Value4 = 2.0
```

----

exp(double x) returns the value of **e** raised to the **xth** power.

Declaration

Following is the declaration for exp() function.

```
double exp(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the exponential value of x.

Example

The following example shows the usage of exp() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   double x = 0;

   printf("The exponential value of %lf is %lf\n", x, exp(x));
   printf("The exponential value of %lf is %lf\n", x+1, exp(x+1));
   printf("The exponential value of %lf is %lf\n", x+2, exp(x+2));

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The exponential value of 0.000000 is 1.000000
The exponential value of 1.000000 is 2.718282
The exponential value of 2.000000 is 7.389056
```

----

fmod(double x, double y) returns the remainder of **x** divided by **y**.

Declaration

Following is the declaration for fmod() function.

```
double fmod(double x, double y)
```

Parameters

- **x** − This is the floating point value with the division numerator i.e. x.
- **y** − This is the floating point value with the division denominator i.e. y.

Return Value

This function returns the remainder of dividing x/y.

Example

The following example shows the usage of fmod() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   float a, b;
   int c;
   a = 9.2;
```

```
   b = 3.7;
   c = 2;
   printf("Remainder of %f / %d is %lf\n", a, c, fmod(a,c));
   printf("Remainder of %f / %f is %lf\n", a, b, fmod(a,b));
   
   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Remainder of 9.200000 / 2 is 1.200000
Remainder of 9.200000 / 3.700000 is 1.800000
```

----

log(double x) returns the natural logarithm (base-e logarithm) of **x**.

Declaration

Following is the declaration for log() function.

```
double log(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns natural logarithm of x.

Example

The following example shows the usage of log() function.

```
#include <stdio.h>
#include <math.h>

int main () {
   double x, ret;
   x = 2.7;

   /* finding log(2.7) */
   ret = log(x);
   printf("log(%lf) = %lf", x, ret);
   
   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
log(2.700000) = 0.993252
```

---

double cos(double x)

Returns the cosine of a radian angle x.

double sin(double x)

Returns the sine of a radian angle x.

```
#include <stdio.h>
#include <math.h>
int main()
{
      float i = 0.314;
      float j = 0.25;
      float k = 6.25;
      float sin_value = sin(i);
      float cos_value = cos(i);
   printf(“ SIN = %f” , sin_value);
   printf(“ COS = %f” , cos_value);

     return 0;
}
```

OP

The value of sin(0.314000) : 0.308866

The value of cos(0.314000) : 0.951106

--

acos(double x) returns the arc cosine of **x** in radians.

Declaration

Following is the declaration for acos() function.

```
double acos(double x)
```

Parameters

- **x** − This is the floating point value in the interval [-1,+1].

Return Value

This function returns principal arc cosine of x, in the interval [0, pi] radians.

Example

The following example shows the usage of acos() function.

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main () {
   double x, ret, val;

   x = 0.9;
   val = 180.0 / PI;

   ret = acos(x) * val;
   printf("The arc cosine of %lf is %lf degrees", x, ret);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The arc cosine of 0.900000 is 25.855040 degrees
```

---

asin(double x) returns the arc sine of **x** in radians.

Declaration

Following is the declaration for asin() function.

```
double asin(double x)
```

Parameters

- **x** − This is the floating point value in the interval [-1,+1].

Return Value

This function returns the arc sine of x, in the interval [-pi/2,+pi/2] radians.

Example

The following example shows the usage of asin() function.

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main () {
   double x, ret, val;
   x = 0.9;
   val = 180.0 / PI;

   ret = asin(x) * val;
   printf("The arc sine of %lf is %lf degrees", x, ret);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The arc sine of 0.900000 is 64.158067 degrees
```

----

atan(double x) returns the arc tangent of **x** in radians.

Declaration

Following is the declaration for atan() function.

```
double atan(double x)
```

Parameters

- **x** − This is the floating point value.

Return Value

This function returns the principal arc tangent of x, in the interval [-pi/2,+pi/2] radians.

Example

The following example shows the usage of atan() function.

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main () {
   double x, ret, val;
   x = 1.0;
   val = 180.0 / PI;

   ret = atan (x) * val;
   printf("The arc tangent of %lf is %lf degrees", x, ret);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The arc tangent of 1.000000 is 45.000000 degrees
```

----

cos(double x) returns the cosine of a radian angle **x**.

Declaration

Following is the declaration for cos() function.

```
double cos(double x)
```

Parameters

- **x** − This is the floating point value representing an angle expressed in radians.

Return Value

This function returns the cosine of x.

Example

The following example shows the usage of cos() function.

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main () {
   double x, ret, val;

   x = 60.0;
   val = PI / 180.0;
   ret = cos( x*val );
   printf("The cosine of %lf is %lf degrees\n", x, ret);

   x = 90.0;
   val = PI / 180.0;
   ret = cos( x*val );
   printf("The cosine of %lf is %lf degrees\n", x, ret);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
The cosine of 60.000000 is 0.500000 degrees
The cosine of 90.000000 is 0.000000 degrees
```

----

div_t div(int numer, int denom) divides **numer (numerator)** by **denom (denominator)**.

Declaration

Following is the declaration for div() function.

```
div_t div(int numer, int denom)
```

Parameters

- **numer** − This is the numerator.
- **denom** − This is the denominator.

Return Value

This function returns the value in a structure defined in <cstdlib>, which has two members. For div_t:*int quot; int rem;*

Example

The following example shows the usage of div() function.

```
#include <stdio.h>
#include <stdlib.h>

int main () {
   div_t output;

   output = div(27, 4);
   printf("Quotient part of (27/ 4) = %d\n", output.quot);
   printf("Remainder part of (27/4) = %d\n", output.rem);

   output = div(27, 3);
   printf("Quotient part of (27/ 3) = %d\n", output.quot);
   printf("Remainder part of (27/3) = %d\n", output.rem);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Quotient part of (27/ 4) = 6
Remainder part of (27/4) = 3
Quotient part of (27/ 3) = 9
Remainder part of (27/3) = 0
```

----

modf(double x, double *integer) returns the fraction component (part after the decimal), and sets integer to the integer component.

Declaration

Following is the declaration for modf() function.

```
double modf(double x, double *integer)
```

Parameters

- **x** − This is the floating point value.
- **integer** − This is the pointer to an object where the integral part is to be stored.

Return Value

This function returns the fractional part of x, with the same sign.

Example

The following example shows the usage of modf() function.

```
#include<stdio.h>
#include<math.h>

int main () {
   double x, fractpart, intpart;

   x = 8.123456;
   fractpart = modf(x, &intpart);

   printf("Integral part = %lf\n", intpart);
   printf("Fraction Part = %lf \n", fractpart);

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
Integral part = 8.000000
Fraction Part = 0.123456
```

----

sin(double x) returns the sine of a radian angle **x**.

Declaration

Following is the declaration for sin() function.

```
double sin(double x)
```

Parameters

- **x** − This is the floating point value representing an angle expressed in radians.

Return Value

This function returns sine of x.

Example

The following example shows the usage of sin() function.

```
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main () {
   double x, ret, val;

   x = 45.0;
   val = PI / 180;
   ret = sin(x*val);
   printf("The sine of %lf is %lf degrees", x, ret);

   return(0);
}
```

Let us compile and run the above program to produce the following result −

```
The sine of 45.000000 is 0.707107 degrees
```
