FUNCȚII ARITMETICE

Se conțin în <math.h>

Valori absolute

int abs(int x);
Returnează un întreg care reprezintă valoarea absoluta a argumentului.

long int labs(long int x);
Analog cu funcția abs, cu deosebirea ca argumentul si valoarea returnata sunt de tip long int .

double fabs (double x);
Returnează un real care reprezintă valoarea absoluta a argumentului real.

C fabs()

The fabs() function returns the absolute value of a number.

Function Prototype of fabs()

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

The fabs() function takes a single argument (in `double`) and returns the absolute value of that number (also in `double`).

```
[Mathematics] |x| = fabs(x) [In C programming]
```

To find absolute value of an integer or a float, you can explicitly convert the number to double.

```
 int x = 0;

 double result;
 result = fabs(double(x));
```

The fabs() function is defined in math.h header file

Example: C fabs() function

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

int main()
{
    double x, result;

    x = -1.5;
    result = fabs(x);
    printf("|%.2lf| =  %.2lf\n", x, result);

    x = 11.3;
    result = fabs(x);
    printf("|%.2lf| =  %.2lf\n", x, result);

    x = 0;
    result = fabs(x);
    printf("|%.2lf| =  %.2lf\n", x, result);

    return 0;

}
```

Output

```
|-1.50| =  1.50
|11.30| =  11.30
|0.00| =  0.00
```

Funcții de rotunjire

double floor(double x);
Returnează un real care reprezintă cel mai apropiat număr, fără zecimale, mai mic sau egal cu x (rotunjire prin lipsa).

C floor()

The floor() function calculates the nearest integer less than the argument passed.

C floor() Prototype

```
double floor(double arg)
```

The floor() function takes a single argument and returns the value in type `double`.

It is defined in <math.h> header file.

For example:

If 2.3 is passed to floor(), it will return 2.

In order to calculate floor() for long double or float, you can use the following prototype.

```
long double floorl( long double arg );

float floorf( float arg );
```

Example: C floor() function

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

int main()
```

```
{
   double num = -8.33;
   int result;

   result = floor(num);

   printf("Floor integer of %.2f = %d", num, result);
   return 0;
}
```

Output

```
Floor integer of -8.33 = -9
```

double ceil(double x);
Returnează un real care reprezintă cel mai apropiat număr, fără zecimale, mai mare sau egal cu x (rotunjire prin adaos).

C ceil()

The ceil() function computes the nearest integer greater than the argument passed.

C ceil() Prototype

```
double ceil( double arg );
```

The ceil() function takes a single argument and returns a value of type `int`.

For example: If 2.3 is passed to ceil(), it will return 3.

The function is defined in <math.h> header file.

```
long double ceill( long double arg );

float ceilf( float arg );
```

In order to find the ceil() of long double or float, you can use the above prototype.

Example: C ceil() function

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

int main()
{
   double num = 8.33;
   int result;

   result = ceil(num);
   printf("Ceiling integer of %.2f = %d", num, result);

   return 0;
}
```

Output

```
Ceiling integer of 8.33 = 9
```

Funcții trigonometrice

double sin (double x)

Returnează valoarea lui sin(x), unde x este dat in radiani. Numărul real returnat se afla in intervalul [-1, 1].

C sin()

The sin() function returns the sine of a number.

Function Prototype of sin()

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

The sin() function returns the sine of an argument (angle in radians).

```
[Mathematics] sinx = sin(x) [In C Programming]
```

It is defined in math.h header file.

The return value of sin() lies between 1 and -1.

Example: C sin() function

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

int main()
{
    double x;
    double result;

    x = 2.3;
    result = sin(x);
    printf("sin(%.2lf) = %.2lf\n", x, result);

    x = -2.3;
    result = sin(x);
    printf("sin(%.2lf) = %.2lf\n", x, result);

    x = 0;
    result = sin(x);
    printf("sin(%.2lf) = %.2lf\n", x, result);


    return 0;
}
```

Output

```
sin(2.30) = 0.75
sin(-2.30) = -0.75
sin(0.00) = 0.00
```

C sinh()

The sinh() function computes the hyperbolic sine of an argument.

C sinh() Prototype

```
double sinh( double arg );
```

The sinh() function takes a single argument and returns the value of type `double`.

```
[Mathematics] sinhx = sinh(x) [In C programming]
```

It is defined in <math.h> header file.

In order to find the sinh() of long double or float numbers, use the following prototype.

```
long double sinhl( long double arg );

float sinhf( float arg );
```

C sinh() range

The arguments passed to the function sinh() can be any number, either negative or positive.

Example: C sinh() function

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
    double angle = 2.50, result;
    result = sinh(angle);
    printf("Sine hyperbolic of %.2lf (in radians) = %.2lf", angle, result);
    return 0;
}
```

Output

```
Sine hyperbolic of 2.50 (in radians) = 6.05
```

double cos (double x)
Returnează valoarea lui cos(x), unde x este dat in radiani. Numărul real returnat se afla in intervalul [-1, 1].

C cos()

The cos() function computes the cosine of an argument.

C cos() Prototype

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

Function cos() takes a single argument in radians and returns a value in type `double`. The value returned by cos() is always in the range: -1 to 1.

It is defined in <math.h> header file.

```
[Mathematics] cosx = cos(x) [In C Programming]
```

In order to use cos() for floats or long double, you can use the following prototype:

```
long double cosl(long double x);

float cosf(float x);
```

C cos() range

The arguments passed to cos() function can be any value, either negative or positive.

Example: C cos() function

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
   double arg = 30, result;

   // Converting to radian
   arg = (arg * PI) / 180;
   result = cos(arg);

   printf("cos of %.2lf radian = %.2lf", arg, result);

   return 0;
}
```

Output

```
cos of 0.52 radian = 0.87
```

C cosh()

The cosh() function computes the hyperbolic cosine of a number.

cosh() Function Prototype

```
double cosh(double x)
```

The cosh() function takes a single argument (angle in radians) and returns the hyperbolic cosine of that angle as type `double`.

The cosh() function is defined in math.h header file.
In order to find the cosh() of long double or float numbers, you can use the following prototype.

```
long double coshl( long double arg);

float coshf( float arg);
```

Example: C cosh()

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

int main ()
{
    double x, result;

    x = 0.5;
    result = cosh(x);
    printf("Hyperbolic cosine of %lf (in radians) = %lf\n", x, result);

    x = -0.5;
    result = cosh(x);
    printf("Hyperbolic cosine of %lf (in radians) = %lf\n", x, result);

    x = 0;
    result = cosh(x);
    printf("Hyperbolic cosine of %lf (in radians) = %lf\n", x, result);

    x = 1.5;
    result = cosh(x);
    printf("Hyperbolic cosine of %lf (in radians) = %lf\n", x, result);

    return 0;
}
```

Output

```
Hyperbolic cosine of 0.500000 (in radians) = 1.127626

Hyperbolic cosine of -0.500000 (in radians) = 1.127626

Hyperbolic cosine of 0.000000 (in radians) = 1.000000
Hyperbolic cosine of 1.500000 (in radians) = 2.352410
```

double tan(double x)
Returnează valoarea lui tg(x), unde x este dat in radiani.

C tan()

The tan() function returns tangent of the argument passed.

Function Prototype of tan()

```
double tan(double x)
```

The tan() function returns tangent of a number (angle in radians).

```
[Mathematics] tanx = tan(x) [In C Programming]
```

It is defined in math.h header file.

Example: C tan() function

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

int main()
{
    double x;
    double result;

    x = 2.3;
    result = tan(x);
    printf("tan(%.2lf) = %.2lf\n", x, result);
    x = -2.3;
    result = tan(x);
    printf("tan(%.2lf) = %.2lf\n", x, result);

    x = 0;
    result = tan(x);
    printf("tan(%.2lf) = %.2lf\n", x, result);

    return 0;
}
```

Output

```
tan(2.30) = -1.12
tan(-2.30) = 1.12
tan(0.00) = 0.00
```

C tanh()

The tanh() function computes the hyperbolic tangent of an argument.

C tanh() Prototype

```
double tanh( double arg );
```

The tanh() function takes a single argument and returns the value in type `double`.

It is defined in <math.h> header file.

```
[Mathematics] tanhx = tanh(x) [In C programming]
```

In order to find the tanh() of long double or float, you can use the following prototype.

```
long double tanhl( long double arg);

float tanhf( float arg);
```

C tanh() range

The arguments passed to the function tanh() can be any number, either negative or positive.

Example: C tanh() function

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
    double angle = 0.40, result;
    result = tanh(angle);
    printf("Tangent hyperbolic of %.2lf (in radians) = %.2lf", angle, result);
    return 0;
}
```

Output

```
Tangent hyperbolic of 0.40 (in radians) = 0.38
```

Funcții trigonometrice inverse

double asin

Returnează valoarea lui arcsin(x), unde x se afla in intervalul [-1, 1]. Numarul real returnat (in radiani) se afla in intervalul [-pi/2, pi/2].

C asin()

The asin() function returns the arc sine (inverse sine) of a number in radians.

The `asin()` function takes a single argument (1 ≥ x ≥ -1), and returns the arc sine in radians.

The `asin()` function is included in `<math.h>` header file.

---

asin() Prototype

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

To find arc sine of type `int`, `float` or `long double`, you can explicitly convert the type to `double` using cast operator.

```
 int x = 0;

 double result;

 result = asin(double(x));
```

Also, two functions asinf() and asinl() were introduced in C99 to work specifically with type `float` and `long double` respectively.

```
float asinf(float x);

long double asinl(long double x);
```

---

asin() Parameter

The `asin()` function takes a single argument in the range of [-1, +1]. It's because the value of sine is in the range of 1 and -1.

Parameter	Description
double value	Required. A double value between - 1 and +1 inclusive.

---

asin() Return Value

The `asin()` functions returns the value in range of [-π/2, +π/2] in radians. If the parameter passed to the `asin()` function is less than -1 or greater than 1, the function returns NaN (not a number).

Parameter (x)	Return Value
x = [-1, +1]	[-π/2, +π/2] in radians
-1 > x or x > 1	NaN (not a number)

Example 1: asin() function with different parameters

```
#include <stdio.h>
#include <math.h>
int main()
{
    // constant PI is defined
    const double PI =  3.1415926;
    double x, result;

    x =  -0.5;
    result = asin(x);
    printf("Inverse of sin(%.2f) = %.2lf in radians\n", x, result);

    // converting radians to degree
    result = asin(x)*180/PI;
    printf("Inverse of sin(%.2f) = %.2lf in degrees\n", x, result);

    // paramter not in range
    x = 1.2;
    result = asin(x);
    printf("Inverse of sin(%.2f) = %.2lf", x, result);

    return 0;
}
```

Output

```
Inverse of sin(-0.50) = -0.52 in radians
Inverse of sin(-0.50) = -30.00 in degrees
Inverse of sin(1.20) = nan
```

Example 2: asinf() and asinl() function

```
#include <stdio.h>
#include <math.h>
int main()
{
    float fx, fasinx;
    long double lx, ldasinx;

    // arc sinine of type float
    fx = -0.505405;
    fasinx = asinf(fx);

    // arc sinine of type long double
    lx = -0.50540593;
    ldasinx = asinf(lx);

    printf("asinf(x) = %f in radians\n", fasinx);
    printf("asinl(x) = %Lf in radians", ldasinx);

    return 0;
}
```

Output

```
asinf(x) = -0.529851 in radians
asinl(x) = -0.529852 in radians
```

C asinh()

The asinh() function computes the hyperbolic of arc sine of an argument.

C asinh() Prototype

```
double asinh (double x);
```

Function asinh() takes a single argument as double and returns the value in radians.

And, the returned value of asinh() is of type `double`.

For better understanding of asinh():

```
[Mathematics] sinh-1x = asinh(x) [In C programming]
```

Two other functions asinhf() and asinhl() are also present to specifically work with `float` and `long double` respectively.

The `asinh()` function is defined in <math.h> header file.

C asinh() range

The range of argument for asinh() can be any value from negative to positive.

Example: C asinh() function

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
        float num = 8.0;
        double result;
        result = asinh(num);

        printf("Inverse of sinh(%.2f) = %.2f in radians", num, result);
        // Converting radians to degree
        result=(result*180)/PI;
        printf("\nInverse of sinh(%.2f) = %.2f in degrees", num, result);
        return 0;
}
```

Output

```
Inverse of sinh(8.00)=2.78 in radians
Inverse of sinh(8.00)=159.08 in degrees
```

double acos(double x)

Returnează valoarea lui arccos(x), unde x se află în intervalul [-1, 1]. Numărul real returnat se afla in intervalul [0, pi].

C acos()

The acos() function returns the arc cosine (inverse cosine) of a number in radians.

The `acos()` function takes a single argument (1 ≥ x ≥ -1), and returns the arc cosine in radians.

The `acos()` function is included in `<math.h>` header file.

acos() Prototype

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

To find arc cosine of type `int`, `float` or `long double`, you can explicitly convert the type to `double` using cast operator.

```
 int x = 0;

 double result;

 result = acos(double(x));
```

Also, two functions acosf() and acosl() were introduced in C99 to work specifically with type `float` and `long double` respectively.

```
float acosf(float x);

long double acosl(long double x);
```

acos() Parameter

The `acos()` function takes a single argument in the range of [-1, +1]. It's because the value of cosine is in the range of 1 and -1.

Parameter	Description
double value	Required. A double value between - 1 and +1 inclusive.

acos() Return Value

The `acos()` functions returns the value in range of [0.0, π] in radians. If the parameter passed to the `acos()` function is less than -1 or greater than 1, the function returns NaN (not a number).

Parameter (x)	Return Value
x = [-1, +1]	[0, π] in radians
-1 > x or x > 1	NaN (not a number)

Example 1: acos() function with different parameters

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

int main()
{
    // constant PI is defined
    const double PI =  3.1415926;
    double x, result;

    x =  -0.5;
    result = acos(x);
    printf("Inverse of cos(%.2f) = %.2lf in radians\n", x, result);

    // converting radians to degree
    result = acos(x)*180/PI;
    printf("Inverse of cos(%.2f) = %.2lf in degrees\n", x, result);

    // paramter not in range
    x = 1.2;
    result = acos(x);
    printf("Inverse of cos(%.2f) = %.2lf", x, result);

    return 0;
}
```

Output

```
Inverse of cos(-0.50) = 2.09 in radians
Inverse of cos(-0.50) = 120.00 in degrees
Inverse of cos(1.20) = nan
```

Example 2: acosf() and acosl() function

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

```
int main()
{
    float fx, facosx;
    long double lx, ldacosx;

    // arc cosine of type float
    fx = -0.505405;
    facosx = acosf(fx);

    // arc cosine of type long double
    lx = -0.50540593;
    ldacosx = acosf(lx);

    printf("acosf(x) = %f in radians\n", facosx);
    printf("acosl(x) = %Lf in radians", ldacosx);

    return 0;
}
```

Output

```
acosf(x) = 2.100648 in radians
acosl(x) = 2.100649 in radians
```

C acosh()

The acosh() function returns the arc hyperbolic cosine (inverse hyperbolic cosine) of a number in radians.

The `acosh()` function takes a single argument (x ≥ 1), and returns the arc hyperbolic cosine in radians.

The `acosh()` function is included in `<math.h>` header file.

---

acosh() Prototype

```
double acosh(double x);
```

To find arc hyperbolic cosine of type `int`, `float` or `long double`, you can explicitly convert the type to `double` using cast operator.

```
int x = 0;

double result;

result = acosh(double(x));
```

Also, two functions acoshf() and acoshl() were introduced in C99 to work specifically with type `float` and `long double` respectively.

```
float acoshf(float x);

long double acoshl(long double x);
```

acosh() Parameter and Return Value

The `acosh()` function takes a single argument greater than or equal to 1.

Parameter	Description
double value	Required. A double value greater than or equal to 1 (x ≥ 1).

acosh() Return Value

The `acosh()` functions returns a number greater than or equal to 0 in radians. If the argument passed is less than 1 (x < 1), the function returns NaN (not a number).

Parameter (x)	Return Value
x ≥ 1	a number greater than or equal to 0 (in radians)
x < 1	NaN (not a number)

Example 1: acosh() function with different parameters

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

int main()
{
    // constant PI is defined
    const double PI =  3.1415926;
    double x, result;

    x =  5.9;
    result = acosh(x);
    printf("acosh(%.2f) = %.2lf in radians\n", x, result);

    // converting radians to degree
    result = acosh(x)*180/PI;
    printf("acosh(%.2f) = %.2lf in degrees\n", x, result);

    // parameter not in range
    x = 0.5;
    result = acosh(x);
    printf("acosh(%.2f) = %.2lf", x, result);

    return 0;
}
```

Output

```
acosh(5.90) = 2.46 in radians
acosh(5.90) = 141.00 in degrees
acosh(0.50) = nan
```

Example 2: acosh() for INFINITY and DBL_MAX

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

int main()
{
    double x, result;

    // maximum representable finite floating-point number
    x =  DBL_MAX;
    result = acosh(x);
    printf("Maximum value of acosh() in radians = %.3lf\n", result);

    // Infinity
    x = INFINITY;
    result = acosh(x);
    printf("When infinity is passed to acosh(), result = %.3lf\n", result);

    return 0;
}
```

Possible Output

```
Maximum value of acosh() in radians = 710.476
When infinity is passed to acosh(), result = inf
```

Here, `DBL_MAX` defined in `float.h` header file is the maximum representable finite floating-point number. And, `INFINITY` defined in `math.h` is a constant expression representing positive infinity.

Example 3: acoshf() and acoshl() function

```
#include <stdio.h>
#include <math.h>
int main()
{
    float fx, facosx;
    long double lx, ldacosx;

    // arc hyperbolic cosine of type float
    fx = 5.5054;
    facosx = acoshf(fx);

    // arc hyperbolic cosine of type long double
    lx = 5.50540593;
    ldacosx = acoshl(lx);

    printf("acoshf(x) = %f in radians\n", facosx);
    printf("acoshl(x) = %Lf in radians", ldacosx);

    return 0;
}
```

Output

```
acoshf(x) = 2.390524 in radians
acoshl(x) = 2.390525 in radians
```

double atan(double x)

Returnează valoarea lui arctg(x), unde x este dat în radiani. Numărul real returnat se afla in intervalul [0, pi].

C atan()

The atan() function computes the arc tangent of an argument.

C atan() Prototype

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

Function atan() takes a single argument as a double and returns the value in radians.

The returned value of atan() is of type `double`.

For better understanding of atan():

```
[Mathematics] tan-1x = atan(x) [In C programming]
```

It is defined in <math.h> header file.

C atan() range

Library function atan() take any value from negative to positive.

Example: C atan() function

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
    double num = 1.0;
    double result;

    result = atan(num);
    printf("Inverse of tan(%.2f) = %.2f in radians", num, result);

    // Converting radians to degrees
    result = (result * 180) / PI;
    printf("\nInverse of tan(%.2f) = %.2f in degrees", num, result);

    return 0;
}
```

Output

```
Inverse of cos(1.00) = 0.79 in radians
Inverse of cos(1.00) = 45 in degrees
```

C atanh()

The atanh() function returns the arc hyperbolic tangent (inverse hyperbolic tangent) of a number in radians.

The `atanh()` function takes a single argument (-1 ≤ x ≥ 1), and returns the arc hyperbolic tangent in radians.

The `atanh()` function is included in `<math.h>` header file.

---

atanh() Prototype

```
double atanh(double x);
```

To find arc hyperbolic tangent of type `int`, `float` or `long double`, you can explicitly convert the type to `double` using cast operator.

```
 int x = 0;

 double result;

 result = atanh(double(x));
```

Also, two functions atanhf() and atanhl() were introduced in C99 to work specifically with type `float` and `long double` respectively.

```
float atanhf(float x);

long double atanhl(long double x);
```

atanh() Parameter

The `atanh()` function takes a single argument greater than or equal to -1 and less than or equal to 1.

Parameter	Description
double value	Required. A double value greater than or equal to 1 $(-1 \leq x \geq 1)$.

Example 1: atanh() function with different parameters

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

int main()
{
    // constant PI is defined
    const double PI =  3.1415926;
    double x, result;

    x =  -0.5;
    result = atanh(x);
    printf("atanh(%.2f) = %.2lf in radians\n", x, result);

    // converting radians to degree
    result = atanh(x)*180/PI;
    printf("atanh(%.2f) = %.2lf in degrees\n", x, result);

    // parameter not in range
    x = 3;
    result = atanh(x);
    printf("atanh(%.2f) = %.2lf", x, result);
    return 0;
}
```

Output

```
atanh(-0.50) = -0.55 in radians
atanh(-0.50) = -31.47 in degrees
atanh(3.00) = nan
```

double atan2(double y, double x)

Returnează valoarea lui tg(y/x), cu excepția faptului ca semnele argumentelor x si y permit stabilirea cadranului si x poate fi zero. Valoarea returnata se afla in intervalul [-pi, pi]. Daca x si y sunt coordonatele unui punct in plan, funcția returnează valoarea unghiului format de dreapta care unește originea axelor carteziene cu punctul, fata de axa absciselor. Funcția folosește, de asemenea, la transformarea coordonatelor carteziene in coordonate polare.

C atan2()

The atan2() function computes the arc tangent of an argument.

C atan2() Prototype

```
double atan2(double y, double x);
```

Function atan2() takes two arguments: x-coordinate and y-coordinate, and calculate the angle in radians for the quadrant.

For better understanding of atan2():

```
[Mathematics] tan-1(y/x) = atan2(y,x) [In C programming]
```

Two other functions atan2f() and atan2l() are also present in C to specifically work with `float` and `long double` respectively.

The `atan2()` function is defined in <math.h> header file.

C atan2() range

The arguments of atan2() can be any number, either positive or negative.

Example: C atan2() function

```c
#include <stdio.h>
#include <math.h>
#define PI 3.141592654

int main()
{
  double x, y, result;
  y = 2.53;
  x = -10.2;
  result = atan2(y, x);
  result = result * 180.0/PI;

  printf("Tangent inverse for(x = %.1lf, y = %.1lf) is %.1lf degrees.", x, y,
result);
  return 0;
}
```

Output

```
Tangent inverse for(x = -10.2, y = 2.53) is 166.1 degrees.
```

Caution while using atan2()

The value of second argument passed should not be 0. If the second argument passed is 0, the program will not run correctly.

Funcții exponențiale logaritmice

double exp (double x)
long double exp(long double x)
Returnează valoarea lui e la puterea x.

C exp()

The exp() function computes the exponential (Euler's number) raised to the given argument.

C exp() Prototype

```
double exp( double arg );
```

The exp(arg) takes a single argument and returns the value in type `double.`

[Mathematics] e^x = exp(x) [In C programming]

It is defined in <math.h> header file.
In order to calculate the exp() for long double or float, you can use the following prototype

```
long double expl( long double arg);

float expf( float arg);
```

C exp() range

The arguments for exp() can be any value, either negative or positive.

Example: C exp() function

```
#include <stdio.h>
#include <math.h>
int main()
{
  double x = 12.0, result;
  result = exp(x);
  printf("Exponential of %.2lf = %.2lf", x, result);
  return 0;
}
```

Output

```
Enter the value of x to find e^x: 12
Exponential of 12.00 = 162754.79
```

double ldexp(double a, int b); long double ldexpl(long double a, int b)

Returneaza valoarea lui 2 la puterea (a*b).

Function **double ldexp(double x, int exponent)** returns **x** multiplied by 2 raised to the power of **exponent**.

Declaration

Following is the declaration for ldexp() function.

```
double ldexp(double x, int exponent)
```

Parameters

- **x** − This is the floating point value representing the significand.
- **exponent** − This is the value of the exponent.

Return Value

This function returns $x * 2^{exp}$

Example

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

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

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

   x = 0.65;
   n = 3;
   ret = ldexp(x ,n);
   printf("%f * 2^%d = %f\n", x, n, ret);

   return(0);
}
```

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

```
0.650000 * 2^3 = 5.200000
```

double frexp(double x, int *y); long double frexp(long double x, int *y)

Returnează valoarea $x*2^y$ calculând de asemenea si valoarea lui y. Exemplu: Daca x=8, operația k=frexp(x, y), calculează numărul real k, care trebuie înmulțit cu 2^y pentru a primi rezultatul egal cu x=8, determinându-l în același timp și pe y (valoarea puterii la care va trebui ridicata cifra 2). Pentru x=8 și k=frexp(x, y) vom obține următoarele rezultate: y=4, k=0,5; adică $0,5=8/2^4$.

function **double frexp(double x, int *exponent)** return value is the mantissa, and the integer pointed to by **exponent** is the exponent. The resultant value is **x = mantissa * 2 ^ exponent**.

Declaration

Following is the declaration for frexp() function.

```
double frexp(double x, int *exponent)
```

Parameters

- **x** − This is the floating point value to be computed.
- **exponent** − This is the pointer to an **int** object where the value of the exponent is to be stored.

Return Value

This function returns the normalized fraction. If the argument x is not zero, the normalized fraction is **x** times a power of two, and its absolute value is always in the range 1/2 (inclusive) to 1 (exclusive). If **x** is zero, then the normalized fraction is zero and zero is stored in exp.

Example

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

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

int main () {
   double x = 1024, fraction;
   int e;
   fraction = frexp(x, &e);
   printf("x = %.2lf = %.2lf * 2^%d\n", x, fraction, e);
   
   return(0);
}
```

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

```
x = 1024.00 = 0.50 * 2^11
```

double log(double x)
Returnează logaritmul natural al argumentului (ln(x)).

C log()

The log() function computes the natural logarithm of an argument.

C log() Prototype

```
double log( double arg );
```

The log() function takes a single argument and returns a value of type `float`.

[Mathematics] log$_e$x = log(x) [In C programming]

It is defined in <math.h> header file.

In order to find the log() of long double or float numbers, you can use the following prototype.

```
long double logl( long double arg);

float logf(float arg);
```

C log() Arguments

Argument	remarks
arg > 0 (Greater than zero)	Finds the log of the argument

Argument	remarks
arg < 0 (Less thn zero)	Shows run-time error

Example: C log() function

```
#include <stdio.h>
#include <math.h>
int main()
{
    double num = 5.6, result;

    result = log(num);
    printf("log(%.1f) = %.2f", num, result);

    return 0;
}
```

Output

```
log(5.6) = 1.72
```

double log10(double x)
Returnează logaritmul zecimal al argumentului (lg(x)).

C log10()

The log10() function computes the base 10 logarithm of an argument.

C log10() Prototype

```
double log10( double arg );
```

It takes a single argument and returns a value of type `float`.

[Mathematics] $\log_{10}x$ = log10(x) [In C programming]

It is defined in <math.h> header file.

In order to find the log10() of long double or float, use the following prototype.

```
long double log10l( long double arg);

float log10f( float arg);
```

C log10() range

Argument	remarks
arg > 0	Finds the log$_{10}$ of the argument
arg < 0	Shows run-time error.

Example: C log10() function

```
#include <stdio.h>
#include <math.h>
int main()
{
    double num = 4.00, result;

    result = log10(num);
    printf("log10(%.1f) = %.2f", num, result);

    return 0;
}
log(4.0) = 0.60
```

double pow (double baza, double exponent);

C pow()

The pow() function computes the power of a number.

The pow() function takes two arguments (base value and power value) and, returns the power raised to the base number. For example,

```
[Mathematics] $x^y$ = pow(x, y) [In programming]
```

The `pow()` function is defined in `math.h` header file.

C pow() Prototype

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

The first argument is a base value and second argument is a power raised to the base value.

To find the power of `int` or a `float` variable, you can explicitly convert the type to `double` using cast operator.

```
int base = 3;

int power = 5;

pow(double(base), double(power));
```

Example: C pow() function

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

int main()
{
    double base, power, result;

    printf("Enter the base number: ");
    scanf("%lf", &base);

    printf("Enter the power raised: ");
    scanf("%lf",&power);

    result = pow(base,power);

    printf("%.1lf^%.1lf = %.2lf", base, power, result);

    return 0;
}
```

Output

```
Enter the base number: 2.5
Enter the power raised: 3.4
2.5^3.4 = 22.54
```

long double pow(long double baza, long double exponent) Returnează un real care reprezintă rezultatul ridicării bazei la exponent (baza la puterea exponent).

double pow10(int x); float pow10f(int x);
long double pow10l(int x)
Returnează valoarea lui 10 la puterea x.

double fmod(double x, double y); long double fmod(long double x, long double y)
Returnează valoarea restului de la împărțirea lui x la y.

function **double 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
```

double sqrt(double x)

Returnează rădăcina pătrata a argumentului x.

C sqrt()

The sqrt() function computes the square root of a number.

Function prototype of sqrt()

```
double sqrt(double arg);
```

The `sqrt()` function takes a single argument (in `double`) and returns its square root (also in `double`).

```
[Mathematics]  √x = sqrt(x) [In C Programming]
```

The `sqrt()` function is defined in math.h header file.

To find the square root of `int`, `float` or `long double` data types, you can explicitly convert the type to `double` using cast operator.

```
int x = 0;

double result;

result = sqrt(double(x));
```

You can also use the `sqrtf()` function to work specifically with float and `sqrtl()` to work with `long double` type.

```
long double sqrtl(long double arg );

float sqrtf(float arg );
```

Example: C sqrt() Function

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

int main() {
   double number, squareRoot;

   printf("Enter a number: ");
   scanf("%lf", &number);

   // computing the square root
   squareRoot = sqrt(number);

   printf("Square root of %.2lf =  %.2lf", number, squareRoot);

   return 0;
}
```

Output

```
Enter a number: 23.4
```

```
Square root of 23.40 =  4.84
```

C cbrt()

The cbrt() function computes the cube root of a number.

Function prototype of cbrt()

```
double cbrt( double arg );
```

The function cbrt() takes a single argument (in `double`) and returns the cube root (also in `double`).

```
[Mathematics]  ∛x = cbrt(x) [In C Programming]
```

The cbrt() function is defined in math.h header file.

To find the cube root of type `int`, `float` or `long double`, you can explicitly convert the type to `double` using cast operator.

```
int x = 0;

double result;

result = cbrt(double(x));
```

Also, you can use `cbrtf()` function to work specifically with float and `cbrtl()` to work with `long double` type.

```
long double cbrtl(long double arg );

float cbrtf(float arg );
```

Example: C cbrt() Function

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

int main()
{
    double num = 6, cubeRoot;

    cubeRoot =  cbrt(num);
    printf("Cube root of %lf =  %lf", num, cubeRoot);

    return 0;
}
```

Output

```
Cube root of 6.000000 =  1.817121
```

C hypot()

The hypotenuse is the longest side of a right-angled triangle. The hypot() function is used to find hypotenuse when other two sides are provided.

hypot() function Prototype

```
double hypot(double p, double b);
```

`h = √(p2+b2)` in mathematics is equivalent to `h = hypot(p, b);` in C Programming.

The hypot() function is defined in math.h header file.

Example: C hypot() Function

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

int main()
{
    double p, b;
    double hypotenuse;

    p = 5.0;
    b = 12.0;

    hypotenuse = hypot(p, b);

    printf("hypot(%.2lf, %.2lf) = %.2lf", p, b, hypotenuse);

    return 0;
}
```

Output

```
hypot(5.00, 12.00) = 13.00
```

int random(int x)
Returnează o valoare aliatoare in intervalul de la 0 la (x-1); Este necesara includerea bibliotecii <stdlib.h>
int rand(void) Generează un număr alegator în intervalul [0, RAND_MAX]. Este necesara includerea bibliotecii <stdlib.h>; Nu este necesară inițializarea.