CL-1002 Programming Fundamentals	LAB - 04 Introduction of operators and math.h library functions

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
Fall 2021

OPERATORS:

There are many operators in C for manipulating data which include arithmetic Operators, Relational Operators, Logical operators and many more which will be discussed accordingly.

C Arithmetic Operators

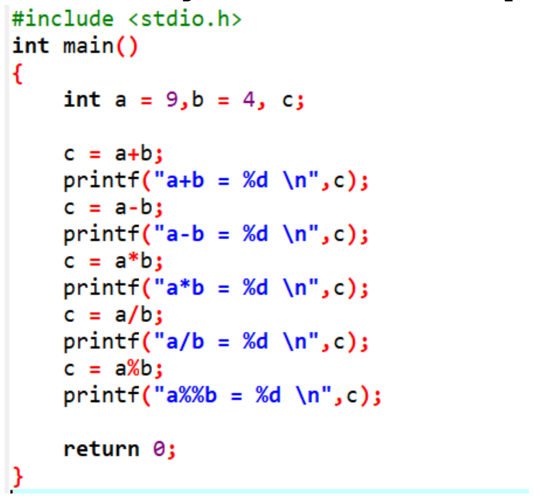
Some of the fundamental operators are:

Operator	Description	Example
+	Adds two operands.	A + B = 30
–	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

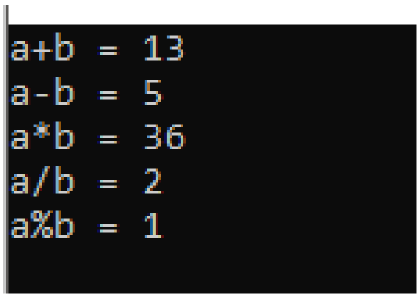
Example code:

```
#include <stdio.h>
int main()
{
   int a = 9,b = 4, c;

   c = a+b;
   printf("a+b = %d \n",c);
   c = a-b;
   printf("a-b = %d \n",c);
   c = a*b;
   printf("a*b = %d \n",c);
   c = a/b;
   printf("a/b = %d \n",c);
   c = a%b;
   printf("a%%b = %d \n",c);

   return 0;
}
```

```
// Working of arithmetic operators
```



```
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);
    c = a-b;
    printf("a-b = %d \n",c);
    c = a*b;
    printf("a*b = %d \n",c);
    c = a/b;
    printf("a/b = %d \n",c);
    c = a%b;
    printf("a%%b = %d \n",c);

    return 0;
}
```



```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
a%b = 1
```

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

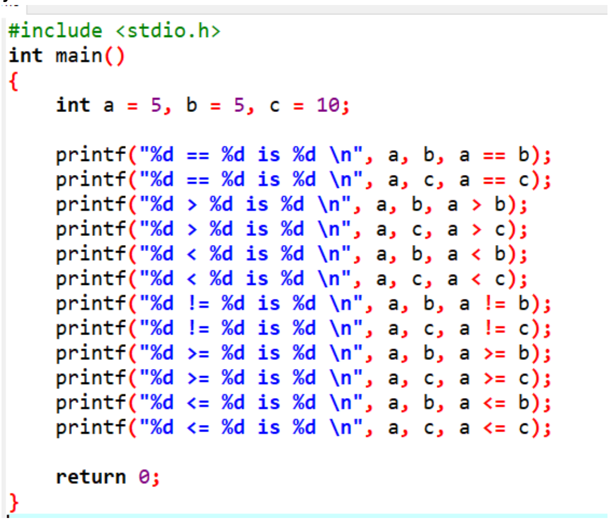
Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

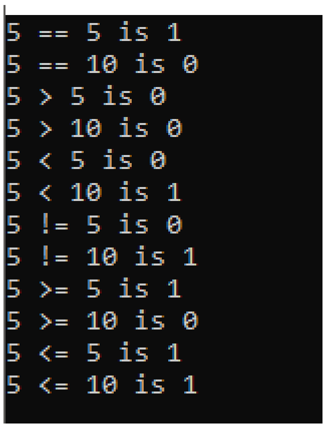
Example code:

```
#include <stdio.h>
int main()
{
   int a = 5, b = 5, c = 10;

   printf("%d == %d is %d \n", a, b, a == b);
   printf("%d == %d is %d \n", a, c, a == c);
   printf("%d > %d is %d \n", a, b, a > b);
   printf("%d > %d is %d \n", a, c, a > c);
   printf("%d < %d is %d \n", a, b, a < b);
   printf("%d < %d is %d \n", a, c, a < c);
```

```
    printf("%d != %d is %d \n", a, b, a != b);
    printf("%d != %d is %d \n", a, c, a != c);
    printf("%d >= %d is %d \n", a, b, a >= b);
    printf("%d >= %d is %d \n", a, c, a >= c);
    printf("%d <= %d is %d \n", a, b, a <= b);
    printf("%d <= %d is %d \n", a, c, a <= c);

    return 0;
}
```



```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;

    printf("%d == %d is %d \n", a, b, a == b);
    printf("%d == %d is %d \n", a, c, a == c);
    printf("%d > %d is %d \n", a, b, a > b);
    printf("%d > %d is %d \n", a, c, a > c);
    printf("%d < %d is %d \n", a, b, a < b);
    printf("%d < %d is %d \n", a, c, a < c);
    printf("%d != %d is %d \n", a, b, a != b);
    printf("%d != %d is %d \n", a, c, a != c);
    printf("%d >= %d is %d \n", a, b, a >= b);
    printf("%d >= %d is %d \n", a, c, a >= c);
    printf("%d <= %d is %d \n", a, b, a <= b);
    printf("%d <= %d is %d \n", a, c, a <= c);

    return 0;
}
```



```
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
5 >= 10 is 0
5 <= 5 is 1
5 <= 10 is 1
```

C Logical Operators

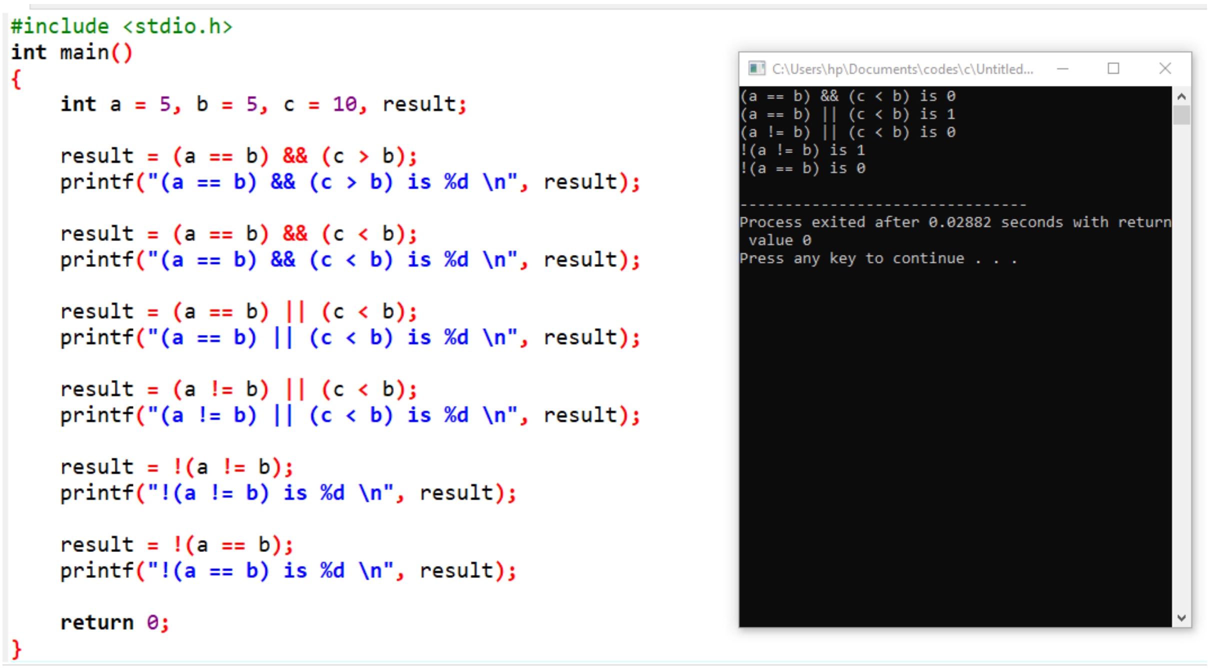
An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression `((c==5) && (d>5))` equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression `((c==5) || (d>5))` equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression `!(c==5)` equals to 0.

Example code:

```
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);

    result = (a == b) && (c < b);
```

```
   printf("(a == b) && (c < b) is %d \n", result);

   result = (a == b) || (c < b);
   printf("(a == b) || (c < b) is %d \n", result);

   result = (a != b) || (c < b);
   printf("(a != b) || (c < b) is %d \n", result);

   result = !(a != b);
   printf("!(a != b) is %d \n", result);

   result = !(a == b);
   printf("!(a == b) is %d \n", result);

   return 0;
}
```



```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) is %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) is %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) is %d \n", result);

    result = !(a != b);
    printf("!(a != b) is %d \n", result);

    result = !(a == b);
    printf("!(a == b) is %d \n", result);

    return 0;
}
```

```
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0

--------------------------------
Process exited after 0.02882 seconds with return
 value 0
Press any key to continue . . .
```

Bitwise Operators C

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power. Bitwise operators are used in C programming to perform bit-level operations.

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

~	Bitwise complement
<<	Shift left
>>	Shift right

```
// C Program to demonstrate use of bitwise operators
#include <stdio.h>
int main()
{
	// a = 5(00000101), b = 9(00001001)
	unsigned char a = 5, b = 9;

	// The result is 00000001
	printf("a = %d, b = %d\n", a, b);
	printf("a&b = %d\n", a & b);

	// The result is 00001101
	printf("a|b = %d\n", a | b);

	// The result is 00001100
	printf("a^b = %d\n", a ^ b);

	// The result is 11111010
	printf("~a = %d\n", a = ~a);

	// The result is 00010010
	printf("b<<1 = %d\n", b << 1);

	// The result is 00000100
	printf("b>>1 = %d\n", b >> 1);
	return 0;
}
```

```
Output: a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>>1 = 4
```

Comma Operator

Comma operators are used to link related expressions together. For example:

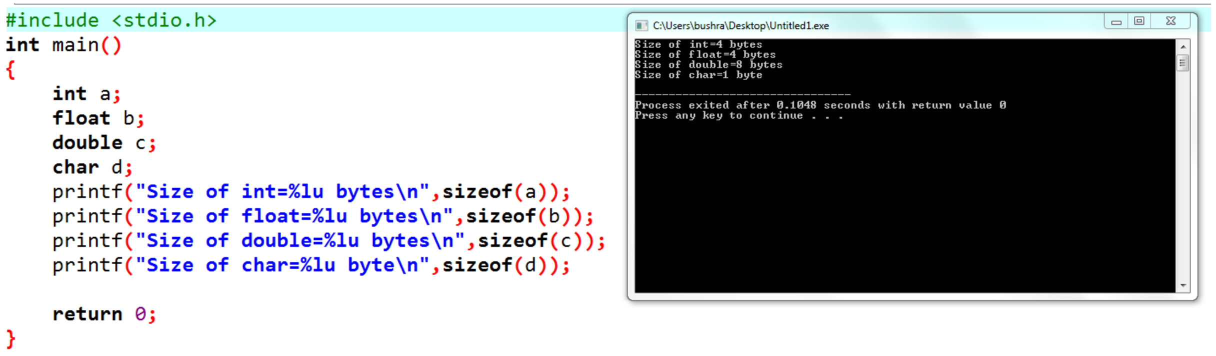
int a, c = 5, d;

The sizeof operator

The sizeof is a unary operator that returns the size of data (constants, variables, array, structure, etc).

Example code:

```
include <stdio.h>
int main()
{
   int a;
   float b;
   double c;
   char d;
   printf("Size of int=%lu bytes\n",sizeof(a));
   printf("Size of float=%lu bytes\n",sizeof(b));
   printf("Size of double=%lu bytes\n",sizeof(c));
   printf("Size of char=%lu byte\n",sizeof(d));

   return 0;
}
```



```
#include <stdio.h>
int main()
{
    int a;
    float b;
    double c;
    char d;
    printf("Size of int=%lu bytes\n",sizeof(a));
    printf("Size of float=%lu bytes\n",sizeof(b));
    printf("Size of double=%lu bytes\n",sizeof(c));
    printf("Size of char=%lu byte\n",sizeof(d));

    return 0;
}
```

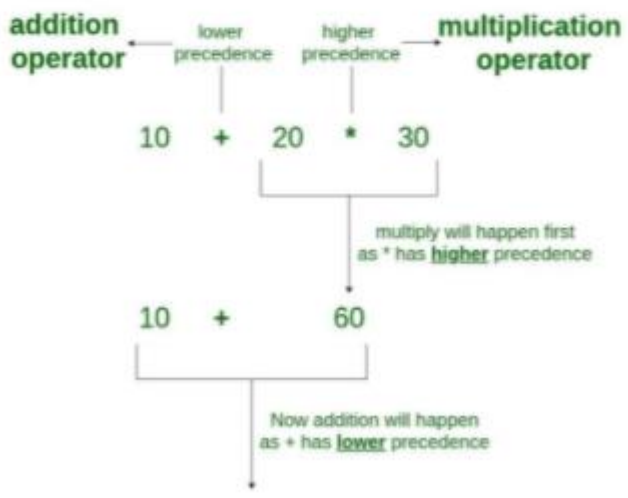
C:\Users\bushra\Desktop\Untitled1.exe

```
Size of int=4 bytes
Size of float=4 bytes
Size of double=8 bytes
Size of char=1 byte

--------------------------------
Process exited after 0.1048 seconds with return value 0
Press any key to continue . . .
```

Operators Precedence in C

- Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence.
- Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated.
- Certain operators have higher precedence than others;
  - **Example**, the multiplication operator has a higher precedence than the addition operator.

Example:



addition operator
lower precedence
higher precedence
multiplication operator
10 + 20 * 30
multiply will happen first as * has higher precedence
10 + 60
Now addition will happen as + has lower precedence

```
70
```

Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either Left to Right or Right to Left.
For example: '*' and '/' have same precedence and their associativity is Left to Right, so the expression "100 / 10 * 10" is treated as "(100 / 10) * 10".

NOTE: in the table given below, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom.

Operator	Description	Associativity
( ) [ ] . -> ++ —	Parentheses Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement	left-to-right
++ — + – ! ~ (*type*) * & **sizeof**	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ –	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right

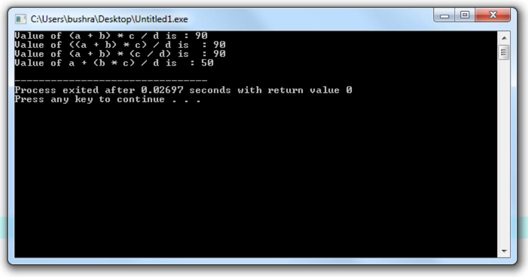
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
| |	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

REFERENCE SITE: https://www.geeksforgeeks.org/operator-precedence-and-associativity-in-c/

Example code:

```
#include <stdio.h>

main() {

  int a = 20;
  int b = 10;
  int c = 15;
  int d = 5;
  int e;

  e = (a + b) * c / d;      // ( 30 * 15 ) / 5
  printf("Value of (a + b) * c / d is : %d\n",  e );

  e = ((a + b) * c) / d;    // (30 * 15 ) / 5
  printf("Value of ((a + b) * c) / d is  : %d\n" ,  e );

  e = (a + b) * (c / d);   // (30) * (15/5)
  printf("Value of (a + b) * (c / d) is  : %d\n",  e );
  e = a + (b * c) / d;     //  20 + (150/5)
  printf("Value of a + (b * c) / d is  : %d\n" ,  e );

  return 0;
}
```

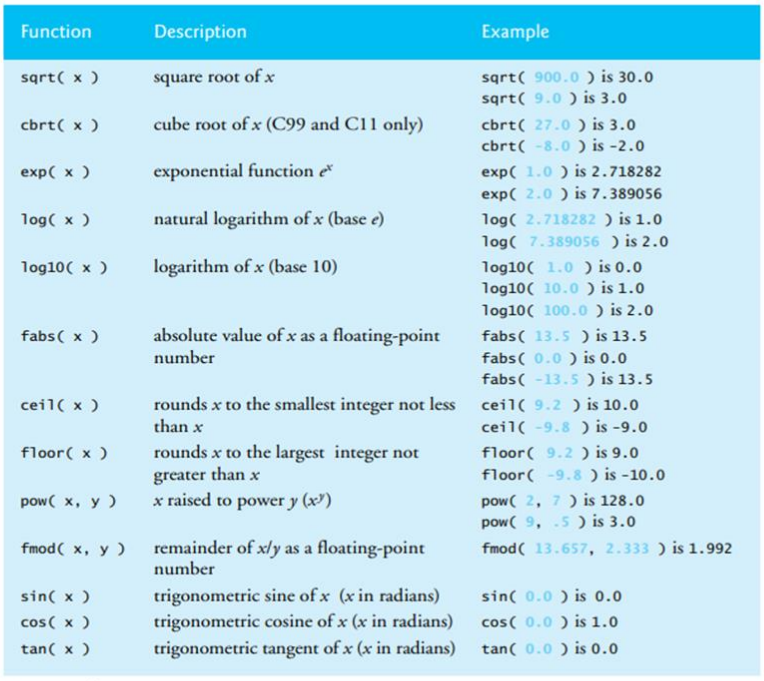
```c
#include <stdio.h>

int main() {

   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;

   e = (a + b) * c / d;
   printf("Value of (a + b) * c / d is : %d\n",  e );

   e = ((a + b) * c) / d;
   printf("Value of ((a + b) * c) / d is  : %d\n" ,  e );

   e = (a + b) * (c / d);
   printf("Value of (a + b) * (c / d) is  : %d\n",  e );

   e = a + (b * c) / d;
   printf("Value of a + (b * c) / d is  : %d\n" ,  e );

   return 0;
}
```



```
Value of (a + b) * c / d is : 90
Value of ((a + b) * c) / d is  : 90
Value of (a + b) * (c / d) is  : 90
Value of a + (b * c) / d is  : 50

--------------------------------
Process exited after 0.02697 seconds with return value 0
Press any key to continue . . .
```

Math library functions

Math library functions allow you to perform certain common mathematical calculations.



Function	Description	Example

`sqrt( x )`	square root of *x*	`sqrt( 900.0 )` is `30.0`
		`sqrt( 9.0 )` is `3.0`

`cbrt( x )`	cube root of *x* (C99 and C11 only)	`cbrt( 27.0 )` is `3.0`
		`cbrt( -8.0 )` is `-2.0`

`exp( x )`	exponential function e^x	`exp( 1.0 )` is `2.718282`
		`exp( 2.0 )` is `7.389056`

`log( x )`	natural logarithm of *x* (base *e*)	`log( 2.718282 )` is `1.0`
		`log( 7.389056 )` is `2.0`

`log10( x )`	logarithm of *x* (base 10)	`log10( 1.0 )` is `0.0`
		`log10( 10.0 )` is `1.0`
		`log10( 100.0 )` is `2.0`

`fabs( x )`	absolute value of *x* as a floating-point number	`fabs( 13.5 )` is `13.5`
		`fabs( 0.0 )` is `0.0`
		`fabs( -13.5 )` is `13.5`

`ceil( x )`	rounds *x* to the smallest integer not less than *x*	`ceil( 9.2 )` is `10.0`
		`ceil( -9.8 )` is `-9.0`

`floor( x )`	rounds *x* to the largest integer not greater than *x*	`floor( 9.2 )` is `9.0`
		`floor( -9.8 )` is `-10.0`

`pow( x, y )`	*x* raised to power *y* (x^y)	`pow( 2, 7 )` is `128.0`
		`pow( 9, .5 )` is `3.0`

`fmod( x, y )`	remainder of *x*/*y* as a floating-point number	`fmod( 13.657, 2.333 )` is `1.992`
`sin( x )`	trigonometric sine of *x* (*x* in radians)	`sin( 0.0 )` is `0.0`
`cos( x )`	trigonometric cosine of *x* (*x* in radians)	`cos( 0.0 )` is `1.0`
`tan( x )`	trigonometric tangent of *x* (*x* in radians)	`tan( 0.0 )` is `0.0`

Fig. 5.2 | Commonly used math library functions.

1	double acos(double x)

	Returns the arc cosine of x in radians.
2	double asin(double x) Returns the arc sine of x in radians.
3	double atan(double x) Returns the arc tangent of x in radians.
4	double atan2(double y, double x) Returns the arc tangent in radians of y/x based on the signs of both values to determine the correct quadrant.
5	double cos(double x) Returns the cosine of a radian angle x.
6	double cosh(double x) Returns the hyperbolic cosine of x.
7	double sin(double x) Returns the sine of a radian angle x.
8	double sinh(double x) Returns the hyperbolic sine of x.
9	double tanh(double x) Returns the hyperbolic tangent of x.
10	double exp(double x) Returns the value of **e** raised to the xth power.
11	double frexp(double x, int *exponent) The returned value is the mantissa and the integer pointed to by exponent is the exponent. The resultant value is x = mantissa * 2 ^ exponent.
12	double ldexp(double x, int exponent) Returns **x** multiplied by 2 raised to the power of exponent.
13	double log(double x) Returns the natural logarithm (base-e logarithm) of **x**.
14	double log10(double x) Returns the common logarithm (base-10 logarithm) of **x**.
15	double modf(double x, double *integer) The returned value is the fraction component (part after the decimal), and sets integer to the integer component.
16	double pow(double x, double y) Returns x raised to the power of **y**.
17	double sqrt(double x) Returns the square root of **x**.

18	double ceil(double x) Returns the smallest integer value greater than or equal to **x**.
19	double fabs(double x) Returns the absolute value of **x**.
20	double floor(double x) Returns the largest integer value less than or equal to **x**.
21	double fmod(double x, double y) Returns the remainder of x divided by **y**

- Functions are normally used in a program by writing the **name of the function** followed by a **left parenthesis** followed by the **argument** (or a comma-separated list of arguments) of the function followed by a **right parenthesis.**
  - For example, to calculate and print the square root of 900.0 you might write When this statement executes,
  - **printf("%.2f", sqrt(900.0));**
  - the math library function sqrt is called to calculate the square root of the number contained in the parentheses (900.0).
  - The number 900.0 is the argument of the sqrt function.
  - The preceding statement would print 30.00.

The sqrt function takes an argument of type double and returns a result of type double.

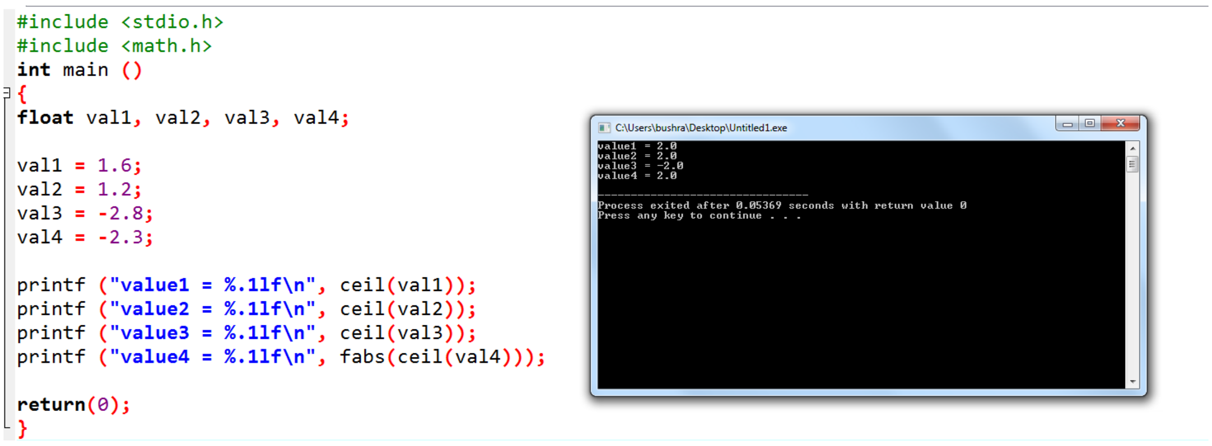
All functions in the math library that return floating-point values return the data type double. Note that double values, like float values, can be output using the %f conversion specification.

Error-Prevention Tip Include the math header by using the preprocessor directive #include when using functions in the math library

REFERENCE SITE: https://www.geeksforgeeks.org/c-library-math-h-functions/

Example code:

```
#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", fabs(ceil(val4)));

return(0);
}
```



```
#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", fabs(ceil(val4)));

return(0);
}
```

C:\Users\bushra\Desktop\Untitled1.exe

```
value1 = 2.0
value2 = 2.0
value3 = -2.0
value4 = 2.0

--------------------------------
Process exited after 0.05369 seconds with return value 0
Press any key to continue . . .
```

TASKS

1. A man wants to go for tour of different countries. He has 10000000 PKR initially, He decided to take a tour of **France, England, Spain, Canada and Japan**. He decided to divide this amount equally, i.e. 2000000 for each country tour. Write a Program to convert the divided amount (which is in Pakistani rupees) into Swiss franc, Pound Sterling, Spanish Euro, Canadian Dollar and Japanese Yen.
2. One of the jobs that Joe Roberts has been given at work is to order special paper for a report for a board meeting. The paper comes in reams of 500 sheets. He always makes five more copies than the number of people that will be there. Joe wants to know how many reams of paper he needs for a meeting. He can order **only whole, not partial**, **reams**. Assume the required number of pages will not equal an exact number of reams. Test your solution using scratch with the following data:

   The report is 140 pages long.

   There will be 25 people at the meeting.
3. Explain the order of evaluation of the operators in each of the following statements, show the value of x after each statement is executed using the C program. Compare your computed answer with the answer shown after execution of C program.

   a= 2, b=4, c=2
   a) a + b / c + b % 2 * a - c
   b) a++*(a/b*c)+b--
   c) a++*(a/b*c)+--b
4. Write a C program that takes temperature in Celsius and converts it into Fahrenheit and Kelvin.
5. Write a C program that takes user year of birth as input and tell him/her his age (in years only) in 2021.

   Sample output:

   ```
   Enter your birth year: 1997
   Your age in 2021 = 24 years
   ```
6. Write a C program to Find out distance, coordinates of midpoint using distance formula, derived from Pythagorean Theorem and value of X by Quadratic formula, as follows:

Distance = √((x2-x1)^2 + (y2-y1)^2)

Midpoint= ((x2+x1/2), (y2+y1/2))

$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ Given (a≠0.)

7. Write c program that find the result of the following operations: you are allowed to initialize any values. Observe and write in comment how that operation produces results.

- True OR False AND NOT True
- “A” > “H”

8. In a cricket match between England and New Zealand, England gave the target of 282 runs. In the first 10 overs of a cricket game, the run rate of New Zealand was only 3.2 runs. Write a C program to calculate the run rate in the remaining 40 overs to reach the target of 282 runs.
9. A shopkeeper sells watches in 45 rupees each. He purchased watches for Rs. 20 per piece. Write a C Program to calculate profit and profit percentage.
10.
   a) 600-meter-long distance was crossed by a man in 5 minutes. Construct a C Program to find out the speed in km per hour.
   b) Calculate the hypogenous for a right-angle triangle in which user inputs base and perpendicular of a triangle. Use functions from Math.h library to calculate square.