Dhaka International University

Knowledge is Power

DIU

Estd. 7th April, 1995

DEPARTMENT OF CSE

LAB REPORT

COURSE NAME : Structured Programming Language Lab

COURSE CODE : 0613-102

REPORT NO : 02

REPORT ON : pow(), ceil(), floor(), trunc() function in c programming

SUBMITTED BY	SUBMITTED TO
NAME : Mir Yeasir Abrar **ROLL** : 35 **REG. NO** : CS-D-98-23-127358 **BATCH** : 98 **SEMESTER** : 1st	Rakib Mahmud Lecturer Department of CSE

DATE OF SUBMISSION : 31 August, 2024

Date of Performance : 24 August, 2024

Title : The 'pow()' function in C programming

Objective : The objective of this lab report is to explore and understand the 'pow()' function in C programming. The 'pow()' function is essential for performing exponentiation operations, which are fundamental in various mathematical and scientific computations. This report aim's to explain the function's 'syntax', 'provide an example code', 'discuss its application', 'limitations', and 'relevence', and offer a comprehensive understanding of its usage.

Introduction : The 'pow()' function is defined in C standard library's 'math.h' header file. It computes the value of a number raised to a specific power. This function is particularly useful in mathematical computations where power operations are required.

- syntax : double pow(double base, double exponent);
  - base - The base number (a 'double' value) that will be raised to the power specified by the exponent.
  - exponent - The exponent (a 'double' value) to which the base number will be raised.

- Return Value : The function returns the result as a 'double'. The result of 'base' raised to the power of 'exponent'.

• code :

```
# include <stdio.h>
# include <math.h>
int main
{
    double base = 2.0;
    double exponent = 3.0;
    double result;

    result = pow (base, exponent);
    printf (" %.2lf raised to the power of %.2lf is
            %.2lf \n", base, exponent, result);

    return 0;
}
```

• Output :

2.00 raised to the power of 3.00 is 8.00

• Explanation :

1. Include the Headers :
   - '#include <stdio.h>' - Includes the standard I/O library for input and output functions.
   - '#include <math.h>' - Includes the math library which contains the 'pow()' function.

2. Declare variables :
   - 'base', 'exponent' and 'result' are declared as 'double' to handle floating-point numbers. which allows for precise calculations.

3. Call 'pow()':

- 'result = pow(base, exponent);' — The 'pow()' function is called with 'base' and 'exponent' as arguments. It computes '2.0' raised to the power of '3.0' and stores the result in the 'result' variable.

4. Print Result:

- 'printf("%.2lf raised to the power of %.2lf is %.2lf\n", base, exponent, result);' — This line prints the result in the console.

In this example, '2.0' raised to the power of '3.0' equals '8.0', which is displayed on the console.

Limitations:

- Integer Precision – While 'pow()' handles floating-point numbers, it may not yield precise results for integer calculations, leading to potential inaccuracies.
- Complex Number – The function does not support complex numbers. For such cases, we have to use libraries like 'complex.h'.
- Extreme Values – For very large or small values, 'pow()' function may return 'INF' (infinity) or 'NAN' (not a number), which could cause errors in calculations.

Applications:

- Scientific Calculations – useful for exponential growth, decay, and other scientific formulas requiring power operations.
- Engineering – Essential for calculating power laws and other engineering related computations.
- Graphics and simulations – Applied in transformations and other mathematical operations involving exponentiation in graphics programming and simulations.
- Algorithms – Facilitates the implemention of algorithms that require power calculations, such as cryptographic functions and numerical methods.

conclusion: The 'pow()' function is a valuable tool in C programming for performing exponentiation. It is widely used in scientific, engineering, and algorithmic contexts. However, it is important to understand its limitations, such as potential precision issue with integers and handling the extreme value, to use it effectively in various applications.

References

- C standard library Documentation
- programiz.com
- scaler.com
- ibm.com
- openai.com
- Math function.pdf (Lab material)

Titel: The 'ceil()' function in C programming

Objective:

To explore and understand the 'ceil()' function in C programming, which rounds up floating-point numbers to the nearest integer. This function is useful in various computational tasks where upward rounding is required.

Introduction:

The 'ceil()' function is provided in the C standard library within the 'math.h' header. It returns the smallest integer value that is greater than or equal to the given floating-point number.

- Syntax – double ceil(double x);
  - x – The floating-point number to be rounded up.
  - Return value – The smallest integer greater than or equal to 'x', returned as a 'double'.

- Code –

```
#include <stdio.h>
#include <math.h>
int main() {
    double num1 = 2.3;
    double num2 = -2.7;
    printf("ceil of %.1lf is %.1lf\n", num1, ceil(num1));
    printf("ceil of %.1lf is %.1lf\n", num2, ceil(num2));
    return 0;
}
```

•output:

ceil of 2.3 is 3.0
ceil of -2.7 is -2.0

Explanation:

- Headers: '#include <stdio.h>' for input/output operations and '#include <math.h>' for mathematical functions like. 'ceil()'.
- Variables: The floating-point numbers 'num1' and 'num2' are used to demonstrate the function.
- Function Call:
  - 'ceil(2.3)' rounds '2.3' up to '3.0'.
  - 'ceil(-2.7)' rounds '-2.7' up to '-2.0'.
- output: The program prints the results, showing how 'ceil()' rounds each number up.

Limitation:

- Exact Integers: Using 'ceil()' on integers is redundant as its designed for floating-point number.
- Upward Rounding Only: 'ceil()' always rounds towards positive infinity. For downward rounding, we use the 'floor' function.
- Returns 'double': Even though it returns as a rounded value of integer, the value is in 'double' format.

Application:

- Mathematical Calculations: Particularly useful when calculations result in a floating-point number, and rounding up to the next whole number is needed.

- Graphics and UI Layouts: Ensures proper placement of elements by rounding up measurements.

- Financial calculations: In scenarios like billing, 'ceil()' helps to rounding up costs to ensure sufficient coverage.

Conclusion:

The 'ceil()' function is an essential rounding tool in C, especially for scenarios where floating-point numbers need to be rounded up to the nearest integers. Its usage spans across various fields such as mathmetics graphics, and finance, making it a versatile function. However understanding its limitation, such as its exclusive upward rounding and returning a 'double' type, is key to effectively applying it in your program.

Reference:

- c standard library Documentation
- programiz.com
- scaler.com
- openai.com
- ibm.com
- math function.pdf (Lab material)

Titel : The 'floor()' Function in C programming

Objective :

To explore and understand the 'floor()' function in C programming, which rounds down floating-point numbers to the nearest integer. This function is essential for tasks where downward rounding is required.

Introduction :

The 'floor()' function is provided in the C standard library within the 'math.h' header. It returns the largest integer value that is less than or equal to to the given floating-point number.

- Syntax - double floor(double x);
- x - The floating point number to be rounded down
- Return value - The largest integer less than or equal to 'x', returned as a 'double'.

- code -

```
#include <stdio.h>
#include <math.h>
int main() {
    double num1 = 3.7;
    double num2 = -3.2;
    printf("Floor of %.1lf is %.1lf\n", num1, floor(num1));
    printf("Floor of %.1lf is %.1lf\n", num2, floor(num2));

    return 0;
}
```

- Output -

```
Floor of 3.7 is 3
Floor of -3.2 is -4
```

Explanation:

- Headers – '#include <stdio.h>' for input/output operations and '#include <math.h>' for mathematical function like floor.

- Variables – The floating-point numbers 'num1' and 'num2' are used to demonstrate the function.

- Function Call –
- floor(3.7) rounds 3.7 down to 3.0
- floor(-3.2) rounds -3.2 down to -4.0

- Output – The program prints the results, illustrating how 'floor()' consistently rounds numbers down.

Limitations:

- Exact Integers: The 'floor()' function is unnecessary for integers, as they do not require rounding.

- Downward rounding only: 'floor()' always rounds towards negative infinity. For upward rounding, we use 'ceil()' function.

- Returns 'double': Although the result is an integer, it is returned in 'double' format.

Applications:

- Mathematical Calculations: Ideal for situations requiring rounding down, such as when calculating the number of whole items that can be produced from a resource

- Financial Calculations: Useful in budgeting where costs are rounded down to ensure estimates do not exceed available funds.

• Array Indexing – Helps in determining the appropriate lower index when dealing with floating-point numbers.

Conclusion:

The 'floor()' function is a valuable tool in C programming for rounding down floating-point numbers to the nearest integers. It is particularly useful in fields such as mathematics, finance, and programming where downward rounding is essential. Understanding its behavior, including its limitation to downward rounding and its return type as a 'double', is crucial for its effective application in various computational tasks.

Reference:

- C Standard Library Documentation
- programiz.com
- scaler.com
- openai.com
- ibm.com
- Math Function.pdf (Lab material)

Titel: The 'trunc()' function in c programming

Objective:

To explore and understand the 'trunc()' function in c programming, which truncates function the decimal part of of a floating-point number, effectively rounding it towards zero. This function is useful when the integer part of a number is needed without rounding up or down.

Introduction:

The 'trunc()' function is part of the c standard library and is included in the 'math.h' header. It discards the fractional part of a floating-point number returning the integer part as a 'double'.

... syntax - double trunc(double x);

- x - The floating point number to be truncated.
- Return value - The integer part of 'x' after truncating the fractional part, returned as a 'double'.

... code -

```
# include <stdio.h>
# include <math.h>
int main (){
  double num1 = 4.8;
  double num2 = -4.8;
  printf("Trunc of %.1lf is %.1lf\n", num1, trunc(num1));
  printf("Trunc of %.1lf is %.1lf\n", num2, trunc(num2));
  return 0;
}
```

... output -

```
Trunc of 4.8 is 4.0
Trunc of -4.8 is -4.0
```

Explanation:

- Headers: '#include <stdio.h>' for input/output operations and '#include <math.h>' for mathematical functions like 'trunc()'.
- Variables: The floating point numbers 'num1' and 'num2' are used to illustrate the function.
- function call
  - 'trunc(4.8)' truncates '4.8' to '4.0'
  - 'trunc(-4.8)' truncates (-4.8) to -4.0
- Returns 'double': Even though the result is an integer, it is returned as a 'double'.
- Output: The program prints the truncated results, demonstrating how 'trunc()' simply removes the fractional part

Limitation:

- Does not round: Unlike 'ceil()' or 'floor()', 'trunc()' does not round numbers, it simply drops the decimal part, making it unsuitable for rounding operations.
- Returns 'double': Eventhough the result is an integer, it is returned as a 'double'.

Application:

- Data processing: useful when only the integer part of a floating-point number is needed, such as in statistical calculations.
- Mathematical operations: Can be applied in scenarios where precision is not required, and only the whole number is of interest.

. Graphics and Game Development : Employed to determine pixel positions or grid place-ments where fractional parts are irrelevent.

Conclusion - The 'trunc()' function in c programming is a straight-forward and effective tool for truncating floating-point numbers by removing their decimal part. This function is particularly useful in situations where rounding is not desired, and the integer part alone is needed. However understand that it does not round but simply cuts off the fractional part is important for its correct application.

Reference -

- c Standard Library Documentation
- ibm.com
- openai.com
- geeksforgeeks.org
- learn.microsoft.com
- Math Function.pdf (Lab material)