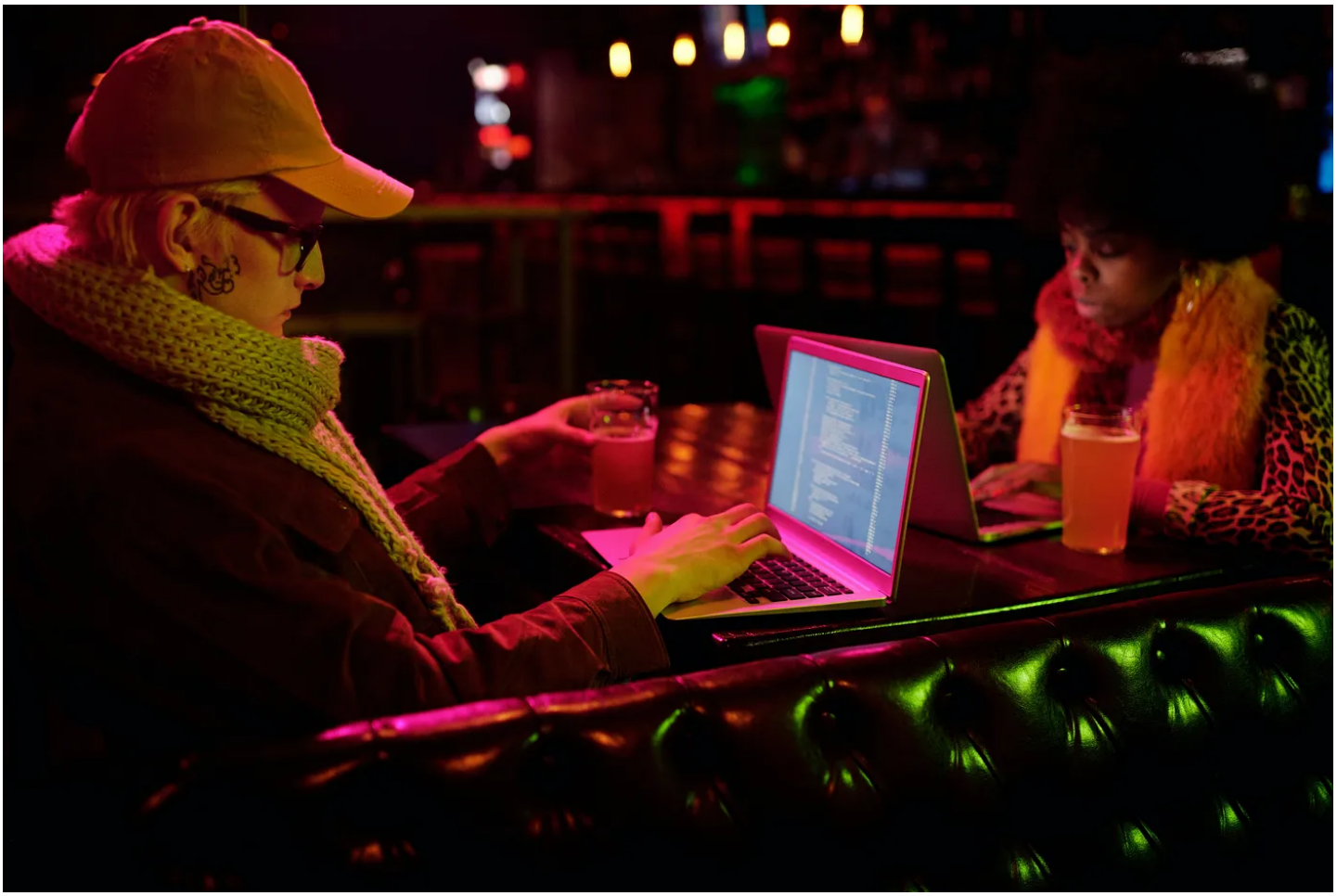
C++ Cheatsheet: Basic Level

Deepak Ranolia · Follow

13 min read · Nov 26, 2023



Welcome to the C++ Cheatsheet designed to enhance your proficiency in C++ programming. This is structured into four levels: Basic, Intermediate, Advanced, and Expert. Each level introduces you to progressively sophisticated C++ concepts and practices. The examples provided in each level are diverse, ensuring a comprehensive understanding of C++.

> ***01 Basic Level***
>
> *The Basic Level covers fundamental C++ concepts, including variable declarations, control structures, and basic functions. It provides a solid foundation for beginners and serves as a refresher for those familiar with C++ basics.*

1 ***Hello World*** The "*Hello, World!*" program in C++ is often the first program written by developers learning a new language. Its simplicity allows beginners to focus on understanding basic syntax, structure, and how to produce program output. When compiled and run, this program prints "Hello, World!" to the console, confirming that the development environment is set up correctly and the basic syntax is understood.

```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Description:

1. `#include <iostream>` : This line is a preprocessor directive that tells the compiler to include the input/output stream library (`iostream`). It provides functionality for handling input and output operations.
2. `int main() { }` : In C++, every program must have a `main` function, which is the entry point of the program. The execution of the program begins from the `main` function.
3. `std::cout << "Hello, World!" << std::endl;` : This line outputs the text "Hello, World!" to the console. `std::cout` is the standard output stream, and `<<` is the stream insertion operator used to output data. `std::endl` inserts a newline character and flushes the output buffer.
4. `return 0;` : Indicates the successful completion of the program. The `0` here is a convention to indicate that the program terminated without errors.

2 ***Variables & Input*** In C++, a variable is a named storage location that can hold data. Variables must be declared before they are used, specifying their data type. In your provided code:

```
#include <iostream>

int main() {
    int age;
    std::cout << "Enter your age: ";
    std::cin >> age;
```

```
    std::cout << "Your age is: " << age << std::endl;
    return 0;
}
```

Description:

1. `int age;` : Declares an integer variable named `age`.
2. `std::cout << "Enter your age: ";` : Outputs the prompt "Enter your age: " to the console.
3. `std::cin >> age;` : Accepts input from the user and stores it in the variable `age`. The `>>` operator is used with `std::cin` for input.
4. `std::cout << "Your age is: " << age << std::endl;` : Outputs the text "Your age is: " followed by the value stored in the `age` variable to the console.
5. `return 0;` : Indicates successful program execution.

Note: So, when you run this program, it prompts the user to enter their age, reads the input, and then displays “Your age is: “ followed by the entered age. The user’s input is stored in the `age` variable, allowing the program to use and display it.

3 *Arithmetic operations* involve performing mathematical calculations on numeric data types. The common arithmetic operations include addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`). Your

provided code demonstrates the use of these operations:

```
#include <iostream>

int main() {
    int a = 5, b = 3;

    // Addition
    std::cout << "Sum: " << a + b << std::endl;

    // Subtraction
    std::cout << "Difference: " << a - b << std::endl;

    // Multiplication
    std::cout << "Product: " << a * b << std::endl;

    // Division
    std::cout << "Quotient: " << a / b << std::endl;

    return 0;
}
```

Description:

1. `int a = 5, b = 3;` : Initializes two integer variables, `a` and `b`, with values 5 and 3, respectively.
2. `std::cout << "Sum: " << a + b << std::endl;` : Calculates and outputs the sum of `a` and `b`.
3. `std::cout << "Difference: " << a - b << std::endl;` : Calculates and outputs the difference of `a` and `b`.

4. `std::cout << "Product: " << a * b << std::endl;` : Calculates and outputs the product of `a` and `b`.
5. `std::cout << "Quotient: " << a / b << std::endl;` : Calculates and outputs the quotient of `a` divided by `b`.

Note: The program demonstrates basic arithmetic operations, displaying the results for addition, subtraction, multiplication, and division. The values are calculated and then printed to the console. Note that in C++, integer division truncates the decimal part, so the result of `a / b` in this case would be `1`.

4 *Conditional statements* in C++ allow you to control the flow of your program based on certain conditions. The most common conditional statement is the `if` statement, which evaluates a condition and executes a block of code if the condition is true. The `else if` and `else` statements can be used to specify additional conditions.

```
#include <iostream>

int main() {
    int num;

    // Prompt user to enter a number
    std::cout << "Enter a number: ";
    std::cin >> num;

    // Check if the entered number is greater than 0
    if (num > 0) {
        std::cout << "Positive number." << std::endl;
    }
```

```
    // If not, check if the number is less than 0
    else if (num < 0) {
        std::cout << "Negative number." << std::endl;
    }
    // If neither condition is true, the number must be 0
    else {
        std::cout << "Zero." << std::endl;
    }

    return 0;
}
```

Description:

1. `int num;` : Declares an integer variable named `num` to store the user-inputted number.
2. `std::cout << "Enter a number: ";` : Outputs a prompt asking the user to enter a number.
3. `std::cin >> num;` : Takes user input and stores it in the `num` variable.
4. `if (num > 0)` : Checks if `num` is greater than 0.
5. `std::cout << "Positive number." << std::endl;` : If the condition in (4) is true, this line is executed, indicating that the number is positive.
6. `else if (num < 0)` : If the condition in (4) is false, checks if `num` is less than 0.
7. `std::cout << "Negative number." << std::endl;` : If the condition in (6) is true, this line is executed, indicating that the number is negative.

8. `else` : If neither of the above conditions is true, this block is executed.
9. `std::cout << "Zero." << std::endl;` : Outputs that the number is zero.

Note: This program demonstrates the use of conditional statements to determine whether a given number is positive, negative, or zero based on user input.

5 ***Loops — While loop*** in C++ are used to repeatedly execute a block of code until a certain condition is met. The `while` loop is one such construct, where the code within the loop is executed as long as the specified condition is true.

```
#include <iostream>

int main() {
    int i = 1;

    // The while loop continues as long as the condition (i <= 5) is true
    while (i <= 5) {
        // Output the current value of i followed by a space
        std::cout << i << " ";

        // Increment the value of i
        i++;
    }

    // Output a newline character after the loop
    std::cout << std::endl;

    // Return 0 to indicate successful execution
    return 0;
}
```

Description:

1. `int i = 1;` : Initializes an integer variable `i` with the value 1. This variable will be used as a counter in the loop.
2. `while (i <= 5)` : This is the condition for the `while` loop. The loop will continue executing as long as the value of `i` is less than or equal to 5.
3. `std::cout << i << " ";` : Outputs the current value of `i` followed by a space.
4. `i++;` : Increments the value of `i` by 1 in each iteration of the loop.
5. The loop repeats steps 3–4 until the condition in (2) becomes false.
6. `std::cout << std::endl;` : Outputs a newline character after the loop, creating a new line in the console.
7. `return 0;` : Indicates successful program execution.

Note: This specific program demonstrates the use of a `while` loop to output the numbers from 1 to 5, separated by spaces. The loop continues until the value of `i` exceeds 5.

6 ***Loops-For Loop*** The `for` loop in C++ is another type of loop that is commonly used for iterative tasks. It consists of three parts: initialization, condition, and iteration expression.

```
#include <iostream>
```

```
int main() {
    // The for loop is initialized with int i = 1;
    // The loop continues as long as the condition i <= 5 is true
    // The expression i++ is executed after each iteration
    for (int i = 1; i <= 5; i++) {
        // Output the current value of i followed by a space
        std::cout << i << " ";
    }

    // Output a newline character after the loop
    std::cout << std::endl;

    // Return 0 to indicate successful execution
    return 0;
}
```

Description:

1. `for (int i = 1; i <= 5; i++)` : This is the `for` loop construct.

- `int i = 1;` : Initializes an integer variable `i` with the value 1.
- `i <= 5` : This is the condition for the loop. The loop will continue executing as long as the value of `i` is less than or equal to 5.
- `i++` : This expression is executed after each iteration of the loop, incrementing the value of `i` by 1.

1. `std::cout << i << " ";` : Outputs the current value of `i` followed by a space in each iteration.
2. The loop repeats steps 2 (output) and 1 (increment) until the condition

in the `for` statement becomes false.

3. `std::cout << std::endl;` : Outputs a newline character after the loop, creating a new line in the console.
4. `return 0;` : Indicates successful program execution.

Note: This specific program demonstrates the use of a `for` loop to output the numbers from 1 to 5, separated by spaces. The loop initialization, condition, and iteration are all specified within the `for` statement.

7 *Arrays* In C++, an array is a collection of elements of the same data type stored in contiguous memory locations. Each element in the array is identified by an index or a subscript. The first element is at index 0, the second at index 1, and so on.

Here are some key points about arrays in C++:

Declaration: To declare an array, you specify the data type of its elements, followed by the array name and the size of the array in square brackets. For example:

```
int numbers[5];
// Declares an integer array with a size of 5
```

Initialization: You can initialize an array at the time of declaration by providing a comma-separated list of values enclosed in curly braces:

```
int numbers[5] = {1, 2, 3, 4, 5};
// Initializes an array with values 1, 2, 3, 4, and 5
```

Accessing Elements: Elements in an array are accessed using square brackets and the index of the element. Array indices start from 0. For example:

```
int thirdElement = numbers[2];
// Accesses the third element (index 2) of the array
```

Size: The size of an array is fixed at the time of declaration and cannot be changed during the program's execution.

Contiguous Memory: The elements in an array are stored in contiguous memory locations. This property allows for efficient memory access and iteration.

Looping Through Arrays: You can use loops, such as `for` or `while`, to iterate through the elements of an array.

```cpp
#include <iostream>

int main() {
    // Declare an integer array named 'numbers' with a size of 5 and initialize
    int numbers[5] = {1, 2, 3, 4, 5};

    // Output the third element (index 2) of the array.
    std::cout << "Third element: " << numbers[2] << std::endl;

    // Return 0 to indicate successful execution.
    return 0;
}
```

Description:

1. `int numbers[5] = {1, 2, 3, 4, 5};` : Declares an array named `numbers` that can hold integers. It is initialized with values 1, 2, 3, 4, and 5. The size of the array is specified as 5.
2. `std::cout << "Third element: " << numbers[2] << std::endl;` : Outputs the text "Third element: " followed by the value stored in the third element of the array. In C++, array indices start from 0, so `numbers[2]` refers to the third element.
3. `return 0;` : Indicates successful program execution.

Note: In this program, the array `numbers` is created, and the third element (index 2) is accessed and printed to the console. Arrays in C++ provide a way to store multiple values of the same data type in a single variable.

8 ***Functions*** In C++, a function is a reusable block of code that performs a specific task. Functions help in modularizing code, making it easier to understand, maintain, and debug.

```
#include <iostream>

int add(int a, int b) {
    return a + b;
}

int main() {
    std::cout << "Sum: " << add(3, 4) << std::endl;
    return 0;
}
```

Description:

- `add` is a function that takes two parameters (`a` and `b`) of type `int` and returns an `int` .
- Inside the function body, it calculates the sum of `a` and `b` using the `+` operator and returns the result.
- The `main` function is the entry point of the program.
- It calls the `add` function with arguments `3` and `4` .
- The result of the `add` function is printed to the console using `std::cout` .
- `std::endl` is used to insert a newline character and flush the output

buffer.

- The program calculates the sum of 3 and 4 using the `add` function and prints the result, which is `7`.

Note: In summary, the code defines a simple function `add` that adds two integers and demonstrates its usage in the `main` function to calculate and print the sum of 3 and 4.

9 ***Strings*** In C++, a `std::string` is a standard library class that represents a sequence of characters. It provides a convenient way to work with text data.

```
#include <iostream>
#include <string>

int main() {
    std::string greeting = "Hello, ";
    std::string name = "John";
    std::cout << greeting + name << std::endl;
    return 0;
}
```

Description:

- Two `std::string` variables, `greeting` and `name`, are declared and initialized with strings.

- `greeting` is assigned the value "Hello, ".
- `name` is assigned the value "John".
- The `+` operator is used for string concatenation. It combines the contents of `greeting` and `name`.
- The result, “Hello, John”, is printed to the console using `std::cout`.
- `std::endl` is used to insert a newline character and flush the output buffer.
- The program concatenates the greeting and the name and prints the combined string.

Note: In summary, the code demonstrates the use of `std::string` for handling strings in C++. It initializes two string variables, concatenates them, and prints the result.

10 *Pointers* In C++, a pointer is a variable that stores the memory address of another variable.

```
#include <iostream>

int main() {
    int num = 42;
    int *ptr = &num;
    std::cout << "Value at pointer: " << *ptr << std::endl;
    return 0;
}
```

Description:

- An integer variable `num` is declared and assigned the value `42`.
- A pointer variable `ptr` is declared, which is capable of storing the memory address of an integer.
- `&num` retrieves the memory address of the variable `num`, and this address is assigned to the pointer `ptr`.
- The `*` operator is used to dereference the pointer, accessing the value stored at the memory address it points to.
- The value at the memory address pointed to by `ptr` (which is the value of `num`, i.e., `42`) is printed to the console.
- The program prints the value stored at the memory address pointed to by the pointer.

Note: In summary, the code demonstrates the concept of pointers in C++, showing how to declare a pointer, assign it the memory address of a variable, and then use it to access the value stored at that memory address.

11 ***Structures*** In C++, a structure is a user-defined data type that allows you to group together variables of different data types under a single name.

```
#include <iostream>
```

```
// Define a structure named Point
struct Point {
    int x; // Member variable for x-coordinate
    int y; // Member variable for y-coordinate
};

int main() {
    // Declare an instance of the Point structure
    Point p;

    // Assign values to the member variables of the Point structure
    p.x = 3;
    p.y = 5;

    // Print the values of the Point structure
    std::cout << "Point: (" << p.x << ", " << p.y << ")" << std::endl;

    return 0;
}
```

Description:

- A structure named `Point` is defined, which contains two member variables: `x` for the x-coordinate and `y` for the y-coordinate.
- An instance of the `Point` structure is declared using the variable name `p`.
- Values are assigned to the member variables of the `Point` structure for the instance `p`.
- The values of the `Point` structure (`p.x` and `p.y`) are printed to the console in the format "Point: (x, y)".

- The program outputs the coordinates of the point stored in the `Point` structure.

Note: In summary, the code demonstrates the use of structures in C++ to represent a point with x and y coordinates. It shows how to define a structure, declare an instance of it, assign values to its members, and print the structure's values.

12 *Enums*, short for enumerations, are a user-defined data type in C++ used to assign names to integral constants, making the code more readable and maintainable. Enums create a set of named integer values, and each name corresponds to a unique integer value. The elements of an enum are listed inside curly braces.

```
#include <iostream>

// Define an enumeration named Day
enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY };

int main() {
    // Declare a variable 'today' of type 'Day' and assign the value 'WEDNESDAY
    Day today = WEDNESDAY;

    // Print the value of 'today'
    std::cout << "Today is day " << today << std::endl;

    return 0;
}
```

Description:

- An enumeration named `Day` is defined, listing the days of the week.
- A variable named `today` of type `Day` is declared and assigned the value `WEDNESDAY`.
- The value of the `today` variable is printed to the console.
- The program outputs the assigned integer value (`WEDNESDAY` is assigned the value 3) for the `today` variable.

Note: In summary, the code demonstrates the use of enums in C++ to represent days of the week. The enum is defined, a variable is declared with an enum type, and the value of the enum variable is printed. The output reflects the integer value assigned to the enum constant `WEDNESDAY`.

13 ***File handling-Writing*** is an essential aspect of programming, allowing you to interact with files on your system. In C++, the `<fstream>` header provides classes and functions for file handling. Specifically, `ofstream` (output file stream) is used for writing to files.

```
#include <iostream>
#include <fstream>

int main() {
    // Create an output file stream and open a file named "example.txt"
    std::ofstream file("example.txt");

    // Write the string "Hello, File!" to the file
```

```
    file << "Hello, File!";

    // Close the file
    file.close();

    return 0;
}
```

Description:

- An object of type `std::ofstream` named `file` is created.
- The constructor is used to open a file named “example.txt” in output mode. If the file doesn’t exist, it will be created; if it exists, its contents will be truncated.
- The string “Hello, File!” is written to the file using the stream insertion (`<<`) operator.
- The `close` method is called to close the file after writing.
- Closing the file is important to ensure that all data is flushed and the file resources are released.
- The program returns 0 to indicate successful execution.

Note: In summary, the code demonstrates how to open a file for writing using `ofstream`, write a string to the file, and then close the file. After running this program, you will find a file named "example.txt" in the same directory as your executable, containing the text "Hello, File!".

14 *File Handling-Reading*

```cpp
#include <iostream>
#include <fstream>
#include <string>

int main() {
    std::ifstream file("example.txt");
    std::string content;
    getline(file, content);
    std::cout << "File Content: " << content << std::endl;
    file.close();
    return 0;
}
```

15 *Dynamic Memory Allocation*

```cpp
#include <iostream>

int main() {
    int *arr = new int[5];
    for (int i = 0; i < 5; i++) {
        arr[i] = i + 1;
    }
    std::cout << "Dynamic Array: ";
    for (int i = 0; i < 5; i++) {
        std::cout << arr[i] << " ";
    }
    delete[] arr;
    std::cout << std::endl;
    return 0;
}
```

This concludes the Basic level of the C++ cheatsheet.

Note: Please check out my C++ cheatsheet if you have not read it yet.

01 c++ Cheatsheet: Basic Level

02 c++ Cheatsheet: Intermediate Level

03 c++ Cheatsheet: Advanced Level

04 c++ Cheatsheet: Expert Level

C Programming C Cheatsheet Cplusplus

Written by Deepak Ranolia

539 Followers · 12 Following

Strong technical skills, such as Coding, Software Engineering, Product Management & Finance. Talk about finance, technology & life
https://rb.gy/9tod91

Follow

Responses (1)

What are your thoughts?

Respond

Md Shoriful Islam Ashiq
6 months ago

Good article. Information 🔥

Reply

More from Deepak Ranolia

Deepak Ranolia

Understanding Proxychains4.conf

Deepak Ranolia

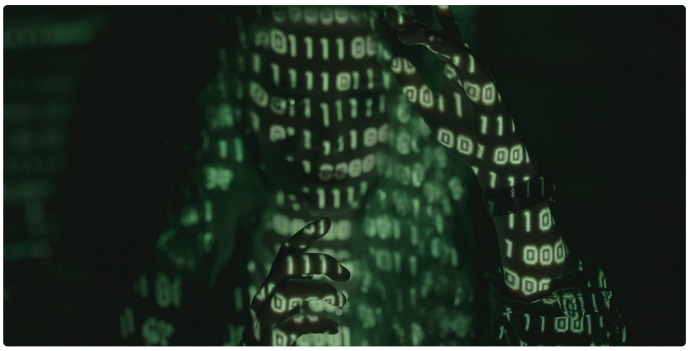
Step-by-Step Guide: Creating a

Proxychains is a powerful tool that enables users to run any application through a prox...

Nov 13, 2023 19

Step 1: Install Prerequisites

Dec 4, 2023 23



Deepak Ranolia

Nikto: Scanning Web Servers for Vulnerabilities

In an increasingly digitized world, web servers have become the backbone of...

Nov 5, 2023 2

Deepak Ranolia

C++ Cheatsheet: Advanced Level

Welcome to the C++ Cheatsheet designed to enhance your proficiency in C++...

Dec 5, 2023 115 2

See all from Deepak Ranolia

Recommended from Medium

In Nerd For Tech by Mohit Mishra

Implementing Shared Pointers in C for Robust Memory Management

C Can Be Smart Too, Taking Control of Memory

Oct 24 52 1

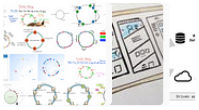
Linking

Advanced Techniques in C Programming—2

23. Using Macros for Code Reusability

Oct 26

Lists

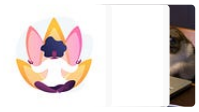


General Coding Knowledge

20 stories · 1807 saves

Coding & Development

11 stories · 935 saves

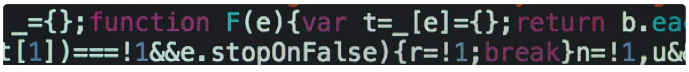


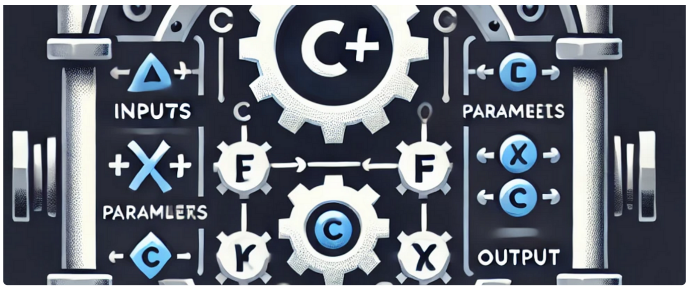
Stories to Help You Grow as a Software Developer

19 stories · 1517 saves

ChatGPT

21 stories · 908 saves



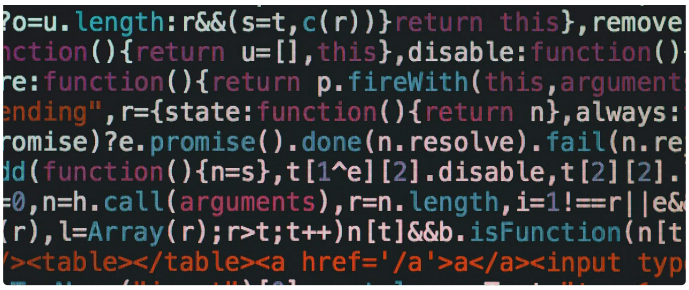


md

Lecture 3: C++ Functions

To create something in C++, you need to use functions. A function is a specific operation...

Dec 3 50 1



Code Link

10 Concepts that you Should know in C Programming

C programming is a foundational language in computer science and software...

Aug 11 2 1

Keith McNulty

Modeling the Movement of Projectiles

What happens when an object is launched into the air and acted upon only by gravity?

4d ago 447 12

anzhi zhu

100 C++ interview questions and answers

Below are 100 C++ interview questions, each with simple answers. Please note that these...

Jul 18 13 1