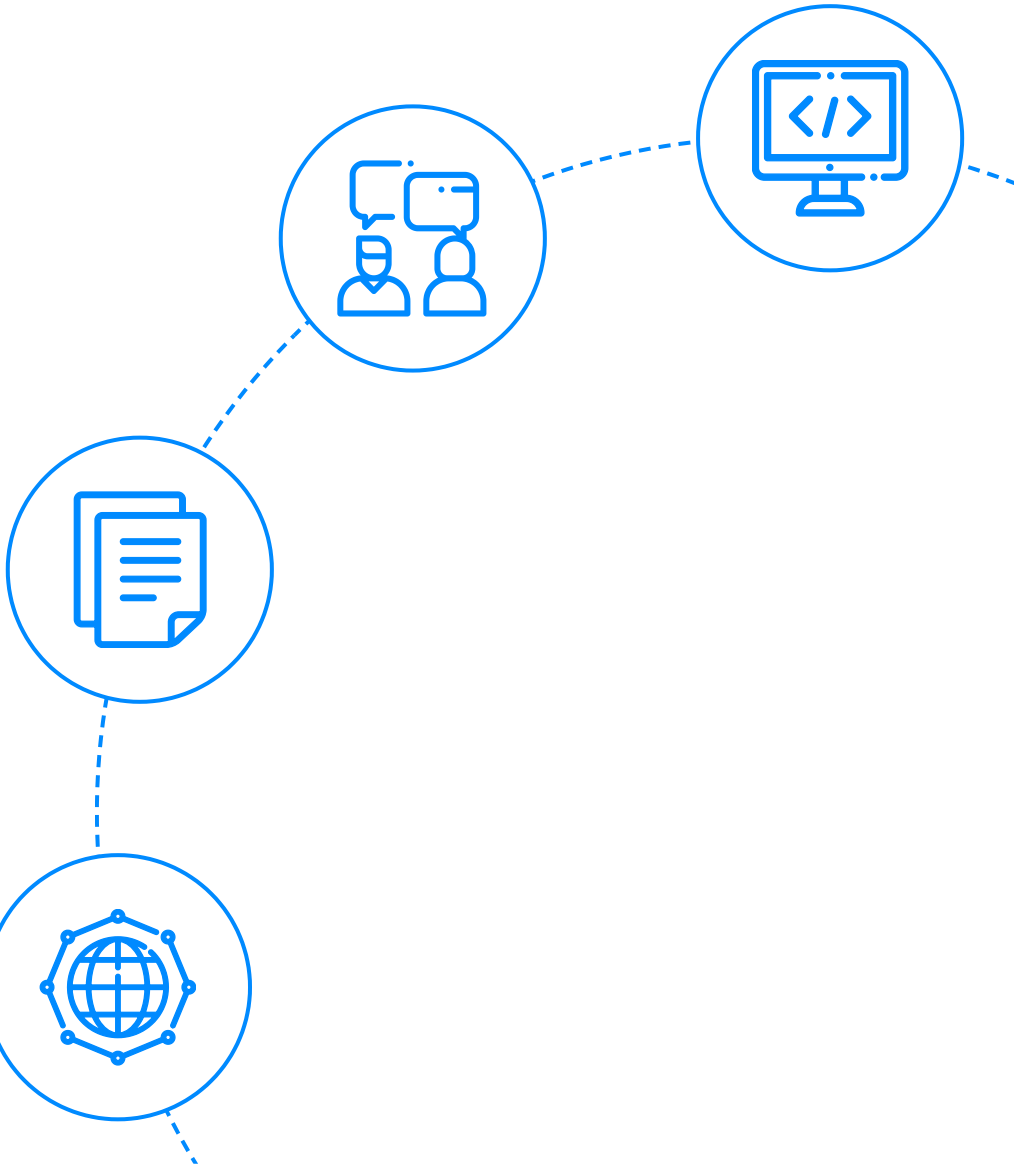
InterviewBit

C++ Cheat Sheet



To view the live version of the page, click here.

Contents

Learn C++ Programming: Basics to Advanced

Let's get Started

It is never easy to learn a new language. You must comprehend a large number of concepts, words, and phrases. It's difficult not to become overwhelmed by the amount of information available. This is why a beginner's cheat sheet is quite useful! You can end up jumping back and forth between different pages or closing one tab after another without realizing it if you don't have it. However, using this C++ cheat sheet, you can finally begin learning C++ quickly. From basic ideas like data types, loops, and functions to more complicated features like strings, pointers, algorithms, and more, everything you need is there at your fingertips. **Learn More**.

But before you jump into the coding section, you must have a little understanding of the language you are going to learn. C++ is a general-purpose programming language that was developed in 1979 by Bjarne Stroustrup. It is currently ranked as the **second most popular programming language** in use today and is used in both small and large programs to create any type of application. Its efficient and powerful syntax can handle both procedural and object-oriented styles of coding, making it a viable solution for any project. C++ can be compiled on any system as its compiler comes with the package and doesn't need any other tools for compilation or linking. This makes C++ a portable language that can be used on any operating system or hardware platform. I guess you're warm enough for the drill now.

Learn C++ Programming: Basics to Advanced

1. Basic Syntax

```
#include <iostream>
using namespace std;
// main() is where program execution begins.
int main() {
 // This is where you write your code
   return 0;
}
```

This is the basic structure of C++. You will have to use this structure almost every time you write a C++ code. It contains the main function where the program execution begins.

2. Output

```
cout << "Hello World";
```

cout prints anything under the “ ” to the screen.

3. Input

int variable;

```
cin >> variable;
```

cin takes the input from the screen and stores it in the variable.

4. Comments

```
//This is a comment
/* C++ comments can also
   * span multiple lines
*/
```

Anything after // is ignored by the compiler. Multiple lines can be commented using /*.....*/.

5. Conditions and If Statements

- If statement

```
if (condition) {
// This block of code will get executed if the condition is True
}
```

If statement belongs to the category of decision-making statements. These statements make decisions based on a condition. If the condition in the condition block is true, the statements in the curly brackets { } are executed. Let's see the example given below.

```
if(2<3){
cout << "2 is less than three";
}
```

- If-else statement

If-else is an extension of the if statement. If the conditions provided with if are not true, the statements in the else block are executed.

```
if(2>3){
cout<< "2 is greater than 3";
}
else{
cout<< "3 is greater than 2";
}
```

- else if

if can be paired with else if for additional conditions.

```
if(2>3){
cout<< "2 is greater than 3";
}
else if(2==3){
cout<< "2 is equal to 3";
}
else{
cout<< "3 is greater than 2";
}
```

- Switch case

```
switch (grade) {
 case 9:
   cout << "Freshman\n";
   break;
 case 10:
  cout << "Sophomore\n";
   break;
 case 11:
   cout << "Junior\n";
   break;
 case 12:
   cout << "Senior\n";
   break;
 default:
   cout << "Invalid\n";
   break;
}
```

A switch statement allows you to test an expression against a variety of cases. If a match is found, the code within begins to run. A case can be ended with the break keyword. When no case matches, default is used.

6. Data Types

- Char

```
char variable_name= 'c';
```

This stores characters such as ‘A’, ‘a’, ‘1’ etc. It takes 1 byte of the system’s memory.

- Integer

```
int variable_name = 123;
```

This is the most commonly used data type. It is used to store integers and takes 4 bytes of memory.

- Float

```
float pi = 3.14;
```

It stores single-precision floating-point numerals.

- Double

```
double root_three = 1.71;
```

It stores double-precision floating-point numbers. It takes 8 bytes of memory.

- Boolean

```
boolean b = false;
```

A boolean variable can either be true or false;

- String

```
string str = "Hello";
```

In C++, the string is a collection of characters, enclosed by double quotes “ ”. It is analogous to a character array. The string is not a built-in data type. Don’t forget to include this line at the top of your code before using string class - #include <cstring>;

7. Arrays

```
int main() {
string str[4] = {"Volvo", "BMW", "Volkswagen", "Ford"};
for(int i=0;i<4;i++){
cout << str[i]+ " ";
}
   return 0;
}
```

Instead of defining individual variables for each item, arrays are used to hold these values of the same data type in a single variable. It stores the values in a contagious block of memory, that’s why we need to specify the number of values it is going to hold beforehand.

Declare an array by specifying the variable type, the array name enclosed in square brackets, and the number of elements it should store.

8. Vectors

```
#include <vector>
int main() {
 vector<int> grade(3);
 grade[0] = 90;
 grade[1] = 80;
 grade[2] = 70;
 return 0;
}
```

A vector in C++ is a dynamic list of things that can expand and shrink in size. It can only hold values of the same type. It is important to #include the vector library in order to use vectors.

```
vector<string> wishlist;
wishlist.push_back("Furniture");
wishlist.push_back("Basket");
wishlist.pop_back();
cout << wishlist.size();  // returns the output 1
```

- push_back() function adds the value at the end of the vector.
- pop_back() function removes the element from the end of the vector.
- size() returns the size of the vector.

9. References and Pointers

```
string variable1 = "Value1"; // a variable
string &variable2 = variable1; // reference to variable1
```

Reference is an alias for an already existing variable. Once it is initialized to a variable, it cannot be changed to refer to another variable. In the above example, variable2 is a reference to variable1.

A pointer is a variable that contains the memory address of another variable.

```
int var = 2, *p;
p = &var;// The variable p holds the address of the variable var
```

```
int *const ptr;
```

ptr in the above code is a constant pointer.

```
const int* ptr;
```

The above code shows how to declare a pointer to a constant.

10. Functions

```
int sum (int a, int b){ // Declaration
  return a+b;
}
int main(){
int first_number= 10;
int second_number = 20;
cout<< sum(first_number, second_number); // Calling a function
}
```

When a function is called, it is a collection of statements that are all executed at the same time. Every function has a name that is used to refer to it when it is called. A function typically contains the following parts:

- **Return value**: Return value is the value that the function return at the end of the function. A function must return the value of the same data type as mentioned in the function declaration.
- **Parameters**: The parameters are the placeholders of the values passed on to them. In the above code, a and b are the function parameters.
- **Declaration**: It contains the name of the function, its return type, and parameter list.

11. String Functions

- append function

```
string firstName = "Ramesh ";
string lastName = "Babu";
string fullName = firstName.append(lastName);
cout << fullName;
```

This function concatenates one string after another string. The output of the above code will be: Ramesh Babu

You can also change any individual character of a string in C++.

```
string variable1 = "Hello World";
variable1[1] = 'i';
cout << variable1;
```

- length function

```
string variable1 = "Find my length";
cout << "The length of the string is: " << variable1.length();
```

It returns the length of the string as an integer.

12. Math Functions

```
cout << max(100, 110); // Returns max value
```

The above function returns the maximum value.

```
cout << min(1, 0); // Returns min value
```

The above function returns the minimum value.

```
cout << sqrt(625); // Returns square root
```

The above function returns the square root of the given value.

```
cout << ceil(a); // Returns ceil value
```

The above function returns the ceil value.

```
cout << floor(a);  // returns floor value
```

The above function returns the floor value.

```
cout <<pow(a, b) // returns a raised to the power b
```

The above function returns a to power b.

13. Iterative statements

Suppose you want to write "Write it hundred times" hundred times. One way to do this is to use cout 100 times. But don't worry, there's an easy way. You can use iterative statements to write the same statements 100 times.

- while loop

```
int i=1;
while(i <=100){
cout << "Write it hundred times \n";
i++;
}
```

The above code will print the statement 100 times (i.e. until the value of i is less than 101).

- do-while loop

```
int i=1;
do{
cout << "Write it hundred times \n";
i++;
}
while(i<=100);
```

The do-while loop is very similar to the while loop. The difference is that in the while loop the condition is checked first; and in the do-while loop, the condition is checked after certain tasks have been performed.

- for loop

A for loop repeats a code block for a set number of times. It is divided into three sections:

1. Initializing a counter.
2. Condition
3. The counter increment/decrement

```
for (int i = 0; i < 10; i++) {
 cout << i << "\n";
 }
```

The numbers 0 to 9 are printed on the screen in this example.

14. Object-Oriented Programming

- Class

```
class City {
 // Attribute
string name;
 int population;
public:
 // Method
 void increase_population() {
   population++;
 }
};
```

In C++, a class is the fundamental building block of Object-Oriented programming. A class typically consists of:

1. Attributes, often called member data, are data about a specific instance of a class.
2. Member functions, often known as methods, are functions that are present in the class.

- Objects

```
City Mumbai;
```

An object is an instance of a class that encapsulates its data and member functions. The above code creates an instance of the class City named Mumbai.

15. File handling

```
int main() {
// Create and open a text file
ofstream MyFile("filename.txt");
// Write to the file
MyFile << "File Handling in C++";
// Close the file
MyFile.close();
}.
```

File handling refers to reading and writing data in files. C++ provides us with functions that allow us to do so. The above code creates a file and writes text in it.

```
getline();
```

The above function allows us to read the file line by line.

```
void open(const char* file_name,ios::openmode mode);
```

The above function opens a file.

Conclusion

There has been a lot of change in the world of C++ programming since its inception, and it is becoming ever more important to be aware of the new syntax that is being introduced. This article provides a summary of the most popular syntaxes in C++ and has been designed to lay out all the basics for those who are early on in their programming journey. For those who are more experienced, this article will provide an overview of what is happening in the world of C++. We wish you a happy coding journey!

Useful Resources

- Practice C++ Programs Online
- C++ MCQ with Answers
- C vs C++

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions