C++ Cheatsheet

A comprehensive cheat sheet for C++ programming, covering syntax, data structures, standard library functions, and common algorithms.

Language Fundamentals

Basic Syntax

Include Header

```cpp
#include <iostream>
```

Main Function

```cpp
int main() {
  // Your code here
  return 0;
}
```

Output to Console

```cpp
std::cout << "Hello, World!"
<< std::endl;
```

Variables Declaration

```cpp
int age = 30;
float pi = 3.14;
std::string name = "John";
```

Comments

```cpp
// Single-line comment
/*
Multi-line
comment
*/
```

Input from Console

```cpp
int input_num;
std::cin >> input_num;
```

Data Types

`int`	Integer numbers
`float`	Floating-point numbers
`double`	Double-precision floating-point numbers
`char`	Single characters
`bool`	Boolean values (`true` or `false`)
`std::string`	Sequence of characters (from `<string>` header)

Operators

Arithmetic Operators: `+` , `-` , `*` , `/` , `%`

Assignment Operators: `=` , `+=` , `-=` , `*=` , `/=` , `%=`

Comparison Operators: `==` , `!=` , `>` , `<` , `>=` , `<=`

Logical Operators: `&&` (AND), `||` (OR), `!` (NOT)

Increment/Decrement Operators: `++` , `--`

Control Flow

Conditional Statements

If Statement

```cpp
if (condition) {
  // Code to execute if
condition is true
}
```

If-Else Statement

```cpp
if (condition) {
  // Code if true
} else {
  // Code if false
}
```

If-Else If-Else Statement

```cpp
if (condition1) {
  // Code if condition1 is
true
} else if (condition2) {
  // Code if condition2 is
true
} else {
  // Code if all
conditions are false
}
```

Switch Statement

```cpp
switch (expression) {
  case value1:
    // Code for value1
    break;
  case value2:
    // Code for value2
    break;
  default:
    // Default code
}
```

Loops

For Loop

```cpp
for (int i = 0; i < 10;
++i) {
  // Code to repeat
}
```

While Loop

```cpp
while (condition) {
  // Code to repeat
}
```

Do-While Loop

```cpp
do {
  // Code to repeat
} while (condition);
```

Break Statement	Exits the loop.
Continue Statement	Skips the current iteration.

Range-based for loop

Used to iterate over elements in a range (e.g., arrays, vectors).

```cpp
std::vector<int> nums = {1, 2, 3, 4, 5};
for (int num : nums) {
  std::cout << num << " ";
}
// Output: 1 2 3 4 5
```

Functions

Function Definition

Basic Structure	`return_type function_name(parameter_list) { // Function body return value; }`
Example: Add two integers	`int add(int a, int b) { return a + b; }`

Basic Structure:

```
return_type
function_name(parameter_li
st) {
  // Function body
  return value;
}
```

Example: Add two integers:

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

Function Overloading

Defining multiple functions with the same name but different parameters.

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

double add(double a, double b) {
  return a + b;
}
```

Function Pointers

Definition Pointers that store the address of a function.

```
int add(int a, int b) { return a +
b; }
int (*func_ptr)(int, int) = add;
int result = func_ptr(3, 4); //
result is 7
```

Lambda Expressions

Anonymous functions defined inline.

```
auto add = [](int a, int b) { return a + b; };
int result = add(5, 6); // result is 11
```

Standard Template Library (STL)

Containers

`std::vector`	Dynamic array (from `<vector>` header)
`std::list`	Doubly-linked list (from `<list>` header)
`std::deque`	Double-ended queue (from `<deque>` header)
`std::set`	Sorted set (from `<set>` header)
`std::map`	Associative array (from `<map>` header)
`std::unordered_map`	Hash table (from `<unordered_map>` header)

Algorithms

`std::sort` Sorts a range of elements (from `<algorithm>` header)

```
std::vector<int> nums = {3, 1, 4, 1, 5, 9};
std::sort(nums.begin(), nums.end());
// nums is now {1, 1, 3, 4, 5, 9}
```

`std::find` Finds the first occurrence of a value in a range (from `<algorithm>` header)

```
auto it = std::find(nums.begin(), nums.end(), 4);
if (it != nums.end()) {
  std::cout << "Found!" << std::endl;
}
```

`std::transform` Applies a function to a range of elements (from `<algorithm>` header)

```
std::transform(nums.begin(), nums.end(), nums.begin(), []
(int n){ return n * 2; });
// nums is now {2, 2, 6, 8, 10, 18}
```