# Complete C++ CheatSheet

**- Developer Shaurya**

## Important Links

- **C++ Cheatsheet :** https://developershaurya.com/cpp-cheatsheet/

- **Website:** https://developershaurya.com/

- **All Cheatsheets :** https://developershaurya.com/cheatsheets/

- **Youtube:** https://www.youtube.com/@DeveloperShaurya

- **(Developer Shaurya) Community :** https://t.me/developer_shaurya (Telegram)

C++ CheatSheet

# C++ Cheatsheet – A Quick Guide for Beginners & Developers

by Developer Shaurya • September 15, 2025 • 0


Download PDF

## Content Table

## 1. Basic Program Structure

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!\n";
    return 0;
}
```

## 2. Data Types

- `int` → 4 bytes
- `float` → 4 bytes (6 decimal places)
- `double` → 8 bytes (15 decimal places)
- `char` → 1 byte
- `bool` → true/false
- `string` → text
- `long`, `long long` → larger ints
- `unsigned` → no negative numbers

```
int x = 10;
double pi = 3.14159;
char c = 'A';
bool flag = true;
string s = "Hello";
```

## 3. **Input & Output**

```
int n;
cin >> n;
cout << "Number: " << n << endl;
```

## 4. **Operators**

- Arithmetic: `+ - * / %`
- Assignment: `= += -= *= /=`
- Relational: `== != < > <= >=`
- Logical: `&& || !`
- Bitwise: `& | ^ ~ << >>`

## 5. **Control Flow**

```
if (x > 0) cout << "Positive";
else if (x < 0) cout << "Negative";
else cout << "Zero";

switch (grade) {
    case 'A': cout << "Excellent"; break;
    default: cout << "Invalid";
}

for (int i=0; i<5; i++) cout << i << " ";
while (n>0) { cout<<n; n--; }
do { cout<<n; } while(n>0);
```

## 6. **Functions**

```
int add(int a, int b) { return a+b; }
void greet() { cout << "Hello"; }
inline int square(int x) { return x*x; }
```

## 7. **Arrays & Strings**

```
int arr[5] = {1,2,3,4,5};
for(int i=0;i<5;i++) cout<<arr[i];

string s = "World";
cout << s.size();
s.append("!");
```

## 8. **Pointers**

```
int x=10;
int *p=&x;
cout << *p; // value
cout << p;  // address
```

## 9. **References**

```
int a=5;
int &ref=a;
ref=10; // changes a
```

## 10. **Dynamic Memory**

```
int *ptr = new int(5);
cout << *ptr;
delete ptr;
```

## 11. **Structures**

```
struct Student {
    string name;
    int age;
};
Student s1={"Ravi",21};
```

## 12. **Classes & OOP**

```
class Car {
public:
    string brand;
    Car(string b) { brand=b; }
    void drive() { cout<<brand<<" driving\n"; }
};
Car c("BMW");
c.drive();
```

### Access Modifiers

- `public` → accessible everywhere
- `private` → accessible inside class only
- `protected` → accessible in derived classes

## 13. **Constructor & Destructor**

```
class A {
public:
    A() { cout<<"Constructor\n"; }
    ~A() { cout<<"Destructor\n"; }
};
```

## 14. **Inheritance**

```
class Parent {
public: void speak(){cout<<"Hello";} };
class Child: public Parent {
public: void run(){cout<<"Running";} };
Child c; c.speak(); c.run();
```

## 15. **Polymorphism**

```
class Base { public: virtual void show(){ cout<<"Base"; } };
class Derived: public Base { public: void show(){ cout<<"Derived"; } };
Base* b = new Derived();
b->show(); // Derived
```

## 16. **Encapsulation**

- Data hidden in private
- Access via getters/setters

## 17. **Abstraction**

```
class Shape {
public: virtual void draw()=0; // pure virtual
};
class Circle: public Shape {
public: void draw(){ cout<<"Circle"; }
};
```

## 18. **Templates**

```
template<typename T>
T add(T a, T b) { return a+b; }
cout<<add(3,4);
cout<<add(3.5,4.5);
```

## 19. **Exception Handling**

```
try {
    throw 10;
} catch (int e) {
    cout << "Error: " << e;
}
```

## 20. **File Handling**

```
ofstream out("data.txt");
out << "Hello File";
out.close();

ifstream in("data.txt");
string line;
while(getline(in,line)) cout<<line;
in.close();
```

## 21. **STL (Standard Template Library)**

### Vectors

```
vector<int> v={1,2,3};
v.push_back(4);
for(int x:v) cout<<x<<" ";
```

### Maps

```
map<string,int> m;
m["Alice"]=20;
cout<<m["Alice"];
```

### Sets

```
set<int> s={1,2,3};
s.insert(2);
for(int x:s) cout<<x;
```

### Algorithms

```
sort(v.begin(), v.end());
reverse(v.begin(), v.end());
```

## 22. **Smart Pointers (C++11)**

```
#include <memory>
unique_ptr<int> p1(new int(10));
shared_ptr<int> p2 = make_shared<int>(20);
```

## 23. **Lambda Functions**

```
auto add = [](int a,int b){ return a+b; };
cout<<add(3,4);
```

## 24. **Namespaces**

```
namespace first { int x=10; }
cout<<first::x;
```

## 25. **Preprocessor**

```
#define PI 3.14
#ifndef PI
#define PI 3.14159
#endif
```

## 26. **C++11/14/17 Features**

- `auto` type deduction
- Range-based loops
- `nullptr`
- `constexpr`
- `move semantics`
- Structured bindings
- `std::optional`, `std::variant`