C++ Programming Cheat Sheet

Complete Edition | Calibri + Consolas | Colorized examples

Nikhil Learn Hub | nikhillearnhub.com

Code color legend

`keyword` `string` `number` `comment` `#preprocess`

1. BASIC STRUCTURE

Part	Role
#include <iostream>	I/O streams
using namespace std;	Optional; brings std names
int main()	Program entry point
return 0;	Success exit

Syntax

```
#include <iostream>
int main() {
    // statements
    return 0;
}
```

Short example

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello C++!" << endl;
    return 0;
}
```

Key points

- Prefer std::cout over using-namespace in larger projects.

2. COMMENTS & VARIABLES

Form	Example
Line comment	// note
Block comment	/* multi line */
Declaration	int x = 10;
auto	auto y = 3.14;
const / constexpr	const int N = 5;

Syntax

```
type name = value;
auto name = expr;
```

Short example

```
int age = 21;
auto pi = 3.14159;      // double
const int MAX = 100;
constexpr int SQ = 4 * 4;
```

3. DATA TYPES

Kind	Examples	Notes
Integer	int, long, short	Whole numbers
Floating	float, double	Decimals
Character	char, wchar_t	Single chars
Boolean	bool	true / false
void	void	No value
User / STL	class, string, vector	Defined / library

Syntax

```
bool ok = true;
std::string name = "Ada";
```

Short example

```
int n = 42;
double d = 2.5;
bool flag = false;
char c = 'A';
```

Key points

- Modifiers: signed, unsigned, short, long, long long.

4. INPUT / OUTPUT

API	Use
cout << x	Print to console
cin >> x	Read from console
endl / '\n'	Newline (endl flushes)
getline(cin, s)	Read whole line into string

Syntax

```
cout << expr << ...;
cin >> var;
```

Short example

```
#include <iostream>
#include <string>
using namespace std;
int n; string name;
cin >> n; getline(cin, name);
cout << n << " " << name << "\n";
```

5. OPERATORS

Type	Operators	Notes
Arithmetic	+ - * / %	Math
Relational	< > <= >= == !=	Compare
Logical	&& || !	Combine
Bitwise	& | ^ ~ << >>	Bits
Assignment	= += -= *= /=	Store
Other	?: :: . -> * & new delete	Misc / memory

Syntax

```
a + b;   a == b;   cond ? t : f;
p = new int(5); delete p;
```

Short example

```
int a = 5, b = 2;
int m = (a > b) ? a : b;
int* p = new int(10);
delete p;
```

6. CONDITIONALS

Form	When
if / else if / else	Branch on conditions
switch	Integer / enum cases
?:	Inline choose value

Syntax

```
if (cond) { }
else if (cond2) { }
else { }
switch (x) { case 1: break; default: break; }
```

Short example

```
int x = 10;
if (x > 0)
    cout << "positive\n";
else
    cout << "not positive\n";
```

7. LOOPS

Loop	Notes
for	init; condition; update
while	Entry-controlled
do-while	Runs at least once
range-for	for (auto& x : container)
break / continue	Exit loop / next iteration

Syntax

```
for (init; cond; upd) { }
for (auto& x : v) { }
```

Short example

```
for (int i = 0; i < 5; i++)
    cout << i << " ";
vector<int> v = {1,2,3};
for (int x : v)
    cout << x;
```

8. ARRAYS

Form	Example
1D	int a[5] = {1,2,3,4,5};
2D	int m[2][3];
std::array	array<int,3> a = {1,2,3};

Syntax

```
type name[size];
type name[r][c];
```

Short example

```
int a[3] = {10, 20, 30};
a[1] = 99;
for (int i = 0; i < 3; i++)
    cout << a[i] << " ";
```

Key points

- Fixed size. Prefer vector for dynamic length.

9. VECTORS (STL)

Method	Role
push_back / emplace_back	Add element
size / empty	Length / check
[] / at	Access (at throws)
begin / end	Iterators
pop_back / clear	Remove / wipe

Syntax

```
#include <vector>
vector<T> v;
```

Short example

```
#include <vector>
using namespace std;
vector<int> v = {1, 2};
v.push_back(3);
cout << v.size() << " " << v[0];
```

10. STRINGS (std::string)

Method	Role
size / length	Character count
+ / append	Concatenate
substr(pos, n)	Substring
find / compare	Search / compare
empty / clear	Check / wipe

Syntax

```
#include <string>
string s = "text";
```

Short example

```
string s = "Hello";
s += " C++";
cout << s.substr(0, 5) << "\n";
cout << s.size();
```

11. REFERENCES vs POINTERS

Feature	Reference	Pointer
Null?	No (must bind)	Yes (nullptr)
Rebind?	No	Yes
Syntax	int& r = x;	int* p = &x;
Access	r	*p

Syntax

```
type& ref = var;
type* ptr = &var;
```

Short example

```
int x = 10;
int& r = x;
r = 20;              // x is 20
int* p = &x;
*p = 30;             // x is 30
```

12. DYNAMIC MEMORY

API	Role
new T	Allocate one object
new T[n]	Allocate array
delete p	Free object
delete[] p	Free array
unique_ptr / shared_ptr	Prefer smart pointers

Syntax

```
T* p = new T(args);
delete p;
```

Short example

```
int* p = new int(5);
cout << *p << "\n";
delete p;
p = nullptr;
```

Key points

- Match new[] with delete[]. Prefer RAII / smart pointers.

13. FUNCTIONS

Feature	Notes
Pass by value	Copy of argument
Pass by reference	type& - can modify caller
Default args	Trailing params only
Overloading	Same name, different params
Inline / constexpr	Optimization / compile-time

Syntax

```
ret name(params);
ret name(params) { body }
```

Short example

```
int sum(int a, int b = 0) {
    return a + b;
}
void bump(int& x) { x++; }
cout << sum(2, 3);
```

14. CLASSES & OBJECTS

Part	Meaning
class	User-defined type
object	Instance of a class
public / private	Access control
constructor	Init object
destructor	~Class() cleanup

Syntax

```
class Name {
public:
    Name();
    ~Name();
};
```

Short example

```
class Box {
public:
    int w;
    Box(int w) : w(w) {}
    int area() const { return w * w; }
};
Box b(4);
cout << b.area();
```

15. OOP PILLARS

Pillar	Idea
Encapsulation	Bundle data + methods; hide details
Abstraction	Show essential interface only
Inheritance	Reuse / extend a base class
Polymorphism	One interface, many forms

Syntax

```
class Derived : public Base { };
virtual void f();  // override in derived
```

Short example

```
class Animal {
public:
    virtual void speak() { cout << "?\n"; }
};
class Dog : public Animal {
public:
    void speak() override { cout << "woof\n"; }
};
```

16. INHERITANCE & ACCESS

Mode	public members become
public	public in derived
protected	protected in derived
private	private in derived

Syntax

```
class D : public B { };
class D : protected B { };
class D : private B { };
```

Short example

```
class B { public: int x = 1; };
class D : public B {};
D d;
cout << d.x;
```

17. TEMPLATES

Kind	Use
Function template	Generic function
Class template	Generic type
typename / class	Type parameter

Syntax

```
template <typename T>
T maxv(T a, T b);
```

Short example

```
template <typename T>
T maxv(T a, T b) {
    return (a > b) ? a : b;
}
cout << maxv(3, 7);
```

18. STL CONTAINERS

Group	Examples
Sequence	vector, deque, list, array
Adapters	stack, queue, priority_queue
Associative	set, map, multiset, multimap
Unordered	unordered_set, unordered_map

Syntax

```
#include <map>
map<string,int> m;
```

Short example

```
#include <map>
#include <string>
using namespace std;
map<string,int> ages;
ages["Ann"] = 20;
cout << ages["Ann"];
```

19. STL ALGORITHMS

Algorithm	Role
sort	Sort range
find	Search value
count	Count matches
max_element	Largest element
for_each	Apply function
binary_search	On sorted range

Syntax

```
#include <algorithm>
sort(v.begin(), v.end());
```

Short example

```
#include <algorithm>
#include <vector>
vector<int> v = {3,1,2};
sort(v.begin(), v.end());
auto it = find(v.begin(), v.end(), 2);
```

20. EXCEPTION HANDLING

Keyword	Role
try	Guard risky code
throw	Raise an exception
catch	Handle exception
catch (...)	Catch any

Syntax

```
try { throw expr; } catch (Type e) { }
```

Short example

```
try {
    if (true) throw 42;
} catch (int e) {
    cout << "err " << e << "\n";
}
```

21. FILE HANDLING

Type	Use
ifstream	Read from file
ofstream	Write to file
fstream	Read + write

Syntax

```
#include <fstream>
ofstream out("f.txt");
```

Short example

```
#include <fstream>
using namespace std;
ofstream out("out.txt");
out << "score=" << 95 << "\n";
out.close();
```

22. NAMESPACES

Form	Meaning
namespace N { }	Declare namespace
N::name	Qualified access
using N::name;	Bring one name
using namespace N;	Bring all (careful)

Syntax

```
namespace util { int add(int a,int b); }
```

Short example

```
namespace mathx {
    int sq(int x) { return x * x; }
}
cout << mathx::sq(5);
```

23. INTERVIEW HOTSPOTS

Topic	Revise
References vs pointers	Binding, null, reassignment
virtual / override	Runtime polymorphism
RAII + smart pointers	unique_ptr, shared_ptr
STL complexity	vector O(1) index; map log n
Rule of 0/3/5	Special member functions
const correctness	const methods, const refs

Syntax

```
virtual void f() = 0;  // pure virtual => abstract
```

Short example

```
class Shape {
public:
    virtual double area() const = 0;
    virtual ~Shape() = default;
};
```

Educational summary for Nikhil Learn Hub. Topic map inspired by GeeksforGeeks C++ Cheatsheet; wording is original.