

C++

Introduction to Programming

*originally developed by Weiguang Cui,
refined by Alexander Knebe*

Fast

Portable

OOP

Memory

C++ Language

High-Performance

Cross-platform language for building fast applications.

Memory Control

Direct control over system resources and memory.

C Extension

Developed by Bjarne Stroustrup as an extension of C — C programs integrate easily.

Portable

Develop once — adapt to Windows, Linux, macOS, embedded.

Object-Oriented

Clear program structure; code reuse lowers development costs.

More: <https://cplusplus.com/doc/tutorial/>

Write, Compile and Run



```
// C++ program to display "Hello World"
// Header file for input output functions
#include <iostream>
using namespace std;

// main() function: where execution begins
int main()
{
    // Prints hello world
    cout << "Hello World";
    return 0;
}
```

Both compiling and running process the code line-by-line from top to bottom.

1

Write

Type your C++ code in a .cpp file using any text editor or IDE.

2

Compile

Translate to machine code (e.g. g++ hello.cpp -o hello).

3

Run

Execute the binary. Output appears line-by-line.

Look Into the Detail



```
// C++ program to display "Hello World"
```

```
#include <iostream>
```

← Standard library for I/O. See: en.cppreference.com

```
using namespace std;
```

← Imports std namespace (cout, cin, endl...)

```
int main()
```

← Entry point. int = return type

```
{
```

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

← std output object — prints to screen

```
    return 0;
```

← 0 = success

```
}
```

What about loading local functions that you write yourself?

namespace

A namespace defines a named scope in which you can declare or define identifiers — variables, methods, and classes — to avoid name collisions.

Scope

A namespace creates a named scope.
Access identifiers as `ns::name` or via using namespace `ns`.

No Collision

Two namespaces can share the same identifier names without conflicting.

Access

Use `ns::func()` to call directly, or using namespace `ns`; to bring everything into scope.

namespace — Try It Yourself



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

namespace first_space {
    void func() {
        cout << "Inside first_space" << endl;
    }
}

namespace second_space {
    void func() {
        cout << "Inside second_space" << endl;
    }
}

using namespace first_space;
int main () {
    func();    // calls first_space::func()
    return 0;
}
```

Practice Questions

1

What if you omit using namespace? Can you still call the functions? How?

2

Can two namespaces share the same name but hold different functions?

3

Try both variations and observe what happens.

The main() Function

main() is the entry point of every C++ program — execution always starts here.



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

int Return type (0 = success)

main Required function name

{ } Function body block

return 0 Signals successful exit

Can you compile a program WITHOUT main()? Try it!

Can the return type be void, float, or double? Try changing it.

Variable Types

```
type  variableName = value;
```

int

```
int x = 42;
```

Integer (whole number)
e.g. 123 or -123

double

```
double pi = 3.14;
```

Floating-point decimal
e.g. 19.99 or -3.5

char

```
char c = 'A';
```

Single character
Surrounded by single quotes

string

```
string s = "Hi";
```

Text / sequence of chars
Surrounded by double quotes

bool

```
bool ok = true;
```

Two values only:
true or false

What is the max value of an int (4 bytes) vs float (4 bytes)? Why are they different?

cout and cin

cout – Output

Standard output object.
Prints to the screen.

```
cout << "Hello" << endl;  
cout << 42;
```

cin – Input

Standard input object.
Reads from the keyboard.

```
int x;  
cin >> x;
```

Questions & Practice

- Is there a difference between "" and " for cout?
- Is there a difference between "\n" and << endl?
- Write a program: read keyboard input with cin and print it with cout.

char and string

char — character array

```
char name[256];    // char array
```

Fixed size (up to 256 chars)
Manual memory management
C-style — older approach

string — auto-managed

```
string name;    // needs #include <string>
```

Auto-managed: grows as needed
A class wrapping a char array
Preferred modern C++ approach

Questions

Can a char array be longer than 256? How do you set the size?

Does string have a limit? How many characters can it hold?

Can you convert char to string or vice versa? If so, how?

More on cout — Output Precision



```
#include <iostream>    // cout, fixed
#include <iomanip>       // setprecision

int main () {
    double f = 3.14159;

    cout << setprecision(5) << f;    // 3.1416
    cout << setprecision(9) << f;    // 3.14159000

    cout << fixed;
    cout << setprecision(5) << f;    // 3.14159
    cout << setprecision(9) << f;    // 3.141590000

    return 0;
}
```

What changes?

setprecision(n)

Sets the number of significant digits in output.

std::fixed

Switches to fixed-point: n digits after the decimal.

Without fixed

setprecision(5) on 3.14159 gives 3.1416 (rounds).

With fixed

setprecision(5) on 3.14159 gives 3.14159 (exact).

Run this — what do you observe?