The C++ Programming Language

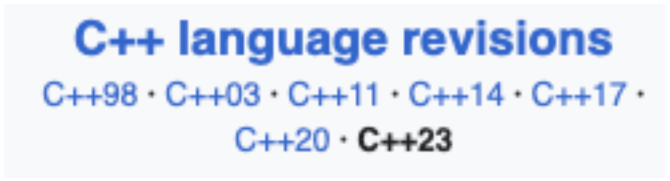
- **C++** is an **imperative**, **object-oriented language** and is an extension of the C programming language.
- C++ has a **static type system** and is considered more **strongly typed** than C.
- **Semicolons** terminate statements while **curly braces** are used to group statements into blocks **(block-structured)**.
- Code is organized into **classes and objects** (which are instantiations of classes).
- Applications include operating systems, desktop applications, video games, servers, and performance-critical applications.

a general-purpose programming language good for 'scaling up'

C++ is standardized

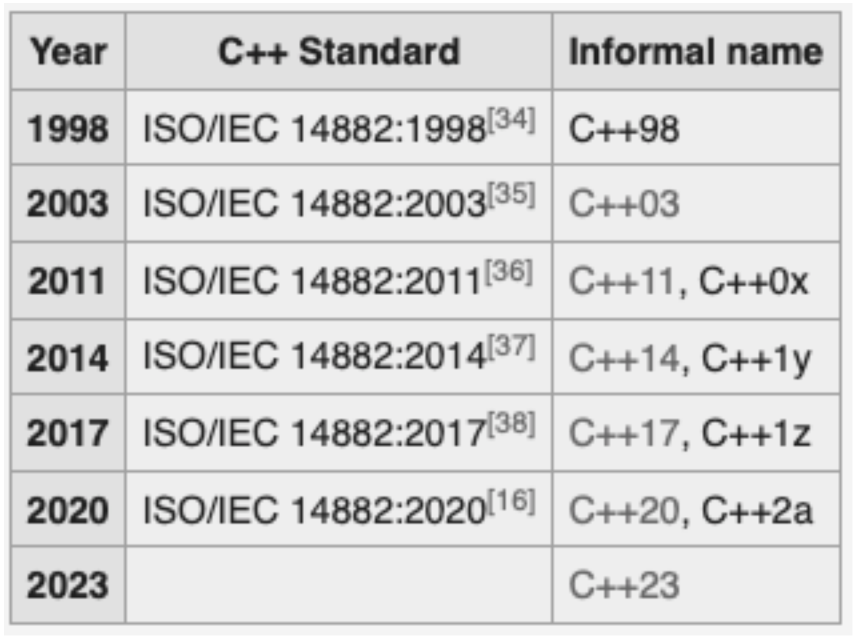
by an ISO working group known as JTC1/SC22/WG21.

So far, it has published six revisions of the C++ standard and is currently working on the next revision, C++23.



C++ language revisions

C++98 · C++03 · C++11 · C++14 · C++17 · C++20 · **C++23**



Year	C++ Standard	Informal name
1998	ISO/IEC 14882:1998[34]	C++98
2003	ISO/IEC 14882:2003[35]	C++03
2011	ISO/IEC 14882:2011[36]	C++11, C++0x
2014	ISO/IEC 14882:2014[37]	C++14, C++1y
2017	ISO/IEC 14882:2017[38]	C++17, C++1z
2020	ISO/IEC 14882:2020[16]	C++20, C++2a
2023		C++23

The **C++ language** provides the **five basic types**:

- char
- int
- float
- double
- bool

```
bool done = false;
```

Recall, a **data type** is:

- a set of possible values, and
- a set of possible operations on these values, and
- a representation of these values for a specific machine.

and the **modifiers**:

- short — 'at least 16 bits'
- long — 'at least 32 (64?) bits'

- signed
- unsigned

It also provides **derived types** based on the **four basic types**:

- **pointers** (to entities of some type)
- **arrays** (of elements of the same type)
- **structs** (of members of possibly different types)
- **unions** (of overlapping members of possibly different types)

It also provides a way to create **user-defined types**.

A **class** is a user-defined type.

- A class is defined in C++ using the keyword `class` followed by the name of the class.
- The body of the class is defined inside the curly brackets and terminated by a semicolon.

```
class Student
{
public:

    . . .

private:

    . . .
};
```

access specifier `public`, for the **interface** portion of the class (usually member functions)

access specifier `private`, for the **implementation** portion of the class (usually member variables)

Example of a class

```
class Student
{
public:
    Student() {};
    Student(string name, double GPA) {this->name = name; this->GPA=GPA;};
    string getName() {return name;};
    double getGPA() {return GPA;};
private:
    string name;
    double GPA;
};



int main(int argc, const char * argv[])
{
    Student s1("Jack", 3.2);

    cout << "Hello " << s1.getName() << endl;
    cout << "Your GPA is: " << s1.getGPA() << endl;

    return 0;
}
```

s1 is an object of type Student
s1 is an instance of the Student class

Hello World in C++

```
#include <iostream>

int main(int argc, const char * argv[])
{
    std::cout << "Hello World!\n”;

    return 0;
}
```

Hello World in C++

the directive #include <iostream> instructs the preprocessor to include a standard C++ header file that has objects like cin and cout

```
#include <iostream>

int main(int argc, const char * argv[])
{
    std::cout << "Hello World!\n”;

    return 0;
}
```

std::cout is an object that represents the standard character output device, and << is the insertion operator which indicates that what follows is passed to std::cout.

Hello World in C++

Everything in the C++ standard library is declared within the namespace `std`.
A program needs to either qualify each use of a library object with `std::` or introduce visibility of the namespace with a `using namespace` declaration.

```cpp
#include <iostream>

using namespace std;

int main(int argc, const char * argv[])
{
    cout << "Hello World!\n”;

    return 0;
}
```

C++ has a `string` type

The C++ standard library has a class `string`. It represents a sequence of characters and has a set of member functions for string manipulation and provides dynamic memory management for the string data.

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
    string blabla;

    blabla = "this is a string";

    cout << blabla << endl;

    return 0;
}
```

insert the `endl` manipulator for a newline character

Try this on the server using gcc...

```
#include <iostream>

using namespace std;

int main ()
{
    string fn;
    cout << "enter your first name: ";
    cin >> fn;
    cout << "you entered " << fn << endl;

    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main ()
{
    int i;
    cout << "enter an integer: ";
    cin >> i;
    cout << "you entered " << i << endl;

    return 0;
}
```

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
    string mystr;
    float price=0.0;
    bool good_input = false;

    while(!good_input)
    {
        cout << "Enter price: ";
        getline(cin, mystr);
        try
        {
            price = stof(mystr);
            good_input = true;
        }
        catch(exception &err)
        {
            cout << "Conversion failure: " << err.what() <<endl;
        }
    }
    cout << "Total price: " << price*5 << endl;

    return 0;
}
```

`getline` gets everything the user typed up to pressing 'enter'

`stof` throws an exception if it cannot do the conversion