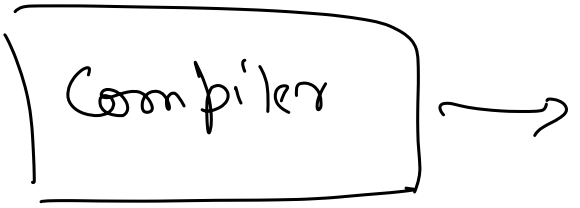
```
int main() {
    return 0;
}
```

first.cpp
(Source file)
↑
portable



executable file (machine dependent)

Interpretor

A Brief Timeline of C Programming History

1967: BPCL developed by Martin Richards.
1970: B developed by Ken Thomson Simplified version of BPCL.
1972: Traditional C developed by Dennis Ritchie.
1978: K&RC developed by Brian Kernighan & Dennis Ritchie.
1989: ANSI C revised by ANSI Committee.
1990: ANSI C / ISO C revised by ISO Committee.
1999: C99 revised by C standards Committee.
2011: revised by C standards Committee.
2018: C18, addressed defects in C11 without introducing new language features.

A Brief Timeline of C++ Programming History

1979: Stroustrup starts to work on C with Classes in Bell Labs
1983: C with Classes renamed to C++
1985: The first edition of "The C++ Programming Language"
1989: C++ 2.0, version was released
1998: The first ISO Standard (C++ 98)
2003: C++03, bugfixes
2011: C++11, a major revision
2014: C++14, bugfixes and small improvements on C++11
2017: C++17, major revision of the C++ programming language.
2020: C++20, major revision of the C++ programming language.

Tokens

The text of a C++ program consists of tokens and white space.

A token is the smallest element of a C++ program that is meaningful to the compiler.

The C++ parser recognizes these kinds of tokens:
Keywords
Identifiers
Numeric, Boolean and Pointer Literals
String and Character Literals
User-Defined Literals
Operators
Punctuators

Tokens are usually separated by white space, which can be one or more:
Blank space, Tabs, New lines,
Comments

Character Set

The basic source character set consists of 96 characters that may be used in source files.

This set includes the space character, horizontal tab, vertical tab, form feed and new-line control characters, and:

Lower case a-z
Upper case A-Z
Digits 0-9
_ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " '

C++ is case sensitive ⇒ Small a is not same as capital A

Identifier

An identifier is a sequence of characters used to denote one of the following:
Object or variable name,
Class, structure, or union name,
Type name, etc

The following characters are allowed as any character of an identifier:

_
Lower case a-z
Upper case A-Z

The following characters are allowed as any character in an identifier except the first:
Digits 0-9

Keywords

Keywords are predefined reserved identifiers that have special meanings.
They can't be used as identifiers in your program.

alignas alignof auto
bool break
case catch char char16_t char32_t class const const_cast constexpr continue
decltype default delete do double dynamic_cast
else enum explicit extern
false float for friend
goto
if inline int
long
mutable
namespace new noexcept nullptr
operator
private protected public
register reinterpret_cast return
short signed sizeof static static_assert static_cast struct switch
template this thread_local throw true try typedef typeid typename
union unsigned using declaration using directive
virtual void volatile
wchar_t while

There will also be compiler specific keywords.

(left to right associativity)
Scope resolution ::

(left to right associativity)
Postfix increment ++
Postfix decrement --
Member selection (object or pointer) . or ->
Array subscript []
Function call ()
Type name typeid
Constant type conversion const_cast
Dynamic type conversion dynamic_cast
Reinterpreted type conversion reinterpret_cast
Static type conversion static_cast

(right to left associativity)
Prefix increment ++
Prefix decrement --
Size of object or type sizeof
One's complement ~
Logical not !
Unary negation -
Unary plus +
Address-of &

Operators

↳ Precedence
↳ Associativity

2 + 5 x 10

2 + 5 - 3

Indirection *
Create object new
Destroy object delete
Cast ()

(left to right associativity)
Pointer-to-member (objects or pointers) .* or ->*

(left to right associativity)
Multiplication *
Division /
Modulus %

(left to right associativity)
Addition +
Subtraction -

(left to right associativity)
Left shift <<
Right shift >>

(left to right associativity)
Less than <
Greater than >

Less than or equal to <=
Greater than or equal to >=

(left to right associativity)
Equality ==
Inequality !=

(left to right associativity)
Bitwise AND &

(left to right associativity)
Bitwise exclusive OR ^

(left to right associativity)
Bitwise inclusive OR |

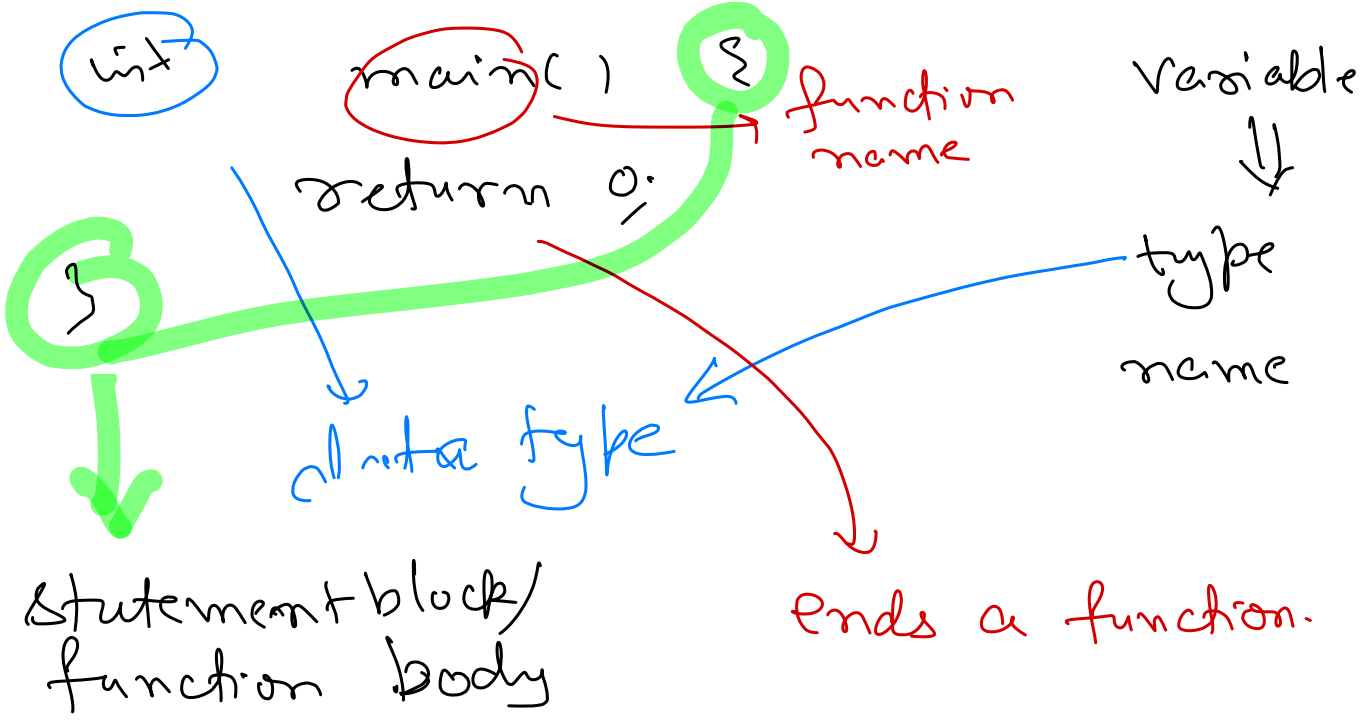
(left to right associativity)
Logical AND &&

(left to right associativity)
Logical OR ||

(right to left associativity)
Conditional ? :

Assignment =
Multiplication assignment *=
Division assignment /=
Modulus assignment %=
Addition assignment +=
Subtraction assignment -=
Left-shift assignment <<=
Right-shift assignment >>=
Bitwise AND assignment &=
Bitwise inclusive OR assignment |=
Bitwise exclusive OR assignment ^=
throw expression throw

(left to right associativity)
Comma ,



→ C/C++ program is a collection of functions.

→ There must be a function called main.

→ main is entry point of your program.

↓

from where program execution starts.

```
#include <iostream>

int main()
{
    std::cout << "Hello world";
    return 0;
}
```

double quotes

↓

" Hello world "

↳ string literal/constant

0

↳ integer literal/constant

```
std :: cout << 100;
```

100 → integer constant

Find sum of 2 numbers ⇒ integers.

1. Get two numbers
2. Add two numbers & find result
3. Display result

Read from user

```
std :: cin >> x;
```

```
int main() {
    int no1;
    int no2;

    std:: cin >> no1;
    std:: cin >> no2;

    int result;
    result = no1 + no2;
```

① ②

Declare variable no1 of type int

variable in which value accepted from user will be stored.

assign-ment operator

addition operator

```
int i;
i = 5;
```

③ [std:: cout << result;

return 0;

}

Exercise

① Power of number

② Swap two numbers.

③ Area of rectangle.

④ Compound Interest.