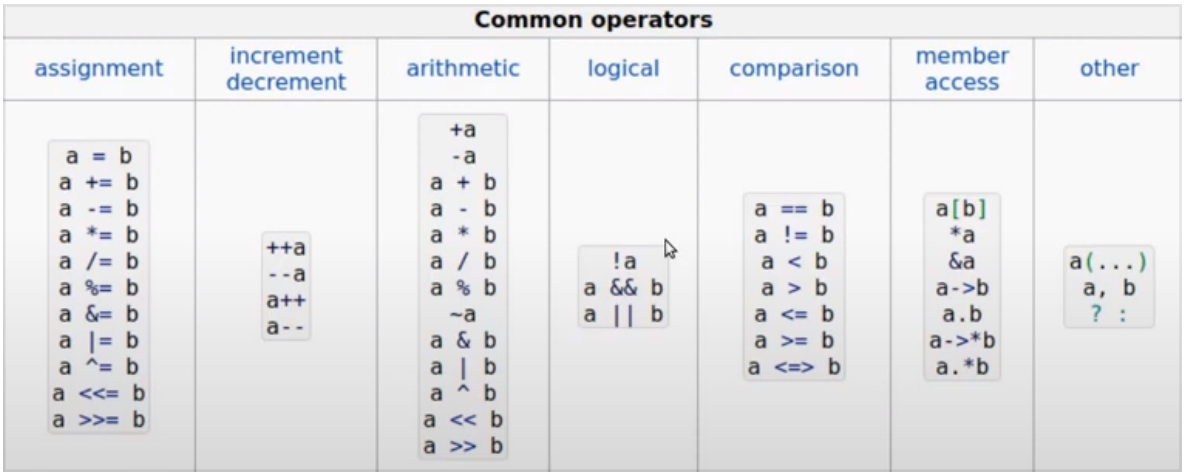
Core C++

C++ Keywords

alignas (since C++11)	default(1)	register(2)
alignof (since C++11)	delete(1)	reinterpret_cast
and	do	requires (since C++20)
and_eq	double	return
asm	dynamic_cast	short
atomic_cancel (TM TS)	else	signed
atomic_commit (TM TS)	enum	sizeof(1)
atomic_noexcept (TM TS)	explicit	static
auto(1)	export(1)(3)	static_assert (since C++11)
bitand	extern(1)	static_cast
bitor	false	struct(1)
bool	float	switch
break	for	synchronized (TM TS)
case	friend	template
catch	goto	this
char	if	thread_local (since C++11)
char8_t (since C++20)	inline(1)	throw
char16_t (since C++11)	int	true
char32_t (since C++11)	long	try
class(1)	mutable(1)	typedef
compl	namespace	typeid
concept (since C++20)	new	typename
const	noexcept (since C++11)	union
consteval (since C++20)	not	unsigned
constexpr (since C++11)	not_eq	using(1)
constinit (since C++20)	nullptr (since C++11)	virtual
const_cast	operator	void
continue	or	volatile
co_await (since C++20)	or_eq	wchar_t
co_return (since C++20)	private	while
co_yield (since C++20)	protected	xor
decltype (since C++11)	public	xor_eq
	reflexpr (reflection TS)	

C++ Expressions



Common operators						
assignment	increment decrement	arithmetic	logical	comparison	member access	other
a = b a += b a -= b a *= b a /= b a %= b a &= b a |= b a ^= b a <<= b a >>= b	++a --a a++ a--	+a -a a + b a - b a * b a / b a % b ~a a & b a | b a ^ b a << b a >> b	!a a && b a || b	a == b a != b a < b a > b a <= b a >= b a <=> b	a[b] *a &a a->b a.b a->*b a.*b	a(...) a, b ? :

Control Structures

If statement

```
if (STATEMENT) {
  // This is executed if STATEMENT == true
} else if (OTHER_STATEMENT) {
  // This is executed if:
  // (STATEMENT == false) && (OTHER_STATEMENT == true)
} else {
  // This is executed if neither is true
}
```

Switch Statement

```
switch(STATEMENT) {
  case CONST_1:
    // This runs if STATEMENT == CONST_1.
    break;
  case CONST_2:
    // This runs if STATEMENT == CONST_2.
    break;
  default:
    // This runs if no other options worked.
}
```

Example:

- C style (C also has enum but just as an example)

```c
#include <stdio.h>
int main() {
  // Color could be:
  // RED   == 1
  // GREEN == 2
  // BLUE  == 3
  int color = 2;
  switch (color) {
    case 1: printf("red\n"); break;
    case 2: printf("green\n"); break;
    case 3: printf("blue\n"); break;
  }
  return 0;
}
```

- C++ style

```cpp
#include <iostream>

int main() {
  enum class RGB { RED, GREEN, BLUE };
  RGB color = RGB::GREEN;

  switch (color) {
  case RGB::RED:   std::cout << "red\n"; break;
  case RGB::GREEN: std::cout << "green\n"; break;
  case RGB::BLUE:  std::cout << "blue\n"; break;
  }
  return 0;
}
```

Loops

While Loop

```
while (STATEMENT) {
  // Loop while STATEMENT == true.
}
```

For Loop

```
for (INITIAL_CONDITION; END_CONDITION; INCREMENT) {
  // This happens until END_CONDITION == false
}
```

Range For Loop

```
for (const auto& value : container) {
  // This happens for each value in the container.
}
```

To Exit Loops and Iterations:

- Use *break* to exit the loop.
- Use *continue* to skip to next iteration

Built-in Types

```
bool this_is_fun = true;     // Boolean: true or false.
char carret_return = '\n';   // Single character.
int meaning_of_life = 42;    // Integer number.
short smaller_int = 42;      // Short number.
long bigger_int = 42;        // Long number.
float fraction = 0.01f;      // Single precision float.
double precise_num = 0.01;   // Double precision float.
auto some_int = 13;          // Automatic type [int].
auto some_float = 13.0f;     // Automatic type [float].
auto some_double = 13.0;     // Automatic type [double].
```