Assignment operators

Operator name	Syntax	Over load able	Prototype examples (for class T)	
			Inside class definition	Outside class definition
simple assignment	`a = b`	Yes	`T& T::operator =(const T2& b);`	N/A
addition assignment	`a += b`	Yes	`T& T::operator +=(const T2& b);`	`T& operator +=(T& a, const T2& b);`
subtraction assignment	`a -= b`	Yes	`T& T::operator -=(const T2& b);`	`T& operator -=(T& a, const T2& b);`
multiplication assignment	`a *= b`	Yes	`T& T::operator *=(const T2& b);`	`T& operator *=(T& a, const T2& b);`
division assignment	`a /= b`	Yes	`T& T::operator /=(const T2& b);`	`T& operator /=(T& a, const T2& b);`
remainder assignment	`a %= b`	Yes	`T& T::operator %=(const T2& b);`	`T& operator %=(T& a, const T2& b);`
bitwise AND assignment	`a &= b`	Yes	`T& T::operator &=(const T2& b);`	`T& operator &=(T& a, const T2& b);`
bitwise OR assignment	`a |= b`	Yes	`T& T::operator |=(const T2& b);`	`T& operator |=(T& a, const T2& b);`
bitwise XOR assignment	`a ^= b`	Yes	`T& T::operator ^=(const T2& b);`	`T& operator ^=(T& a, const T2& b);`
bitwise left shift assignment	`a <<= b`	Yes	`T& T::operator <<=(const T2& b);`	`T& operator <<=(T& a, const T2& b);`
bitwise right shift assignment	`a >>= b`	Yes	`T& T::operator >>=(const T2& b);`	`T& operator >>=(T& a, const T2& b);`

Increment/decrement operators

Operator name	Syntax	Overloadable	Prototype examples (for class T)	
			Inside class definition	Outside class definition
pre-increment	`++a`	Yes	`T& T::operator++();`	`T& operator++(T& a);`
pre-decrement	`--a`	Yes	`T& T::operator--();`	`T& operator--(T& a);`
post-increment	`a++`	Yes	`T T::operator++(int);`	`T operator++(T& a, int);`
post-decrement	`a--`	Yes	`T T::operator--(int);`	`T operator--(T& a, int);`

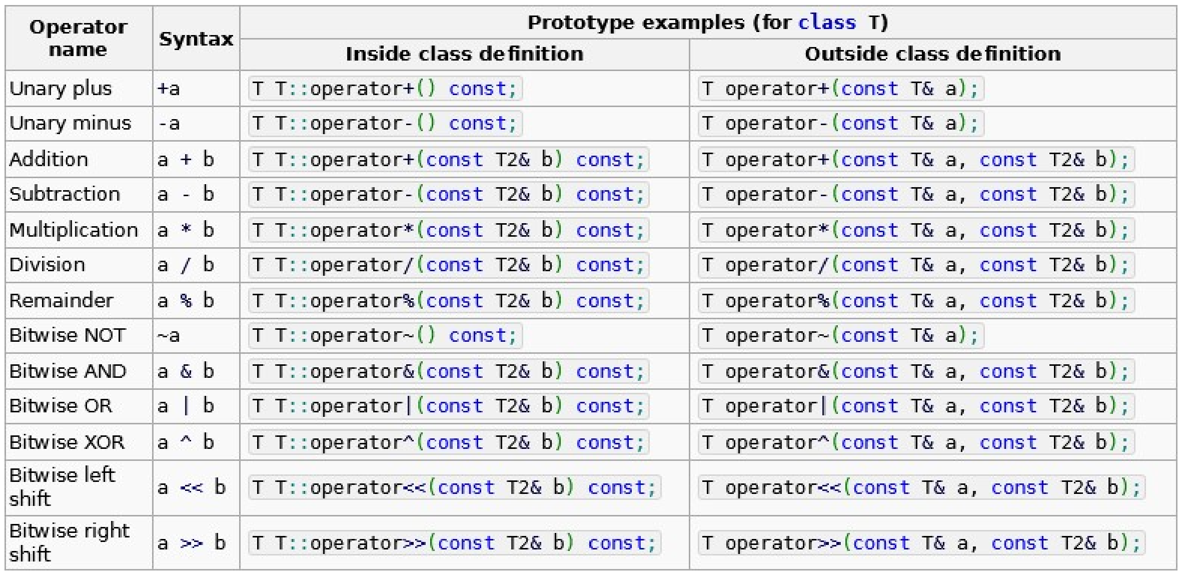
Logical operators

Operator name	Syntax	Over load able	Prototype examples (for class T)	
			Inside class definition	Outside class definition
negation	`not a` `!a`	Yes	`bool T::operator!() const;`	`bool operator!(const T &a);`
AND	`a and b` `a && b`	Yes	`bool T::operator&&(const T2 &b) const;`	`bool operator&&(const T &a, const T2 &b);`
inclusive OR	`a or b` `a || b`	Yes	`bool T::operator||(const T2 &b) const;`	`bool operator||(const T &a, const T2 &b);`

Other operators

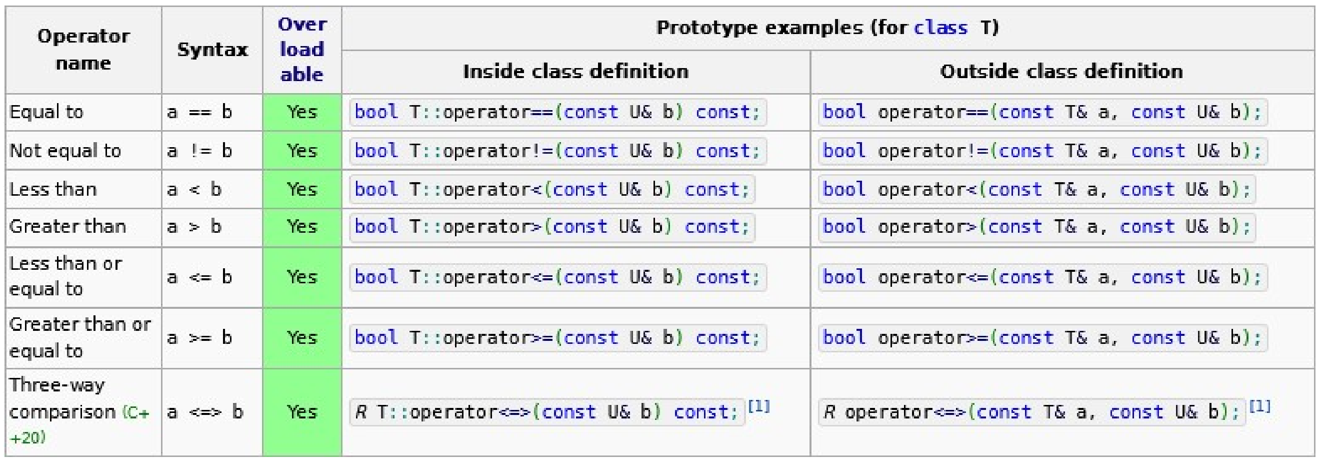
Operator name	Syntax	Overload able	Prototype examples (for class T)	
			Inside class definition	Outside class definition
function call	`a(a1, a2)`	Yes	`R T::operator()(Arg1 &a1, Arg2 &a2, ...);`	N/A
comma	`a, b`	Yes	`T2& T::operator,(T2 &b);`	`T2& operator,(const T &a, T2 &b);`
conditional operator	`a ? b : c`	No	N/A	N/A

Arithmetic operators



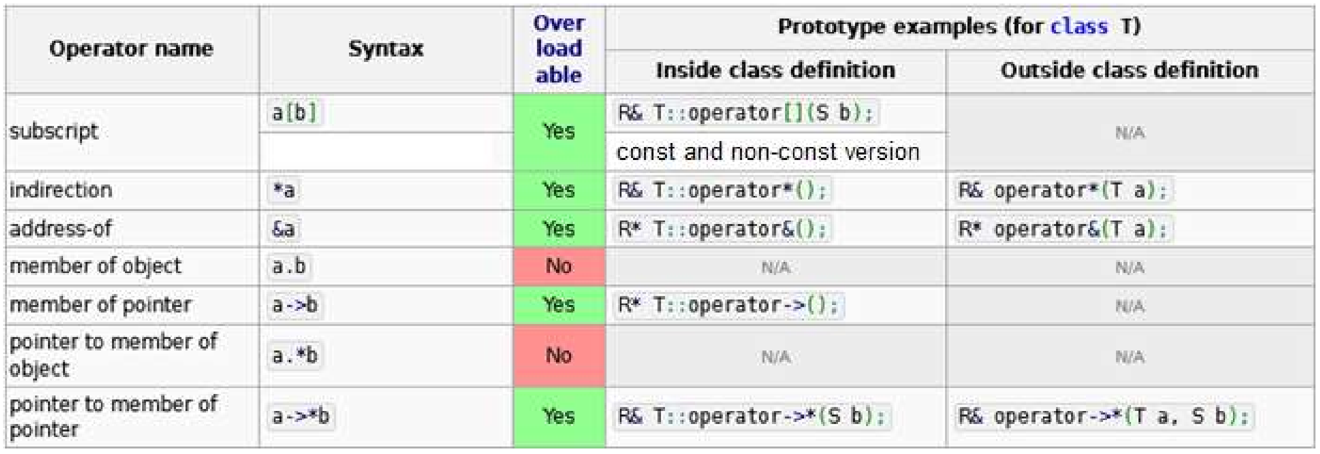
Operator name	Syntax	Prototype examples (for class T)	
		Inside class definition	Outside class definition
Unary plus	+a	T T::operator+() const;	T operator+(const T& a);
Unary minus	-a	T T::operator-() const;	T operator-(const T& a);
Addition	a + b	T T::operator+(const T2& b) const;	T operator+(const T& a, const T2& b);
Subtraction	a - b	T T::operator-(const T2& b) const;	T operator-(const T& a, const T2& b);
Multiplication	a * b	T T::operator*(const T2& b) const;	T operator*(const T& a, const T2& b);
Division	a / b	T T::operator/(const T2& b) const;	T operator/(const T& a, const T2& b);
Remainder	a % b	T T::operator%(const T2& b) const;	T operator%(const T& a, const T2& b);
Bitwise NOT	~a	T T::operator~() const;	T operator~(const T& a);
Bitwise AND	a & b	T T::operator&(const T2& b) const;	T operator&(const T& a, const T2& b);
Bitwise OR	a | b	T T::operator|(const T2& b) const;	T operator|(const T& a, const T2& b);
Bitwise XOR	a ^ b	T T::operator^(const T2& b) const;	T operator^(const T& a, const T2& b);
Bitwise left shift	a << b	T T::operator<<(const T2& b) const;	T operator<<(const T& a, const T2& b);
Bitwise right shift	a >> b	T T::operator>>(const T2& b) const;	T operator>>(const T& a, const T2& b);

Comparison operators



Operator name	Syntax	Over load able	Prototype examples (for class T)	
			Inside class definition	Outside class definition
Equal to	a == b	Yes	bool T::operator==(const U& b) const;	bool operator==(const T& a, const U& b);
Not equal to	a != b	Yes	bool T::operator!=(const U& b) const;	bool operator!=(const T& a, const U& b);
Less than	a < b	Yes	bool T::operator<(const U& b) const;	bool operator<(const T& a, const U& b);
Greater than	a > b	Yes	bool T::operator>(const U& b) const;	bool operator>(const T& a, const U& b);
Less than or equal to	a <= b	Yes	bool T::operator<=(const U& b) const;	bool operator<=(const T& a, const U& b);
Greater than or equal to	a >= b	Yes	bool T::operator>=(const U& b) const;	bool operator>=(const T& a, const U& b);
Three-way comparison (C++20)	a <=> b	Yes	R T::operator<=>(const U& b) const;[1]	R operator<=>(const T& a, const U& b);[1]

Member access operators



Operator name	Syntax	Over load able	Prototype examples (for class T)	
			Inside class definition	Outside class definition
subscript	a[b]	Yes	R& T::operator[](S b);	N/A
			const and non-const version	
indirection	*a	Yes	R& T::operator*();	R& operator*(T a);
address-of	&a	Yes	R* T::operator&();	R* operator&(T a);
member of object	a.b	No	N/A	N/A
member of pointer	a->b	Yes	R* T::operator->();	N/A
pointer to member of object	a.*b	No	N/A	N/A
pointer to member of pointer	a->*b	Yes	R& T::operator->*(S b);	R& operator->*(T a, S b);