

Time Class Requirements Specification

Douglas Wilhelm Harder

October 3, 2026

1 Purpose

The purpose of the `Time_hms` and `Time_s` classes is to represent durations in hours, minutes, and seconds, and to provide operations for creating, querying, comparing, displaying, adding, and subtracting durations.

A duration is not a time of day. Consequently, the number of hours is not restricted to the range 0 through 23. For example, `37:15:42` represents a duration of 37 hours, 15 minutes, and 42 seconds. Durations may be positive, negative, or zero.

Both classes shall provide the same behavior and corresponding public operations, with their respective class names substituted in declarations. This document specifies observable behavior, not an internal representation or algorithms.

2 Scope

Each class shall support construction of non-negative durations from hours, minutes, and seconds; access to components and sign; comparison; unary plus and minus; addition and subtraction of durations; compound assignment; prefix and postfix increment and decrement by one second; and formatted output.

Arithmetic directly between a duration and an integer, multiplication, division, and remainder are outside the scope of this specification.

3 Construction

REQ-1.1 Each class shall provide a public constructor with the following form, where `Time` stands for the applicable class name:

```
Time( std::uint32_t hours,  
      unsigned int  minutes,  
      unsigned int  seconds );
```

REQ-1.2 The constructor shall create a non-negative duration having the specified hour, minute, and second components.

REQ-1.3 The hours argument shall be representable as a `std::uint32_t`; hours need not be less than 24. The specification does not require arithmetic to support results outside the range representable by a 32-bit unsigned hour component.

REQ-1.4 Minutes and seconds shall each be between 0 and 59, inclusive. Arguments outside these ranges shall cause `std::invalid_argument` to be thrown.

For example, `Time{ 37, 15, 42 }` shall construct a duration of 37 hours, 15 minutes, and 42 seconds.

4 Components and sign

REQ-2.1 Each class shall provide:

```
bool          negative() const;  
std::uint32_t hours()   const;  
std::uint8_t  minutes() const;  
std::uint8_t  seconds() const;
```

REQ-2.2 The hour, minute, and second accessors shall return the components of the *magnitude* of the duration. Minutes and seconds shall be in the range 0 through 59.

REQ-2.3 `negative()` shall return `true` exactly when the duration is negative. Zero shall not be negative. None of these operations shall modify the object.

5 Comparisons

REQ-3.1 Each class shall support all six comparisons of two durations:

```
bool operator==( Time const &other ) const;  
bool operator!=( Time const &other ) const;  
bool operator< ( Time const &other ) const;  
bool operator<=( Time const &other ) const;  
bool operator>=( Time const &other ) const;  
bool operator> ( Time const &other ) const;
```

REQ-3.2 Comparisons shall reflect the numerical ordering of signed durations. Equal durations shall compare equal, including any duration equal to zero.

6 Unary operators

REQ-4.1 Unary plus shall return an unchanged duration, and unary minus shall return its additive inverse:

```
Time operator+() const;  
Time operator-() const;
```

REQ-4.2 Applying unary minus to zero shall produce zero, not a negative zero. Neither operator shall modify its operand.

7 Addition and subtraction

REQ-5.1 Each class shall support:

```
Time operator+( Time const &other ) const;  
Time operator-( Time const &other ) const;
```

REQ-5.2 These operators shall return the mathematical sum and difference of the operands, including for negative operands. Neither operand shall be modified.

REQ-5.3 The results shall be correctly normalized into hours, minutes, and seconds. For example:

```
2:45:30 + 1:30:45  ->  4:16:15  
1:30:00 - 2:00:00  -> -0:30:00  
-1:00:00 + 2:00:00 ->  1:00:00
```

These requirements do not prescribe how carrying, borrowing, or sign handling is implemented.

8 Compound assignment

REQ-6.1 Each class shall support:

```
Time &operator+=( Time const &other );  
Time &operator-=( Time const &other );
```

REQ-6.2 These operators shall modify the left-hand object to contain the sum or difference and shall return a reference to that object.

9 Increment and decrement

REQ-7.1 Each class shall support prefix and postfix increment and decrement:

```
Time &operator++();  
Time operator++( int );  
Time &operator--();  
Time operator--( int );
```

REQ-7.2 Increment shall increase a duration by one second; decrement shall decrease it by one second. These operations shall work across minute and hour boundaries and across zero.

REQ-7.3 Prefix operators shall return a reference to the modified object. Postfix operators shall return the value of the object before modification.

10 String representation and output

REQ-8.1 Each class shall provide:

```
std::string to_string() const;
```

The string shall use the form `H:MM:SS`, with one or more digits for hours and exactly two digits each for minutes and seconds.

REQ-8.2 A negative duration shall have a leading minus sign. A non-negative duration shall not have a leading plus sign. Zero shall be formatted as `0:00:00`.

For example, `2:05:09`, `-0:30:00`, and `37:15:42` are valid representations.

REQ-8.3 Each class shall provide a non-member output-stream insertion operator:

```
std::ostream &operator<<( std::ostream &out,  
                          Time const &time );
```

It shall insert the same text returned by `to_string()` and return a reference to the output stream.

11 Required public interface

The following summarizes the common interface. Replace `Time` consistently with `Time_hms` or `Time_s`.

```
class Time {
public:
    Time( std::uint32_t hours,
          unsigned int  minutes,
          unsigned int  seconds );

    bool          negative() const;
    std::uint32_t hours()    const;
    std::uint8_t  minutes() const;
    std::uint8_t  seconds() const;

    bool operator==( Time const &other ) const;
    bool operator!=( Time const &other ) const;
    bool operator< ( Time const &other ) const;
    bool operator<=( Time const &other ) const;
    bool operator>=( Time const &other ) const;
    bool operator> ( Time const &other ) const;

    Time operator+() const;
    Time operator-() const;

    Time operator+( Time const &other ) const;
    Time operator-( Time const &other ) const;

    Time &operator+=( Time const &other );
    Time &operator-=( Time const &other );

    Time &operator++();
    Time  operator++( int );
    Time &operator--();
    Time  operator--( int );

    std::string to_string() const;
};

std::ostream &operator<<( std::ostream &out,
                          Time const &time );
```

12 Examples of required behavior

These examples apply to either class, with `Time` replaced by its actual name.

```
Time t{ 1, 59, 59 };
std::cout << t;           // 1:59:59
std::cout << ++t;         // 2:00:00
std::cout << t++;         // 2:00:00
std::cout << t;           // 2:00:01

t -= Time{ 3, 0, 1 };
std::cout << t;           // -1:00:00
std::cout << -t;          // 1:00:00
std::cout << t.negative(); // 1 (without boolalpha)

Time zero{ 0, 0, 0 };
std::cout << -zero;       // 0:00:00
```

13 Implementation independence

The classes shall satisfy the same externally observable requirements. No requirement prescribes how a duration is stored, how arithmetic is performed, or how components are obtained. The implementations may therefore differ while providing equivalent behavior for the operations specified above.