1 Class source_location [reflection.src_loc]

1.1 Header <source_location> Synopsis [reflection.src_loc.intro]

namespace std {
  struct source_location {
    constexpr source_location() noexcept;

    constexpr uint_least32_t line() const noexcept;
    constexpr uint_least32_t column() const noexcept;
    constexpr const char* file_name() const noexcept;
    constexpr const char* function_name() const noexcept;

    static constexpr source_location current() noexcept;
  };
}

[Note: The intent of source_location is to have a small size and efficient copying.– end note ]

constexpr source_location() noexcept;

Effects: Constructs an object of class source_location.

Remark: The values are implementation-defined.

constexpr uint_least32_t line() const noexcept;

Returns: The presumed line number (16.8) represented by this object.

Remark: Line numbers are presumed to be 1-indexed however an implementation is encouraged to return 0 when the line number is unknown.
constexpr uint_least32_t column() const noexcept;

Returns: An implementation-defined value representing some offset from the start of the line represented by this object.

Remark: Column numbers are presumed to be 1-indexed however an implementation is encouraged to return 0 when the column number is unknown.

constexpr const char* file_name() const noexcept;

Returns: The presumed name of the current source file (14.2) represented by this object as an NTBS.

constexpr const char* function_name() const noexcept;

Returns: If this object represents a position in the body of a function, returns an implementation-defined NTBS that should correspond to the function name. Otherwise, returns an empty string.

static consteval source_location current() noexcept;

Returns: When invoked by a function call whose postfix-expression is a (possibly parenthesized) id-expression naming current, returns a source_location with an implementation-defined value. The value should be affected by `#line` (14.4) in the same manner as for `__LINE__` and `__FILE__`. If invoked in some other way, the value returned is unspecified.

Remark: When a brace-or-equal-initializer is used to initialize a non-static data member, any calls to current should correspond to the location of the constructor or aggregate initialization that initializes the member.

[Note: When used as a default argument (9.3.6), the value of the source_location will be the location of the call to current at the call site. – end note ]

[Example:

```cpp
struct s {
    source_location member = source_location::current();
    int other_member;
    s(source_location loc = source_location::current())
        : member(loc) // values of member will be from call-site
    {}
    s(int blather) : // values of member should be hereabouts
        other_member(blather)
    {}
    s(double) // values of member should be hereabouts
    {}
};
void f(source_location a = source_location::current()) {
    source_location b = source_location::current(); // values in b represent this line
```]
void g() {
    f(); // f’s first argument corresponds to this line of code
    source_location c = source_location::current();
    f(c); // f’s first argument gets the same values as c, above
}

– end example ]

2 Feature macro

We recommend the feature macro __cpp_lib_source_location for this feature.