Proposal for C2y
WG14 N3483

Title:	Mandatory diagnostic message on missing return statements
Author, affiliation:	Igor S. Gerasimov, BME, Hungary
Date:	2025-02-09
Proposal category:	Language changes
Target audience:	General Developers, Compiler Developers

Abstract: Missing return statements in functions lead to undefined behaviour, that may lead to unpredictable and potentially dangerous behavior. To improve code and language safety, this Paper specifies behaviour of missing return statements.

Mandatory diagnostic message on missing return statements

Reply-to: Igor S. Gerasimov (foxtranigor@gmail.com)
Document No: 3483
Date: 2025-02-09

1 Introduction

In C23, if the } that terminates the function body is reached, and the value of the function call is used by the caller, the behavior is undefined (6.9.1.12). This paper aims to improve code and, in general, language safety by requiring that all control flows have either a `return` statement or, alternatively, a `[[noreturn]]` function call. In the case if this limitation is not met, the program is ill-formed and the implementation shall produce a diagnostic message.

2 Current status

Most modern C compilers (GCC with `-Wreturn-type`[1], Clang by default[1], MSVC by default) already provide warnings for missing `return` statements, and this change ensures that such cases are always errors. This should be straightforward to implement by elevating existing warnings to diagnostic messages. However, extra analysis for `[[noreturn]]` can be required.

3 Proposed wording

The wording proposed is a diff from the committee draft of ISO/IEC 9899:2023 dated April 1, 2023. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.9.1p12:

Unless otherwise specified, if the } that terminates the function body is reached, and ~~the value of the function call is used by the caller, the behavior is undefined.~~the return type of the function is non-**void** type, and the function does not contain `return` statements or `[[noreturn]]` function calls in all control flows, the program is ill-formed and the implementation shall produce a diagnostic message.

Updating of 7.21.1p4 **EXAMPLE 1** is not required since all control flows have return statements.

4 Known issues

An additional complication arises with `unreachable()` macro, which is not defined with the `[[noreturn]]` attribute. As a result, the impementation may still expect a `return` statement after `unreachable();`.

5 Examples

EXAMPLE 1 return is not required for `void` functions:

```
1 void foo(int n) {
2   /* ... */ // valid: return type is void
3 }
```

EXAMPLE 2 Missed return statement for $n \leq 0$:

```
int foo(int n) {
  if (n > 0) return n;
  // ill-formed: n <= 0 does not return anything
}
```

EXAMPLE 3 `[[noreturn]]` call or `return` statement in all control flows:

```
[[noreturn]] void exit(int);
int foo(int n) {
  if (n > 0) {
    exit(0);
  } else {
    return n;
  } // valid: return value or [[noreturn]]
}
```

EXAMPLE 4 Interaction with `unreachable()` macro:

```
int foo(int n) {
  if (n > 0) {
    unreachable(); // ill-formed unless unreachable() is not marked with [[noreturn]]
  } else {
    return n;
  }
}
```

6 References

1. `https://godbolt.org/z/W75Y9K618`