MISRA C Rules

Gokula Kannan R

27-02-2025

Table of Contents

Contents

1 Introduction

MISRA C (Motor Industry Software Reliability Association) provides a set of mandatory and required coding guidelines for the C programming language aimed at improving safety, security, and reliability in embedded systems.

2 Mandatory Rules

Mandatory rules are the most stringent category of **MISRA C** rules. They must be followed without exception because violating these rules can lead to undefined behavior, serious safety risks, or software failures. Unlike required rules, mandatory rules do not allow deviations, meaning that all code must comply with them in all cases.

- **Rule 14.1** - All non-empty statements must be enclosed in braces.

```c
#include <stdio.h>

int main(void) {
    if (1) {
        printf("Hello, MISRA C!\n"); // Braces required
    }
    return 0;
}
```

- **Rule 17.3** - NULL Pointer Dereferencing Must Be Avoided.

```c
#include <stdio.h>

int main(void) {
    int num = 10;
    int *ptr = &num; // Pointer assigned a valid address
    if (ptr != NULL) {
        int value = *ptr; // Safe dereferencing
        printf("Value: %d\n", value);
    }
    return 0;
}
```

- **Rule 4.1** – Only ISO C Standard Features Shall Be Used.

```c
#include <stdio.h>

int main(void) {
    int a = 10;
    int b = 20;
    a = a + b; // Standard C operation
    printf("Sum: %d\n", a);
    return 0;
}
```

3 Required Rules

Required rules are critical coding standards that must be followed unless there is a documented deviation. Unlike mandatory rules, required rules allow developers to justify exceptions, but these deviations must be reviewed and approved. Required rules help improve safety, security, and maintainability while allowing some flexibility in specific cases.

- **Rule 10.1** - Implicit conversions should be avoided.

```c
#include <stdio.h>

int main(void) {
    int x = 100;
    float y = (float)x; // Explicit conversion
    printf("Value: %.2f\n", y);
    return 0;
}
```

- **Rule 12.4** - Avoid side effects in expressions.

```c
#include <stdio.h>

int main(void) {
    int a = 5;
    a = 10;
    int b = a + 10; // No side effects in expressions
    printf("Result: %d\n", b);
    return 0;
}
```

4 Optional Rules

Optional rules, also known as Advisory Rules, are recommendations rather than strict requirements. These rules do not need to be followed but are highly encouraged because they help improve code readability, maintainability, and performance.

- **Rule 15.7** - Return values of functions shall be checked where applicable.

```c
#include <stdio.h>

int divide(int a, int b, int *result) {
    if (b == 0) {
        return -1; // Error: Division by zero
    }
    *result = a / b;
    return 0; // Success
}

int main(void) {
    int res;
    if (divide(10, 2, &res) == 0) {
        printf("Result: %d\n", res);
    } else {
        printf("Error: Division by zero\n");
    }
    return 0;
}
```

5 Conclusion

MISRA C is essential for ensuring high reliability and safety in C programming, especially in embedded systems. Following mandatory, required, and optional rules helps developers write more secure and maintainable code.