C Coding Standards

www.robopathshala.com

@alokm014

What Are Coding Rules and Guidelines?

Coding rules and guidelines ensure that software is:

- **Safe**: It can be used without causing harm.
- **Secure**: It can’t be hacked.
- **Reliable**: It functions as it should, every time.
- **Testable**: It can be tested at the code level.
- **Maintainable**: It can be maintained, even as your codebase grows.
- **Portable**: It works the same in every environment.

Why Apply Coding Standard?

There are four key benefits of using coding standards:

- Reduce Code Bugs
- Improve Code Readability
- Ease Code Review process
- Easy to Maintain
- Cost Efficient

General Principles

- Explicit is better than implicit.
- Be consistent.
- It is easier to prevent a bug than to find it and fix it.
- Write as if you are writing for someone else to use and maintain code.
- Use C99.
- Avoid proprietary compiler language keyword extensions.
- Avoid complicated statements.
- Use 4 spaces per indent level.

What : Line Width

How?

All lines must be limited to 80 characters.

Why?

1080p is still one of the most popular resolutions for monitors and it just so happens that with most code editors you can comfortably fit 2 code windows at 80 characters side by side, and even have room for a sidebar if you like that sort of thing.

What : Indentation

How?

Indent level is 4 spaces.

Why?

Greatly improves readability.

What : Braces

How?

Braces must surround each code block, even single line blocks and empty blocks.

Why?

This prevents bugs when near by code is changed or commented out.

What : &&, ||

How?

Unless it is a single identifier each operand of logical AND and logical OR shall be surrounded by parentheses.

Why?

Do not depend on C operator precedence rules, those who maintain the code in the future might miss this.

What : static

How?

‘static’ should be used to declare all variables and function that are unused outside of the modules in which they are declared

Why?

This reduces bugs.

What : volatile

How?

- It should be used to declare global variables accessible by interrupt service.
- It should be used to declare pointer to a memory-mapped I/O peripheral register set.
- It should be used to declare a global variable accessible by multiple threads.
- ‘volatile’ should be used to declare delay loop counters.

Why?

This reduces bugs

What : const

How?

- ‘const’ should be used to declare variables that should not change after initialization.
- It should be used as an alternative to #define for numeric constants.

Why?

This reduces bugs

What : Comment markers

How?

WARNING: Risk in changing block of code.

TODO: Area of code still under construction.

NOTE: Descriptive comment about why.

Why?

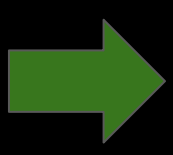
Improves code maintainability

Safe

```
i f ((len > 0 ) && ( i t r < MAX))
{
    ...do something
}
```

Risky

```
i f (len > 0  && i t r < MAX)
{
    ...do something
}
```

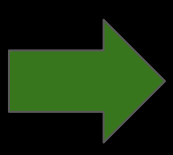


Safe

```
char * x;
char y;
```

Risky

```
char * x; char y;
```

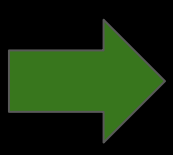


Safe

```
i f ( i t r > 9)
{
    state = END;
}
```

Risky

```
i f ( i t r > 9 ) state = END;
```



Safe

```
i f ( NULL == count)
{
    return true;
}
```

Risky

```
i f ( count == NULL )
{
    return true;
}
```

Do

```
uint8_t num;
num = 9 + 7;
```

Do Not

```
uint8_t num;
num = 9+7;
```

Do

```
#ifdef USE_CRC32
    # define MUL_SIZE 152
#else
    # define MUL_SIZE 254
#endif
```

Do Not

```
#ifdef USE_CRC32
# define MUL_SIZE 152
#else
# define MUL_SIZE 254
#endif
```

Do

```
inline int max( int num1 , int num2 )
```

Do Not

```
#define MAX(A, B) ((A) > (B) ? (A) : (B))
```

Do

```
uint8_t find_shape(uint8_t val)
{
	switch(val)
	{
		case RECT:
			...do something
		break;

		case TRIA:
			...do something
		break;

		default:
			...do something
		break;
	}
}
```

Do Not

```
uint8_t find_shape(uint8_t val)
{
	switch(val)
	{
		case RECT:
		...do something
		break;

		case TRIA:
		...do something
		break;

		default:
		...do something
		break;
	}
}
```

What : if, while, for, switch, and return

How?

Shall be followed by one space when there is additional program text on the same line

Why?

Improves code readability

What : =, +=, *=, /=, %=, &=, |=, ^=, ~=, and !=

How?

Assignment operators shall always be preceded and followed by one space

Why?

Improves code readability

What : Function parameters

How?

Each comma separating function parameters shall always be followed by one space

Why?

Improves code readability

What : for loop

How?

Each semicolon separating the elements of a for statement shall always be followed by one space.

Why?

Improves code readability

What : Statements

How?

- No line should contain more than one statement.
- Each semicolon shall follow the statement it terminates without a preceding space.

Why?

Reduces bugs, Improves code readability

What : Naming

How?

Module names shall consist entirely of lowercase letters, numbers, and underscores. No spaces.

Why?

Reduces bugs

What : Variable Naming

Variable type	Starting characters
Global variable	gVar
Pointer variable	pVar
Boolean variable	bVar
Data Array	aData[ ]

What : Popularly accepted abbreviations

Term	Abbreviation
Minimum	min
Manager	mgr
Maximum	max
Mailbox	mbox
Interrupt Service Routine	isr
Initialize	init
Input/output	io
Handle	h
Error	err

What : Popularly accepted abbreviations

Term	Abbreviation
global	g
current	curr
configuration	cfg
buffer	buf
average	avg
millisecond	msec
message	msg
nanosecond	nsec
number	num

What : Popularly accepted abbreviations

Term	Abbreviation
transmit	tx
receive	rx
temperature	temp
temporary	tmp
synchronize	sync
string	str
register	reg
previous	prev
priority	prio

Do You Find It Helpful?

Save the post, in case you want to see it again

Share your thoughts in the comment section

www.robopathshala.com