THE ULTIMATE 41

GOLDEN RULES OF C PROGRAMMING

INTRODUCTION:

This document describes most of the golden rules that every C developer should take into account.

If someone doesn't totally agree with some of these rules, no problem, the main rule of C programming should always be the same in order to achieve the same goal:

« Write a readable, simple and maintainable software »

GOLDEN RULES:

RULE_01: *Use formatting without type underscores like Pascal or Camelcase for naming variables and functions.*

RULE_02: *Always add a space after each statement: if ,switch, for or while.*

RULE_03: *Use parentheses and braces in all your conditions and this for reasons of clarity and readability of the code.*

RULE_04: *Calling functions in conditions is forbidden so what to do is to call the function before, save its result in a variable and test it in the condition using this variable.*

RULE_05: *For more clarity and readability during mathematical processing and calculations, it is recommended to add a space before and after each operator: "+, -, *, =, /".*

RULE_06: *Use a const type variable in your infinite loop because in unit testing it is very difficult or even impossible to test an infinite* ***while(1)*** *loop.*

RULE_07: *In the decision conditions type: "if, while, for" is forbidden to assign or perform processing operations on the variables.*

RULE_08: *In all conditional "if-else" blocks you must always put the "else" block even if it doesn't contain any processing.*

RULE_09: *In the control conditions, you should never put more than four logical operations "&& or ||" to not have a cyclomatic complexity greater than 20.*

RULE_10: *In switch case blocks, each block that contains a statement must end with a break.*

RULE_11: *Don’t forget also to end with the default case "default" in the conditional block of "switch case".*

RULE_12: *In a “**for**” loop, it is strongly advised not to initialize other variables other than the counter of the loop.*

RULE_13: *Inside a “**for**” loop. Never change the value of its counter.*

RULE_14: *In "**while**" conditions testing an inequality is preferable to a permissive equality test only in case of an equality with a constant **enum**.*

RULE_15: *Local variables contain random data when they are created to avoid any problem related to this it is mandatory to initialize them before their use.*

RULE_16: *The variables except the const must never be reinitialized at the time of the declaration we first make the declaration then in another step their initialization.*

RULE_17: *Variables typed float should never be tested to their exact equality and it is better to add some tolerance (+-%) and that is for reasons of portability and robustness of the code.*

RULE_18: *Be very careful, the volatile keyword indicates that the content of the variable can change at any time in the code during its execution, which implicitly induces at the c compiler level to deactivate any optimization of the code on this resource.*

RULE_19: *Any numeric constant that has no functional meaning in the program should not be declared with a “**#define**”.*

RULE_20: *To avoid any surprises, it is strongly recommended to surround the “**#define**” constants with parentheses.*

RULE_21: *Must be very careful when using the sizeof instruction which can give unexpected results related to the non-alignment of certain data structures.*

RULE_22: *Always initialize the pointer to NULL before using it.*

RULE_23: *Global pointer should never be assigned to an address of a local variable or function.*

RULE_24: *In a "bit field" structure, it is necessary that:*

- *each element has a number of bits greater than 0*
- *the total number of bits is equal to the default type*

RULE_25: *In an “enum” statement, either:*

- *the first element is initialized only*
- *all elements are initialized*

RULE_26: *The use of the "continue" statement is prohibited.*

RULE_27: *The use of the "?" " forbidden.*

RULE_28: *The use of the "goto" statement is prohibited.*

RULE_29: *The use of the break statement is prohibited outside the "switch-case".*

RULE_30: *The use of the "macro" function is very limited to certain cases, otherwise prefer the definition of the classic function.*

RULE_31: *To avoid having memory problems, it is not recommended to use recursive functions: it is the functions that call themselves.*

RULE_32: *It is forbidden to have several "return" in different places of the function.*

RULE_33: *Use the symbol “/* */” for comments instead of “//” which can in some cases have side effects.*

RULE_34: *Comments must give added value they must be precise, clear and short.*

RULE_35: *The line of code must not exceed 80 characters.*

RULE_36: *Use spaces for indentation and never use tabs and for more clarity and visibility at the code level it is necessary to use indentation of 4 spaces.*

RULE_37: *For more coherence and homogeneity in your program, it is recommended to use template cartridges for the structure of the .c and .h files as well as cartridges for the declaration of functions.*

RULE_38: *It is strongly recommended to standardize collaborative tools such as development IDE, set up templates for notes or specification documents, design and test.*

RULE_39: *It is strongly recommended to use a configuration and versioning manager like Git.*

RULE_40: *Using a tool like doxygen allows both to have comments in the code a little more advanced and also to generate documentation of your code which can be easily communicated to third parties for example during code reviews.*

RULE_41: *Use a lint (PC-LINT or MISRA) to check and report warnings during compilation.*

So, those are my top ***41 golden rules to C programmer*** in order to produce a clean, simple and maintenable code free from bug.

I want to end with this:

You can be a successful programmer. Do you believe that? If you don't believe it, you also have the right.

Because your belief associated with commitment determines whether you will succeed or not.

If you don't believe you can do it but you try hard enough to practice, chances are you will fail the first time, but don't stop there.

So do all you can to educate yourself and start believing you can do it!

I hope this has given you some help and instilled in you some good practices for better coding.

These best practices have made my life as a coder so much easier and it can make yours easier and better!

With love,
@Freecoder.dev

THE ULTIMATE 41 GOLDEN RULES OF C PROGRAMMING