Bryce Jensen
CS-350-R4870
4-2 Journal: Best Coding Practices
3/30/2024

Describe **best coding practices** in Embedded C

Some of the best coding practices I can think of include:

Making code modular. When code is structured into smaller functions it allows it to be reusable. One example of how these can be organized in C is by using structs. If the code is made to be as generic as possible (no hardcoded data, etc.) – while still accomplishing the desired goal – it allows it to be used by other programs instead of rewriting a function that one already essentially has.

Creating good comments and documentation. Adding meaningful comments to explain the purpose of functions, variables, and complex logic can dramatically increase the readability of one's code. I will add to this, comments on everything are not necessary. Instead writing the code so that it "reads" like a comment would be far better. For example, using variable names that make sense such as, `timerCounter` instead of just the letters `tC`, is one of the best practices to get in to. Also, follow consistent commenting styles, this also helps with general readability.

Memory management. C does not manage memory the same as other, more advanced languages. Always be mindful of memory constraints. Few embedded systems have an "unlimited" amount of memory. Avoid unnecessary memory allocation and be sure to destroy objects that are no longer needed as soon as it makes sense to do so. Some other ways to manage memory are by using static and const tags for data.

Bit manipulation. This is one we have specifically covered in class. Use bitwise operations for efficient manipulation of individual bits. Of course, these should also be documented clearly as they can get complicated and confusing at times.

Bryce Jensen
CS-350-R4870
4-2 Journal: Best Coding Practices
3/30/2024

Describe **common pitfalls** in Embedded C

Some common pitfalls that come to mind are:

Declaring multiple variables on a single line. It would be far better for each variable to be declared on its own line.

Forgetting to free memory (Bhageria, 2020). As stated above, C does not self-manage memory. This means that the code must account for this by freeing that memory manually. Otherwise, that variable will be stored indefinitely, even if the code does not need it again for the rest of the program.

Bryce Jensen
CS-350-R4870
4-2 Journal: Best Coding Practices
3/30/2024

References

Bhageria, V. (2020, December 11). *Most common pitfalls in C Programming Language and how to avoid them*. Retrieved from nerdyelectronics.com: https://nerdyelectronics.com/most-common-pitfalls-in-c/