Cameron, Jay, Nisha

CS 4900

10/12/16

**Comparisons between C Programming Styles**

Among many different programming styles for C, we chose three main C coding standards i.e NASA’s C style guide, C11, and C99. These coding and styles and characteristics have been discussed in this report. We’ve also made an attempt to discuss their cost factor, availability, and easiness to read and understand. The first one is NASA’s C style guide.

**NASA C Style Guide:**

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created to investigate the effectiveness of software engineering technologies when applied to the development of applications software. They define their coding standards based on Organisation, Easy to read, Easy to understand, Maintainability, and Efficiency.

For maximizing the readability and maintainability of C code:

1. Organize programs using encapsulation and information hiding techniques.
2. Enhance readability through the use of white space.
3. Add comments to help others understand your program.
4. Create names that are meaningful and readable.
5. Follow ANSI C standards, when available.

Organization of Program:

1. Create a README file to document what the program does.
2. Group the main function with other logically related functions in a program file.
3. Use module files to group logically related functions (not including the main function).
4. Use header files to encapsulate related definitions and declarations of variables and functions.
5. Write a Makefile to make recompiles more efficient.

NASA’s C style guide is available for free online. They have described the coding in very simpler terms, and even non technical person can understand it very well. In terms of reading, it’s actually very easy to follow as they have provided examples with each topic.Formatting standards are included within the style guide. There’s not a separate software for formatting. The second in line is C11.

**References**:

<http://homepages.inf.ed.ac.uk/dts/pm/Papers/nasa-c-style.pdf>

**C11:**

C11 is the newest C programming standard to date. The standard is an extension to the C99 standard and adds more functionality to C. One big feature that the C11 standard allows for is extensions for decimal floating point arithmetic to allow for higher degrees of accuracy. It also allows for bounds checking as well as creating a better memory model for multithreading to be more efficient. Security in the standard include new features such as getting rid of the get() function, which has been regarded as unsafe for use, making functions such as fopen() more safe with IO, and including static assertions which allows for catching of errors in the preprocessing phase that were difficult to catch before. Lastly, the C11 standard is free to use for anyone to use and does not require any purchasing of materials to learn. Now that we’ve discussed about C11, the last one is C99.

**References:**

* <http://www.open-std.org/jtc1/sc22/wg14/>’
* <https://en.wikipedia.org/wiki/C11_(C_standard_revision)>
* <http://blog.smartbear.com/codereviewer/c11-a-new-c-standard-aiming-at-safer-programming/>

**C99:**

C99 widely known as ISO is the next standard introduced in 1990 after ANSI. Changes in ISO standard have been made for years and the new edition was published in 1999. In particular, for basic C programs this standard is not so complicated to use and GCC compiler has a complete support for this standard. ISO is designed to promote the portability of C programs among a variety of data-processing systems. It is intended for use by implementers and programmers. It also, specifies the form and establishes the interpretation of programs written in the C programming language. Following are the coding standards for ISO:

|  |  |  |
| --- | --- | --- |
| • [Formatting](https://www.gnu.org/prep/standards/html_node/Formatting.html#Formatting): |  | Formatting the source code. Such as, indentation for easy readability. |
| • [Comments](https://www.gnu.org/prep/standards/html_node/Comments.html#Comments): |  | Commenting the work so user can know the purpose of the program. |
| • [Syntactic](https://www.gnu.org/prep/standards/html_node/Syntactic-Conventions.html#Syntactic-Conventions)  [Conventions](https://www.gnu.org/prep/standards/html_node/Syntactic-Conventions.html#Syntactic-Conventions): |  | Clean use of C constructs which helps finding bugs and unclear code. |
| • [Names](https://www.gnu.org/prep/standards/html_node/Names.html#Names): |  | Naming variables, functions, and files gives useful information about them |
| • [System Portability](https://www.gnu.org/prep/standards/html_node/System-Portability.html#System-Portability): |  | Portability among different operating systems ensures the correct output. |
| • [CPU Portability](https://www.gnu.org/prep/standards/html_node/CPU-Portability.html#CPU-Portability): |  | Supporting the range of CPU types to handle the differences among them |
| • [System Functions](https://www.gnu.org/prep/standards/html_node/System-Functions.html#System-Functions): |  | Portability and “standard” library functions to write clearer and faster code. |
| • [Internationalization](https://www.gnu.org/prep/standards/html_node/Internationalization.html#Internationalization): |  | Techniques to interpret the text in a program into various languages |
| • [Character Set](https://www.gnu.org/prep/standards/html_node/Character-Set.html#Character-Set): |  | Use ASCII by default. |
| • [Quote Characters](https://www.gnu.org/prep/standards/html_node/Quote-Characters.html#Quote-Characters): |  | Use "..." or ’...’ in the C locale to make programs look better. |
| • [Mmap](https://www.gnu.org/prep/standards/html_node/Mmap.html#Mmap): |  | How you can safely use mmap to handle different kind of files. |

However, there are many different things that ISO doesn’t specify such as:

* the mechanism by which C programs are transformed and invoked for use by a data-processing system;
* the mechanism by which input and output data are transformed by a C program;
* the size or complexity of a program and its data that will exceed the capacity of any specific data-processing system or the capacity of a particular processor;
* all minimal requirements of a data-processing system that is capable of supporting a conforming implementation.

Overall, all the discussed standards stand solid and are widely used by all kind of users. We’ve decided to use the C11 for our Quad Solver and other C projects as it’s the most latest C standard guide. Also, there are extensions for decimal floating point arithmetic to allow for higher degrees of accuracy.

**References**:

<https://gcc.gnu.org/onlinedocs/gcc/Standards.html>

<https://www.gnu.org/prep/standards/html_node/Writing-C.html>

<http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=57853>