**The Power of C**

***1st Edition***

Author: Thomas K.Caston

\_Mandarin: 阮玉嘉康

\_Korean: 즈하캥

\_Vietnamese: Quyền Thư Gia (Nick Name)

CodeName: Krnl\_D0cum3nt3dCL01ng

Published: 16/07/2025

* **Getting started (what is C, and what its implications are)**

*C is a statically typed , general-purpose, high-performance procedural programming language ,created in the early 70’s*

*by Dennis Ritchie at Bell labs, it was originally developed to build the UNIX ( UNiplexed Information Computing System ) operating systems,and overtime it became one of the most widely used programming languages in the world*

* **Evolution of C Standards**

Whilst not changing drastically in terms of syntax and code implementations, the evolution of C standards provides crucial historical context and highlights the language's continued development as following:

***>C89/C90 (ANSI C / ISO C90): Being the first standardized version of the C language,*** it was largely based on the book *The C Programming Language* by Kernighan and Ritchie *K&R*. It established a common ground for C compilers, promoting portability. Key features include function prototypes and the void keyword.

***>C99 (ISO C99): Published in 1999, C99 introduced several new features to the language.*** Notable additions include // comments (borrowed from C++), long long int, variable-length arrays (VLAs), flexible array members, designated initializers, restrict keyword for pointers, and improvements to floating-point arithmetic. Aiming to improve support for scientific and numeric programming (or floating point of arithmetic).

***>C11 (ISO C11): Released in 2011, C11 added core programming features (Foundational Concepts for Modern Languages)*** like anonymous structs and unions, type-generic expressions (\_Generic), alignment specifications (\_Alignof, \_Alignas), and support for multi-threading (<threads.h>) and atomic operations (<stdatomic.h>). This standard brought C closer to modern programming implications, especially that for concurrent programming.

***>C18 (ISO C18): Published in 2018, C18 (also known as C17)*** is primarily a minor revision of C11. Its main purpose was to address defects, clarify ambiguities, and make minor corrections to the C11 standard but didn't introduce significant new language features but focused on maintaining stability and correctness.

⇒Showing that C is *not a static language* but has adapted over time to include new functionalities and address issues, though often at a slower pace than languages like C++.

* **Key features of C**

***Compiling process***: C code can be translated into machine understanding levels of code,making it fast for realtime implications as following:

.GNU \*GNU's Not Unix (includes GCC, GDB, Glibc, coreutils...) widely used

in 98% applications nowadays.

.Can be used to write compiler or interpreter for something more high level like Python, JavaScript *\_(Python and JavaScript cannot be directly translated to binary 0101011 type of machine’s understanding)\_.*

***Whilst maybe mixed up with C++ (most majority usually refers to C/C++) C itself is a procedural language having core features like:***

***.Low-level access:*** *you as a developer can directly manipulate memory (usually the RandomMemoryAccess) and hardware (with the help of pointers), more closer to ASM than something like Java but still possess the core programming concepts like:*

***→Procedural Programming***: Code is structured in functions and procedures *(C runs like Python / Python runs like C by compiling/interpreting lines by lines)*.

***→Portable AF:*** Lightweight, machine/kernel-level controlling gives C the power to be compiled on different systems with minimal changes.

***→Efficient:*** Great for performance-critical realtime applications (High Performance Games / Applications, Servers, CLi Apps , Operating Systems and most definitely Kernels like Linux,...)

***🡺Fun fact: Most of applications you use/see/interact with on a server nowadays were written on but they are now getting*** ***replaced with more modern language alternatives like:***

***→C++:*** *It’s C but OOP , it functions with a C-like performace whilst with newer, more robust features for nowadays programming languages standards.*

***→Go:*** *More functional than OOP C++ but less of a procedural thing like C, its replacing C/C++ as a alternative to host server.*

***→Rust:*** *Same for Golang but more low-level touching than Golang (faster compile-time), people compare C to these new languages but here is the catch, they are newer than C, and they are OOP friendly****. That being said*** *most programming languages can function and perform about the same IF YOU don’t look at the compilation time, with every library nowadays you can code even a website in such ancient languages like C.*

***→Experiments like Carbon*** *(the supposedly replacement for C/ C++ made by GOOGLE)*

***→Zig:*** *Another modern alternative (no specific comments provided) but it would be a good option*

* **Example Codes for C**
* ***Hello World! (Programmer’s First Programme)***

**#include <stdio.h>**

**int main() {**

**printf("Hello, World!\n");**

**return 0;**

**}**

* ***Basic Input & Output (Processing basic string input and giving basic string output)***

**#include <stdio.h>**

**int main() {**

**int age;**

**printf("Enter your age: ");**

**scanf("%d", &age);**

**printf("You are %d years old!\n", age);**

**return 0;**

**}**

* ***Else if (Core Concept, like 0&1 but in C and every programming language ever made, it’s a function that guide the computer to do a set of things when True or False or somewhere in between)***

**#include <stdio.h>**

**int main(){**

**int score;**

**printf("Enter your score (0–100): ");**

**scanf("%d", &score);**

**if (score >= 90) {printf("Grade: A\n");}**

**else if (score >= 80)**

**{printf("Grade: B\n");}**

**else if (score >= 70)**

**{printf("Grade: C\n");}**

**else if (score >= 60)**

**{printf("Grade: D\n");}**

**else**

**{printf("Grade: F (uh oh!)\n");}**

**return 0;**

**}**

* ***For loop***

**#include <stdio.h>**

**int main()**

**{**

**for (int i = 1; i <= 100; i++) {**

**if (i % 15 == 0) printf("FizzBuzz\n");**

**else if (i % 3 == 0) printf("Fizz\n");**

**else if (i % 5 == 0) printf("Buzz\n");**

**else {printf("%d\n", i)};**

**}**

**return 0;**

**}**