# C (Programming language)

**Efficiency**

C programming language is known for its efficiency, this is mostly due to its variety of data types and powerful operators. Compilers, libraries and interpreters of other programming languages are often implemented in C. For example, interpreted languages like Python, Ruby, and PHP have their primary implementations written in C, which shows its efficiency. Another aspect of C is the speed, since C is the lowest-level language among all high-level languages (closer to hardware, it combines the capabilities of an assembly language with features of high-level language), it is much less complex for the human and for the computer, meaning it is more efficient (faster).
As code written in C cannot be executed directly, it needs to be compiled, which means every time the code is modified, it has to be compiled again. This means testing and debugging requires additional time, depending on the size of the code. On the other hand, if the code is compiled, it runs very efficiently.
However, there are also downsides of C language. The compiled executable is not cross-platform, meaning that the executable on one machine does not work on the other machine, as the executable is made for every platform differently, depending on the architecture of computer (hardware and operating system). [1][2][3][4]

**Simplicity**

C is somewhat easy to use as it uses structured approach. It has vast collection of keywords, operators, built-in functions and data types which make it also efficient and powerful. C is not the easiest language to start learning, as it takes a bit of time to get used to the syntax, compared to Python, for example, where syntax is comparatively easier in the starting phase. However, after learning C as the first language, learning any other language is relatively simple.
Simplicity can be also demonstrated in a comparison of the main books in programming language, as “The C programming language” by B. Kernigham and D. Ritchie is fairly short, compared to its extension C++ “C++ programming language”, which is about 4 times longer. [4][5]

**Orthogonality**

In computer programming, orthogonality means that operations do not have side effects, they change just one thing without affecting others. There should be only one way to change each property.
C programming language is not orthogonal, meaning it is not the simplest and easiest language to learn and use, as it is inconsistent in its treatment of concepts and language structure. As this language is not orthogonal, user and/or programmer has to worry about side effects of instructions, meaning extra work has to be done to suppress or work around the side effects.
Examples of why C is non- orthogonal language:

* Structures (but not arrays) may be returned from a function.
* An array can be returned if it is inside a structure.
* A member of a structure can be any data type (except void, or the structure of the same type).
* An array element can be any data type (except void). Everything is [passed by value](https://en.wikipedia.org/wiki/Evaluation_strategy#Call_by_value) (except arrays).

[6][7]

**Definiteness (Syntax and semantics)**

ISO/IEC 9899:2018, also known as C18 is the most recent standard for C programming language, which defines the syntax and semantics of the C programming language. It leaves some behaviours of many C constructors as undefined, which frees the compiler to do whatever is easiest or most efficient. For example, the following code displays different outputs, depending on the compiler (outputs for the following program can be: 0, 1, 2, or 1, 0, 2, etc.).

#include<stdio.h>

int main() {

    int i = 0;

    printf("%d %d %d\n", i++, i++, i);

    return 0;

}

As a programmer, it is never good idea to use programming constructs whose behaviour is undefined or unspecified.
C allows the declared variables to be uninitialized, which result in different outputs, depending on the compiler, memory etc, when used incorrectly. C also has some context-dependant syntax, for example, symbols \* and &. The first one may refer to multiplication, pointer as a data type and it is also used to dereference a pointer. The second symbol is the “address-of”, which is a unary operator which returns the address of its operand in memory (a pointer to it), it can also be used as bitwise AND operation.
Some other basic syntax rules for C program are:

* C is a case sensitive language so all C instructions must be written in lower case letter.
* All C statement must end with a semicolon.
* Whitespace is used in C to describe blanks and tabs.
* Whitespace is required between keywords and identifiers.
* C operators can be classified into following types (Arithmetic, Relational Logical, Bitwise, Assignment, Conditional, Special operators).
* There are 32 keywords, that are preserverd words that have special meaning in C language.
* There are rules for identifiers (names for variables, constants, functions etc.).
* C language supports 2 different data types: primary (int, float, etc) and derived (array, structure, etc.).

[8][9][10]

**Reliability**

C programming language is not considered reliable. Firstly, it does not support error/exception handling directly, which means various anomalies and bugs can occur and catching the errors or taking appropriate responses is not easy.
C is also prone to have:

* Buffer overflows: which means while writing data to a buffer, program can overrun the buffer's boundary and overwrite adjacent memory locations.
* Dangling pointers: which means pointers can point to an invalid object of the appropriate type, when the object is itself deleted from memory or destroyed by the stack frame on return.
* Double frees: when free() is called more than once on the same value, which causes memory leak. This can lead to the program to crash or in some circumstances, alter the execution flow. By overwriting particular registers or memory spaces, an attacker can trick the program into executing code of his/her own choosing, often resulting in an interactive shell with elevated permissions.
* Memory leaks: this occurs, when a computer program incorrectly manages memory allocations. This can lead to reducing the performance of the computer or in worst case scenario application fails due to memory thrashing.
* Race conditions: application depends on the sequence or timing of processes or threads. A concept called a “data race” is a a type of race condition, which especially can occur in C programming language, as the memory model defined in C has undefined behaviour for programs that contain data race. This anomaly or bug is difficult to reproduce or even fix, as it depends on relative timing between interfering threads.
* Undefined behaviour (More of it in Definiteness (Syntax and semantics) chapter).

[11][12][13][14][15][16]

**Program verification**

Without any official tools or mechanisms, it is really hard to verify the C code, to prove that the program satisfies a formal specification of its behaviour, because the code can run months or even years with faults in it, until these faults become errors.
On the other hand, there are third party tools to implement the formal verification, for example, CompCert is a formally verified optimizing compiler for a large subset of the C99 programming language. [17][18]

**Abstraction faclities (data and procedural)**

C is a procedural language, as the code is written as a sequence of instructions. Procedural language means that the user has to specify “what to do” and also “how to do” (step by step procedure). This also means that this language only returns restricted data types and allowed values. This data type determines the type of data a variable will hold. As mentioned before, C programming language supports 2 different data types, primary (int, float, etc) and derived (array, structure, etc.). Compared to other programming language, for example Python, every variable which is used in the program must be declared as what data-type it is. [10][19]

**Portability**

C is considered portable language, as the source code developed on one machine (operating system and hardware architecture), can be compiled and run almost identically on other machine with different environment (with different operating system and hardware architecture) with little to no modifications. This is possible due to C compilers, that are available for almost every operating system and hardware architecture, as the compilers are supplied by hardware vendors, operating systems etc.

*References:*

[1] <https://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming>
[2] <https://www.gnu.org/software/gnuastro/manual/html_node/Why-C.html>
[3]<http://www.tud.ttu.ee/im/Vladimir.Viies/materials/IAS0450_Prog.lang.anal/progLangAnal/progLangAnal19/>
[4] <https://www.quora.com/What-are-the-main-characteristics-of-c-programming>
[5] <https://www.quora.com/Which-is-easier-to-start-with-programming-C-or-Python>
[6] <http://www.catb.org/~esr/writings/taoup/html/ch04s02.html#orthogonality>
[7] [https://en.wikipedia.org/wiki/Orthogonality\_(programming)](https://en.wikipedia.org/wiki/Orthogonality_%28programming%29)
[8] <https://www.iso.org/standard/74528.html>
[9] <https://www.geeksforgeeks.org/c-programming-language-standard/>
[10] <https://www.studytonight.com/c/operators-in-c.php>
[11] <https://en.wikipedia.org/wiki/Buffer_overflow>
[12] https://en.wikipedia.org/wiki/Dangling\_pointer
[13] <https://www.owasp.org/index.php/Doubly_freeing_memory>
[14] <https://en.wikipedia.org/wiki/Memory_leak>
[15] <https://blog.regehr.org/archives/761>
[16] <https://en.wikipedia.org/wiki/Race_condition>
[17] <https://ts.data61.csiro.au/publications/nicta_full_text/7629.pdf>
[18] <https://en.wikipedia.org/wiki/CompCert>
[19] <https://www.geeksforgeeks.org/difference-between-procedural-and-non-procedural-language/>