KU LEUVEN

C: an introduction

Some history

1

introductory concepts

C, n.: A programming language that is sort of like Pascal except more like assembly except that it isn't very much like either one, or anything else. It is either the best language available to the art today, or it isn't.

— Ray Simard

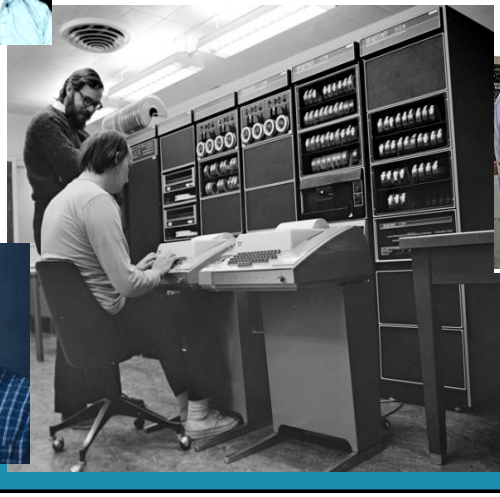
4

C: history

- C: was developed in the years 1969 to 1973 by Dennis Ritchie at Bell Labs based on the work of Ken Thompson
  - strongly influenced by a language called BCPL which itself was a derivative of Algol.
  - closely linked to the development of the Unix operating system.
- 1978: Kernighan & Ritchie
  "*The C Programming Language*"
  first published description (a pseudo standard)
- 80's: C has also gained substantially in use and availability from the explosive expansion of the Personal Computer market

Dennis Ritchie

Brian Kernighan



Ken
Thompson

C: time frame

- C was originally created as a language with which one could write operating systems. (like Unix).
- It was created in the 70's:
  - Computers were much slower.
  - Computers were much more expensive
  - Memory was incredibly expensive (> $0.01/bit!)
  - Few engineers/programmers could type fast
  - Serious coding was done in assembler

C: standard

- The informal description of the language in Kernighan & Ritchie's book was not good enough.
- Many C-dialects were developed – compatibility problems
- ANSI established a committee known as X3J11 in 1983 to set a standard.
  "*unambiguous, machine independent*"
- A report defining the language at the end of 1989 was produced. report X3.159 but the standard was soon taken over by ISO with the designation ISO/IEC 9899-1990.
- https://en.wikipedia.org/wiki/C_(programming_language)#History

C: standard

year	standard
1972	C invented
1978	The C Programming Language published; first specification of language (pseudo standard)
1989	C89 standard (known as ANSI C or Standard C)
1990	ISO C, ANSI C adopted by ISO, known as C90 (slightly modified)
1999	C99 standard • mostly backward-compatible • not completely implemented in many compilers
2011	C standard C11 (small step up from C99)
2017/2018	C standard C17 / C18 under development in 2017, and officially published in 2018, C17 is also commonly referred to as C18.

programming languages

- https://www.levenez.com/lang/
- Programming index (tiobe)
  - https://www.tiobe.com/tiobe-index/
- See also
  - https://www.pluralsight.com/blog/software-development/why-every-programmer-should-learn-c
  - https://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming

C: characteristics

- C is a *general purpose programming language*
- *high level*:
  - supports different data types
  - fundamental control flow constructions
  - functions
  - standard libraries
- *low level*:
  - macro
  - pointers
- *https://en.wikipedia.org/wiki/C_(programming_language)*
- *https://www.geeksforgeeks.org/features-of-c-programming-language/*

C: characteristics

- Language that “bridges” concepts from high-level programming languages and hardware
  - Assembly = low level
  - Python = Very high level
    Abstracts hardware almost completely
- C maintains control over much of the processor
- C can be dangerous
  - Type system error checks only at compile-time
  - No garbage collector for memory management
    Programmer must manage heap memory manually
- Source: https://redirect.cs.umbc.edu/~tinoosh/cmpe311/

C: characteristics

- C is a procedural language
  - Problem solving with focus on defining functions that perform a single service
  - Data is global or passed to functions as parameters
- C libraries consist of predefined functions.
  - Char/string functions (`strcpy, strcmp, …`)
  - Math functions (`floor, sin, …`)
  - Input/Output functions (`printf, scanf, …`)

C: pro

- C is good for:
  - dealing with devices & operating systems,
  - doing raw number-crunching or graphics-crunching,
  - for run-time systems
- C is much more structured and expressive than assembly language
- C is portable: move programs by making a few changes and recompiling, not rewriting

C: pro

- when working under Linux/UNIX, you will encounter C
  OS is mainly written in C.
- C supports logical constructions and data structures. Knowing C helps in stepping up to other languages
- C++ is an extension of C, everything you learn in C, you can use it in C++.
- A good compiler, `gcc`, is available for free, open source and generally excellent
  https://gcc.gnu.org/

C: pro

- *Low-level nature* helps in writing efficient code.
- C is a *small* language (small number of keywords)
- C is *common* (C code is easy to find).
- C is *mature* (*stable)* (features are well-tested and well-understood) Standard helps
- C can be very readable
- availability of different compilers on a broad scale of platforms.

C: contra

- lacks extensive standard libraries (composed of predefined functions)
- freedom in programming style can result in unreadable programs
- not that high level (OO)
- can be dangerous
  - low level nature can crash the system
  - easy to have:
    - infinite loops
    - illegal memory access

C: contra

- 1 line of code can contain a lot of information
- compiler will not easily track down programming errors, as does a Fortran or pascal compiler
- 'C takes the point of view that the programmer is always right' -- Michael DeCorte
- 'C is a razor-sharp tool, with which one can create an elegant and efficient program or a bloody mess.' -- B. Kernighan
- http://www.ioccc.org/
  (*The International Obfuscated C Code Contest)*