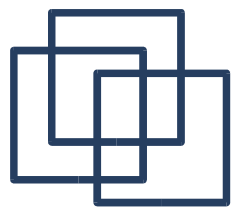


## Lecture 06: Introduction to C Programming

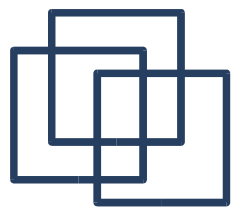
**Marshall McMullen**  
**Computer Science Department**  
**University of Arizona**



# Outline

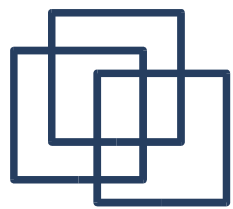
---

- History of C
- Strengths and weaknesses of C
- C vs C++
- C vs Java



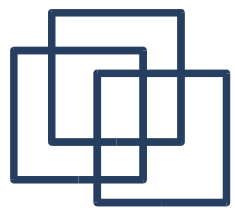
# History of C

- UNIX was originally written in assembly code, which was difficult to debug and develop, and horribly painful to enhance
- A higher-level language with greater abstraction was needed
- Ken Thompson (original author of UNIX) thus developed a small language named B, based on BCPL, as systems programming language from the 1960s, which is descended from Algol-60
- Dennis Ritchie soon joined the development of the UNIX project and began developing programs in B
- In 1970, Bell Labs acquired a PDP-11 and Thompson rewrote a portion of UNIX in B
- By 1971 it was clear that B was not well suited to the PDP-11, so Ritchie began to develop a modified version of B which he named NB (“new B”)



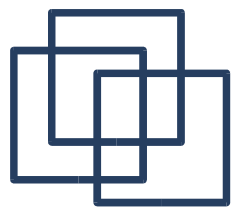
# History of C (cont)

- NB soon began to diverge greatly from B, so Ritchie renamed it C
- By 1973 the language was stable enough that UNIX was completely rewritten in C
- In 1978, the first milestone towards standardization of C was achieved when *The C Programming Language*, by Brian Kernighan and Dennis Ritchie was published
  - Known as the “K&R Book” or “The White Book”
  - Absent a formal standardization, this served as the *de facto* standard
- Unfortunately K&R book was silent and/or fuzzy on certain critical language features, so compilers implemented them differently
- American National Standards Institute (ANSI) approved formal standardization in 1989. Approved by International Standards Organization (ISO) in 1990. Known as “ANSI C” “ANSI/ISO C” or just “Standard C”



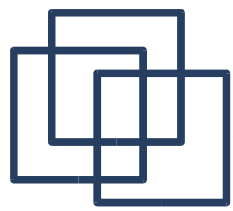
# History of C (cont)

- Original K&R version of C often called “K&R C” or “Classic C”
- Standard C is completely backwards compatible with Classic C and you may still see some programs using Classic C conventions



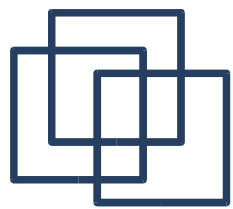
# C's Design Philosophy

- C was designed *specifically* for writing Operating Systems and other system software
- C is a low-level language because can access machine level concepts such as memory addresses, bytes, and even bits
- C is a small language with more limited set of features than other languages. Additional features provided through standard libraries.
- C is a permissive language that allows you to do absolutely anything you want because it assumes you know what you are doing!



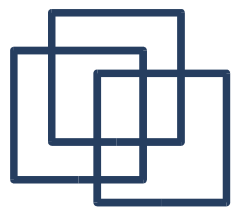
# Strengths & Weaknesses of C

- C's strengths and weaknesses direct result of its design philosophy
- Strengths:
  - Efficiency – designed to replace assembly code where speed critical
  - Portability – due to tight coupling with UNIX and ANSI/ISO standards
  - Power – extensive data types and operators and direct memory access
  - Flexibility – very few restrictions, allows programmer to do anything
  - Standard Library – extensive library of additional features provides power
  - Integration with UNIX – very powerful when combined with UNIX
- Weaknesses
  - Error-prone due to high flexibility and latitude given to programmer
  - Difficult to understand because terse and extremely flexible
  - Difficult to modify – no “packages” or “modules”



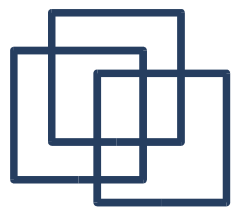
# C vs C++

- C++ is an **Object-Oriented** language while C is an **Imperative (or Procedural)** Programming Language
  - In C, functions and variables are not associated with objects
- C++ is basically a superset of C that adds Object Oriented Programming and other more advanced language features
- C++ code is generally as fast and as efficient as C code but
- C++ is far more complex than C
- Both C and C++ are compiled into *machine code* (platform dependent)
- So why learn C instead of C++?
  - C++ is a superset of C, and is far more complex, so best to learn C first
  - **Lots** of C code exists – odds are you will have to use/maintain it
  - Small programs do not generally benefit from C++



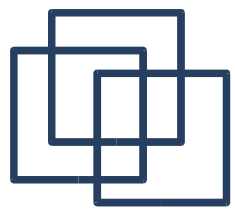
# C vs Java

- Java developed directly from C++
- Like C++, Java is an **Object-Oriented** language
- Syntax of Java programs *very similar* to C programs – so what you know about Java programming will help you program in C
- C provides direct access to memory and pointers to primitives not just references and therefore pointer arithmetic can be performed
- C has no string type – programmers handle manually and through adherence to standard conventions
- C is compiled into machine code, Java produces byte code
  - Java is platform independent (“write once, run everywhere”) while C must be recompiled for each platform
  - Advantages/Disadvantages?



# C vs Java (cont)

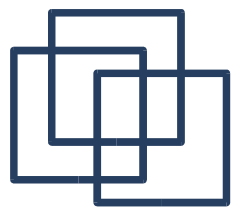
- C has no garbage collection, so all memory management is done directly by the programmer
- C types can differ in size on different platforms, while Java made this uniform
- Java comes with a rich set of classes for nearly all common data structures, while in C you have to write it yourself
- C programs generally *faster* and more *efficient* than Java counterparts due to fundamentally opposite design differences
  - C's primary focus on *speed* and *efficiency* – *no concern for safety*
  - Java's primary focus on *portability* and *safety* – less focus on *speed/efficiency*



# Reading Assignments

---

- *C Programming: A Modern Approach*. By K.N. King.
  - Chapter 1



# Acknowledgments

---

- *C Programming: A Modern Approach*. By K.N. King.