ORIGIN ACADEMY · COURSE SYLLABUS

C Programming

Learn C properly from scratch: memory and pointers, data structures, processes, POSIX threads and network sockets, with the tools professionals use.

14 weeks · about 10 hours a week · live cohort of up to 10
Curriculum last updated 19 November 2026

Availability

This course opens soon. Join the waitlist on the course page to hear first about dates and prices.

Before you start

- No programming experience needed
- Command-line basics help; Phase 0 shows you what you need
- English good enough to follow technical documentation and live sessions: about a credit (C6 or better) in WASSCE or NECO English Language, or an equivalent such as KCSE or IGCSE English (roughly CEFR level B1). You don't need the certificate: Phase 0 is free, so you can check for yourself
- A laptop running Linux, macOS, or Windows with WSL

How the course works

- A live cohort of up to 10 students, with a mentor.
- At least half your time is hands-on: every module ends with a lab.
- Live sessions are recorded, so you can rewatch them or catch up.
- Questions between sessions are answered within 24 hours on weekdays.
- A mentor reviews every phase project, like a pull request at work.

Phase 0 is free for everyone, before you enroll.

Curriculum

PHASE 0 · WEEK 1 · FREE

Getting Started & Tooling

Install a toolchain, see how source becomes a program, and set up Make, GDB and Git from day one.

- Why C, and the C standards (C99 to C23)
- GCC and Clang: compile, assemble, link
- Your first program and main()
- Make and a debugger from day one

You build: Hello world with your own Makefile

PHASE 1 · WEEKS 2–3

Core Language

Types, operators, control flow and functions, scope and storage duration, written the way professionals write them.

- Variables and data types
- Fixed-width integers and type conversion
- Operators: arithmetic, logical and bitwise
- Control flow
- Functions and recursion
- Scope, storage duration, static and extern

You build: Command-line calculator

PHASE 2 · WEEK 4

Arrays & Strings

Arrays, C strings, command-line arguments and the classic string-handling pitfalls.

- Arrays
- C strings and string.h
- Command-line arguments
- Buffer overflows and safe string handling

You build: Text statistics tool

PHASE 3 · WEEKS 5–6

Pointers & Memory

Pointers and pointer arithmetic, void and function pointers, the stack and the heap, and finding memory bugs.

- Pointer basics
- Pointer arithmetic and arrays
- void pointers and function pointers
- Stack vs heap
- malloc, calloc, realloc and free
- Finding leaks with Valgrind and AddressSanitizer

You build: Dynamic array library

PHASE 4 · WEEK 7

Structs & Data Structures

Structs, unions, enums and typedef; linked lists, dynamic arrays, hash maps and ring buffers; opaque types.

- Structs, unions, enums and typedef
- Linked lists
- Hash maps
- Ring buffers
- Opaque pointers and clean APIs
- Unit testing with Unity

You build: Hash map library with tests

PHASE 5 · WEEK 8

Preprocessor, Linkage & Undefined Behavior

Header files, macros and conditional compilation, declaration vs definition, linkage, const, volatile and restrict, and undefined behavior.

- Header files and include guards
- Macros and conditional compilation
- Declarations, definitions and linkage
- const, volatile and restrict
- Undefined behavior and how to avoid it

You build: Multi-file library with a clean public header

PHASE 6 · WEEK 9

Files, Processes & Signals

File I/O, errno and exit codes, fork/exec/wait, pipes and signals, and the debugging toolkit: gdb, strace, ltrace, perf and Valgrind.

- File I/O and streams
- errno and exit codes
- Processes: fork, exec and wait
- Pipes
- Signals
- The debugging toolkit: gdb, strace, ltrace, perf and Valgrind

You build: Mini shell with pipes

PHASE 7 · WEEKS 10–11

Concurrency with POSIX Threads

Threads, mutexes and condition variables, race conditions and deadlocks, C11 atomics and the memory model, and thread pools.

- Why threads: processes vs threads
- Creating and joining pthreads
- Mutexes and race conditions
- Condition variables and producer/consumer
- Deadlocks and how to avoid them
- C11 atomics and the memory model
- Finding races with ThreadSanitizer

You build: Thread-pool job runner

PHASE 8 · WEEK 12

Network Programming with Sockets

The BSD sockets API, TCP and UDP clients and servers, byte order, getaddrinfo, and serving many clients with threads or poll().

- How TCP and UDP work
- The sockets API: socket, bind, listen, accept, connect
- Byte order and getaddrinfo

- A TCP echo server and client
- UDP sockets
- Many clients: one thread each, or poll()

You build: Multi-client TCP chat server

PHASE 9 · WEEKS 13–14

Capstone

Build a complete C application to a written brief, with code review.

- Design review
- Build and test
- Code review and hardening
- Demo

You build: Multi-threaded TCP key-value server

Your certificate

You earn it by doing two things:

- Attend every live session. If you miss one, make it up by watching the recording and submitting that week's lab within 7 days. You can do this up to 2 times per course.
- Score more than 50% on the final evaluation at the end of the course.

Your Enterprise Readiness Rating is separate: it scores your projects, capstone and assessments so employers can see how job-ready you are.

Questions

Email academy@originclouds.io, or see the course online: https://academy.originclouds.io/courses/c-programming