Syllabus for CS-8001 “Introduction to C”

Course Summary

The objective of this course is for you to learn C—the language, its philosophy, its tool chain, and techniques commonly found in low-level programming. You will apply these concepts by building an interpreter for a language called PSI.

Textbook

Your assigned readings will be from materials that I have written. For more depth, I will recommend external resources including chapters from Brian Hall’s “Beej’s Guide to C Programming”, freely available here

Schedule (Tentative)

Office Hours will be held at 8:00pm Atlanta (Eastern) time starting **August 25**. Except for **Wednesday, September 3**, all office hours will fall on **Monday**.

These dates are tentative and unofficial—the official deadlines are in Canvas.

Week of:	Topic		Due:
18 Aug	Module 1: Why Use C? How C Works; What Compilers Do Fri 22: Phase 0 (“Stackman”) released		
25 Aug	Module 2: Base Types; Functions and Program Structure Fri 29: PSI Phase 1 Released		A1 28 Aug AoE
1 Sep	Module 3: Pointers; Arrays and Strings; Undefined Behavior Is Bad/Good/Undefined		A2 4 Sep AoE
8 Sep	Module 4: Structs; Stack and Heap		A3 11 Sep AoE
15 Sep	Module 5: Objects and Ownership; the FILE Object		A4 18 Sep AoE

22 Sep	Module 6: Security; Enums; Function Pointers		A5 25 Sep AoE *(includes "Phase 0")*
29 Sep	Module 7: Resizeable Arrays; Unions; Bits Fri 3: PSI Phase 2 Released		A6 2 Oct AoE
6 Oct	Module 8: Debugging!		A7 9 Oct AoE
13 Oct	Module 9: What malloc Actually Does; Ring Buffers; Error Handling; setjmp and longjmp and goto		P1 Check-In 16 Oct AoE
20 Oct	Module 10: The C Preprocessor; Programming in the Large		
27 Oct	Module 11: Case Study—Reference Counting Fri 31: PSI Phase 3 Released (Halloween lol)		
3 Nov	Module 12: Case Study—Hash Table		P2 Check-In 6 Nov AoE
10 Nov	Module 13: Concurrency		A8 13 Nov AoE
17 Nov			A9 20 Nov AoE
24 Nov			

1 Dec			2 Dec: Final Report Rough Draft
8 Dec			***9 Dec***: Final Report—**NO EXTENSIONS**

Course Project

You will be implementing an interpreter for PSI, a Lisp dialect, in C.

In **Phase 1**, you will implement the core logic for a four-function calculator. This will require you to design some of its basic data structures, as well as interact with the user at the command line:

```
psi> (+ 6 7)

13

psi> (* (+ 8 9) 10)

170

psi> (= (+ 11 12) 23)

#t

psi> (quit)

<exits the interpreter>
```

You are expected to delete data structures used for intermediate computations, and to be able to prove your program is free of memory leaks.

In **Phase 2**, you will introduce to this "calculator" a memory by implementing variables and the ability to bind them.

```
psi> (def a 13)

13

psi> (+ a 14)
```

```
27

psi> (def b (+ a 15))

28
```

You will also implement conditional execution (if-then-else)...

```
psi> (if (< b 100) 1 2)    ;; recall: b = 28

1

psi> (if (> b 100) (print "Pann G. was here") #f)

#f    ;; doesn't print

psi> (if (= 3 3) 0 (quit))

0     ;; doesn't quit

psi> ((if (= 3 4) + *) 6 7)

42
```

... as well as basic list operations:

```
psi> (head '(16 17 18))

16

psi> (tail '(19 20 21))

(20 21)

psi> (cons 22 '(23 24))

(22 23 24)
```

You will also implement quoting, which allows you to represent (and work with) unexecuted code, and eval, which invokes the evaluator:

```
psi> (+ 26 27)

53

psi> '(+ 28 29)

(+ 28 29)

psi> (def e '(+ 30 31 32))

(+ 30 31 32)
```

```
psi> (eval e)

93
```

In **Phase 3**, your program will come into its own as a full-fledged Lisp interpreter—with functions, local variables, and more control-flow primitives—enabling you to use it for whole programs:

```
;; prog1.psi
(def prime (n)
 (if (< n 2)
   #f
   (let c (cell 2)
        r (cell #f)
     (do
       (loop
         (let d (! c)
           (if (> (* d d) n)
             (do (:= r #t) #f)
             (if (= (% n d) 0)
               (do (:= r #f)     #f)
               (do (:= c (+ 1 d) #t)))))
        (! r)))))
(def parse-number (str)
 (let x (read str)
    (if (= (type x) 'number) x #f)))
(def main ()
 (do
   (output "Enter a number: ")
   (let s   (input)
        num (parse-number s)
     (if num
       (if (prime num)
         (output (str num) " is prime.\n")
         (output (str num) " is not prime.\n")
        (error 'not-a-number)))))
(main)
----
psi> (load "prog1.psi")
Enter a number: 23
23 is prime.
```

Don't worry—this is still a C course. You won't be required to write any PSI. Test cases and programs will be provided for you.

This is a challenging project. The reason I have chosen it is:

- to teach you what high-level languages do to support your code.

- to show the advantages and disadvantages of dynamic typing and automatic memory management ("garbage collection.")

The course project isn't easy—my implementation contains **2113 lines of C code**. I've been programming for 20+ years, and it still took me several days. You **absolutely cannot** leave it to the last minute.

AI Policy

You may use tools like Grammarly and ProWritingAid for grammar, although you will not be graded (except in extreme cases) on it. You may use natural language AI as a search tool—therefore, you may read AI-generated text, but you cannot submit it.

You may ***not*** request or intentionally look at AI-generated code—no exceptions. The course materials should be sufficient for you to learn C. If they're not, you should tell me to improve them, because that's literally my job.

Grading

You need **9 points** or at least **3/8** (C-) on your Final Report to pass (get an "S"). Note that:

- the course is pass/fail, so don't worry about your transcript.
- the late policy, in recognition of your other obligations, is deliberately generous.
- no student who submitted a Final Report has ever received a failing grade ("U").

Weekly Assignments (1 point each, x9)

Weekly assignments are graded as:

- 1.0 = ✓ — correct, or with very minor mistakes.
- 0.9 = **S**ignal — partially incorrect, but not suggestive of serious misunderstanding.
- 0.7 = **W**arning — incorrect; student seems not to understand the material.
- 0.5 = ac**K**nowledgement — assignment was more than 10 days late, not graded.
- 0.0 — no submission, or a submission in bad faith; total non-engagement.

Signals and warnings aren't cumulative. A submission scoring two signals will be graded at 0.9 (not 0.8). The purpose of a signal is to be just that—not an evaluation.

Course Project Check-Ins (0.5 point each, x2)

The Phase 1 Check-In is due **October 16**.

The Phase 2 Check-In is due **November 6**.

Submissions will be graded on a 1/2/3/4 scale for progress toward completion and scored as follows: 1 = 0.25; 2 = 0.35; 3 = 0.45; 4 = 0.50.

Rough Draft Receipt (0.25 points)

You will get 0.25 points extra for submitting *something* by the Rough Draft Deadline of December 2. Only your last submission will be graded, so don't worry if it's not in the best shape yet—submit something by this date, and keep going. You're encouraged to resubmit as your project and paper improve.

Course Project Report (8 points)

You will be graded on the completion of your implementation as well as the quality of your writeup. Your implementation grade will be determined based on how much you have done:

- 0.0 — Phase 1 incomplete.
- 0.5 — Phase 1 partially complete, with major bugs.
- 1.0 — Phase 1 complete with only minor bugs.
- 1.5 — Phase 2 partially complete, with major bugs.
- 2.0 — Phase 2 complete with only minor bugs or mostly complete.
- 2.5 — Phase 3 mostly incomplete, but some functionality.
- 3.0 — Phase 3 partially complete, with major bugs.
- 3.5 — Phase 3 mostly complete with only minor bugs.
- 4.0 — Phase 3 complete, with only insignificant bugs, *and thoroughly tested*.

Your writing grade will be assessed as follows:

- 1.0 — Well below the standard of graduate level work.
- 1.5 — Passing. Admissible for a course project.
- 2.0 — Satisfactory. Could be presented with some editing.
- 2.5 — (~30%) Strong. Achievement visible, claims well-supported.
- 3.0 — (~10%) Very strong. Clear, convincing writing—enjoyable to read.
- 3.5 — (~1%) Exceptional. Model paper.
- 4.0 — (<<1%) Cormac McCarthy.

Late Policy

As this is a 1-credit seminar designed for part-time students who are concurrently taking other courses, the late policy is deliberately lenient.

For weekly homework, submissions that are 0.02 to 3.01 days late get a Signal. Submissions that are 3.02 to 10.01 days late get a Warning. Submissions that are 10.02+ days late will not be graded or receive feedback and are assigned an Ack ("K").

The course project check-ins are not a major part of your grade, but their purpose is to deter procrastination. Therefore, the late policies are more strict:

- Check-Ins: -0.05 per eight hours; 3.02+ days late not accepted.

The final report's deadline—**December 9**—is **non-negotiable.** I want all of you to succeed, but I have to set rules and policies.

Final Grade Assignment

Letter grades will not show on your transcript, but you can think of your total score as a letter grade like so:

```
 15.00 (90%) to 18.25    = A

 12.00 (72%) to 14.95    = B

  9.00 (54%) to 11.95    = C
```

If you complete all the assignments, even late, it's nearly impossible to fail. Still, take this course seriously. You will be building an interpreter from scratch in C, verifying against a comprehensive test suite (which I will provide), debugging it for dozens of hours, and writing a paper in which you must convince me that all features have been implemented correctly, with resource leaks (e.g., memory) nonexistent or extremely rare. This is no small achievement.

Code of Conduct / Academic Integrity

Please don't plagiarize. This includes output of generative AI. Everything you submit must be your own work. This course is subject to the entirety of Georgia Tech's Code of Conduct, including its Academic Honor Code, so if you have any questions, please review the Code of Conduct here.

Disability Statement

If you have learning needs that require special accommodation, I am happy to make adjustments, but please contact the Office of Disability Services at contact the Office of Disability Services at (404) 894-2563 or http://disabilityservices.gatech.edu/