CS 8001 OIC Syllabus

Introduction to C Programming

Summer 2026

Instructor Information

Instructor of Record: Ana Rusch, ARusch3@gatech.edu

Primary Instructor: Michael Church, mchurch8@gatech.edu

General Course Information

Description

This seminar, intended for students used to high-level languages such as Ruby and Python, will teach C, the lingua franca for systems programming over the past forty years due to its elegance, efficiency, and low-level (close to the hardware) capabilities. We will tackle C's syntax, type system, and program layout, as well as arrays, pointers, and control structures, giving students the ability to translate familiar techniques into this lower-level language. In this project-based course, students will be guided through the implementation of an interpreter for a very basic Lisp, designed to give them a sense of what the computer, whether they use low-level languages or continue with high-level ones, is doing. We'll also cover file I/O, memory management, C's standard library, and the tools necessary to implement common data structures like hash tables and resizable arrays.

Course Learning Outcomes

By the semester's end, the student will develop an understanding of the following concepts:

- Learn C Programming, the language
    - Learn its philosophy
    - Learn its tool chain
    - Learn its techniques commonly found in low-level programming.
- You will be to apply these concepts by building an interpreter for a language called PSI.

Required Course Materials

Yur 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.

Grading Policy:

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").

Description of Graded Components

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.

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 **March 12**.
The Phase 2 Check-In is due **April 2**.
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.

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—**May 6**—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 18.00 = 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.

Course Policies

Attendance and/or Participation

Office Hours will be held at 8:00pm Atlanta (Eastern) time at the start of the semester. Except for **Institute Holidays,** all office hours will fall on **Monday**.

The official dates are on Canvas.

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.

Academic Integrity

Georgia Tech aims to cultivate a community based on trust, academic integrity, and honor. Students are expected to act according to the highest ethical standards. Review Georgia Tech's Honor Code and the student Code of Conduct.

Any student suspected of cheating or plagiarism on a quiz, exam, or assignment will be reported to the Office of Student Integrity, who will investigate the incident and identify the appropriate penalty for violations.

Accommodations for Students with Disabilities

If you are a student with learning needs that require special accommodation, contact the Office of Disability Services (404-894-2563) as soon as possible to make an appointment to discuss your special needs and to obtain an accommodations letter. Please also e-mail me as soon as possible in order to set up a time to discuss your learning needs.

Student-Faculty Expectations Agreement

At Georgia Tech, we believe that it is important to strive for an atmosphere of mutual respect, acknowledgement, and responsibility between faculty members and the student body. The Student-Faculty Expectations articulate some basic expectations that you can have of me and that I have of you. In the end, simple respect for knowledge, hard work, and cordial interactions will help build the environment we seek. Therefore, I encourage you to remain committed to the ideals of Georgia Tech while in this class.