COMPSCI 1XC3

Lab 6-2 — C Control Flow

Winter 2026

Lab Goals

By the end of this lab, you should be able to:

- Write conditional logic in C
- Write and control loops in C
- Compare different loop forms
- Reason about control-flow execution

This lab is **practice-focused**, not grading-focused.

Before You Start

For this lab, you may need concepts from the lecture notes **3-3-C-ControlFlow**.

Topics that may come up include:

- `if`, `if-else`, `if-else if-else`
- `for` and `while` loops
- `break` and `continue`
- Loop termination conditions
- Labels and `goto` (awareness only)

You are encouraged to **open the lecture notes** and use **experimentation** to reason about behavior.

Organization Tip

As we've done before, it's a good idea to stay organized with your files.

Consider creating a main folder for this course, for example:

- `1xc3/`
  - `lab6-2/`
  - or `c-practice/` (your preference)

This will make it easier to find your work later and reuse it for practice.

Progress Check (Required)

You will complete **three tasks**.

For each task:

- Write or run the code
- Compile with `gcc`
- Show the output to a TA

Progress Check 1 — Multi-Branch Conditional

Task

Write code that prints:

- `"SMALL"` if `x < 10`
- `"MEDIUM"` if `10 ≤ x < 100`
- `"LARGE"` otherwise

```
int x = 42;

/* your code here */
```

You must use an `if-else if-else` chain.

Progress Check 2 — Loop + Continue (Two Ways)

Task

Print the numbers **1 through 10**,
but **skip printing the number 5**.

You must complete this task **twice**:

A. Using a `for` loop

```
/* for-loop version */
```

B. Using a `while` loop

```
/* while-loop version */
```

Constraints:

- Both versions must use `continue`
- Output should be identical in both cases

Progress Check 3 — Loop Reasoning

Task

Run the following code.

```c
for (int i = 0; i <= 5; i++) {
    printf("%d\n", i);
}
```

Explain to a TA:

- How many times the loop runs
- What the final value of `i` is after the loop finishes

Optional Practice & Experiment

(Not Required)

The following tasks are **not required** for check-off,
but are good practice if you finish early.

These are meant to be treated like the
"**C Practice & Experiment**" sections in the lecture notes.

You do **not** need to show these to a TA.

Optional Task A — `switch` Statement

Task

Complete the code so that it prints:

- `"Monday"` if `day == 1`
- `"Tuesday"` if `day == 2`
- `"Wednesday"` if `day == 3`
- `"Other"` otherwise

```
int day = 3;

/* your code here */
```

Optional Task B — Loop with `break`

Task

Complete the loop so that it prints numbers starting at `1`,
but **stops once the value reaches 7**.

```
for (int i = 1; i <= 10; i++) {
    /* your code here */
    printf("%d\n", i);
}
```

Optional Task C — Nested Control Flow

Task

Complete the code so that it prints **only even numbers** between `1` and `20`.

```
for (int i = 1; i <= 20; i++) {
    /* your code here */
}
```

Optional Task D — `for` vs `while`

Task

Rewrite the following `for` loop using a `while` loop.

```c
for (int i = 0; i < 5; i++) {
    printf("%d\n", i);
}
```

Starter code:

```c
int i = 0;
/* your code here */
```

Optional Task E — `goto` and Labels (Awareness)

Task

Run the following code and observe the output.

```c
int i = 0;

start:
if (i < 3) {
    printf("%d\n", i);
    i++;
    goto start;
}
```

Questions to think about:

- What control structure does this resemble?
- Why is `goto` generally avoided in modern C code?

You are **not expected** to use `goto` elsewhere.

What TAs Are Checking

- Code compiles
- Output is correct
- Explanation shows understanding

Style and formatting do not matter for this lab.

Wrap-Up

This lab focused on **writing and reasoning about control flow in C**.

Use your lecture notes and experiments to:

- compare loop constructs
- understand execution order
- recognize less common control-flow features

Next lab will continue building on these ideas.