# Module 2: Making Decisions and Repeating Actions (Days 6-9)

## 1. Philosophy Focus: Knowledge-graph-based learning

So far, our programs run in a straight line from top to bottom. That's very limited. To write useful software, we need to be able to make decisions ("*if* the user is an admin, show the delete button") and repeat actions ("*for* each item in the shopping cart, add its price to the total"). This module, "Control Flow," is the next logical step, building directly on your knowledge of variables.

### **DAY 6: Making Decisions with if, else if, else**

**Goal:** Learn how to execute different blocks of code based on a condition.

1. Inductive Example: The Number Checker

Compile and run the following. Try entering a positive number, a negative number, and zero.

#include <stdio.h>  
  
int main(void) {  
 int number;  
  
 printf("Enter an integer: ");  
 scanf("%d", &number);  
  
 if (number > 0) {  
 printf("You entered a positive number.\n");  
 }   
 else if (number < 0) {  
 printf("You entered a negative number.\n");  
 }   
 else {  
 printf("You entered zero.\n");  
 }  
  
 return 0;  
}

**2. Discovering the Pattern**

* **The if statement:** It checks a condition inside the parentheses (). If the condition is true, the code block inside the curly braces {} is executed.
* **Comparison Operators:**
  + > (greater than)
  + < (less than)
  + >= (greater than or equal to)
  + <= (less than or equal to)
  + == (equal to - **use a double equals sign for comparison!** A single = is for assignment.)
  + != (not equal to)
* **else if and else:** These are optional. The code flows from top to bottom. The first condition that is true gets executed, and the rest of the chain is skipped. The else block runs if *none* of the preceding conditions were true.

**3. Day 6 Practice**

1. **Grade Calculator:** Write a program that asks for a numerical score (0-100).
   * If the score is 90 or above, print "Grade: A".
   * If the score is 80-89, print "Grade: B".
   * If the score is 70-79, print "Grade: C".
   * If the score is 60-69, print "Grade: D".
   * Otherwise, print "Grade: F".  
     (Hint: You'll need logical operators: score >= 80 && score < 90. The && means "AND".)

### **DAY 7: Repeating Actions with while Loops**

**Goal:** Learn how to repeat a block of code as long as a condition is true.

**1. Inductive Example: Countdown**

#include <stdio.h>  
  
int main(void) {  
 int count = 10;  
  
 while (count > 0) {  
 printf("%d...\n", count);  
 count = count - 1; // Or more commonly: count--;  
 }  
  
 printf("Liftoff!\n");  
 return 0;  
}

2. Discovering the while Loop Pattern

A while loop has three key parts:

1. **Initialization:** A variable is set up *before* the loop starts (int count = 10;).
2. **Condition:** The loop checks the condition in the () before each repetition. If it's true, the loop body runs. If it's false, the loop is skipped.
3. **Update:** Inside the loop, something must change the variable from the condition. If you forget count--, you will have an **infinite loop**!

**3. Day 7 Practice**

1. **Password Guesser:** Create a simple "secret number" game.
   * Declare an int secret\_number = 7; and an int guess;.
   * In a while loop, continuously ask the user "Guess the number: ".
   * The loop should continue as long as guess != secret\_number.
   * When the user guesses correctly, the loop will terminate. After the loop, print "You guessed it!".

### **DAY 8: Repeating a Known Number of Times with for Loops**

**Goal:** Learn a more compact syntax for loops that need to run a specific number of times.

1. Inductive Example: Countdown Revisited

The countdown is a perfect example of a loop that runs a known number of times (10). The for loop is designed for this.

#include <stdio.h>  
  
int main(void) {  
 // The for loop combines initialization, condition, and update in one line!  
 for (int count = 10; count > 0; count--) {  
 printf("%d...\n", count);  
 }  
  
 printf("Liftoff!\n");  
 return 0;  
}

2. Discovering the for Loop Pattern

The parentheses of a for loop have three parts, separated by semicolons: for (initialization; condition; update)

1. **int count = 10;**: This runs **once** at the very beginning of the loop.
2. **count > 0;**: This is the condition, checked **before** each iteration.
3. **count--**: This is the update, which runs **after** each iteration.

This structure is less error-prone than a while loop for simple counting because you are unlikely to forget the update step.

**3. Day 8 Practice**

1. **Times Table:** Ask the user for a number. Then use a for loop that runs 10 times (e.g., from i = 1 to i <= 10) to print out the multiplication table for their number.
   * Example output for input 5:  
     5 x 1 = 5  
     5 x 2 = 10  
     ...  
     5 x 10 = 50

### **DAY 9: Spaced Repetition - Review**

**Goal:** Combine if statements and loops to solve a more complex problem.

1. Done-for-you Training Plan: FizzBuzz

This is a classic programming interview question that tests your understanding of loops and conditional logic.

**Project Requirements:**

* Write a program that prints the numbers from 1 to 100, each on a new line.
* But there are rules:
  + For numbers that are multiples of 3, print "Fizz" instead of the number.
  + For numbers that are multiples of 5, print "Buzz" instead of the number.
  + For numbers that are multiples of both 3 and 5, print "FizzBuzz".

**Hint:** The "modulo" operator (%) gives you the remainder of a division. For example, 10 % 3 is 1, because 10 divided by 3 is 3 with a remainder of 1. If number % 3 == 0, then you know the number is a multiple of 3.

**Strategy:**

* Use a for loop to iterate from 1 to 100.
* Inside the loop, use a chain of if, else if, else statements.
* **Important:** You need to check for the "multiple of both 3 and 5" case *first*. Why? Think about what would happen if you checked for "multiple of 3" first for the number 15.