# Module 1: Variables and Basic I/O (Days 3-5)

## 1. Philosophy Focus: Inductive Teaching

A pattern you discover on your own will stick far more than one you are told. In this module, you'll learn about variables by *using* them first. We'll give you a series of small, related problems that force you to see the patterns of how data is stored and manipulated, leading you to an intuitive understanding of "data types."

### **DAY 3: Storing Information - Variables and Data Types**

**Goal:** Learn how to store and use data within a C program.

1. The Need for Storage

Programs are all about processing data. We need a way to store data temporarily in memory so we can work with it. We do this with variables. A variable is simply a named storage location.

2. Inductive Example: The Age Calculator

Type, compile, and run the following code:

#include <stdio.h>  
  
int main(void) {  
 int current\_year = 2024;  
 int birth\_year = 1995;  
 int age; // We declare a variable but don't give it a value yet  
  
 age = current\_year - birth\_year; // Now we assign a value to it  
  
 printf("Someone born in %d is %d years old in %d.\n", birth\_year, age, current\_year);  
  
 return 0;  
}

**3. Discovering the Pattern**

* **Declaration:** Notice the pattern: type name; or type name = value;. Before you can use a variable, you must **declare** it by telling the compiler its **type** and its **name**.
* **int Type:** The int keyword means this variable holds an integer (a whole number).
* **Assignment:** The = sign is the assignment operator. It takes the value on the right and stores it in the variable on the left.
* **Using Variables:** The printf function is using %d as a placeholder for an int. The variables birth\_year, age, and current\_year are passed to printf, and their values are slotted into the placeholders in order.

4. Common Data Types

C has several fundamental data types. Here are the most common ones:

* int: Integers (e.g., -5, 0, 120).
* double: Floating-point numbers (numbers with a decimal point, e.g., 3.14, -0.001, 99.99). Use %f in printf.
* char: A single character (e.g., 'A', '!', 'z'). Use %c in printf.

**5. Day 3 Practice**

1. **Item Pricer:** Write a program that declares an int variable for quantity and a double variable for price\_per\_item. Assign them values (e.g., 3 and 5.99). Create a third double variable total\_price and calculate the result. Print the result in a sentence like: "3 items at $5.99 each will cost a total of $11.98". (Hint: use %f for the doubles).
2. **Initial:** Create a char variable and store your first initial in it. Print it to the screen.

### **DAY 4: Getting User Input with scanf**

**Goal:** Make your programs interactive by reading data typed by the user.

1. Inductive Example: Interactive Age Calculator

Let's modify yesterday's program to ask the user for their birth year.

#include <stdio.h>  
  
int main(void) {  
 int current\_year = 2024;  
 int birth\_year;  
 int age;   
  
 printf("What year were you born? ");  
  
 // Read an integer from the user and store it in birth\_year  
 scanf("%d", &birth\_year);   
  
 age = current\_year - birth\_year;  
  
 printf("You are approximately %d years old.\n", age);  
  
 return 0;  
}

**2. Discovering the Pattern (scanf)**

* scanf is the counterpart to printf. It reads formatted input from the user.
* The format string ("%d") tells scanf what kind of data to expect (in this case, an integer).
* **The Ampersand &:** This is the most important and confusing part. scanf needs to know the **memory address** of the variable where it should store the data. The & operator gets the address of a variable. For now, just remember the rule: **When using scanf with basic types like int and double, you must put an & before the variable name.**

**3. Day 4 Practice**

1. **Area Calculator:** Write a program that asks the user for the width and height of a rectangle (as doubles). Read both inputs using scanf. Calculate the area and print it.
2. **Multiple Inputs:** You can read multiple values with one scanf call. Try this: scanf("%lf %lf", &width, &height);. Note the %lf for double with scanf.
3. **A Note on Types:** For printf, you can use %f for both float and double. For scanf, you **must** use %f for float and %lf for double. It's a common C pitfall. We'll stick to double for simplicity.

### **DAY 5: Spaced Repetition - Review**

**Goal:** Solidify the concepts of variables, printf, and scanf by building a simple, complete program.

1. Done-for-you Training Plan: The Tip Calculator

This project will use everything you've learned in this module.

**Project Requirements:**

1. Welcome the user to the tip calculator.
2. Ask the user for the total bill amount (a double).
3. Ask the user what percentage they want to tip (e.g., 15, 20). This should be an int.
4. Calculate the tip amount. Remember to convert the percentage to a decimal (e.g., 15 becomes 0.15).
5. Calculate the total amount (bill + tip).
6. Print the tip amount and the final total amount, formatted nicely to two decimal places.

**Hint:** To print a double with exactly two decimal places, use %.2f in printf.

**Example Interaction:**

Welcome to the tip calculator!  
What was the total bill? $67.50  
What percentage tip would you like to give? 15  
Tip amount: $10.13  
Total bill: $77.63

Take your time with this. Try to write it without looking back at the previous days' code. If you get stuck, it's okay to look, but the effort of trying to recall it first is what builds the memory.