Module 2: Variables, Data Types & Operators — ready for inclusion in your C Programming Course.

Learn2Code Academy

C Programming Course – Module 2

📅 **Date:** November 11, 2025

👨‍🏫 **Instructor:** *Foffe Lili (DevLili)*

📘 **Course Title:** *Introduction to Programming with C*

Module 2: Variables, Data Types & Operators

🎯 Learning Objectives

By the end of this module, students should be able to:

- Understand what **variables** are and their role in programming.
- Identify and use different **data types** in C.
- Declare and initialize variables correctly.
- Understand **constants** and their purpose.
- Apply various **operators** (arithmetic, relational, logical, assignment, etc.) in C programs.
- Write programs that perform calculations and comparisons using variables and operators.

1. What is a Variable?

A **variable** is a name given to a memory location where data can be stored, modified, and retrieved during program execution.

Think of a variable as a **box** that holds information — such as a number, a letter, or a word.

Example:

```
int age = 20;
```

Here:

- `int` → data type (integer)
- `age` → variable name
- `20` → value stored in the variable

🛍️ You can change the value of a variable anytime:

```
age = 25;
```

🔠 2. Rules for Naming Variables

When naming a variable in C:

✅ Valid rules:

1. Must begin with a **letter** or **underscore (_)**.
2. Can contain letters, digits, and underscores.
3. **No spaces or special characters** allowed.
4. C is **case-sensitive** (`Age` ≠ `age`).
5. Avoid using **reserved keywords** like `int`, `if`, `while`, etc.

✅ Examples of valid variable names:

```
count, total_marks, number1, _result
```

❌ Invalid examples:

```
1stNumber, my-variable, float, total marks
```

3. Declaring and Initializing Variables

Declaration → telling the compiler the type and name of a variable.
Initialization → assigning an initial value.

Examples:

```
int age;         // declaration
age = 20;        // initialization

float salary = 55000.50;  // declaration + initialization
char grade = 'A';
```

🟫 4. Data Types in C

C has different **data types** to define the kind of data a variable can store.

Data Type	Keyword	Size (Bytes)	Example Value	Description
Integer	`int`	4	25	Whole numbers
Float	`float`	4	3.14	Decimal numbers
Double	`double`	8	10.45678	Large/precise decimals
Character	`char`	1	'A'	Single character
String	`char[]`	varies	"Hello"	Sequence of characters

Example Program:

```
#include <stdio.h>

int main() {
```

```
    int age = 18;
    float height = 1.75;
    char grade = 'B';

    printf("Age: %d\n", age);
    printf("Height: %.2f\n", height);
    printf("Grade: %c\n", grade);

    return 0;
}
```

Output:

```
Age: 18
Height: 1.75
Grade: B
```

🔢 5. Constants

A **constant** is a fixed value that does not change during program execution.

Two ways to define constants:

1. **Using `#define`:**

```
#define PI 3.14159
```

2. **Using `const` keyword:**

```
const int DAYS_IN_WEEK = 7;
```

Example:

```
#include <stdio.h>
```

```c
#define PI 3.14

int main() {
    const int days = 7;
    printf("PI = %.2f\n", PI);
    printf("Days in a week = %d\n", days);
    return 0;
}
```

⚙ 6. Operators in C

Operators are **symbols** used to perform operations on variables and values.

A. Arithmetic Operators

Operator	Description	Example	Result
+	Addition	a + b	30
-	Subtraction	a - b	10
*	Multiplication	a * b	200
/	Division	a / b	2
%	Modulus (remainder)	a % b	0

Example:

```c
int a = 20, b = 10;
printf("Sum = %d\n", a + b);
printf("Difference = %d\n", a - b);
```

B. Relational Operators

Used to compare values.

Operator	Meaning	Example	Result
==	Equal to	a == b	False
!=	Not equal	a != b	True
>	Greater than	a > b	True
<	Less than	a < b	False
>=	Greater or equal	a >= b	True
<=	Less or equal	a <= b	False

Example:

```
if (a > b)
    printf("a is greater than b");
```

⚖️ C. Logical Operators

Used to combine conditions.

Operator	Meaning	Example	Result
&&	Logical AND	(a > 5 && b < 10)	True if both true
`		`	Logical OR
!	Logical NOT	!(a > 5)	Reverses result

💾 D. Assignment Operators

Operator	Meaning	Example	Equivalent
=	Assign	a = 10	—
+=	Add and assign	a += 5	a = a + 5
-=	Subtract and assign	a -= 3	a = a - 3

Operator	Meaning	Example	Equivalent
*=	Multiply and assign	a *= 2	a = a * 2
/=	Divide and assign	a /= 4	a = a / 4

7. Example Program Using Operators

```
#include <stdio.h>

int main() {
    int a = 10, b = 20;

    printf("Addition: %d\n", a + b);
    printf("Subtraction: %d\n", a - b);
    printf("Product: %d\n", a * b);
    printf("Division: %d\n", b / a);
    printf("Remainder: %d\n", b % a);

    return 0;
}
```

Output:

```
Addition: 30
Subtraction: -10
Product: 200
Division: 2
Remainder: 0
```

8. Key Takeaways

✓ A **variable** stores data that can change during program execution.

✓ A **data type** tells the compiler what kind of data a variable holds.

✓ A **constant** stores a value that cannot change.

✓ **Operators** perform mathematical, comparison, and logical operations.

✓ Understanding variables and operators is essential for problem-solving in C.

📝 Assignments for Module 2

🎯 Part A – Practice and Recall

1. What is the difference between a **variable** and a **constant**?
2. Explain the importance of data types in C.
3. Write a program that declares variables of each data type and displays their values.
4. Write a program that performs and prints all arithmetic operations between two numbers.
5. Find and fix the errors:

```
6. int 1num = 10;
7. float average = 4.5
8. printf("Average: %d", average);
```

9. What happens if you divide two integers like `5 / 2`? Why?
10. Explain the use of the modulus operator `%` with an example.

Part B – Mini Project

Create a program that:

- Declares two integers: `x` and `y`.
- Performs all **arithmetic**, **relational**, and **logical** operations on them.
- Displays the results clearly with `printf()` statements.

Expected Output Example:

```
x = 10, y = 5
Sum = 15
Difference = 5
x > y: True
x && y: True
```

🔍 Part C – Research & Prepare for Next Module

To prepare for **Module 3: Control Structures (Decision Making & Loops)**, research and write notes on:

1. What are **conditional statements** in C?
2. How does an **if-else** statement work?
3. What is the difference between `for`, `while`, and `do-while` loops?
4. Write pseudocode that checks if a number is even or odd.

👨‍💻 *Prepared by:* **Foffe Lili (DevLili)**

💼 *Instructor – Learn2Code Academy*

📅 *Date: November 11, 2025*