Week 1: Introduction to C Programming

Overview

Welcome to your first week of learning C programming! This week, we will cover the basics of writing and running a simple C program. By the end of this tutorial, you will:

- Understand the structure of a C program
- Learn how to use `printf()` for output
- Take user input using `scanf()`
- Work with basic data types and variables

Time Breakdown

- **Introduction & Explanation (10 min)**
- **Writing a Simple C Program (10 min)**
- **Working with Variables and Data Types (15 min)**
- **Taking User Input (20 min)**
- **Exercise & Q/A (15 min)**

---

1. Writing Your First C Program (10 min)

A C program consists of functions and statements enclosed in `{}`. Every program starts with a `main()` function. Below is a simple C program that prints "Hello, World!" to the screen.

Example: Hello, World!

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

PROF

Explanation:

- `#include <stdio.h>`: This is a preprocessor directive that includes the standard input-output library.
- `int main()`: This is the main function where program execution starts.
- `printf("Hello, World!\n");`: This prints "Hello, World!" to the screen.
- `return 0;`: Ends the program successfully.

Exercise 1 (5 min):

Modify the above program to print your name instead of "Hello, World!".

2. Variables and Data Types (15 min)

Variables are used to store data. In C, you must declare a variable before using it.

Common Data Types:

Data Type	Description	Example
`int`	Stores integers	`int age = 25;`
`float`	Stores decimal numbers	`float pi = 3.14;`
`char`	Stores a single character	`char letter = 'A';`

Example: Using Variables

```c
#include <stdio.h>

int main() {
    int age = 20;
    float height = 5.9;
    char grade = 'A';

    printf("Age: %d\n", age);
    printf("Height: %.1f\n", height);
    printf("Grade: %c\n", grade);

    return 0;
}
```

Exercise 2 (10 min):

PROF

Create a program that declares variables for your name, age, and favorite number, then prints them.

3. Taking User Input (20 min)

You can take input from the user using `scanf()`.

Example: User Input

```c
#include <stdio.h>

int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("You are %d years old.\n", age);
```

```
    return 0;
}
```

Exercise 3 (10 min):

Write a program that asks the user for their name and age, then prints a greeting message.

---

4. Assignment (10 min): Medical Checkup Program

Inspired by previous assignments, let's create a structured task:

Task: Write a C program that simulates a basic medical checkup. Your program should:

1. Ask the user for their age, gender (M/F), height (in meters), weight (in kg), blood type (A, B, or O), and body temperature.
2. Print all this information in a single `printf()` statement, using proper formatting.

Example Output:

```
You are 24 years old and your gender is "M".
Your height and weight are 1.74 m, 74.00 kg.
Your blood type is 'A'.
Your body temperature is 36.50 degrees Celsius.
```

Hints:

- Use `printf()` for output formatting.
- Use `\n` for line breaks and `\t` for spacing.
- Store values in variables and use `%d`, `%f`, and `%c` format specifiers accordingly.

PROF

Summary & Wrap-Up (10 min)

This week, we covered:

- Writing a simple C program
- Using `printf()` for output
- Declaring variables and using data types
- Taking user input with `scanf()`

Next Week: Basic C Programming - Arithmetic Operations and Control Statements

Happy coding! 🚀