C PROGRAMMING

By Lungsom Lamnio

```c
int main() {
    int i;
    boysHostel:
        printf("Hello World \n");
        i++;
    for(i=1; i<=10; i++) {
        if(i%2 == 0) {
            goto boysHostel;
        }
        printf("%d \n", i);
        // boysHostel:
        //      i++;
    }
```

1. The program prints "Hello World \n" and increments `i`.
2. The `for` loop starts with `i = 1`.
3. When `i` is 2, the condition `i % 2 == 0` is true (since 2 is divisible by 2), and the program jumps to the `boysHostel` label.
4. Once it jumps to `boysHostel`, it prints "Hello World \n" again and increments `i` within the label.
5. However, after incrementing `i`, the `for` loop is effectively restarted from `i = 2` again (because the `for` loop initialization doesn't happen again, but the loop checks condition `i <= 10` keeps holding true for i = 2).
6. Every time `i` reaches 2, the `goto` statement is triggered, and the cycle repeats, causing the program to continuously loop between printing "Hello World \n" and incrementing `i`.

Copy code

```c
#include <stdio.h>

int main() {
    int i = 1; // Initialize i here

boysHostel:
    printf("Hello World \n");
    i++; // Increment i after printing "Hello World"

    for (; i <= 10; i++) { // Continue the loop from the current value of i
        if (i % 2 == 0) {
            goto boysHostel; // Go back to the label if i is even
        }
        printf("%d \n", i); // Print i if it's not even
    }

    return 0;
}
```

```c
int main() {
    int choice, num;

    do {
        // Display the menu
        printf("\nMenu:\n");
        printf("1. Check Even or Odd\n");
        printf("2. Check Positive or Negative\n");
        printf("3. Square of the Number\n");
        printf("4. Exit\n");
        printf("Enter your choice (1-4): ");
        scanf("%d", &choice);
```

```c
        switch (choice) {
            case 1:
                printf("Enter a number: ");
                scanf("%d", &num);
                if (num % 2 == 0)
                    printf("%d is Even\n", num);
                else
                    printf("%d is Odd\n", num);
                break;

            case 2:
                printf("Enter a number: ");
                scanf("%d", &num);
                if (num >= 0)
                    printf("%d is Positive\n", num);
                else
                    printf("%d is Negative\n", num);
                break;
```

```
            case 3:
                printf("Enter a number: ");
                scanf("%d", &num);
                printf("Square of %d is %d\n", num, num * num);
                break;

            case 4:
                printf("Exiting the program...\n");
                break;

            default:
                printf("Invalid choice! Please enter a valid option (1-4).\n");
        }
    } while (choice != 4);

    return 0;
}
```

Arrays in C

Notes:

- An **array** is a collection of elements of the same data type stored in contiguous memory locations.
- Arrays are indexed starting from `0`, so the first element is at index `0`, the second at `1`, and so on.
- The size of the array is fixed and needs to be specified during declaration.
- Arrays can be **single-dimensional** or **multi-dimensional** (like 2D arrays).

Syntax:

- **Declaration of 1D Array:**

c Copy code

```c
data_type array_name[array_size];
```

Example:

c Copy code

```c
int numbers[5];   // Declares an integer array of size 5
```

Functions in C

Notes:

- A **function** is a block of code that performs a specific task and can be called when needed.
- Functions allow code reusability and modular programming.
- There are **two types of functions**: **Library functions** (e.g., `printf()`, `scanf()`) and **User-defined functions**.
- Functions can take **parameters** (inputs) and **return values** (outputs).
- **Function prototypes** are declarations of functions that inform the compiler about the function's name, return type, and parameters.

Syntax:

- **Declaration (Prototype):**

c Copy code

```c
return_type function_name(parameter_type1, parameter_type2, ...);
```

Example:

c Copy code

```c
int add(int, int);  // Function prototype
```

No Assignments Today

Complete your pending assignments.

Steps to submit the assignments

Step 1: Make a GitHub repo with the name as YourMentor.
Step 2: Inside the repo YourMentor, make a folder with the name as C_BootCamp.
Step 3: Submit the assignment with the day number (eg: Day1_Assignment.pdf).
Step 4: DM me the link of repo.

NOTE:

1. Active members will get the chance to make a real-life project with me.
2. Advantages with Upcoming BootCamps.

THANK YOU