

Module: 2

Day1:

Trainer: Asif Nafees

do while:

do-while loop functions similarly to other programming languages, where the loop guarantees at least one execution of the code block before evaluating the condition.

Syntax of do-while Loop in C

```
do {  
    // Statements to be executed  
} while (condition);
```

do: Marks the beginning of the loop block.

Statements: Code that is executed once before the condition is checked.

while: Continues looping as long as the condition is true.

condition: An expression that evaluates to true or false.

Execution Flow of the do-while Loop

1. Step 1: The code block inside the do is executed.
2. Step 2: The condition is evaluated.
3. Step 3: If the condition is true, the loop repeats by executing the block again.
4. Step 4: If the condition is false, the loop exits, and the program moves on to the next statement after the loop.

Key Features

The do-while loop always runs at least once, even if the condition is false from the start.

It is useful when the block of code must run before the condition is tested.

Example of a do-while Loop in C

```
#include <stdio.h>  
int main() {  
    int i = 1;  
    // Example of do-while loop  
    do {  
        printf("Iteration: %d\n", i);  
        i++;  
    } while (i <= 5);  
    return 0;  
}
```

Explanation of the Code

1. The variable i is initialized to 1.
2. The block of code inside the do executes, printing the current value of i and incrementing it.

3. After each iteration, the condition $i \leq 5$ is checked. If it is true, the loop repeats; otherwise, it exits the loop.

Output:

Iteration: 1

Iteration: 2

Iteration: 3

Iteration: 4

Iteration: 5

Use Cases of do-while Loop in C

1. Input Validation A do-while loop can be used to continuously prompt the user for input until valid data is entered.

```
#include <stdio.h>
int main() {
    int number;
    do {
        printf("Enter a number between 1 and 10: ");
        scanf("%d", &number);
    } while (number < 1 || number > 10);
    printf("You entered a valid number: %d\n", number);
    return 0;
}
```

Explanation: This code repeatedly prompts the user for a number until the input is between 1 and 10. The loop will execute at least once, ensuring the prompt is shown.

2. Menu-Driven Program A do-while loop can be used in a menu-driven program, where the menu options are displayed repeatedly until the user decides to exit.

```
#include <stdio.h>
int main() {
    int choice;
    do {
        printf("\nMenu:\n");
        printf("1. Option 1\n");
        printf("2. Option 2\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                printf("You selected Option 1\n");
                break;
            case 2:
                printf("You selected Option 2\n");
                break;
        }
    } while (choice != 3);
}
```

```

        case 3:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
} while (choice != 3);

return 0;
}

```

Explanation: The do-while loop keeps displaying the menu and taking input from the user until the user selects option 3, which exits the program.

3. Game Loop In simple text-based games, the do-while loop can be used to repeatedly run the game logic until a win/lose condition is met.

```

#include <stdio.h>
int main() {
    int target = 5;
    int guess;
    do {
        printf("Guess the number (between 1 and 10): ");
        scanf("%d", &guess);
        if (guess < target) {
            printf("Too low!\n");
        } else if (guess > target) {
            printf("Too high!\n");
        } else {
            printf("Correct! You guessed the number.\n");
        }
    } while (guess != target);
    return 0;
}

```

Explanation: The player keeps guessing the number until they guess correctly. The do-while ensures the game runs at least once before checking the condition.