Loop & Switch-Case

Loop & Switch-Case

1. While Loop

A `while` loop is a function used to execute the same block of code repeatedly. The loop continues execution as long as the given condition evaluates to `1` (TRUE) or more. When the condition evaluates to `0` (FALSE), the loop stops and the program proceeds to the next lines of code.

Similar to an `if` statement, the `while` loop is built into the C programming language, meaning there is no need to declare or return its value explicitly.

Syntax:

```
while (condition) {
    // Code to be executed repeatedly
}
```

The `condition` is checked before executing the loop body:

- If `condition` is `TRUE`, the code inside the loop is executed.
- If `condition` is `FALSE`, the loop terminates.

Example 1: Counting from 1 to 10

```
#include <stdio.h>

int main() {
    int n = 1;

    while (n <= 10) {  // Loop runs while n is less than or equal to 10
```

```
        printf("%d\n", n);
        n++;  // Increment n by 1 in each iteration
    }

    return 0;
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Infinite Loop

If the condition never changes or is always `TRUE`, the loop will run indefinitely.

Example 2: Infinite Loop (Press Ctrl+C to Stop)

```
#include <stdio.h>

int main() {
    while (1) {  // The condition is always TRUE
        printf("This loop will run forever!\n");
    }

    return 0;
}
```

Output (Repeats Forever):

```
This loop will run forever!
This loop will run forever!
This loop will run forever!
...
```

Using `break` to Exit a While Loop

The `break` statement can be used to exit a while loop forcefully.

Example 3: Using `break` to Stop the Loop

```c
#include <stdio.h>

int main() {
    int n = 1;

    while (1) {  // Infinite loop
        printf("%d\n", n);
        if (n == 5) {
            break;  // Exit the loop when n reaches 5
        }
        n++;
    }

    return 0;
}
```

Output:

```
1
2
3
4
5
```

Using `continue` to Skip an Iteration

The `continue` statement is used to skip the remaining code in the loop for a specific iteration.

Example 4: Skipping a Number (Skipping 5)

```
#include <stdio.h>

int main() {
    int n = 0;

    while (n < 10) {
        n++;

        if (n == 5) {
            continue;  // Skip printing 5
        }

        printf("%d\n", n);
    }

    return 0;
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```

2. DO-WHILE LOOP

The `do-while` loop is similar to the `while` loop. The difference lies in the execution order:

- In a `while` loop, the condition is checked **before** executing the code.

- In a `do-while` loop, the code is executed **at least once** before checking the condition.

Syntax:

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

Example 5: Difference Between While and Do-While

```
#include <stdio.h>

int main() {
    int n = 0;

    printf("Using while loop:\n");
    while (n > 0) {
        printf("This will NOT be printed.\n");
    }

    printf("\nUsing do-while loop:\n");
    do {
        printf("This WILL be printed at least once.\n");
    } while (n > 0);

    return 0;
}
```

Output:

```
Using while loop:


Using do-while loop:
This WILL be printed at least once.
```

Explanation:

- The `while` loop does **not** execute because `n > 0` is `FALSE`.
- The `do-while` loop runs **once** before checking the condition.

---

3. FOR LOOP

A `for` loop is an advanced version of the `while` loop. It allows for a **specific range** and **controlled iterations**.

The `for` loop consists of **three components**:

1. **Initialization (`init`)** → Sets the starting value. (e.g., `i = 1;`)
2. **Condition (`condition`)** → Determines when the loop stops. (e.g., `i <= 10;`)
3. **Increment (`increment`)** → Updates the loop variable. (e.g., `i++`)

Syntax:

```
for (initialization; condition; increment) {
    // Code to be executed
}
```

Example 6: Printing Numbers 1 to 10

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }

    return 0;
}
```

Output:

```
1
2
3
4
5
6
7
```

```
8
9
10
```

Example 7: Loop with Step Size of 2

```c
#include <stdio.h>

int main() {
    for (int i = 0; i <= 10; i += 2) {
        printf("%d\n", i);
    }

    return 0;
}
```

Output:

```
0
2
4
6
8
10
```

---

4. SWITCH-CASE STATEMENT

A `switch-case` statement is an alternative to `if-else-if` for comparing a variable against multiple **fixed values**.

- If the variable matches a `case`, the corresponding block of code is executed.
- If no cases match, the `default` case is executed (if present).
- The `break` statement **prevents fall-through**, meaning once a match is found, execution stops.

Syntax:

```
switch (variable) {
    case value1:
        // Code to execute
        break;
    case value2:
        // Code to execute
        break;
    default:
        // Code if no cases match
}
```

Example 8: Simple Menu System

```
#include <stdio.h>

int main() {
    int choice;

    printf("Select an option:\n");
    printf("1. Start\n");
    printf("2. Settings\n");
    printf("3. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Game Starting...\n");
            break;
        case 2:
            printf("Opening Settings...\n");
            break;
        case 3:
            printf("Exiting Program...\n");
            break;
        default:
            printf("Invalid Choice!\n");
    }
```

```
    return 0;
}
```

Example Output:

```
Select an option:
1. Start
2. Settings
3. Exit
Enter your choice: 2
Opening Settings...
```

Example 9: Days of the Week

```
#include <stdio.h>

int main() {
    int day;

    printf("Enter a number (1-7) for the day of the week: ");
    scanf("%d", &day);

    switch (day) {
        case 1:
            printf("Sunday\n");
            break;
        case 2:
            printf("Monday\n");
            break;
        case 3:
            printf("Tuesday\n");
            break;
        case 4:
            printf("Wednesday\n");
            break;
        case 5:
            printf("Thursday\n");
            break;
        case 6:
```

```
            printf("Friday\n");
            break;
        case 7:
            printf("Saturday\n");
            break;
        default:
            printf("Invalid input! Enter a number between 1 and 7.\n");
    }

    return 0;
}
```

Example Output:

```
Enter a number (1-7) for the day of the week: 5
Thursday
```

Revision #2
Created 7 February 2025 16:13:51 by BH
Updated 7 February 2025 16:30:35 by BH