Download PDF

While Loop

do..while Loop

What is a `do-while` loop?

A `do-while` loop **executes the code at least once**, then checks the condition.

Syntax

```
do {
    // code to execute
} while (condition);
```

Key point: The condition is checked **after** the loop body.

Example 1: Print Numbers from 1 to 5

```
#include <iostream>
using namespace std;

int main() {
    int i = 1;

    do {
        cout << i << " ";
        i++;
    } while (i <= 5);

    return 0;
}
```

Output

```
1 2 3 4 5
```

Explanation:

- `i` starts at 1
- The loop prints the number
- `i` increases by 1
- The loop runs until `i` becomes greater than 5

Example 2: Print “Hello” 3 Times

```cpp
#include <iostream>
using namespace std;

int main() {
    int count = 1;

    do {
        cout << "Hello\n";
        count++;
    } while (count <= 3);

    return 0;
}
```

Explanation:

- The loop runs exactly 3 times
- Useful when you know how many times to repeat something

Example 3: Sum of First 5 Natural Numbers

```cpp
#include <iostream>
using namespace std;

int main() {
    int i = 1, sum = 0;

    do {
        sum += i;
        i++;
    } while (i <= 5);

    cout << "Sum = " << sum;
    return 0;
}
```

Output

```
Sum = 15
```

Example 4: Menu-Driven Program (Real-Life Use)

```cpp
#include <iostream>
using namespace std;

int main() {
    int choice;

    do {
        cout << "\n1. Say Hello";
        cout << "\n2. Say Bye";
        cout << "\n3. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        if (choice == 1)
            cout << "Hello!\n";
        else if (choice == 2)
            cout << "Bye!\n";

    } while (choice != 3);

    return 0;
}
```

Why `do-while` here?

- The menu **must appear at least once**
- Perfect for menus and user input programs

Example 5: Loop Runs At Least Once (Important Concept)

```cpp
#include <iostream>
using namespace std;

int main() {
    int i = 10;

    do {
        cout << "This will print once\n";
    } while (i < 5);

    return 0;
}
```

Explanation:

- Condition is **false**
- Still, the message prints **once**
- This is the main difference from `while` loop

Comparison: `while` vs `do-while`

Feature	while	do-while
Condition checked	Before loop	After loop
Runs at least once	❌ No	☑ Yes
Best for	Unknown executions	Menus / user input

When to Use `do-while`

☑ When the loop **must execute at least once** ☑ For **menus, password checks, user input**

for loop

Syntax of a for loop:

```
for (initialization; condition; update) {
    // code to execute in each iteration
}
```

- **Initialization**: Runs **once** at the start of the loop. Usually used to declare a loop variable.
- **Condition**: Checked **before each iteration**. If true, the loop continues; if false, the loop stops.
- **Update**: Runs **after each iteration**. Usually increments/decrements the loop variable.

Example 1: Print numbers 1 to 5

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 5; i++) { // i starts at 1, loop until i <= 5, i
increments by 1
        cout << i << " ";
    }
    return 0;
}
```

Output:

```
1 2 3 4 5
```

☑ **Explanation:**

- `int i = 1` → start counting from 1
- `i <= 5` → stop after 5
- `i++` → increment i by 1 each time

Example 2: Print even numbers from 2 to 10

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 2; i <= 10; i += 2) { // i increases by 2 each time
        cout << i << " ";
    }
    return 0;
}
```

Output:

```
2 4 6 8 10
```

☑ **Explanation:**

- `i += 2` means **add 2 to i each time**, so we get even numbers.

Example 3: Sum of first 10 natural numbers

```
#include <iostream>
using namespace std;

int main() {
    int sum = 0;
    for (int i = 1; i <= 10; i++) {
        sum += i; // sum = sum + i
    }
    cout << "Sum = " << sum;
    return 0;
}
```

Output:

```
Sum = 55
```

☑ **Explanation:**

- `sum += i` adds each number to the total sum.
- After the loop finishes, `sum` contains the total.

Key Points for Beginners

1. A `for` loop is great for **known number of iterations**.
2. You can control the start, end, and increment.
3. The loop can run **forward** (i++) or **backward** (i--).

Example 4: Basic Multiplication Table

Question: Write a C++ program that asks the user to enter a number and then displays its multiplication table from 1 to 10 using a **for loop**.

Example:

- If the user enters `7`, the program should display:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
...
7 x 10 = 70
```

Requirements:

1. Use `cin` to take input from the user.
2. Use a `for` loop to generate the multiplication table.
3. Display the results in a readable format.

```cpp
#include <iostream>
using namespace std;

int main() {
    int number;

    // Ask the user for the number
    cout << "Enter a number to display its multiplication table: ";
    cin >> number;

    // Loop from 1 to 10
    for (int i = 1; i <= 10; i++) {
        cout << number << " x " << i << " = " << number * i << endl;
    }
```

```
    return 0;
}
```

☑ How it works:

1. `cin >> number;` → User inputs the number for which they want the table.
2. `for (int i = 1; i <= 10; i++)` → Loop runs from 1 to 10.
3. Inside the loop, `number * i` calculates each multiplication.
4. `cout` prints the result in a readable format.

Example Output (if user enters 5):

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
...
5 x 10 = 50
```

Example 5: Print Even Numbers from 1 to N

Question: Write a C++ program that asks the user to enter a positive number `N` and then prints **all even numbers from 1 to N** using a **for loop**.

Example:

- Input: `10`
- Output:

```
Even numbers from 1 to 10 are: 2 4 6 8 10
```

Requirements:

1. Use `cin` to take input from the user.
2. Use a `for` loop to iterate from 1 to `N`.
3. Use an `if` statement to check if a number is even.
4. Display the even numbers in a single line, separated by spaces.

```
#include <iostream>
using namespace std;

int main() {
    int n;
```

```
    // Ask the user for a number
    cout << "Enter a positive number: ";
    cin >> n;

    cout << "Even numbers from 1 to " << n << " are: ";

    // For loop to print even numbers
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 0) {  // Check if number is even
            cout << i << " ";
        }
    }

    cout << endl;
    return 0;
}
```

☑ How it works:

1. User inputs a number `n`.
2. Loop `for (int i = 1; i <= n; i++)` runs from 1 to `n`.
3. `if (i % 2 == 0)` checks if the number is even.
4. Prints only even numbers.

Example Output:

```
Enter a positive number: 10
Even numbers from 1 to 10 are: 2 4 6 8 10
```

Example 6: Print a Simple Star Pattern

Question: Write a C++ program that asks the user to enter a positive number `N` and then prints a **right-angled triangle star pattern** with `N` rows using **nested for loops**.

Example:

- Input: `5`
- Output:

```
*
**
***
****
*****
```

Requirements:

1. Use `cin` to take input from the user.
2. Use a **nested for loop**:
    - Outer loop to control the rows.
    - Inner loop to print stars in each row.
3. Each row should have an increasing number of stars, starting from 1 star in the first row.

```
#include <iostream>
using namespace std;

int main() {
    int n;

    // Ask the user for the number of rows
    cout << "Enter the number of rows: ";
    cin >> n;

    // Outer loop for each row
    for (int i = 1; i <= n; i++) {
        // Inner loop to print stars in each row
        for (int j = 1; j <= i; j++) {
            cout << "*";
        }
        cout << endl; // Move to next row
    }

    return 0;
}
```

☑ How it works:

1. User inputs the number of rows `n`.
2. Outer loop `for (int i = 1; i <= n; i++)` → Controls the rows.
3. Inner loop `for (int j = 1; j <= i; j++)` → Prints stars in each row.
4. Each row has an increasing number of stars.

Example Output:

```
Enter the number of rows: 5
*
**
***
```

```
****
*****
```
