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--).

---