C++ 101 – Session 4

✅ 1. Introduction to the `for` Loop

When you know exactly how many times you want to repeat a block of code, the `for` loop is the best tool to use. It is commonly used for counting and iterating through ranges or collections.

📌 Syntax of a `for` Loop:

```
for (statement 1; statement 2; statement 3) {
    // code block to be executed
}
```

Part	Description
Statement 1	Runs **once** before the loop starts. Usually used to initialize a counter.
Statement 2	The **condition** that must be true for the loop to continue.
Statement 3	Executes **after each loop iteration**. Typically used to update the counter.

✅ 2. Example: Print Numbers 0 to 4

```
for (int i = 0; i < 5; i++) {
    cout << i << "\n";
}
```

🔍 Explanation:

- `int i = 0;` → Start counting from 0.
- `i < 5;` → Continue as long as `i` is less than 5.
- `i++` → Increase `i` by 1 each loop.
- Output:

```
0
1
2
3
4
```

✅ 3. Example: Print Even Numbers (0 to 10)

```
for (int i = 0; i <= 10; i = i + 2) {
    cout << i << "\n";
}
```

🔍 Explanation:

- The loop starts from 0 and increments by 2 each time.
- It only prints even numbers.
- Output:

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

✅ 4. Common `for` Loop Pattern with `if` Statement

```
cpp
CopyEdit
for (int i = 0; i < 10; i++) {
    if (i % 2 != 0) {
        continue; // skip odd numbers
    }
    cout << i << endl;
}
```

💡 Purpose:

- Skips odd numbers using `continue`.
- Only prints even numbers.

🛠️ 5. Task (In-Class Practice)

✅ Your Task:

Write a C++ program using a `for` loop that does the following:

1. **Print a number grid** using **nested loops**
   - Outer loop = rows
   - Inner loop = columns
   - Example Output (3x3 grid):

```
1 2 3
1 2 3
1 2 3
```

2. **Explore and implement a `for-each` loop (range-based loop)**
   - Create an array of 5 numbers
   - Use a `for-each` loop to print each number
   - Example:

```
int numbers[] = {10, 20, 30, 40, 50};

for (int num : numbers) {
    cout << num << endl;
}
```

💬 Notes:

- Use `\t` for spacing in the number grid if needed.
- Nested loops are useful for working with patterns, grids, and matrices.
- `for-each` loops are used for cleaner, simpler access to array elements.