C++ LESSON MODULE: THE FOR-EACH LOOP (RANGE-BASED FOR LOOP)

INTRODUCTION

The for-each loop (introduced in C++11) simplifies iteration through arrays, containers, and collections. It is also known as the range-based for loop. This loop automatically handles initialization, condition checking, and incrementing, making it easier to read and less error-prone than traditional for loops.

SYNTAX AND STRUCTURE

```
for (datatype
variable : container)
{
    // body of loop
}
```

Explanation:

- datatype: Type of each element in the container. You can also use the auto keyword for automatic deduction.
- variable: Represents the current element in each iteration.
- container: Refers to the collection (array, string, vector, set, map, etc.) being traversed.

Unlike classic loops, the range-based loop doesn't require explicit counters or iterators.

EXAMPLE 1: BASIC FOR-EACH LOOP

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

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    for (int value : arr) {
        cout << value << " ";
    }

    return 0;
}
```

OUTPUT: 10 20 30 40 50

Explanation: The loop automatically iterates over arr, assigning each element to value in turn – starting at 10 and ending at 50.

VISUALIZATION OF FOR-EACH EXECUTION

Visualization steps for the range-based for loop:

1. Initialize the loop variable with the first element of the collection.
2. Execute the loop body using that element.
3. Move to the next element automatically.
4. Continue until the end of the container is reached.

EXAMPLE 2: USING AUTO KEYWORD WITH VECTORS

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

int main() {
    vector<string> names =
{"Alice", "Bob", "Charlie"};

    for (auto name : names) {
        cout << name << endl;
    }

    return 0;
}
```

OUTPUT:

Alice
Bob
Charlie

Explanation: Using auto allows C++ to automatically detect the element type inside the vector, making the code concise and clean.

EXAMPLE 3: ITERATING OVER A MAP

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

int main() {
   map<int, string> students = {{1, "Liam"}, {2,
"Emma"}, {3, "Noah"}};

   for (auto [id, name] : students) {
      cout << "ID: " << id << " Name: " << name
<< endl;
   }

   return 0;
}
```

OUTPUT:

ID: 1 Name: Liam
ID: 2 Name: Emma
ID: 3 Name: Noah

Explanation: C++17 introduced structured bindings using [key, value] syntax, making it easier to access map pairs.

EXAMPLE 4: ITERATING A SET

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

int main() {
   set<int> values = {3, 1, 4, 1,
5, 9};

   for (const auto& val :
values) {
      cout << val << " ";
   }

   return 0;
}
```

OUTPUT: 1 3 4 5 9

Explanation: The range-based loop works with all STL containers, including sets. The keyword const auto& ensures elements are not copied or modified unnecessarily.

REFERENCES:

- GEEKSFORGEEKS: HTTPS://WWW.GEEKSFORGEEKS.ORG/CPP/RANGE-BASED-LOOP-C/
- GEEKSFORGEEKS: HTTPS://WWW.GEEKSFORGEEKS.ORG/CPP/G-FACT-40-FOREACH-IN-C-AND-JAVA/
- GEEKSFORGEEKS: HTTPS://WWW.GEEKSFORGEEKS.ORG/CPP/FOR_EACH-LOOP-C/
- GEEKSFORGEEKS: HTTPS://WWW.GEEKSFORGEEKS.ORG/CPP/RANGE-BASED-FOR-LOOP-WITH-SET-IN-CPP/

THANK YOU