**Range Based for Loop**

Range-based for loop in C++ is added since C++ 11. It executes a for loop over a range. Used as a more readable equivalent to the traditional for loop operating over a range of values, such as all elements in a container.

**Syntax :**

**for ( range\_declaration : range\_expression )**

**loop\_statement**

**Parameters :**

**range\_declaration :**

a declaration of a named variable, whose type is the

type of the element of the sequence represented by

range\_expression, or a reference to that type.

Often uses the auto specifier for automatic type

deduction.

**range\_expression :**

any expression that represents a suitable sequence

or a braced-init-list.

**loop\_statement :**

any statement, typically a compound statement, which

is the body of the loop.

C++ implementation :

C++

// Illustration of range-for loop

// using CPP code

#include <iostream>

#include <map>

#include <vector>

// Driver

int main()

{

// Iterating over whole array

std::vector<int> v = { 0, 1, 2, 3, 4, 5 };

for (auto i : v)

std::cout << i << ' ';

std::cout << '\n';

// the initializer may be a braced-init-list

for (int n : { 0, 1, 2, 3, 4, 5 })

std::cout << n << ' ';

std::cout << '\n';

// Iterating over array

int a[] = { 0, 1, 2, 3, 4, 5 };

for (int n : a)

std::cout << n << ' ';

std::cout << '\n';

// Just running a loop for every array

// element

for (int n : a)

std::cout << "In loop" << ' ';

std::cout << '\n';

// Printing string characters

std::string str = "Geeks";

for (char c : str)

std::cout << c << ' ';

std::cout << '\n';

// Printing keys and values of a map

std::map<int, int> MAP(

{ { 1, 1 }, { 2, 2 }, { 3, 3 } });

for (auto i : MAP)

std::cout << '{' << i.first << ", " << i.second

<< "}\n";

}

**Output**

0 1 2 3 4 5

0 1 2 3 4 5

0 1 2 3 4 5

In loop In loop In loop In loop In loop In loop

G e e k s

{1, 1}

{2, 2}

{3, 3}

**C++ 17 or higher:**Range-based loops can also be used with maps like this:

for (auto& [key, value]: myMap) {

cout << key << " has value " << value << std::endl;

}

**Here [key, value] works like elements of pair**which can be directly accessed without specifying first or second keyword.