Laboratory 07B
Range Based For Loop
&
Use of Vectors

CS-102

The Range-Based `for` Loop

- C++ 11 provides a specialized version of the `for` loop that, in many circumstances, simplifies array processing.
- *The range-based `for` loop is a loop that iterates once for each element in an array.*
- *Each time the loop iterates, it copies an element from the array to a built-in variable, known as the range variable.*
- The range-based `for` loop automatically knows the number of elements in an array.
  - You do not have to use a counter variable.
  - You do not have to worry about stepping outside the bounds of the array.

The Range-Based `for` Loop

- Here is the general format of the range-based for loop:

```
      for (dataType rangeVariable : array)
                statement;
```

- ***dataType*** is the data type of the range variable.
- ***rangeVariable*** is the name of the range variable. This variable will receive the value of a different array element during each loop iteration.
- ***array*** is the name of an array on which you wish the loop to operate.
- ***statement*** is a statement that executes during a loop iteration. If you need to execute more than one statement in the loop, enclose the statements in a set of braces.

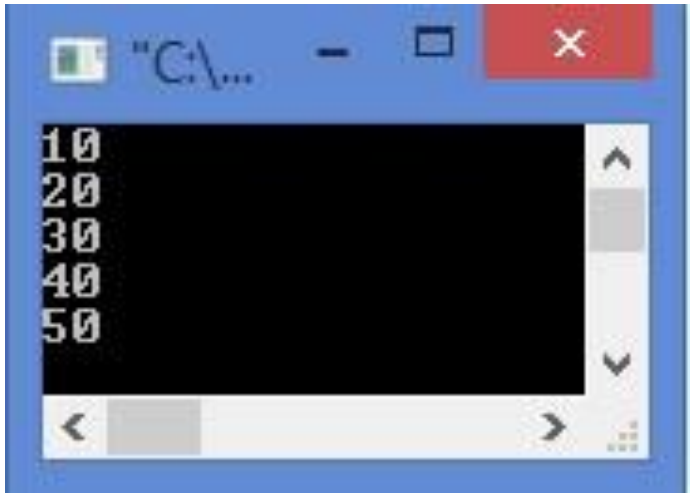
The range-based `for` loop in Program 7-10

```
// This program demonstrates the range-based for loop.
#include <iostream>
using namespace std;

int main()
{
   // Define an array of integers.
   int numbers[] = { 10, 20, 30, 40, 50 };

   // Display the values in the array.
   for (int val : numbers)
      cout << val << endl;

   return 0;
}
```



```
10
20
30
40
50
```

Program 7-11 Demonstrates Range-Based for loop with Strings

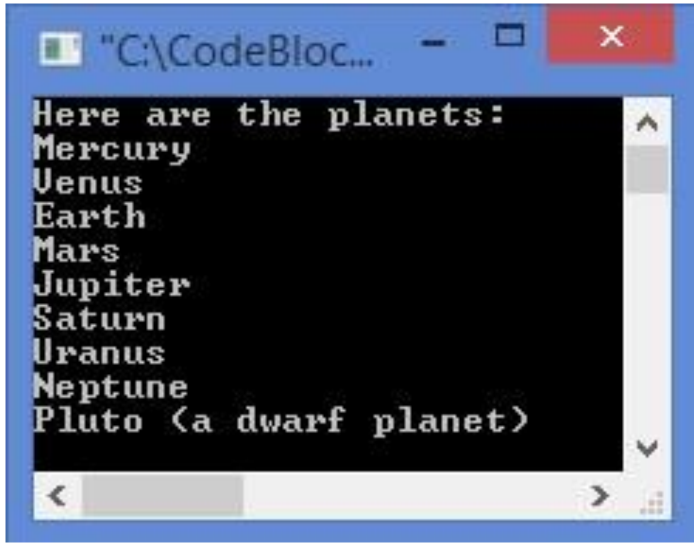
```
// This program demonstrates the range-based for loop.
#include <iostream>
#include <string>
using namespace std;

int main()
{
   string planets[] = { "Mercury", "Venus", "Earth", "Mars",
                        "Jupiter", "Saturn", "Uranus",
                        "Neptune", "Pluto (a dwarf planet)" };

   cout << "Here are the planets:\n";

   // Display the values in the array.
   for (string val : planets)   // Or you can say: for (auto val : planets)
      cout << val << endl;

   return 0;
}
```



```
Here are the planets:
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
Pluto (a dwarf planet)
```

Printing the Contents of an Array

- In C++ 11 you can use the range-based `for` loop to display an array's contents, as shown here:

```
for (int val : numbers)
   cout << val << endl;
```

Summing and Averaging Array Elements

- In C++ 11 you can use the range-based `for` loop, as shown here:

```
double total = 0;   // Initialize accumulator
double average;     // Will hold the average
for (int val : scores)
    total += val;
average = total / NUM_SCORES;
```

Laboratory 07B – Part 1

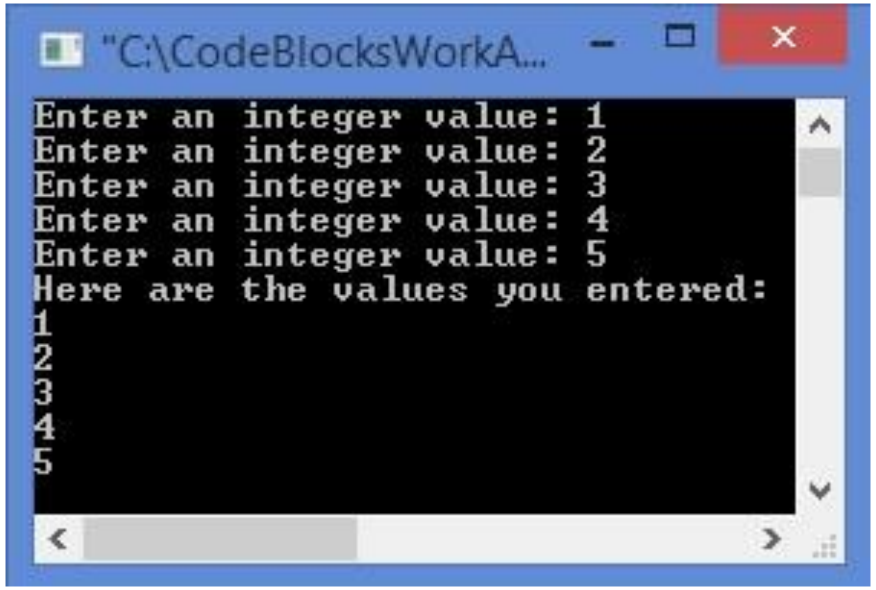
- Enter the programs, 7-10 and 7-11 into your IDE, and get them working. Study the mechanism so you are clear on how it works.
- Note that this particular syntax does not allow you to use the element subscripts. In cases where you need to use the element subscripts, you must use the regular for loop.
- Rewrite programs 7-10 and 7-11, using a regular **for** loop:
  - Call your programs: *YourName*-Lab07B-1a.cpp and *YourName*-Lab07B-1b.cpp
  - If you are doing this lab synchronously, call your instructor so that you can receive credit for this part of the assignment.
  - If you are doing this lab asynchronously, please submit your program to Canvas.

Modifying an Array with a Range-Based **`for`** Loop

- As the range-based `for` loop executes, its range variable contains only a copy of an array element.
- You cannot use a range-based `for` loop to modify the contents of an array unless you declare the range variable as a reference variable.
- To declare the range variable as a reference variable, simply write an ampersand (`&`) in front of its name in the loop header.
- Program 7-12 demonstrates

Program 7-12: Using a range-based for loop to modify contents of Array

```
// This program uses a range-based for loop to modify the contents of an array.
#include <iostream>
using namespace std;
int main()
{
  const int SIZE = 5;
  int numbers[5];
  // Get values for the array.
  for (int &val : numbers)   // Or you can say: for (auto &val : numbers)
  {
      cout << "Enter an integer value: ";
      cin >> val;
  }
  // Display the values in the array.
  cout << "Here are the values you entered:\n";
  for (int val : numbers)
      cout << val << endl;
  return 0;
}
```



```
Enter an integer value: 1
Enter an integer value: 2
Enter an integer value: 3
Enter an integer value: 4
Enter an integer value: 5
Here are the values you entered:
1
2
3
4
5
```

Laboratory 07B – Part 2

- Now we wish to modify the contents using the Range Based for loop. In order to do this we must use Reference variables.
  - This means that we will need to use the & in front of the variable in the loop header.
  - Examine Program 7-12, and note how the header of the Range Based for loop is changed when we wish to input new data into the array, by introducing the & into the syntax.
  - Enter Program 7-12, and run it so that you are clear how it works.
    - Call this program: *YourName*-Lab07B-2.cpp
    - Try omitting the & to see what happens when you don't use it.
- Call the instructor:
  - If you are doing the Lab synchronously, show what happens when you use the & and when you don't.
  - If you are doing the Lab asynchronously, submit your program to Canvas.

Laboratory 07B – Part 3

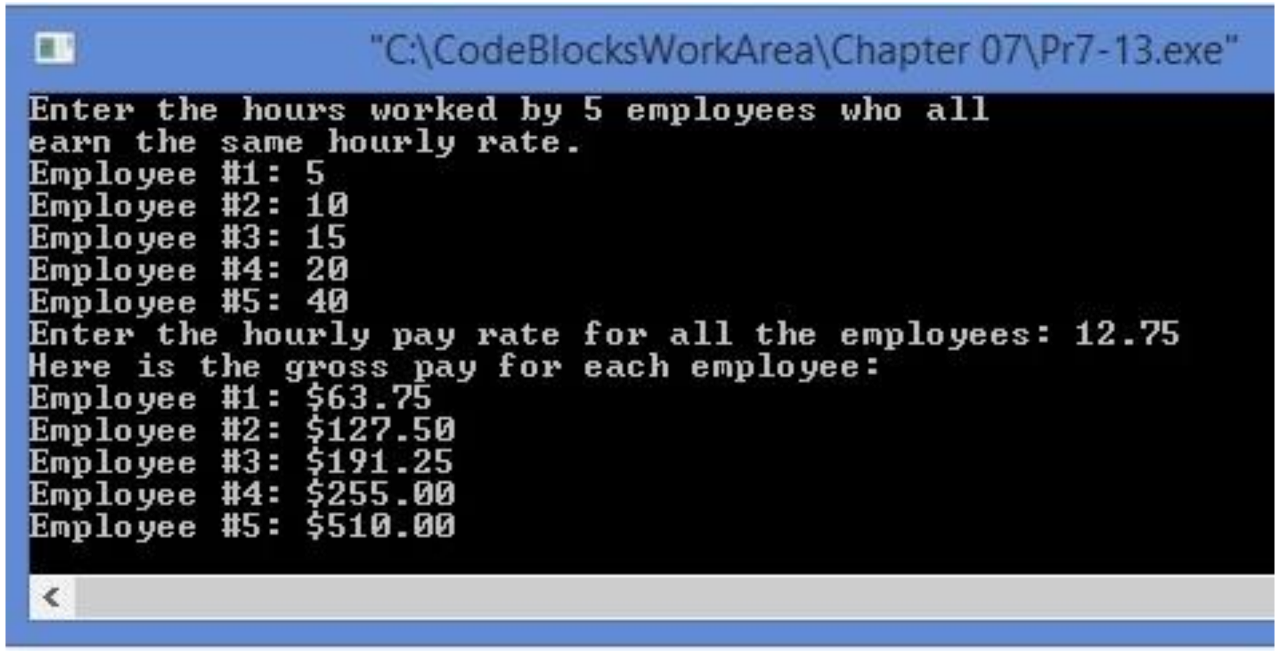
- Now, rewrite Program 7-13 to take advantage of the Range Based for loop.
- Call your program: *YourName*-Lab07B-3.cpp
- Once you have it up and running,
  - If you are doing the Lab synchronously, call the Instructor so that you can be given the proper credit for your work.
  - If you are doing the Lab asynchronously, submit your program into Canvas.
- As you can see, because we needed to preserve the index in order to display the same result, our Range Based solution actually took more steps than the regular for loop solution that our Range Based solution replaced.
- Clearly use of the Range Based for loop is of no advantage when we need to make use of the loop counting variable within the block of statements controlled by the loop.

Program 7-13

```
// This program stores, in an array, the hours worked by
// employees who all make the same hourly wage.
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   const int NUM_EMPLOYEES = 5;   // Number of employees
   int hours[NUM_EMPLOYEES];      // Array to hold hours
   double payrate;                // Hourly pay rate
   double grossPay;               // To hold the gross pay
   // Input the hours worked.
   cout << "Enter the hours worked by ";
   cout << NUM_EMPLOYEES << " employees who all\n";
   cout << "earn the same hourly rate.\n";
```

Program 7-13 Concluded

```
 for (int index = 0; index < NUM_EMPLOYEES; index++)
 {
      cout << "Employee #" << (index + 1) << ": ";
      cin >> hours[index];
 }
 // Input the hourly rate for all employees.
 cout << "Enter the hourly pay rate for all the employees: ";
 cin >> payrate;
 // Display each employee's gross pay.
 cout << "Here is the gross pay for each employee:\n";
 cout << fixed << showpoint << setprecision(2);
 for (int index = 0; index < NUM_EMPLOYEES; index++)
 {
      grossPay = hours[index] * payrate;
      cout << "Employee #" << (index + 1);
      cout << ": $" << grossPay << endl;
  }
  return 0;
}
```



```
Enter the hours worked by 5 employees who all
earn the same hourly rate.
Employee #1: 5
Employee #2: 10
Employee #3: 15
Employee #4: 20
Employee #5: 40
Enter the hourly pay rate for all the employees: 12.75
Here is the gross pay for each employee:
Employee #1: $63.75
Employee #2: $127.50
Employee #3: $191.25
Employee #4: $255.00
Employee #5: $510.00
```

Laboratory 7B – Part 4

- Whereas Arrays are passed by Reference to and from functions, by default, with Vectors, it is the other way around.
- Vectors are passed by Value by default.
- The following program will measure the time that it takes to pass a Vector of up to 100 million integers to a function.
- Enter and run this program obtaining the time it takes to pass a vector of this size by value.
- Then change the program so that it passes the Vector by Reference.
- Measure the time it takes and compare.
  - If you are doing this Lab synchronously, show your result to the Instructor.
  - If you are doing this Lab asynchronously, submit your result to Canvas.
- A program you can use to do this is shown on the next page.

Passing Vectors by Value to a function, and measuring the time it takes.

```
// Laboratory07B Passing Vectors by Value
// Creates a Vector with the number of elements input from the keyboard
// And reports the length of time it takes to pass the Vector by Value
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
void showStats(vector<int>);
int main()
{
    int index, num, startTime, endTime;
    vector<int>   bigVector;
    cout << "How many elements do you want in your Vector: ";
    cin >> num;
    for (index=0; index < num; index++)
        bigVector.push_back(index);                       // This loads the vector up with data
    auto started = chrono::high_resolution_clock::now();  // Starts the clock
    showStats(bigVector);
    auto done = chrono::high_resolution_clock::now();     // Stops the clock
    cout << chrono::duration_cast<chrono::milliseconds>(done-started).count();
    cout << " milliseconds required to pass by value ";
    cout << "a Vector having " << num << " elements!" << endl;
    return 0;
}
void showStats(vector<int> bigV)
{
     cout << "Passed all the data by Value!" << endl << endl;
}
```

```
// Laboratory07B Passing Vectors by Reference
// Creates a Vector with the number of elements input from the keyboard
// And reports the length of time it takes to pass the Vector by Reference
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
void showStats(vector<int> &);
int main()
{
    int index, num, startTime, endTime;
    vector<int>   bigVector;
    cout << "How many elements do you want in your Vector: ";
    cin >> num;
    for (index=0; index < num; index++)
        bigVector.push_back(index);                    // This loads the vector up with data
    auto started = chrono::high_resolution_clock::now();  // Starts the clock
    showStats(bigVector);
    auto done = chrono::high_resolution_clock::now();     // Stops the clock
    cout << chrono::duration_cast<chrono::milliseconds>(done-started).count();
    cout << " milliseconds required to pass by value ";
    cout << "a Vector having " << num << " elements!" << endl;
    return 0;
}
void showStats(vector<int> & bigV)
{
     cout << "Passed all the data by Reference" << endl << endl;
}
```

Passing Vectors by Reference to a function, and measuring the time it takes.

Laboratory 7B – Part 5

- Now fill out the function, showStats(), with code that will compute and then display the mean value of the data
- Note: If you return the value, mean, as reference variables, and display the results in main(), then the chrono clock timer will tell you exactly how long it took to do this computation.
- Try returning the mean by value, instead of by reference. What's the difference?
- If you are doing this Lab synchronously, show your results to the instructor.
- If you are doing this Lab asynchronously, submit your mean, median and times to Canvas.