Module 6: Range-based loops and Objects

Learning Objectives

1. Write code that uses range-based loops for std::vectors.
2. Write code that uses range-based loops for std::maps.
3. Write code that loops through and uses objects in containers (std::vector and std::map).

Process Skills

1. Information processing. Extract structural patterns from sample code.
2. Oral and written communication. Summarize the results of a discussion.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. If there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: ______________________________________ Date: _______________

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	

For virtual activities: Once you select your roles, change your Zoom name using the format and example below.

Format: Group X: First name, Last name initial / Role

Example: Group 1: Paul I / Presenter

Model 1. Range-based loop and std::vector (6 min)

Start time: _______

```
#include <iostream>
#include <vector>

int main() {
  std::vector<double> donations {
    100.0,
    224.25,
    5.75
  };
  // Range-based loop
  for (double donation: donations)
{
    std::cout << donation << "\n";
  }
  return 0;
}
```

Iteration	donation
1	100.0
2	224.25
3	5.75

Screen output:

```
100
224.25
5.75
```

1. The code shows a range-based for loop going over the elements of the donations std::vector and performing the code block that follows. An *iteration* refers to the process of performing a code block once. How many iterations were performed over the donations std::vector?

3

2. We see the donation variable's value change in every iteration. Where do you think these values were taken from? Place a check (✓) beside your answer.

   a. Elements of the donations std::vector ✓
   b. Randomly generated values
   c. Default values of the double data type

3. Why do you think we declare the donation variable with a double type? Place a check (✓) beside your answer.

   a. doubles can store any value that a std::vector might contain
   b. The std::vector contains elements with a double data type ✓
   c. std::vectors contain doubles by default

4. In what order are donations' elements assigned to donation as the range-based loop iterates? Place a check (✓) beside your answer.

    a. donation takes in values randomly
    b. donation takes in values from smallest to largest
    c. donation takes in the values sequentially (index 0, 1, …) ✓

5. Complete the code below to display all values inside the customers_per_hour std::vector. Display each value in a separate line (using std::endl or "\n").

```
#include <iostream>
#include <vector>

int main() {
  std::vector<int> customers_per_hour {2, 4, 1, 4, 3, 2, 1, 1};

  for (int customers: customers_per_hour) {
    std::cout << customers << "\n";
  }
  return 0;
}
```

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**

Model 2. Range-based loop and std::map (10 min)

Start time: ______

std::pair

T1 first;
T2 second;

pair(const T1& x, const T2& y);
 	
// member functions not shown
// here

std::pair is a template class, where we provide the contained elements' data types during construction. We use T1 to refer to the data type of its first element, and T2 for its second element.

Member variables
T1 first; - first value with T1 type

T2 second - second value with T2 type

Member functions
pair(const T1& x, const T2& y); - Initializes first with

x and second with y.

```
#include <iostream>
#include <map>

int main() {
  std::map<int, std::string> products {
    {112, "Apple"},
    {113, "Milk"},
    {109, "Banana"}
  };
  // Range-based loop
  for (std::pair<int, std::string> product: products) {
    std::cout << product.first << ": "
              << product.second << "\n";
  }
  return 0;
}
```

Iteration	product

1	**product: std::pair**
	first = 109
	second = "Banana"

2	**product: std::pair**
	first = 112
	second = "Apple"

3	**product: std::pair**
	first = 113
	second = "Milk"

Screen output:

```
109: Banana
112: Apple
113: Milk
```

6. How many iterations were performed over the products std::map?

3

7. Are the std::map's elements accessed in the same order as they were added using the initialization list constructor? Place a check (✓) beside your answer.

    a. Yes
    b. No ✓

8. In the range-based loop for std::maps, we see that a std::pair object was used to store values in each iteration. Why do you think we use a std::pair instead of a single int or std::string?

The products map contains a key and a value so they can be stored in a std::pair, but not an int or std::string

9. Why do we use the dot notation to access the values of first and second?

std::pair is an object and first and second are its member variables.

10. What information does the product object's first member variable contain? Place a check (✓) beside your answer.

    a. key ✓
    b. value
    c. index

11. What information does the product object's second member variable contain? Place a check (✓) beside your answer.

    a. key
    b. value ✓
    c. index

12. Complete the code below to display all elements inside the food_inventory std::map. Use the format, <quantity> x <name> for each element. For example, 1 x Apple.

```
#include <iostream>
#include <map>

int main() {
  std::map<std::string, int> food_inventory {
    {"Apple", 1},
    {"Banana", 5},
    {"Milk", 3}
  };

  for (std::pair<std::string, int> food: food_inventory) {
    std::cout << food.second << " x " << food.first << "\n";
  }

  return 0;
}
```

🛑 **STOP HERE AND WAIT FOR FURTHER INSTRUCTIONS**