# Module 5: Maps

## Learning Objectives

1. Write code to call member functions given member function declarations.
2. Write code that creates and uses maps.

## Process Skills

1. Information processing. Extract structural patterns from sample code.
2. Critical thinking. Interpret member function declarations.

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. std::map (3 min)

Start time: ______

The class diagram below shows a simplified version of the std::map class that highlights common methods. You can find the full map class at https://en.cppreference.com/w/cpp/container/map.

| **std::map** |
|---|
| // member variables not<br>// shown here |
| map();<br>map(const std::map& other);<br>map(std::initializer_list<value_type> init);<br>void insert(std::initializer_list<value_type> ilist);<br>T& at(const Key& key);<br>void clear();<br>bool empty() const;<br>size_type size() const;<br><br>// other member functions not<br>// shown here |

*std::map is a template class, where we provide the contained elements' data type during construction. We use Key and T to refer to the template data types.*

**std::map Member functions**
**map();** - constructs an empty container.

**map(const std::map& other);** - constructs a container and copies the contents of other

**map(std::initializer_list<value_type> init);** - constructs a container with the contents of the initializer list init

**void insert(std::initializer_list<value_type> ilist);** - inserts elements from initializer list ilist.

**T& at(const Key& key);** - returns a reference to the mapped value of the element with key equivalent to key.

**void clear();** - erases all elements from the container. After this call, size() returns zero.

**size_type size();** - returns the number of elements in the container.

**bool empty() const;** - checks if the container has no elements.

1. How many constructors are shown in the std::map class diagram?

| |
|---|
| 3 |

2. Does the insert member function return a value? Place a check (✓) beside your answer.

   a. Yes
   b. No ✓

3. What value is returned by a std::map object's size member function after calling its clear member function?

| |
|---|
| 0 |

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

## Model 2. Using std::map (15 min)

Start time: ______

| Line | Code | Visualization |
|---|---|---|
| 01 | `std::map<int, std::string> products;` | products: std::map |
| 02 | `products.insert({112, "Apple"});` | products: std::map<br>**Key** \| **Value**<br>112 \| Apple |
| 03 | `products.insert({113, "Milk"});` | products: std::vector<br>**Key** \| **Value**<br>112 \| Apple |

| | | |
|---|---|---|
| | | 113 Milk |
| 04 | `products.at(113) = "Milk (1gal)";` | products: std::map<br>**Key** **Value**<br>112 Apple<br>113 Milk (1 gal) |
| 05 | `std::cout << products.at(112);` | Screen output: Apple |
| 06<br><br>07<br>08 | `std::size_t product_count =`<br>`  products.size();`<br>`std::cout << product_count`<br>`          << "\n";` | Screen output: 2 |
| 09 | `products.clear();` | products: std::map |
| 10<br>11<br>12<br>13<br>14<br>15<br>16 | `bool is_empty = products.empty();`<br>`if (is_empty) {`<br>`  std::cout << "No products.\n";`<br>`} else {`<br>`  std::cout << "Products added to the"`<br>`            << "system.\n";`<br>`}` | Screen output: No donations |

4. Which std::map constructor did we use in line 01? Write the constructor's function declaration below.

map(); also known as the default constructor

5. After performing line 03, how many elements are in the std::map container? *Take note that C++ uses a hashing algorithm on the key to store elements internally. Elements may not always be stored in the order you insert them*.

| |
|---|
| 2 |

6. After performing line 04, what value is assigned to the key 113?

| |
|---|
| `Milk (1gal)` |

7. Analyze as a group line 05. Explain why passing 112 as the argument gives back the value "Apple". Write your explanation below.

| |
|---|
| The at member function takes a key as an argument that it uses to locate and return a reference to the corresponding value. In this case, the value "Apple" is associated with the key 112. |

8. What does the size member function return? Place a check (✓) beside your answer.

   a. The position of the last element in the std::map.
   b. The number of elements in the std::map. ✓
   c. The total number of keys and values in the std::map combined.

9. When using a std::map, what values do we expect to get from the size and empty member functions after calling the clear member function? Complete the table below with the expected values.

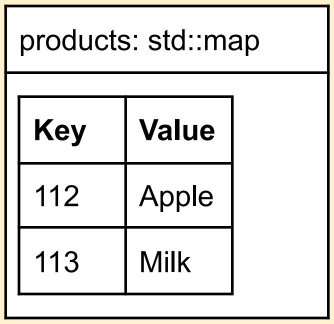
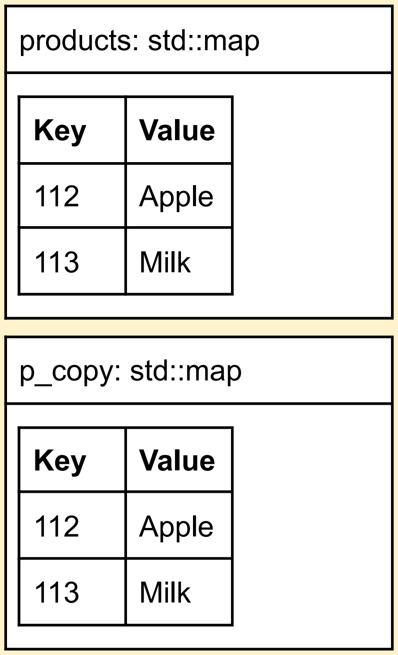
| **member function** | **value after products.clear()** |
|---|---|
| products.size() | 0 |
| products.empty() | true |

10. Write code that uses a std::map to create a food inventory. We will use a food's name as the key to store the corresponding quantity. Add the following information to the map: 1 Apple, 5 bananas, and 3 milk. Ask the user for the updated number of bananas in the inventory and update the map accordingly. Use the map to display the quantities of each item.

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

int main() {
  int new_banana_count = 0;
  // TODO: Create std::map to represent a food inventory.
  std::map<std::string, int> inventory;

  // TODO: Add 1 Apple, 5 Bananas, and 3 Milk to food inventory.
  inventory.insert({"Apple", 1});
  inventory.insert({"Banana", 5});
  inventory.insert({"Milk", 3});

  std::cout << "Updated amount of bananas: ";
  std::cin >> new_banana_count;
  // TODO: Update banana count
  inventory.at("Banana") = new_banana_count;

  std::cout << "Number of apples: ";
  // TODO: Display number of apples
  std::cout << inventory.at("Apple");

  std::cout << "\nNumber of bananas: ";
  // TODO: Display number of bananas
  std::cout << inventory.at("Banana");

  std::cout << "\nNumber of milk: ";
  // TODO: Display number of milk
  std::cout << inventory.at("Milk");

  return 0;
}
```

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

## Model 3. std::map initialization (5 min) Start time: ______

| Line | Code | Visualization |
|---|---|---|
| 01<br>02<br>03<br>04 | `std::map<int, std::string> products {`<br>`  {112, "Apple"},`<br>`  {113, "Milk"}`<br>`};` | products: std::map<br>Key / Value<br>112 / Apple<br>113 / Milk |
| 05 | `std::map<int, std::string> p_copy(products);` | products: std::map<br>Key / Value<br>112 / Apple<br>113 / Milk<br>p_copy: std::map<br>Key / Value<br>112 / Apple<br>113 / Milk |

11. Match the constructor's declaration with the most likely code that used it for instantiating the std::map. Write the line number of the corresponding code in the table below.

| Constructor declaration | Line # (01 - 4, or 05) |
|---|---|
| map(const std::map& other); | 05 |
| map(std::initializer_list<value_type> init); | 01 - 04 |

12. Rewrite your code for creating and filling a food inventory in question 10 with the initialization list constructor (see lines 01 - 04 for an example). Use the std::map copy constructor to create a copy of the food inventory.

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

int main() {
  // TODO: Write code create a food inventory and fill it with the following information:
  // 1 Apple, 5 bananas, and 3 milk. Use the initialization list constructor.
  std::map<std::string, int> inventory {
    {"Apple", 1},
    {"Banana", 5},
    {"Milk", 3}
  };

  // TODO: Create a copy of the food inventory you just created using the copy constructor.
  std::map<std::string, int> inventory_copy(inventory);
  return 0;
}
```

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

## Extra challenge (6 min)

Start time: ______

13. Analyze the code below as a group. Is the code below valid? Explain your reasoning in the box below. Assume the Food object has already been defined and it provides a constructor that takes in the name of the food and its quantity.

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

int main() {
  std::map<int , Food> product_inventory;

  Food food1("Apple", 1);
  Food food2("Banana", 5);
  Food food3("Milk", 3);

  product_inventory.insert({112, food1});
  product_inventory.insert({113, food2});
  product_inventory.insert({114, food3});

  std::map<int, Food> inventory_copy(product_inventory);
  return 0;
}
```

Yes, because the std::map does not limit the data types you can use as keys and values. In this case, it uses an int for the key and a Food object as its value.

## Reflector questions

1. What was the most useful thing your team learned during this session?

2. What previous discussion helped in learning about creating and using std::map?

3. Did you change your strategy for working as a team in this session?

4. Did you notice an improvement in your team's performance?