# Recipe Management System

## Overview

This system is a **Recipe Management System** that allows users to manage and organize a collection of recipes. It includes functionalities for adding, removing, sorting, searching, and displaying recipes. Additionally, the system features colored outputs to improve the user experience. Below are the key features and uses of the system:

### Key Features

* **Add Recipe**: Allows users to add recipes with a name, ingredients, and preparation time.
* **Delete Recipe:** Enables users to remove a recipe by its name.
* **Search Recipe**: Lets users search for a recipe by name, displaying the details with colored output.
* **Display All Recipes**: Lists all available recipes with their details.
* **Sort Recipes**: Sorts recipes by preparation time in ascending order.

## Algorithms Used

### 1. Linear Search Algorithm (searchRecipe)

* **Task:** Search for a recipe by its name in the system.
* **Algorithm**:  
  1. Start from the first recipe in the collection.  
  2. Compare the recipe's name with the search term.  
  3. If a match is found, display the recipe details.  
  4. If no match is found after checking all recipes, display an error message.
* **Time Complexity:** O(n), where n is the number of recipes.

### 2. Insertion Algorithm (addRecipe)

* **Task:** Add a new recipe to the system.
* **Algorithm:**  
  1. Take input for the recipe's name, ingredients, and preparation time.  
  2. Add the recipe to both a map (for fast lookup) and a vector (for display and sorting).
* **Time Complexity:** O(1) for adding to both the map and the vector.

### 3. Deletion Algorithm (deleteRecipes)

**Task:** Remove a recipe from the system by its name.  
**Algorithm:**1. Search for the recipe in the map using its name

2. If found, remove it from both the map and the vector.  
**Time Complexity:** O(n), for removing from the vector, O(1) for removing from the map.

### 4. Sorting Algorithm (sortRecipes)

**Task:** Sort recipes in ascending order of preparation time.

**Algorithm:**  
1. Use the std::sort function with a custom comparator.  
2. Compare recipes based on their preparation time.  
3. Rearrange the recipes in ascending order.  
**Time Complexity**: O(n log n), where n is the number of recipes.

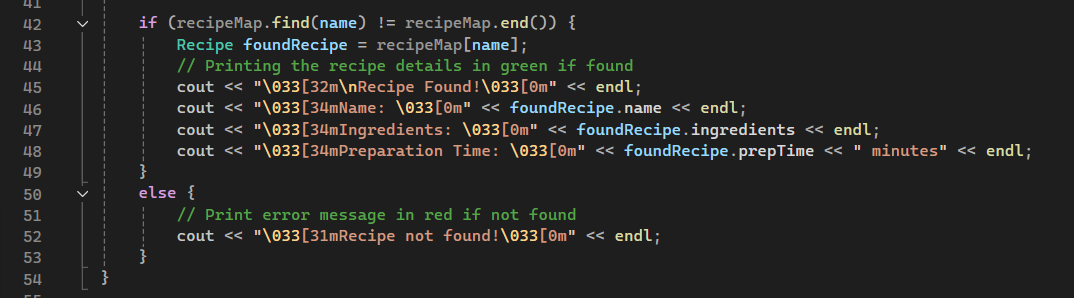
### 5. Display Algorithm (displayRecipes)

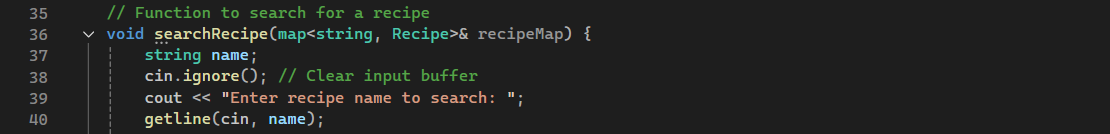
**Task:** Display all recipes in the system.  
**Algorithm:**1. Iterate through the vector containing all recipes.  
2. Print each recipe's name, ingredients, and preparation time.  
**Time Complexity:** O(n), where n is the number of recipes.

## Data Structures Used in the Code

### 1. Struct (Recipe)

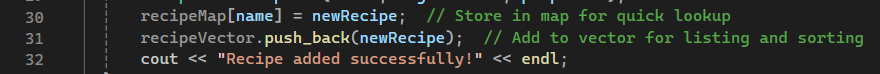
**Usage:** The Recipe structure is used to group related details of a recipe, such as name, ingredients, and preparation time.

  
**Operations:** Used for defining and managing individual recipes.

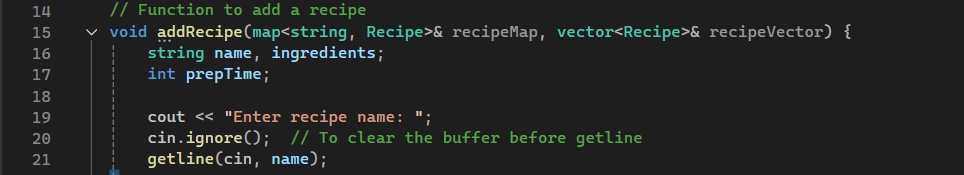


### 2. Map (std::map)

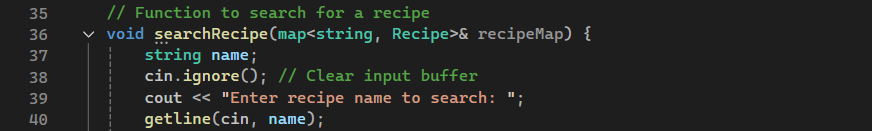
**Us age:** Stores recipes with the recipe name as the key and recipe details as the value.

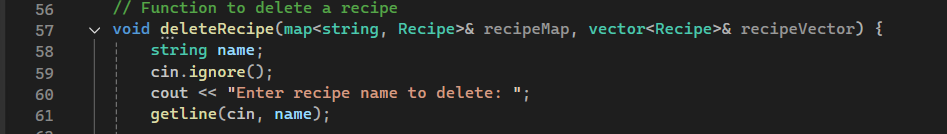


**Operations:**  
- Add Recipe: O(1)



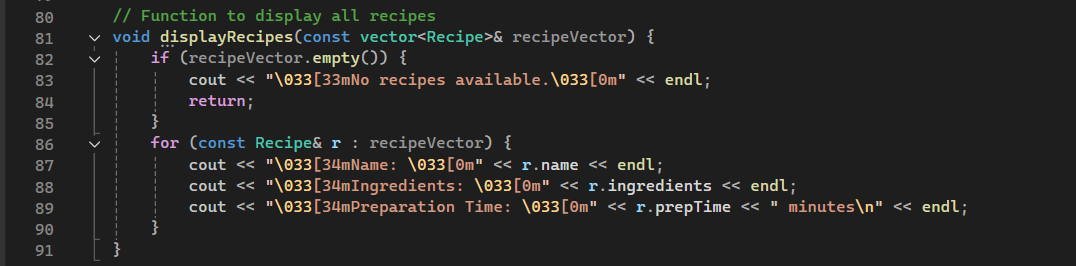
- Search Recipe: O(1)

  
- Delete Recipe: O(1)

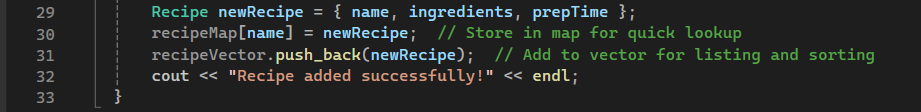


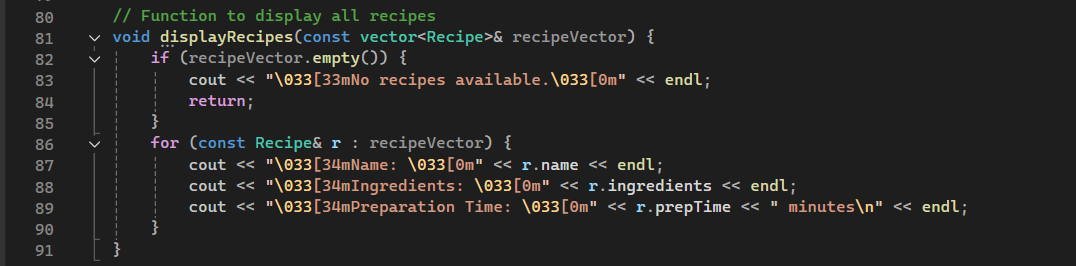
### 3. Vector (std::vector)

**Usage:** Maintains a dynamic list of recipes for display and sorting.

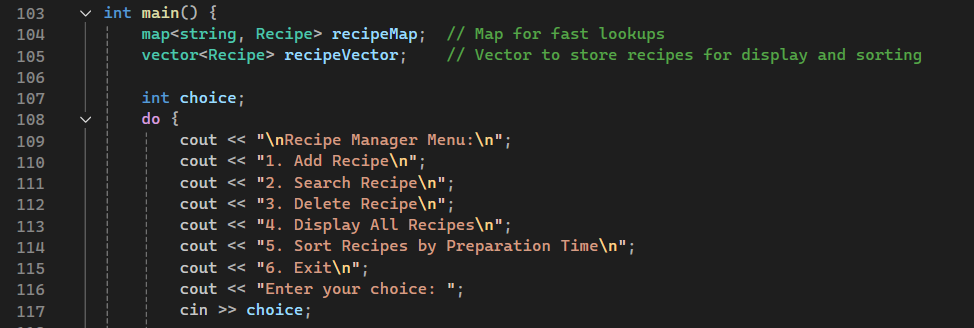


**Operations:**  
- Add Recipe: O(1)



- Display Recipes: O(n) 

- Sort Recipes: O(n log n)



- Delete Recipe: O(n)

