C++ 容器类 `<array>`

C++11 标准引入了 `<array>` 头文件，它提供了**一种固定大小的数组容器***，与 C 语言中的数组相比，具有更好的类型安全和内存管理特性。

`std::array` 是 C++ 标准库中的一个模板类，它定义在 `<array>` 头文件中。`std::array` 模板类提供了一个固定大小的数组，其大小在编译时确定，并且不允许动态改变。

语法

`std::array` 的基本语法如下：

```
#include <array>

std::array<T, N> array_name;
```

- `T` 是数组中元素的类型。
- `N` 是数组的大小，必须是一个非负整数。

特点

- **类型安全**：`std::array` 强制类型检查，避免了 C 语言数组的类型不安全问题。
- **固定大小**：数组的大小在编译时确定，不能在运行时改变。
- **内存连续**：`std::array` 的元素在内存中是连续存储的，这使得它可以高效地访问元素。
- **标准容器**：`std::array` 提供了与 `std::vector` 类似的接口，如 `size()`，`at()`，`front()`，`back()` 等。

实例

下面是一个使用 `std::array` 的简单示例，包括输出结果。

```
#include <iostream>
#include <array>

int main() {
  // 创建一个包含 5 个整数的 std::array
  std::array<int, 5> myArray = {1, 2, 3, 4, 5};

  // 使用范围 for 循环遍历数组
  for (const auto& value : myArray) {
    std::cout << value << " ";
  }
  std::cout << std::endl;

  // 使用索引访问数组元素
  std::cout << "Element at index 2: " << myArray.at(2) << std::endl;

  // 获取数组的大小
  std::cout << "Array size: " << myArray.size() << std::endl;
```

```
    // 修改数组元素
    myArray[3] = 10;

    // 再次遍历数组以显示修改后的元素
    for (const auto& value : myArray) {
        std::cout << value << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

输出结果

```
1 2 3 4 5
Element at index 2: 3
Array size: 5
1 2 3 10 5
```

`std::array` 是 C++ 标准库中一个非常有用的容器，它提供了固定大小的数组，具有类型安全和内存管理的优势。通过上述示例，我们可以看到如何创建、访问和修改 `std::array` 的元素。对于需要固定大小数组的场景，`std::array` 是一个非常好的选择。