


原创 行码棋 已于 2023-12-11 15:28:18 修改 阅读量1.1w 收藏 71 点赞数 27

分类专栏: stl 文章标签: c++ 开发语言 STL array

 stl 专栏收录该内容

236 订阅 17 篇文章

返回STL主目录

1 介绍 1

头文件

```
1 | #include<array>
```

array 是C++11新增的容器，效率与普通数据相差无几，比 vector 效率要高，自身添加了一些成员函数。

和其它容器不同， array 容器的大小是固定的，无法动态的扩展或收缩，只允许访问或者替换存储的元素。

注意：

array 的使用要在 std 命名空间里

2 简实使用 2

1.声明与初始化

1.1.基础数据类型

声明一个大小为100的 int 型数组，元素的值不确定

```
1 | array<int, 100> a;
```

声明一个大小为100的 int 型数组， 初始值 均为 0 (初始值与默认 元素类型 等效)

```
1 | array<int, 100> a{};
```

声明一个大小为100的 int 型数组，初始化部分值，其余全部为 0

```
1 | array<int, 100> a{1, 2, 3};
```

或者可以用等号

```
1 | array<int, 100> a = {1, 2, 3};
```

1.2.高级数据类型

不同于数组的是对元素类型不做要求，可以套结构体

```
1 | array<string, 2> s = {"ha", string("haha")};
2 | array<node, 2> a;
```

2. 取存元素值

★修改元素

```
1 | array<int, 4> a = {1, 2, 3, 4};
2 | a[0] = 4;
```

★访问元素

★★下标访问

 行码棋 已关注

👍 27 🗨 71

```
3 | cout << a[i] << " \n"[i == 3];
```

☆☆利用 auto 访问

```
1 | for(auto i : a)
2 |     cout << i << " ";
```

☆☆迭代器访问

```
1 | auto it = a.begin();
2 | for(; it != a.end(); it++)
3 |     cout << *it << " ";
```

☆☆ at() 函数访问

下标为 1 的元素加上下标为 2 的元素，答案为 5

```
1 | array<int, 4> a = {1, 2, 3, 4};
2 | int res = a.at(1) + a.at(2);
3 | cout << res << "\n";
```

☆☆ get 方法访问

将 a 数组下标为 1 位置处的值改为 x

💧注意💧 获取的下标只能写数字，不能填变量

```
1 | get<1>(a) = x;
```

3 成员函数 3

成员函数	功能
begin()	返回容器中第一个元素的访问迭代器（地址）
end()	返回容器最后一个元素之后一个位置的访问迭代器（地址）
rbegin()	返回最后一个元素的访问迭代器（地址）
rend()	返回第一个元素之前一个位置的访问迭代器（地址）
size()	返回容器中元素的数量，其值等于初始化 array 类的第二个模板参数 N
max_size()	返回容器可容纳元素的最大数量，其值始终等于初始化 array 类的第二个模板参数 N
empty()	判断容器是否为空
at(n)	返回容器中 n 位置处元素的引用，函数会自动检查 n 是否在有效的范围内，如果不是则抛出 out_of_range 异常
front()	返回容器中第一个元素的直接引用，函数不适用于空的 array 容器
back()	返回容器中最后一个元素的直接引用，函数不适用于空的 array 容器。
data()	返回一个指向容器首个元素的指针。利用该指针，可实现复制容器中所有元素等类似功能
fill(x)	将 x 这个值赋值给容器中的每个元素,相当于初始化
array1.swap(array2)	交换 array1 和 array2 容器中的所有元素，但前提是它们具有相同的长度和类型

4 部分用法示例 4

data()

指向底层元素存储的指针。对于非空容器，返回的指针与首元素地址比较相等。

at()

下标为 1 的元素加上下标为 2 的元素，答案

行码棋

已关注

27



71