【KATA 练习日记】关于 std::accumulate 的使用

作者：skjsnb
原文链接：https://ld246.com/article/1674961398082
来源网站：链滴

```cpp
#include <iostream>
#include <vector>
#include <numeric>

void foo1(const std::vector<int>& nums){
    int res = std::accumulate(nums.begin(),nums.end(),0,[](int a,int b) -> int {
        return a*b;
    });
    std::cout << "foo1: " << res << std::endl;
}

void foo2(const std::vector<int>& nums){
    std::string res = std::accumulate(std::next(nums.begin()),nums.end(),std::to_string(nums[0]),[](const std::
        return a + ";" + std::to_string(b);
    });
    std::cout << "foo2: " << res << std::endl;
}

std::string foo3_operation(const std::string& a ,int b){
    return a + ";" + std::to_string(b);
}

void foo3(const std::vector<int>& nums){
    std::string res = std::accumulate(std::next(nums.begin()),nums.end(),std::to_string(nums[0]),foo3_operation
    std::cout << "foo3: " << res << std::endl;
}

void print_vector(const std::vector<int>& nums){
    for (auto i :int : nums){
```

前言

今天的练习是6段难度的题目-> Find the odd int | Codewars

Given an array of integers, find the one that appears an odd number of times.

There will always be only one integer that appears an odd number of times.

Examples

[7] should return 7, because it occurs 1 time (which is odd).

[0] should return 0, because it occurs 1 time (which is odd).

[1,1,2] should return 2, because it occurs 1 time (which is odd).

[0,1,0,1,0] should return 0, because it occurs 3 times (which is odd).

[1,2,2,3,3,3,4,3,3,3,2,2,1] should return 4, because it appears 1 time (which is odd).

看起来比较简单，大体上需要做的就是寻找这一段数列中奇数项，最简单的方式是两个循环嵌套，但这样效率太差，鉴于我的C++水平有限，还在学习中，我用排序的方式实现了这个题目，代码如下：

```cpp
#include <algorithm>
#include <vector>

int findOdd(const std::vector<int>& numbers){
```

```
    std::vector<int> num(numbers);
    std::sort(num.begin(), num.end());
    for (int i = 0; i < num.size(); ++i) {
        if (num[i] != num[i+1] && (i+1) % 2 != 0) {
            return num[i];
        }
    }
    return 0;
}
```

这段代码可以执行并通过测试，速度也还可以。我习惯性的看了下别人的解决方案，结果不出我所料大佬们用了非常简洁的方式去实现，代码如下：

```
#include <functional>
#include <numeric>
#include <vector>

int findOdd(const std::vector<int>& numbers) {
  return std::accumulate(numbers.cbegin(), numbers.cend(), 0, std::bit_xor<>());
}
```

可以发现，这段代码关键在于 std::accumulate 这个函数。

---

std::accumulate

在cppreference网站中，是这样解释的：

计算给定值 **init** 与给定范围 [first, last) 中元素的和。第一版本用 operator+ ，第二版本用二元函数 **p** 求和元素，两个版本都会将 std::move 应用到它们的左侧运算数 **(C++20 起)**。

op 不能使涉及范围的任何迭代器（包含尾迭代器）失效，且不修改它所涉及范围的任何元素及 *****list。

大概就是将容器内的数列累加求和，并支持自定义运算。

1. 函数原型：

C++ 20前

```
// 1
template< class InputIt, class T >
T accumulate( InputIt first, InputIt last, T init );

// 2
template< class InputIt, class T, class BinaryOperation >
T accumulate( InputIt first, InputIt last, T init, BinaryOperation op );
```

从C++ 20开始

```
// 1
```

```
template< class InputIt, class T >
constexpr T accumulate( InputIt first, InputIt last, T init );

// 2
template< class InputIt, class T, class BinaryOperation >
constexpr T accumulate( InputIt first, InputIt last, T init, BinaryOperation op );
```

参数及返回值定义

- **first, last**：元素范围
- **init**：迭代初值
- **op**：自定义运算函数，接收当前积累值 **a**（初始化为 **init**）和当前元素 **b** 。

return: 给定值与给定范围中的元素的和 或者 给定范围在 **op** 上左折叠的结果。

注意事项

1. InputIt 必须符合老式输入迭代器 (LegacyInputIterator) 的要求。
2. T 必须符合可复制赋值 (CopyAssignable) 和 可复制构造 (CopyConstructible) 的要求。

2. 用法实例

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

void foo1(const std::vector<int>& nums){
    int res = std::accumulate(nums.begin(),nums.end(),0,[](int a,int b){
        return a*b;
    });
    std::cout << "foo1: " << res << std::endl;
}

void foo2(const std::vector<int>& nums){
    std::string res = std::accumulate(std::next(nums.begin()),nums.end(),std::to_string(nums[0]),[
(const std::string& a,int b){
        return a + ";" + std::to_string(b);
    });
    std::cout << "foo2: " << res << std::endl;
}

std::string foo3_operation(const std::string& a ,int b){
    return a + ";" + std::to_string(b);
}

void foo3(const std::vector<int>& nums){
    std::string res = std::accumulate(std::next(nums.begin()),nums.end(),std::to_string(nums[0]),
oo3_operation);
    std::cout << "foo3: " << res << std::endl;
}
```

```
void print_vector(const std::vector<int>& nums){
    for (auto i : nums){
        std::cout << i << " ";
    }
    std::cout << std::endl;
}
```

参考资料

- std::accumulate - cppreference.com
- std::accumulate_林多的博客-CSDN博客_std::accumulate

代码仓库

- kata/main.cpp at main · skjsnb/kata (github.com)
- my_cpp_learn/main.cpp at main · skjsnb/my_cpp_learn (github.com)

我的博客站点：Jelin's Blog (skjsnb.com)