

现代C++题目（答案与解析）

卢瑟帝国

2023 年 11 月 11 日

目录

1	实现管道运算符	2
1.1	答案	2
1.2	解析	3
2	实现自定义字面量 <code>_f</code>	3
2.1	答案	4
2.2	解析	4
3	实现 <code>print</code> 以及特化 <code>std::formatter</code>	5
4	给定模板类修改，让其对每一个不同类型实例化有不同 ID	5
5	实现 <code>scope_guard</code> 类型	5
6	解释 <code>std::atomic</code> 初始化	5
7	<code>throw new MyException</code>	5
8	定义 <code>array</code> 推导指引	5
9	名字查找的问题	5
10	遍历任意聚合类数据成员	5
11	<code>emplace_back()</code> 的问题	5
12	实现 <code>make_vector()</code>	5
13	关于 <code>return std::move(expr)</code>	5

暂时只有 13 道题目，并无特别难度，有疑问可看[视频教程](#)或答案解析。

1 实现管道运算符

日期：2023/7/21 出题人：mq白

给出以下代码，在不修改已给出代码的前提下使它满足运行结果。

```
1  int main(){
2      std::vector v{1, 2, 3};
3      std::function f {[](const int& i) {std::cout << i << ' '; } };
4      auto f2 = [](int& i) {i *= i; };
5      v | f2 | f;
6  }
```

要求运行结果

1 4 9

• 难度： ★ ★ ☆ ☆ ☆

提示：T& operator|(T& v, const T& f) 。

1.1 答案

```
1  template<typename U, typename F>
2      requires std::regular_invocable<F, U> //可加可不加，不会就不加
3  std::vector<U>& operator|(std::vector<U>& v1, F f) {
4      for (auto& i : v1) {
5          f(i);
6      }
7      return v1;
8  }
```

不使用模板：

```
1  std::vector<int>& operator|(std::vector<int>& v1, const std::function<void(int&)>& f) {
2      for (auto& i : v1) {
```

```

3         f(i);
4     }
5     return v1;
6 }

```

不使用范围 for，使用 C++20 简写函数模板：

```

1 std::vector<int>& operator|(auto& v1, const auto& f) {
2     std::ranges::for_each(v1, f);
3     return v1;
4 }

```

各种其他答案的范式无非就是这些改来改去了，没必要再写。

1.2 解析

很明显我们需要重载管道运算符 `|`，根据我们的调用形式 `v | f2 | f`，这种链式的调用，以及根据给出运行结果，我们可以知道，重载函数应当返回 `v` 的引用，并且 `v` 会被修改。

`v | f2` 调用 `operator |`，`operator |` 中使用 `f2` 遍历了 `v` 中的每一个元素，然后返回 `v` 的引用，再 `|` `f`。

2 实现自定义字面量 `_f`

日期：2023/7/22 出题人：mq白

给出以下代码，在不修改已给出代码的前提下使它满足运行结果。6 为输入，决定 π 的小数点后的位数，可自行输入更大或更小数字。

```

1 int main(){
2     std::cout << "乐 :{} *\n"_f(5);
3     std::cout << "乐 :{0} {0} *\n"_f(5);
4     std::cout << "乐 :{:b} *\n"_f(0b01010101);
5     std::cout << "{:<10}"_f("卢瑟");
6     std::cout << '\n';
7     int n{};
8     std::cin >> n;

```

```

9      std::cout << "π: {:.{}f}\n"_f(std::numbers::pi_v<double>, n);
10 }

```

要求运行结果

```

乐 :5 *
乐 :5 5 *
乐 :1010101 *
卢瑟*****
6
π : 3.141593

```

- 难度： ★ ★ ☆ ☆ ☆

提示：C++11 用户定义字面量、C++20 format 库。

2.1 答案

```

1  constexpr auto operator""_f(const char* fmt, size_t) {
2      return [=]<typename... T>(T&&... Args) {
3          return std::vformat(fmt, std::make_format_args(std::forward<T>(Args)...));
4      };
5  }

```

2.2 解析

我们需要使用到 C++11 用户定义字面量，`""_f` 正是用户自定义字面量，但字面量运算符（用户定义字面量所调用的函数被称为字面量运算符）的形参列表有一些限制，我们需要的是 `(const char *, std::size)` 这样的形参列表，恰好这是允许的；字面量运算符的返回类型，我们需要自定义，这个类型需要在内部重载 `()` 运算符，以满足上述字面量像函数一样调用的要求。

-
- 3 实现 `print` 以及特化 `std::formatter`
 - 4 给定模板类修改，让其对每一个不同类型实例化有不同 ID
 - 5 实现 `scope_guard` 类型
 - 6 解释 `std::atomic` 初始化
 - 7 `throw new MyException`
 - 8 定义 `array` 推导指引
 - 9 名字查找的问题
 - 10 遍历任意聚合类数据成员
 - 11 `emplace_back()` 的问题
 - 12 实现 `make_vector()`
 - 13 关于 `return std::move(expr)`