C++ 17 可变参数模板包到运行时

迈克尔·乔达基斯

2018 年 1 月 8 日 警察

使用可变参数模板减少函数递归

介绍

在我之前的 C++ 17 文章中， 我不太喜欢**std::any**。但是，我想到了一个有趣的技巧，可以减少可变参数模板函数中所需的递归。

C++ 11 和 C++ 17

先说老办法。从 Cake Processor关于可变参数 printf 的好文章我们有这个：

C++ 缩小▲ 复制代码

```
void safe_printf(const char *s)
{
    while (*s) {
        if (*s == '%') {
            if (*(s + 1) == '%') {
                ++s;
            }
            else {
                throw std::runtime_error("invalid format string: missing arguments");
            }
        }
        std::cout << *s++;
    }
}

template<typename T, typename... Args>
void safe_printf(const char *s, T value, Args... args)
{
    while (*s) {
        if (*s == '%') {
            if (*(s + 1) == '%') {
                ++s;
            }
            else {
                std::cout << value;
```

```
                safe_printf(s + 1, args...); // call even when *s == 0 to detect extra
arguments
                return;
            }
        }
        std::cout << *s++;
    }
    throw std::logic_error("extra arguments provided to printf");
}
```

两个功能。可变参数模板一和最后的 const char* 一，当包中的所有参数都已扩展时最后调用。最重要的困难是必须在所有此类模板中出现的编译级递归。

但是，请记住，参数包使用逗号分隔符扩展参数：

复制代码

```
template <typename ... Args>
void foo(Args ... args)
{

}

foo(1,2,3,"hello"); // means that args is unpacked as such.
```

为什么不使用**initializer_list**将这些值存储在**vector 中**，以便可以在运行时使用 for 循环访问它们？

复制代码

```
template <typename ... Args>
void foo(Args ... args)
{
    std::vector<std::any> a = {args ...};
}
```

很酷。项目存储在**std::any 中**的事实意味着任何东西都可以传递给它，并且现在可以在运行时访问它。请注意，这是有效的，因为 std::any 本身不是模板；它将包含的对象存储为通用指针，并通过**typeinfo**测试它是否与传递给它的类型匹配。顺便说一句，我认为应该放宽类型测试，例如，如果您有一个包含 int 的 std::any，为什么不使用 any_cast<long>() 返回它？

现在一个函数中的新**printf**（好吧，为了效率，可以循环向量，但现在无关紧要）：

C++ 复制代码

```
template<typename ... many>
void safe_printf2(const char *s, many ... args)
{
    vector<any> a = {args ...};

    while (*s) {
        if (*s == '%') {
            if (*(s + 1) == '%') {
                ++s;
            }
            else {

                if (a.empty())
                    throw std::logic_error("Fewer arguments provided to printf");

                if (a[0].type() == typeid(string)) cout << any_cast<string>(a[0]);
                if (a[0].type() == typeid(int)) cout << any_cast<int>(a[0]);
                if (a[0].type() == typeid(double)) cout << any_cast<double>(a[0]);

                a.erase(a.begin());
```

```
                s++;
            }
        }
        std::cout << *s++;
    }
}
```

复制代码

```
// ----
int main()
{
 safe_printf2("Hello % how are you today? I have % eggs and your height is %","Jack"s, 32,5.7);
 // Hello Jack how are you today? I have 32 eggs and your height is 5.7
}
```

通常，使用 vector<any> 允许您将参数包扩展到运行时并使用 for 循环对其进行操作。

当然，初始化列表会复制所有参数。所以你可能想使用指针：

复制代码

```
vector<any> a = {&args...};

if (a[0].type() == typeid(string*)) cout << *any_cast<string*>(a[0]);
```

或者，在 std::ref 和 std::reference_wrapper 的帮助下的引用（我更喜欢指针。此外， std::reference_wrapper 需要更长的时间来编写、理解和使用内部指针，因为它不能使用任何东西否则。我可耻）：

复制代码

```
vector<any> a = {std::ref(args)...};

if (a[0].type() == typeid(std::reference_wrapper<string>))  cout <<
any_cast<std::reference_wrapper<string>>(a[0]).get();
```

历史

2018 年 1 月 6 日：首次发布

执照

分享

关于作者

迈克尔·乔达基斯
软件开发人员
希腊
手表该会员

我正在使用 C++、PHP、Java、Windows、iOS、Android 和 Web（HTML/Javascript/CSS）。

我拥有数字信号处理和人工智能博士学位，专攻专业音频和人工智能应用。

我的主页：https://www.turbo-play.com

评论和讨论

添加评论或问题 | 电子邮件提醒 | Search Comments

第一 页上一页 下一页

std::any 运行时
geoyar 9-Jan-18 1:55

回复：运行时的 std::any
Michael Chourdakis 9-Jan-18 5:46

回复：运行时的 std::any
geoyar 9-Jan-18 9:16

回复：运行时的 std::any
Michael Chourdakis 9-Jan-18 13:02

回复：运行时的 std::any
truthadjustr 13-Jul-18 3:06

回复：运行时的 std::any
Michael Chourdakis 14-Jul-18 5:49

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。