编程语言 函数式编程 Haskell C / C++ 修改

关注者 238 被浏览 19,438

他们也关注了该问题

Haskell等语言中的模式匹配在C++中如何实现? 修改

如果仅仅是字面量的匹配，用条件语句还是可以比较容易地实现。但如果是List，Tuple之类的匹配，如Haskell趣学指南中的例子 [图片] [图片] 这种应该如何在c++...显示全部

关注问题 写回答 邀请回答 添加评论 分享 邀请回答 举报 ...

查看全部 8 个回答

Felis sapiens
函数式编程、编程语言、编程 话题的优秀回答者

开源哥、Belleve、祖与占、草莓大福、刘雨培等 54 人赞同了该回答

Scott encoding 告诉我们，代数数据类型/模式匹配是可以用基本类型和一等函数实现的语法糖。C++ 有 lambda，有 std::function，可以用同样的思路来搞。

比较麻烦的情况是递归的代数数据类型。C++ 没有 isorecursive type，换句话说，不能直接 typedef 一个 std::function，然后让他的参数列表里提到他自己。不过可以用继承的方式来 workaround 一下。

以下是 C++ 中使用 Scott encoding 编码单链表的例子。所有地方 pass by value，不考虑空间效率和缺乏尾递归优化的问题：

```
#include <functional>
#include <iostream>

template <typename T, typename R>
struct SList
    : std::function<R(std::function<R()>, std::function<R(T, SList<T, R>)>)> {
  template <typename F>
  SList(F f)
      : std::function<R(std::function<R()>, std::function<R(T, SList<T, R>)>)>(
            f){};
};

template <typename T, typename R>
SList<T, R> Nil = SList<T, R>([](auto n, auto c) { return n(); });

template <typename T, typename R>
SList<T, R> Cons(T x, SList<T, R> xs) {
  return SList<T, R>([=](auto fx, auto fxs) { return fxs(x, xs); });
}

int main() {
  auto l = Nil<int, void>;
  for (;;) {
    int x;
    std::cin >> x;
    if (x > 0)
      l = Cons<int, void>(x, l);
    else
      break;
  }
  std::function<void()> fx = []() { std::cout << "End of list!" << std::endl; };
```

关于作者

Felis sapiens

函数式编程、编程语言、编程 话题的优秀回答者

电影旅行敲代码、Antokha Yuuki、暮无井见铃也关注了她

回答 624 文章 40 关注者 14,871

已关注 发私信

被收藏 27 次

编程语言与编译原理 5,338 人关注
酿酿酿酿酿泉 创建

l'Illumination de l'Ori 4 人关注
Yutong Zhang 创建

PL 2 人关注
lsdsjy 创建

C++ 1 人关注
sepiggy 创建

```
  std::function<void(int, SList<int, void>)> fxs = [&](auto x, auto xs) {
    std::cout << "List element " << x << std::endl;
    xs(fx, fxs);
  };
  l(fx, fxs);
  return 0;
}
```

以上代码使用 g++ 6.3.0 -std=c++14 编译通过。代数数据类型 SList 的两个模板参数 T 和 R 分别代表列表元素类型，以及在这个列表上进行计算的最终返回类型。定义了两个构造器 Nil 和 Cons，可以用于组装列表；要对 SList 进行模式匹配，只需要传入两个 std::function，分别匹配两种构造器，而构造器 Cons 的两个参数，将被绑定到后一个 std::function 的参数上。

基本原理就是这样。scope-safe，type-safe，除了特别丑和慢以外他是能 work 的。实现函数式语言的模式匹配的话，使用 Scott encoding 是可以考虑的。不过 C++ 还是想办法用 tagged union 是正经。

关于用 encoding 表示代数数据类型和模式匹配的原理，可以看我的另一个回答：zhihu.com/question/3993...

编辑于 2017-03-31

▲ 赞同 54 ▼ 3 条评论 分享 收藏 感谢 ··· 收起

更多回答

watashi
ゆっくりでいいさ

13 人赞同了该回答

std::variant 已经确定要进C++17了，至于原生的 lvariant 和 pattern matching 支持也有提案，不过估计得赶C++23了

Pattern Matching and Language Variants: open-std.org/jtc1/sc22/...

CppCon 2016: David Sankel “Variants: Past, Present, and Future": youtube.com/watch?...

```
// This lvariant implements a value representing the various commands
// available in a hypothetical shooter game.
lvariant command {
  std::size_t set_score; // Set the score to the specified value
  std::monostate fire_missile; // Fire a missile
  unsigned fire_laser; // Fire a laser with specified intensity
```

展开阅读全文

▲ 赞同 13 ▼ 5 条评论 分享 收藏 感谢 ···

Weixin Zhang

2 人赞同了该回答

可以用Visitor设计模式来模拟模式匹配，Java代码（C++类似）如下：

```
interface ListVisitor<E,O> {
  O Nil();
  O Cons(E x, List<E> xs);
```

FP 1 人关
sepiggy 创建

相关问题

在Haskell里，每个类型都可以构造出来一个此类型的表达式吗？ 4 个回答

OCaml，Haskell，standard ML，F#这几门语言之间有什么区别和联系？ 1 个回答

haskell中的类型类是相当于面向对象语言的接口吗？ 8 个回答

阻挡你学会 Haskell 最大的两个问题是什么? 38 个回答

Haskell 以后，就没有其他新语言默认采用惰性求值了，这是否说明（默认）惰性求值是个错误的决定？ 22 个回答

相关推荐

淼懂物理学：理解世界的极简指南
共 31 节课 ▶ 试听

语言专业海外就业：外派翻译
★★★★½ 30 人参与

数据科学导论：Python 语言实现
489 人读过 阅读

```
}
interface List<E> {
  <O> O accept(ListVisitor<E,O> v);
}
class Nil<E> implements List<E> {
  public <O> O accept(ListVisitor<E,O> v) { return v.Nil(); }
}
class Cons<E> implements List<E> {
  E x;
  List<E> xs;
```

展开阅读全文