切換導航 Farseerfc的小窩

- 文A
  - 繁體
  - 简体
  - English
  - 日本語
  - 文A繁體
  - 文A简体
  - 文AEnglish
  - 文A日本語
-
  - About
  - Links
  - About
  - Links
-
  - Import
  - Life
  - Tech

關於C++模板的類型轉換的討論

目錄

- 討論地址
- 原問題
- 我的解答
    - 首先看ff的情況。
    - 再來看f的情況。

這兩天在飲水思源的C板，關於C++模板的類型轉換的一個討論，後面是我的解答。

討論地址

http://bbs.sjtu.edu.cn/bbstcon,board,C,reid,1330078933,file,M.1330078933.A.html

原問題

今天在書上看到模板演繹的時候可以允許cast-down，於是我寫了個東西：

```
template <bool _Test, class _Type = void>
struct enable_if { };

template<class _Type>
struct enable_if<true, _Type> {
    typedef _Type type;
```

```
7  };
8
9  class A { };
10 class B : A { };
11
12 template <typename T>
13 struct traits { static int const value =
14
15 template <>
16 struct traits<A> { static int const val
17
18 template <typename T>
19 void f(T, typename enable_if<traits<T>::v
20
21 template <>
22 void f<A>(A, enable_if<traits<A>::value>:
23
24
25
26 template <typename T>
27 class BB {};
28
29 template <typename T>
30 class DD : public BB<T> {};
31
32 template <typename T> void ff(BB<T>) {}
33
34 int main(int argc, char * argv[])
35 {
36     A a; B b;
37     DD<long> dd;
38     //f(b);
39     ff(dd);
```

```
40  }
```

奇怪的是重載決議的時候， `f` 的情況下它就不讓我特化的 `f<A>` 進來。

但是在 `ff` 的情況下， `ff<BB<long>>` 卻進來了。

在VC10和GCC3.4下測試

我的解答

我們來設身處地地作爲編譯器，看一遍到底發生了什麼。

約定符號 `#` : `A#B` 是把 `B` 帶入 `A<T>` 的參數 `T` 之後實例化得到的結果。

首先看ff的情況。

```
1  DD<long> dd;
```

處理到這句的時候，編譯器看到了 `DD<long>` 的實例化，於是去實例化 `DD#long` ，繼而實例 化了 `BB#long` 。

```
1  ff(dd);
```

這句，首先計算重載函數集合。

第一步，需要從參數 `DD#long -> BB<T>` 推斷 `ff<T>` 的 `T` 。根據函數模板參數推斷規則：

```
:code:`class_template_name<T>` 類型的參數，可以用
```

於是編譯器推斷 `T` 爲 `long` 。這裏就算不是 `BB` 而是完全無關的 `CC` 都可以推斷成功，只要 `CC` 也 是一個 `CC<T>` 形式的模板。

第二步，模板特化匹配。因爲只有一個模板，所以匹配了最泛化的 `ff<T>` 。

第三步，模板實例化。

推斷了 `long -> T` 之後，編譯器實例化 `ff#long` 。

重載函數集合： `{ff#long}`

然後重載抉擇找到唯一的可匹配的實例 `ff#long` ，檢查實際參數 `DD#long` 可以隱式轉換到 形式參數 `BB#long` ，從而生成了這次函數調用。

再來看f的情況。

```
1 f(b);
```

計算候選重載函數集合。

第一步，對所有 `f` 模板推斷實參。根據函數模板參數推斷規則：

> 帶有 :code:`T` 類型的參數，可以用於推斷 :code:`T`

於是 `B -> T` 被推斷出來了。

第二步，模板特化匹配。

這裏 `B` 不是 `A` ，所以不能用 `f<A>` 特化，只能用 `f<T>` 模板。

第三步，模板實例化。

`B` 帶入 `f<T>` 實例化成 `f#B` 的過程中，實例化 `traits#B` 。

由於沒有針對 `B` 的特化，所以用 `traits<T>` 模板，`traits#B::value=false` ，進而 `enable_if#false` 沒有 `type` ，出錯。

唯一的模板匹配出錯，重載函數集合爲空，SFINAE 原則不能找到合適的匹配，於是報錯。