Search: Go Not logged

Reference <utility> forward register log in

C++
Information
Tutorials
Reference
Articles
Forum

Reference C library: Containers: Input/Output: Multi-threading: Other: <algorithm> <hitset> <chrono> <codecvt> <complex> <exception> <functional> <initializer_list> <iterator> dimits> <locale> <memory> <new> <numeric> <random> <ratio> <regex> <stdexcept> <string> <system_error> <tuple> <typeindex> <typeinfo> <type_traits> <utility> <valarray>

<utility> classes: pair functions: declval forward make_pair move move_if_noexcept swap types: piecewise construct t constants: piecewise_construct namespaces: rel_ops



function template
std::forward

<utility>

```
//value (1) template <class T> T&& forward (typename remove_reference<T>::type& arg) noexcept;
//value (2) template <class T> T&& forward (typename remove_reference<T>::type&& arg) noexcept;
```

Forward argument

Returns an rvalue reference to arg if arg is not an Ivalue reference.

If arg is an Ivalue reference, the function returns arg without modifying its type.

This is a helper function to allow *perfect forwarding* of arguments taken as rvalue references to *deduced types*, preserving any potential *move semantics* involved.

The need for this function stems from the fact that all named values (such as function parameters) always evaluate as Ivalues (even those declared as *rvalue references*), and this poses difficulties in preserving potential *move semantics* on template functions that forward arguments to other functions.

Both signatures return the same as:

```
static_cast<decltype(arg)&&>(arg)
```

By providing two signatures and using remove_reference on T, any instantiation is forced to explicitly specify the type of T (any implicitly deduced T would have no match).

Parameters

arg

An object.

Return value

If arg is an $\ensuremath{\textit{Ivalue reference}}$, the function returns $\ensuremath{\textit{arg}}$ with its type unchanged.

Otherwise, the function returns an rvalue reference (T&&) that refers to arg that can be used to pass an rvalue.

Example

```
1 // forward example
 2 #include <utility>
                               // std::forward
 3 #include <iostream>
                               // std::cout
 5 // function with lvalue and rvalue reference overloads:
 6 void overloaded (const int& x) {std::cout << "[lvalue]";}
7 void overloaded (int&& x) {std::cout << "[rvalue]";}</pre>
 8
 ^{9} // function template taking rvalue reference to deduced type:
10 template <class T> void fn (T&& x) {
                                             // always an lvalue
11
     overloaded (x);
12
     overloaded (std::forward<T>(x)); // rvalue if argument is rvalue
13 |}
                                                                                  Edit
14
15 int main () {
16
                                                                                  Run
     int a;
17
     std::cout << "calling fn with lvalue: ";</pre>
18
19
     fn (a);
     std::cout << '\n';
20
21
22
     std::cout << "calling fn with rvalue: ";</pre>
23
     fn (0);
24
     std::cout << '\n';
25
26
     return 0:
27 }
```

Output:

```
calling fn with lvalue: [lvalue][lvalue] calling fn with rvalue: [lvalue][rvalue]
```

Data races

none

Exceptions

No-throw guarantee: this function never throws exceptions.

See also

move Move as rvalue (function template)

Home page | Privacy policy © cplusplus.com, 2000-2020 - All rights reserved - v3.2 Spotted an error? contact us