function template

std::copy

<algorithm>

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```

Copy range of elements

Copies the elements in the range `[first,last)` into the range beginning at *result*.

The function returns an iterator to the end of the destination range (which points to the element following the last element copied).

The ranges shall not overlap in such a way that *result* points to an element in the range *[first,last)*. For such cases, see `copy_backward`.

The behavior of this function template is equivalent to:

```
template<class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result)
{
  while (first!=last) {
    *result = *first;
    ++result; ++first;
  }
  return result;
}
```

Parameters

`first, last`
: Input iterators to the initial and final positions in a sequence to be copied. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.

`result`
: Output iterator to the initial position in the destination sequence.
This shall not point to any element in the range `[first,last)`.

Return value

An iterator to the end of the destination range where elements have been copied.

Example

```cpp
// copy algorithm example
#include <iostream>     // std::cout
#include <algorithm>    // std::copy
#include <vector>       // std::vector

int main () {
  int myints[]={10,20,30,40,50,60,70};
  std::vector<int> myvector (7);

  std::copy ( myints, myints+7, myvector.begin() );

  std::cout << "myvector contains:";
  for (std::vector<int>::iterator it = myvector.begin(); it!=myvector.end(); ++it)
    std::cout << ' ' << *it;

  std::cout << '\n';

  return 0;
}
```

Output:

```
myvector contains: 10 20 30 40 50 60 70
```

Complexity

Linear in the distance between *first* and *last*: Performs an assignment operation for each element in the range.

Data races

The objects in the range `[first,last)` are accessed (each object is accessed exactly once).
The objects in the range between *result* and the returned value are modified (each object is modified exactly once).

Exceptions

Throws if either an element assignment or an operation on iterators throws.
Note that invalid arguments cause *undefined behavior*.

See also

copy_backward	Copy range of elements backward (function template)
fill	Fill range with value (function template)

replace	Replace value in range (function template)

Spotted an error? contact us