
11.3. Auxiliary Functions

The rest of this chapter discusses the algorithms in detail and includes at least one example of each algorithm. To simplify the examples, I use some auxiliary functions so that you can concentrate on the essence of the examples:

Click here to view code image

```
// algo/algostuff.hpp

#ifndef ALGOSTUFF_HPP
#define ALGOSTUFF_HPP

#include <array>
#include <vector>
#include <deque>
#include <list>
#include <forward_list>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <algorithm>
#include <iterator>
#include <functional>
#include <numeric>
#include <iostream>
#include <string>

// INSERT_ELEMENTS (collection, first, last)
// - fill values from first to last into the collection
// - NOTE: NO half-open range
template <typename T>
inline void INSERT_ELEMENTS (T& coll, int first, int last)
{
    for (int i=first; i<=last; ++i) {
        coll.insert(coll.end(),i);
    }
}

// PRINT_ELEMENTS()
// - prints optional string optcstr followed by
// - all elements of the collection coll
// - separated by spaces
template <typename T>
inline void PRINT_ELEMENTS (const T& coll,
                            const std::string& optcstr="")
{
    std::cout << optcstr;
    for (auto elem : coll) {
        std::cout << elem << ' ';
    }
    std::cout << std::endl;
}

// PRINT_MAPPED_ELEMENTS()
// - prints optional string optcstr followed by
// - all elements of the key/value collection coll
// - separated by spaces
template <typename T>
inline void PRINT_MAPPED_ELEMENTS (const T& coll,
                                   const std::string& optcstr="")
{
    std::cout << optcstr;
    for (auto elem : coll) {
        std::cout << '[' << elem.first
                  << ',' << elem.second << "] ";
    }
    std::cout << std::endl;
}

#endif /*ALGOSTUFF_HPP*/
```

First, `algostuff.hpp` includes all header files that may be necessary to implement the examples, so the program doesn't have to do it. Second, it defines three auxiliary functions:[2]

[2] Since C++11, `PRINT_MAPPED_ELEMENTS()` could also be defined as partial specialization of `PRINT_ELEMENTS()`. However, to avoid requiring too many new language features, both functions are defined separately.

1. `INSERT_ELEMENTS()` inserts elements into the container that is passed as the first argument. These elements get the values from the value passed as the second argument up to the value passed as the third argument. Both argument values are included, so this is *not* a half-open range.
2. `PRINT_ELEMENTS()` prints all elements of the container that is passed as the first argument, separated by spaces. You can pass a second argument optionally for a string that is used as a prefix in front of the elements (see Section 6.6, page 216).
3. `PRINT_MAPPED_ELEMENTS()` is the same for containers with a key/value pair: map, multimap, unordered map, and unordered multimap.