

689: Generic Programming — Assignment 2

September 24, 2004

1. To efficiently do things like sorting and searching with random access iterators often requires an algorithm to find the middle of an iterator range. If `begin` and `end` model `RANDOM ACCESS ITERATOR`, then the correct way to find the middle of the range is to calculate `begin + (end - begin) / 2`. Why can't we calculate the middle of the range more simply with `(begin + end) / 2`?
2. The following code example illustrates a common error and will fail to compile.

```
template <typename Iterator>
Iterator::value_type
look_at_first(Iterator begin, Iterator end)
{
    Iterator::value_type v;
    v = *begin;
    return v;
}
```

- (a) What is wrong with this example?
 - (b) Without changing any of the existing code (i.e., by adding code only), how can you fix this?
 - (c) What are the requirements of the `Iterator` in this example?
 - (d) How would you make it more generic?
3. Consider the following example.

```
template <typename X>
void foo(X x) {
    typedef typename x_traits<X>::cost_type cost_type;
    typedef typename x_traits<X>::color_type color_type;

    cost_type y = bar(x);
    color_type z = baz(x);

    std::cout << y << " " << z << std::endl;
}
```

- (a) What are the requirements on template parameter `X`? (What are the valid expressions and associated types?)
- (b) Define an `x_traits` class that can work for user-defined as well as built-in types. The traits class should have a reasonable default (non-specialized) behavior – which is what will be used for user-defined types.
- (c) What do you need to do (for `x_traits`) to be able to call `foo` with an object of a built-in type. Demonstrate using the type `int`.

4. Write a program to sort the lines of text in a specified file. The program should take as input a filename. It should open the file, read it in, sort the lines, and print the result. The program should behave gracefully if the file is not found or cannot be read.

If you choose the right kinds of SL data types and the right SL algorithms, you should not need to use more than one (or a few) lines of code for each of these operations. Browse through the algorithms and function objects in the SL to see which ones might be helpful.

5. Write a function `print_category` that can be used to print the category of an iterator. The function should be prototyped as follows:

```
template <class Iterator>
void print_category(Iterator x);
```

If `x` is an `INPUTITERATOR`, this should print “Input Iterator”, for `FORWARDITERATOR`, it should print “Forward Iterator”, and so on.

- (a) What is printed in response to

```
char a[10];
print_category(a + 10);
```

- (b) What is printed in response to

```
std::list<int> a;
print_category(a.begin());
```

- (c) Create a test program that will cause each type of iterator to be printed.

6. Write a generic algorithm to find the value and the location of the maximum element in a container. The prototype for the algorithm should be

```
template <typename Iterator>
std::pair<Iterator, typename Iterator::value_type>
find_max(Iterator begin, Iterator end);
```

The return value of the function should be pair, containing the max value as well as an iterator pointing to the maximum element in the container. Your implementation should re-use other STL algorithms if possible.

- (a) Test your implementation with at least three different container types, including a container of complex numbers.
- (b) Here, we didn’t indicate any particular requirements on `Iterator`. In your implementation, what are the requirements on the `Iterator` type?
- (c) Are there any other concepts (perhaps implied)? If so, what are their requirements?
- (d) Do these requirements correspond to any particular iterator concept in the standard library?
- (e) Can you make this algorithm more generic? (Hint: Does the prototype make sense if you pass in an array?)
- (f) (Extra credit.) Try compiling a function with a prototype as above and pass in an array for the iterator type. What happens? Explain.

7. Now generalize your `find_max` algorithm to be able to use a function object to define comparison.

- (a) Test your implementation with a container of complex numbers.
- (b) What are the requirements on your function object?
- (c) Do these requirements correspond to any particular concepts in the standard library?

What to turn in

Follow the directions from assignment 1.