Chapter 21 Review

Ulysses Carlos

<2020-05-28 Thu>

1 Questions

1.1 What are examples of useful STL algorithms?

Some examples include sort(), find _if, accumulate, and copy().

1.2 What does find() do? Give at least five examples.

find() searches a specific container (with a begin() and an end()) for the first occurrence of an item. Some examples include:

- Using find() to search for a string in an vector
- Using find() to search for the first occurrence of "Apple" in a list.
- Using find() to search for the first occurrence of an number to remove it.
- Using find() to find a string that is then replaced by another string
- Using find() to insert "Steak" after "Ramen" in a linked list.

1.3 What does count _if() do?

count_if() returns the number of occurrences of a value **x** that is true according to the predicate **p** that was passed as an argument.

1.4 What does sort(b,e) use as its sorting criterion?

sort(b,e) by default compares whether an item i in the sequence is less than another item in the sequence. A predicate p can be passed as an argument that can change the sorting criterion.

1.5 How does an STL algorithm take a container as an input argument?

An STL algorithm takes a container as an input arugment by requiring that two iterators (usually to the beginning and end of the container) as parameters.

1.6 How does an STL algorithm take a container as an output argument?

An STL algorithm takes a container as an output by requiring that the iterator points to some location in the output container (Usually begin()).

1.7 How does an STL algorithm usually indicate "not found" or "failure"?

This is done by returning the end of the sequence (For a sequence s, then s.end() is returned).

1.8 What is a function object?

An object of a class that through operator overloading can simulate a function call.

1.9 In which ways does a function object differ from a function?

A function object can simulate a function all by defining the () operator. If the contents in the operator() are short or simple enough, then the time it to call the function object is much shorter than calling the function itself(Since the stack has to be set up to make a function call).

1.10 What is a predicate?

A predicate is a statement or list of statements that can be evalauted as true or false.

1.11 What does accumulate() do?

accumulate() allows an specified initial value to be added to over a range [a, b) which are set as parameters. For example,

```
vector<int> v { //Whatever };
int result = accumulate(v.begin(), v.end(), int{0});
```

accumulate() traverses the entire container and adds each member to the specified value (in this case 0).

1.12 What does inner_product() do?

inner_product() returns a value obtaining by using a binary operation (Such as multiplication or division) on two different containers.

1.13 What is an associative container? Give at least three examples.

An Associative container is a container where entries are found using "keys". Examples (in C++) include

- Maps (Essentialy Red-black trees)
- Unordered Maps (Hash tables)
- Sets (Like Maps but with only the first field)

1.14 Is list an associative container? Why not?

List is not an associative container since the user does not specify what key is used to sort a specfic node in the list. Instead, by inserting a value, the value will be placed somewhere in the list. Also, List iterators do not allow random-access iteration.

1.15 What is the basic ordering property of binary tree?

The basic ordering property of a binary tree is that values are ordered based whether a value is less than or greater to a root value. In an non-balanced binary search tree, the first value is set as the root and the others are placed depending on whether the value is less than or greater than the root.

1.16 What (roughly) does it mean for a tree to be balanced?

If a tree is said to be balanced, it means that the number of branches on both the left and right are the same so that its distance from the root is the same. It also means the level of branches(1, 2, 4, 8) on both sides are the same.

1.17 How much space per element does a map take up?

A map contains a key and a value, alongside a left and right node, so assuming that we are dealing with a map<int, int>, we can say that the key and value are 8 bytes total, and 2 iterators to the left and right node on x86-64 come up to 16 bytes, so in total, we can say that it takes up 24 bytes per element (Assuming no other members).

1.18 How much space per element does a vector take up?

In comparison, since vectors are continuous, each element takes up the size of the type. For example, each element in a vector<int> should take up 4 bytes.

1.19 Why would anyone use an unordered _map when an (ordered) map is available?

An unordered_map should be used if fast insertion/deletion/searching is critical for a program. Most operations on the unordered_map have a time complexity of O(1) in the average case.

1.20 How does a set differ from a map ?

A set only contains a first value (The key) instead of both a key and value in a map.

1.21 How does a multimap differ from a map ?

A multimap differs from a map in that a key can occur multiple times in the map and each key is considered unique.

1.22 Why use a copy() algorithm when we could "just write a simple loop"?

Using the copy() algorithm allows ease of access for all containers and prevents having to implement loops in order to copy from one container to another.

1.23 What is a binary search?

Binary search is a search preformed on a sorted container that searches for a specific item by first checking if the item is located in the middle of the range. If the value is found there, then a iterator is returned. If the item is less than the middle value, the left half is searched through, while if the item is greater than the middle value, then the search continues in the right half of the container.