Chapter 19 Review

Ulysses Carlos

February 8, 2020

1 Questions

1. Why would we want to change the size of a vector ?

   (a) Changing the size of a vector allows us to have a dynamic block of data that does not rely on having to a set size which prevents additional input.

2. Why would we want to have different element types for different vector s?

   (a) So that we are able to work with different data types with the advantages of using a vector instead of a raw array of the same data type.

3. Why don't we just always define a vector with a large enough size for all eventualities?

   (a) Because we may just allocate too much space at a given time, which may not be possible.

4. How much spare space do we allocate for a new vector ?

   (a) If the vector constructed is called with a space parameter, then that amount is allocated for the vector. If the default constructor is called, then only no space is allocated for the vector.

5. When must we copy vector elements to a new location?

   (a) When we have run out of space for the vector and have to increase its size.

6. Which vector operations can change the size of a vector after construction?

   (a) reallocate() and resize()

7. What is the value of a vector after a copy?

   (a) I'm honestly not sure; I don't really know what is meant by that. I mean, if you use the copy assignment **operator (vector& operator=(const vector &))**, then it return a this pointer (to this object).

8. Which two operations define copy for vector ?

   (a) The copy constructor and assignment operator.

9. What is the default meaning of copy for class objects?

   (a) A copy of a class object is a deep copy (Both objects are distinct from each other)

10. What is a template?

    - A template is a way to specify a class into dealing with a specified element type. I know that's an awful definition, but it allows us to create version of a class that can work with a specified type.
    - Stroustrup defines it as a mechanism that allows a programmer to use types as class or function.
    - The specified type is then generated by the compiler.

11. What are the two most useful types of template arguments?

    (a) Element<E> which can be a container, and Container<C> which can hold Elements and can be accessed as a [begin() , end()) sequence.

12. What is generic programming?

    (a) Stroustrup defines it as a way of writing code that works with a variety of types presented as arguments, as long as those argument types meet specific syntactic and semantic requirements.

13. How does generic programming differ from object-oriented programming?

    - Generic programming is supported by templates which rely on compile-time resolution, while Object-oriented programming relies on class hierarchies and virtual functions which are relying on run-time resolution.
    - Compile-time resolutions are solved at compile time, before the program is executed. Run-time resolution occurs at run-time when the derived class/class functions are called/constructed.

14. How does array differ from vector ?

    (a) array<T> is an immutable data structure of an type T. It contains much of the safety mechanisms of vector (known size, destructor, etc)

15. How does array differ from the built-in array?

    (a) As said earlier, array<T> keeps track of its size and its destructors are called when it leaves the scope (or some error occurs). Raw arrays have none of those safety measures.

16. How does resize() differ from reserve() ?

    (a) reverse() handles the actual reallocation of memory for the vector, while resize() determines to initialize the space between sz (The current size of the vector) and size (the capacity of the vector).

17. What is a resource? Define and give examples.

(a) A resource is something that is acquired and then released back to the system or reclaimed by the operating system. An example of a resource is a file(controlled by fstreams), memory allocated from the free store, etc.

18. What is a resource leak?

    (a) A resource leak is when a resource is allocated/acquired and is not returned back to the system. Normally, operating systems return the memory used by a program, but this would be disastrous for a embedded system.

19. What is RAII? What problem does it address?

    (a) RAII is **Resource Acquisition is Initialization**, which is defined as acquiring resources in the constructor for some object that manages it, and release it again in the matching destructor. This is address the issue of resource leaks when dealing with exceptions and deallocation. Messy code is often written to handle memory deallocation in various situations, and RAII is used in lieu of this.

20. What is **unique_ptr** good for?

    (a) A **unique_ptr** is a wrapper around a raw pointer which deallocates memory assigned to a pointer when the **unique_ptr** leaves the scope or when a exception/error occurs.