LAB 06 - Data Structure Basics

Hayden Whiteford
104001272
COS30031

Wednesday 23rd August

Release Notes:

Answers to each Question

Q.1 [line 102] There are inside array_demo_1 - answer them there.
Q.2 [line 105] In array_demo_2, explain what a4(a1) does
It passes in the a1 array to the a4 constructor, which initialises the a4 array with a copy of the elements in the a1 array.
Q.3 [line 108] No questions for array_demo_3, it's just a demo of Struct/Class use with array.
Q.4 [line 111] How do we (what methods) add and remove items to a stack?
Using the push() and pop() functions.
Q.5 [line 112] A stack has no no [] or at() method - why?
With a stack, only the item at the front of the stack can be accessed at a given time
Q.6 [line 115] What is the difference between a stack.pop() and a queue.pop() ?
A stack pops the last item added, while a queue pops the first item added
Q.7 [line 118] Can we access a list value using and int index? Explain.
No, we can't as an std::list is internally a doubly linked list, which we would need to iterate through to get to a certain element.
Q.8 [line 119] Is there a reason to use a list instead of a vector?
Lists work great in scenarios where there a constant insertions and deletions. Lists provide stable iterators that will still remain valid even when elements get deleted. Certain data structures that need to be set up with doubly linked lists can benefit from using lists, as internally it is a doubly linked list.
Q.9 [line 122] Was max_size and size the same? (Can they be different?)
No. Size was 3, and Max_size was 4611686018427387903. This is to do with the fact that vector sizes aren't fixed. Size indicated the actual number of elements in the vector, while Max size showed the theoretical amount of elements that could be held based on system limitation.
Q.10 [line 123] Which ParticleClass constructor was called?
ParticleClass(int x, int y), the constructor that takes integers as parameters was called
Q.11 [line 124] Were the ParticleClass instances deleted? If so, how?
When the vector went out of scope ie. at the closing bracket of the if statement, the destructor was called for the vector, and this subsequently called the destructors of each particleClass element inside the vector.
Q.12 [line 125] Was the vector instance deleted? If so, how do you know this?
Again, its destructor was called as soon as it went out of scope.
Q.13 [line 126] Your IDE might suggest to use emplace_back instead of push_back. What does this mean?

Push back created a copy of the constructed element to move into the vector. Here it might be more efficient to construct the element directly into the vector so it doesn't need a temporary copy to be made and moved. Emplace_back does this.

Q.1.1 [line 164] What do the < and > mean or indicate?

1.1.1: the arrows indicate the usage of a template.

Q.1.2 [line 165] Why don't we need to write std:array here? (Is this good?)

1.1.2: the template line is an instantiation of the std::array class

Q.1.3 [line 166] Explain what the int and 3 indicate in this case?

1.1.3: that the array will contain objects of type Int and will be 3 indexes big.

Q.1.4 [line 204] In the code above, what is the type of itr2?

1.1.4: it should be an iterator pointing to the first item in the array

Q.1.5 [line 211] In the code above, what is the type of v?

1.1.5: auto, meaning that the compiler will determine what type suits best, in this case type int& as it is a reference to an integer

Q.1.6 [line 212] In the code above, what does the & mean in (auto &v : a1)

1.1.6: the use of the & symbol means that we are using a reference for each number in the array to get and modify the original values, instead of making a copy.

Q.1.7 [line 220] Try this. Why does a1[3] work but at(3) does not?

1.1.7: .at(3) throws an exception because part of the .at() function is out of bounds checking. It recognised unlike using [], that the index was outside the size of the array.

Q.1.8 [line 233] auto is awesome. What is the actual type of v that it works out for us?

1.1.8: the actual type should be an iterator, to be precise std::array<int, 3>::iterator

Q.1.9 [line 240] auto is still awesome. What is the actual type of v here?

1.9: here v is a reference denoted by the & symbol. Its referencing integers here so the precise type should be int&.

Q.1.10 [line 250] How would you do a forward (not reverse) sort?

1.10: the iterators that you pass into the sort() function would be a1.begin() and a1.end() to make it go forward.