iClicker

The preferred method of reducing the need for the scope resolution operator "::" is:

A. using namespace std;
B. "using std::cin;" for each item that needs resolving
C. It doesn't matter
D. You should not use the scope resolution operator

Which of the following is correct?

A. int& intRefArray[20];
B. int* intPtrArray[20];
C. Both are correct
D. Neither is correct

C++ STL - Standard Template Library

Sequence Containers:

- Array
- Vector
- Deque (Double ended queue)
- Forward List
- List

Associative Containers:

- Set
- Multiset
- Map
- Multimap

Container “Adapters”:

- Stack
- Queue
- Priority queue

Sequence Containers

Array

Sequence

Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence.

Contiguous storage

The elements are stored in contiguous memory locations, allowing constant time random access to elements. Pointers to an element can be offset to access other elements.

Fixed-size aggregate

The container uses implicit constructors and destructors to allocate the required space statically. Its size is compile-time constant. No memory or time overhead.

Vector

Sequence

Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence.

Dynamic array

Allows direct access to any element in the sequence, even through pointer arithmetics, and provides relatively fast addition/removal of elements at the end of the sequence.

Allocator-aware

The container uses an allocator object to dynamically handle its storage needs.

Deque (double ended queue)

Sequence

Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence.

Dynamic array

Generally implemented as a dynamic array, it allows direct access to any element in the sequence and provides relatively fast addition/removal of elements at the beginning or the end of the sequence.

Allocator-aware

The container uses an allocator object to dynamically handle its storage needs.

- Provides functionality similar to vectors, but with efficient insertion and deletion of elements at the beginning and the end of the sequence.
- Unlike vectors, deques are not guaranteed to store all its elements in contiguous storage locations: accessing elements in a deque by offsetting a pointer to another element causes *undefined behavior*.

List

Sequence

Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence.

Doubly-linked list

Each element keeps information on how to locate the next and the previous elements, allowing constant time insert and erase operations before or after a specific element (even of entire ranges), but no direct random access.

Allocator-aware

The container uses an allocator object to dynamically handle its storage needs.

(cplusplus.com)

***Forward List**

A list where each element only has a reference to the next element in the list.

Want to understand pointers better? Think of pointers as the indices of an array. In fact, you can interchange the syntax of the two.

Code Example: STL Vector, cin/cout

iClicker

Why did we use a vector instead of an array to capture input from cin?

A. Vector is faster
B. Vector has simpler notation
C. Vector is dynamically allocated
D. There was no advantage

What was one advantage of using the array that was declared as an array over the pointer that was treated like an array?

A. Array was faster
B. Array was easier to read
C. Array allowed use of “auto” in output loop
D. There was no advantage

Array, List, Deque, and Vector all:

A. Store their data in the same way
B. Are containers where ordering of items is a feature
C. Dynamically allocate memory at runtime
D. Have elements that keep track of where to find the next element

The pointers “north”, “south”, “east” and “west” in the geoTile struct from the code example provide which type of functionality?

A. Array
B. List
C. Forward list
D. Vector

Associative Containers

Set

- Sets are containers that store unique elements following a specific order.
- In a set, the value of an element also identifies it (the value is itself the *key*, of type T), and each value must be unique. The value of the elements in a set cannot be modified once in the container (the elements are always const), but they can be inserted or removed from the container.
- Internally, the elements in a set are always sorted following a specific *strict weak ordering* criterion indicated by its internal comparison object (of type Compare).

https://thispointer.com/stdset-tutorial-part-1-set-usage-details-with-default-sorting-criteria/

Multiset

Multiple elements in the container can have equivalent *keys*.

Map

- User accesses a value by using a “key” to access the value
- Internally, the elements in a map are always sorted by its *key* following a specific *strict weak ordering* criterion indicated by its internal comparison object (of type Compare).

Usage example: counting the occurences of all words in a book.

```
map<string, int> wordCount;
```

Key: string
Value: int

This is a concept which you are familiar with from JSON key/value pairs

Multimap

Associative

Elements in associative containers are referenced by their *key* and not by their absolute position in the container.

Ordered

The elements in the container follow a strict order at all times. All inserted elements are given a position in this order.

Map

Each element associates a *key* to a *mapped value*: Keys are meant to identify the elements whose main content is the *mapped value*.

Multiple equivalent keys

Multiple elements in the container can have equivalent *keys*.

Allocator-aware

The container uses an allocator object to dynamically handle its storage needs.

Code Example: STL Vector, cin/cout

- Improved user interface
- Demonstration of a practical difference between array and vector
- Relationship between array and pointer (Multiple examples)
- “Auto”
- Two ways of accessing an element of vector

iClicker

The use of “#include” to insert or delete pieces of code is:

A. Industry standard
B. Academia standard
C. The preferred method
D. For demonstration purposes only

A Map:

A. Is an associative container
B. Uses key:value pairs
C. Can access a mapped value directly by corresponding key
D. All of the above

A set:

A. Like a map but the “key” and “value” are the same
B. Is ordered so that you can index items directly
C. Requires all items be inserted in order
D. Only allows primitive data types

Associative containers:

A. Are used for ordered data
B. Allow retrieval of values by their associated “key”
C. Are only used to get values by string keys
D. They are best for inserting items at desired locations in the collection

Container “Adapters”

Stack

LIFO - last in, first out

- are implemented as *container adaptors*, which are classes that use an encapsulated object of a specific container class as its *underlying container*, **providing a specific set of member functions to access its elements**.
- Elements are *pushed* and *popped* from the *"back"* of the specific container, which is known as the *top* of the stack.

Queue

FIFO - first in, first out

- queues are implemented as *containers adaptors*, which are classes that use an encapsulated object of a specific container class as its *underlying container*, providing a specific set of member functions to access its elements.
- Elements are
    - *pushed* into the *"back"* of the specific container and
    - *popped* from its *"front"*.

Priority queue

Priority queues are a type of container adaptors, specifically designed such that its **first element is always the greatest of the elements it contains**, according to some *strict weak ordering* criterion.