CPP Module 08

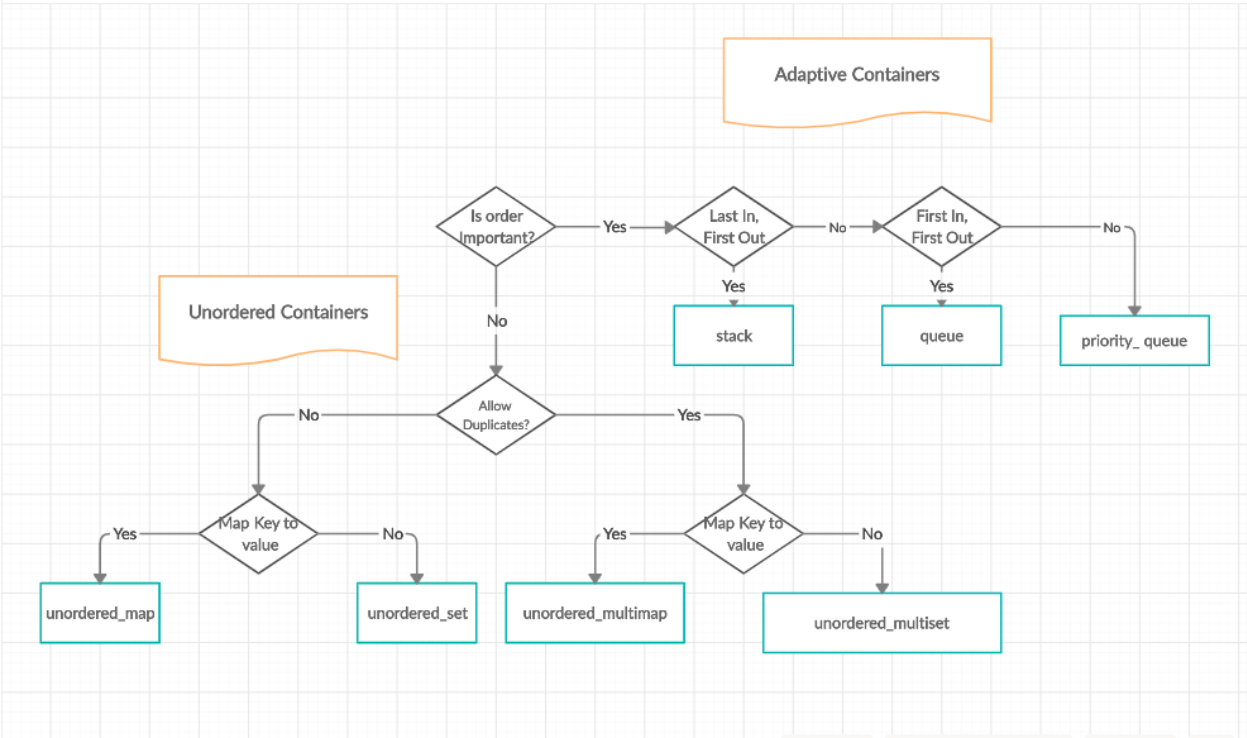
C++ dilinde, bir dizi veya veri yapısındaki öğelerin üzerinde dolaşmak için iterator adı verilen bir yapı kullanılır. Iterator, veri yapısındaki öğelerin başlangıç noktasını işaret eder ve bu öğeler arasında dolaşarak ilerleyebilir.

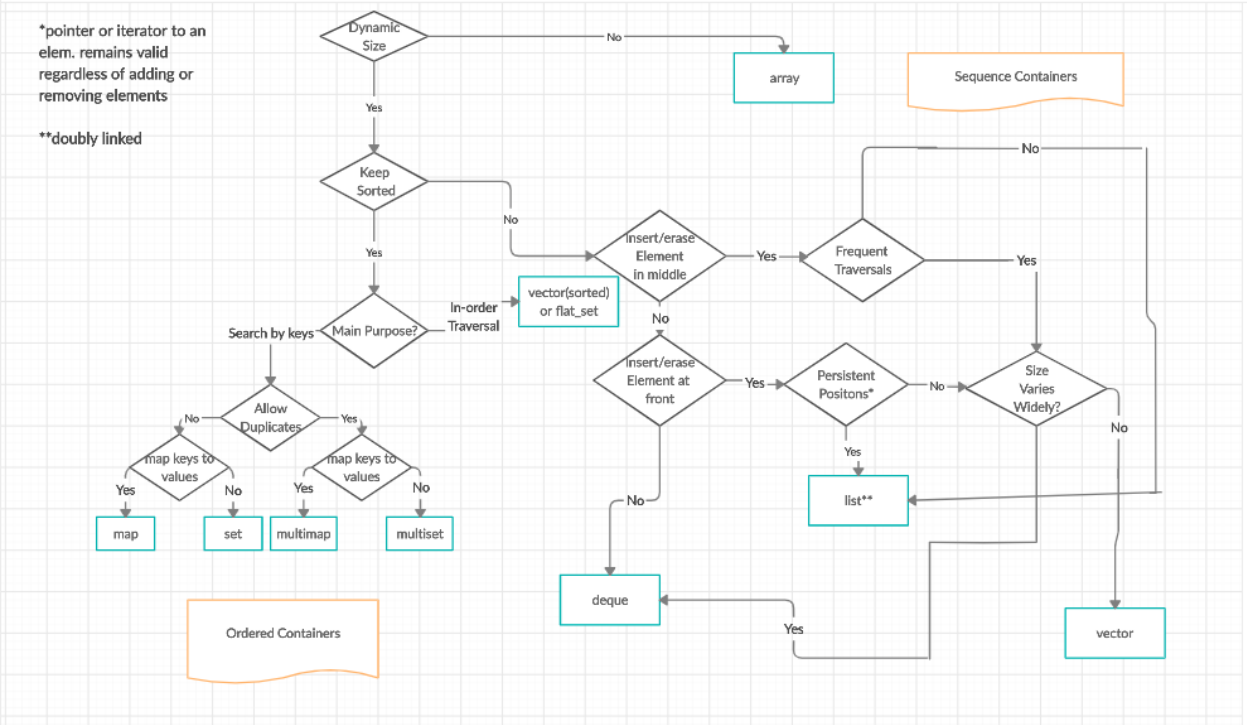
Örneğin, bir `vector` nesnesi oluşturabiliriz ve bu nesneyi dolaşmak için bir iterator kullanabiliriz. Aşağıdaki örnek, bir `vector` oluşturur ve tüm öğeleri yazdırmak için bir iterator kullanır:

```
#include <iostream>
#include <vector>

int main()

{
// Bir vector oluştur
std::vector<int> v = {1, 2, 3, 4, 5};

// Iterator ile vector'ün öğelerini yazdır
for (auto it = v.begin(); it != v.end(); ++it) {
std::cout << *it << " ";
}

return 0;
```

```
}
```

vector is growing in size each time when adding something. It must be contiguous. When space is not enough to grow it changes the entire vector to a larger space. when we add value to the vector, no pointers to the vector remain valid.





https://cplusplus.com/reference/stl/

```
#include <stack>

template<typename T>
class MutantStack : public std::stack<T> {
public:
    // Define iterator type
    typedef typename std::stack<T>::container_type::iterator iterator;

    // Define const iterator type
    typedef typename std::stack<T>::container_type::const_iterator const_iterator;

    // Add begin and end member functions that return iterators
    iterator begin() { return std::stack<T>::c.begin(); }
    iterator end() { return std::stack<T>::c.end(); }

    const_iterator begin() const { return std::stack<T>::c.begin(); }
```

```
    const_iterator end() const { return std::stack<T>::c.end(); }
};
```

In the implementation of `MutantStack` class, we are using `c.end()` in the `begin()` and `end()` member functions to obtain the iterator that points to the beginning and end of the underlying container of the `std::stack` class.

`c` is a member variable of `std::stack` that represents the underlying container. `std::stack` is implemented using a container adapter pattern, where a container type is specified as a template argument and `std::stack` provides a specific interface for that container type. In the case of `std::stack`, the default underlying container type is `std::deque`.

So, to get the iterator that points to the beginning and end of the underlying container, we need to use `c.begin()` and `c.end()`, respectively, where `c` is the member variable that represents the underlying container.

In summary, we are using `c.end()` in the `MutantStack` class to obtain the iterator that points to the end of the underlying container of the `std::stack` class, so that we can provide iterator functionality to the `MutantStack` class.