

C++ ITERATORS

Problem Solving with Computers-II

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



C++STL

- The C++ Standard Template Library is a very handy set of three built-in components:
 - Containers: Data structures
 - Iterators: Standard way to move through elements of containers
 - Algorithms: These are what we ultimately use to solve problems

C++ Iterators behave like pointers

- Let's consider how we generally use pointers to parse an array

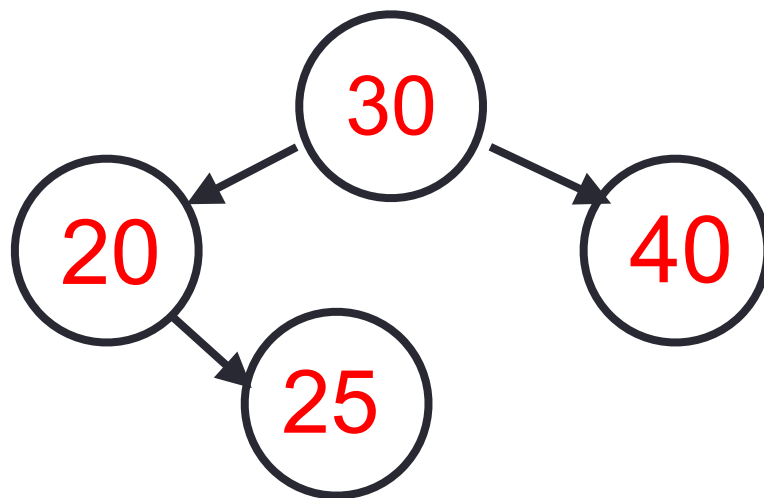
10	20	25	30	46	50	55	60
----	----	----	----	----	----	----	----

```
void printElements(int arr[], int size) {  
    int* p= arr;  
    for(int i=0; i<size; i++) {  
        std::cout << *p << std::endl;  
        ++p;  
    }  
}
```

-

- We would like our print “algorithm” to also work with other data structures
- E,g Linked list or BST

Can a similar pattern work with a BST? Why or Why not?

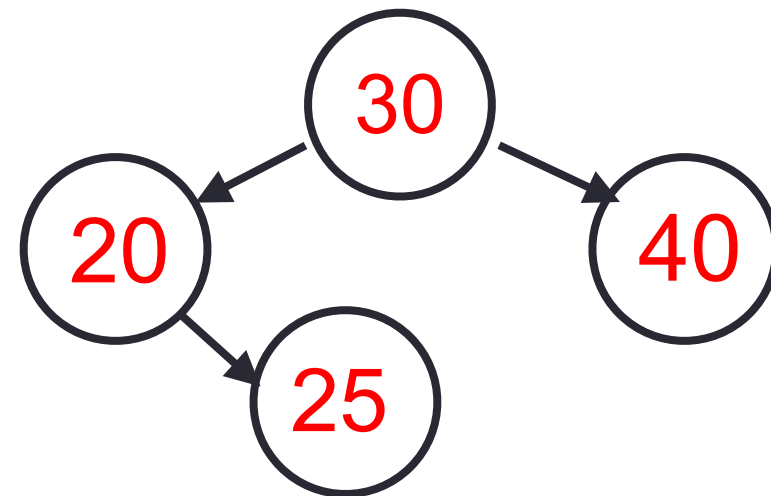


```
void printElements(set<int>& s) {  
    _____ //How should we define p?  
    for(int i=0; i<s.size(); i++) {  
        std::cout << *p << std::endl;  
        ++p;  
    }  
}
```

Iterators are objects that behave like pointers

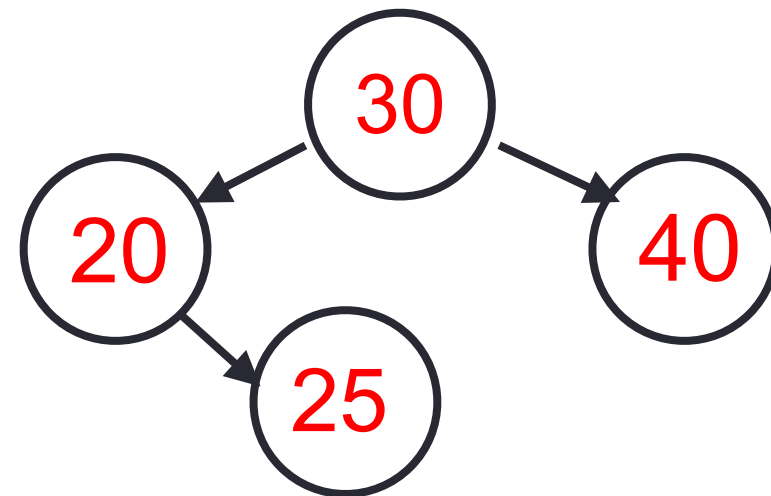
```
set<int> s;  
//insert some elements
```

```
_____ = s.find(25);
```



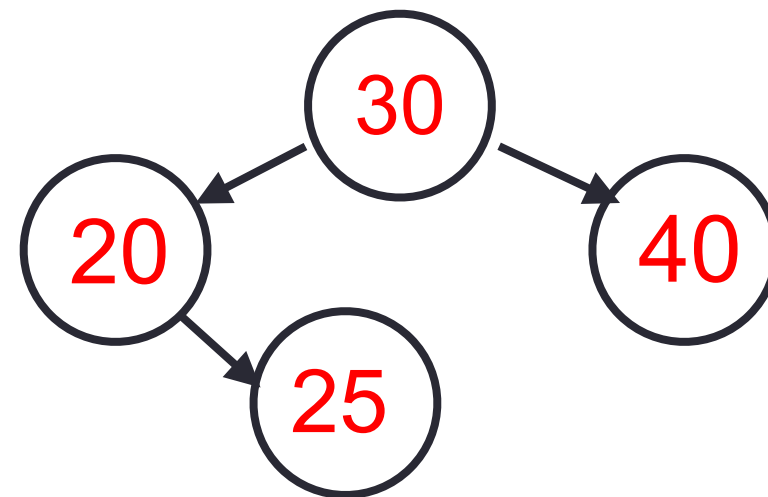
Iterators are objects that behave like pointers

```
set<int> s;  
//insert keys 20, 30, 35, 40  
  
_____ = s.find(25);
```



- “it” is an iterator object which can be used to access data in the container sequentially, without exposing the underlying details of the class

```
set<int> s;  
//insert keys 20, 30, 35, 40  
set<int>::iterator it;  
it = s.find(25);  
cout<<*it;
```



it

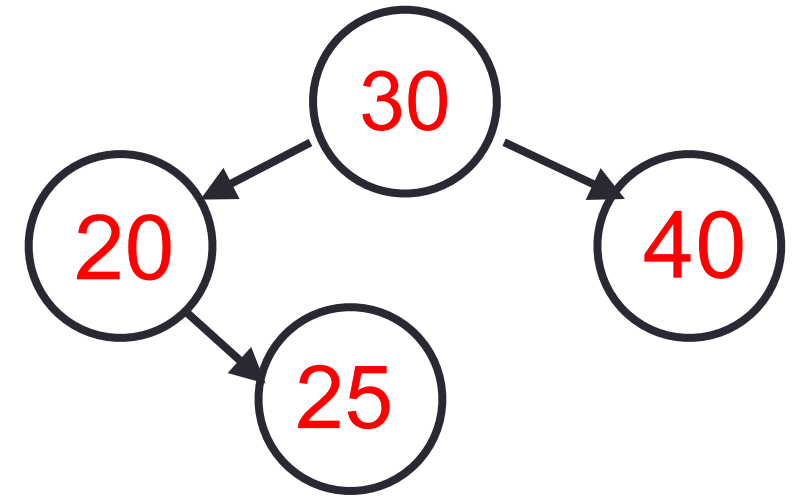


- “it” is an iterator object which can be used to access data in the container sequentially, without exposing the underlying details of the class

```
set<int> s;  
//insert keys 20, 30, 35, 40  
set<int>::iterator it;  
it = s.find(25);  
cout<<*it;  
it++;
```

List the operators that must be overloaded for iterator type?

- A. *
- B. ++
- C. <<
- D. All of the above
- E. A and B

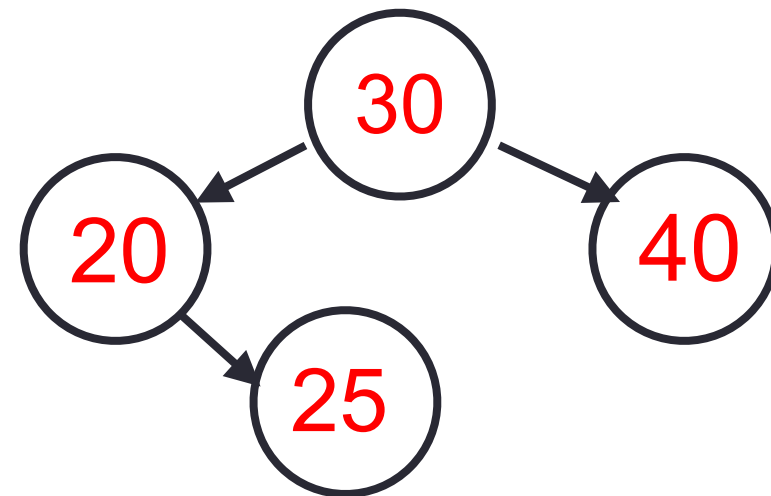


it



C++ Iterators

```
void printElements(set<int>& s) {  
    set<int>::iterator it = s.begin();  
    set<int>::iterator en = s.end();  
    while(it!=en){  
        std::cout << *it <<" ";  
        it++;  
    }  
    cout<<endl;  
}
```



C++ shorthand: auto

```
void printElements(set<int>& s) {  
    auto it = s.begin();  
    auto en = s.end();  
    while(it!=en) {  
        std::cout << *it <<" ";  
        it++;  
    }  
    cout<<endl;  
}
```

Finally: unveiling the range based for-loop

```
void printElements(set<int>& s)  {  
    for(auto item:s){  
        std::cout << item <<" ";  
    }  
    cout<<endl;  
}
```

PA02 Resources

Get familiarized with the STL documentation

Check out the member functions of set and vector

<https://www.cplusplus.com/reference/set/set/set/>

<https://www.cplusplus.com/reference/vector/vector/?kw=vector>

The complexity of each of the member functions is provided:

<https://www.cplusplus.com/reference/set/set/find/>