C++ ITERATORS

Problem Solving with Computers-II

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

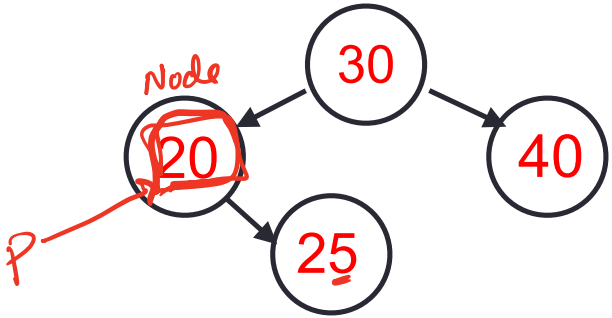
- Let's consider how we generally use pointers to parse an array

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?



20 25 30 40

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

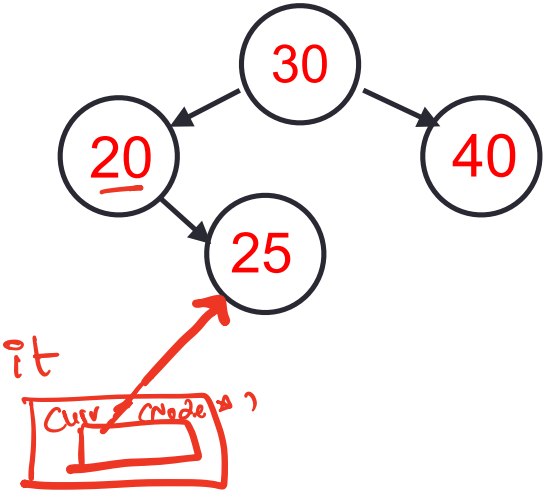
A. set<int>* X
B. Node
C. Node*
D. Iterator
E. int* X

Iterators are objects that behave like pointers

```
set<int> s;
//insert keys 20, 30, 35, 40

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

cout << *it ; // print 25



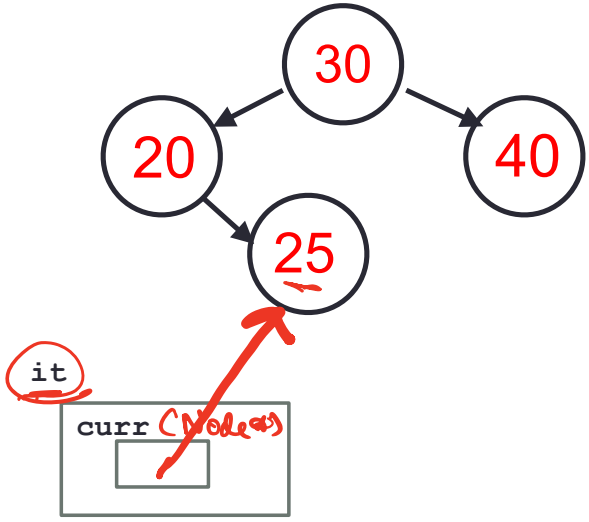
Node*

it is not a Node* int *

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

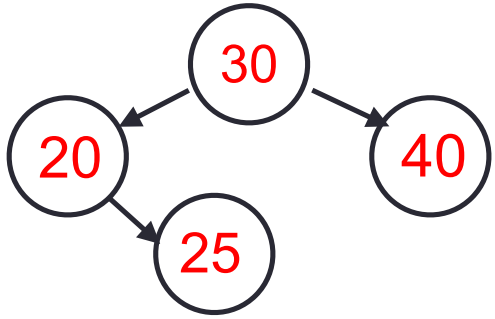
```
set<int> s;
//insert keys 20, 30, 35, 40
set<int>::iterator it;
it =  s.find(25);
cout<<*it;
```

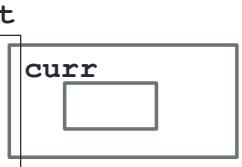
operator ++ ( ) {
curr = getSuccessor(curr);
}



- "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++;
```



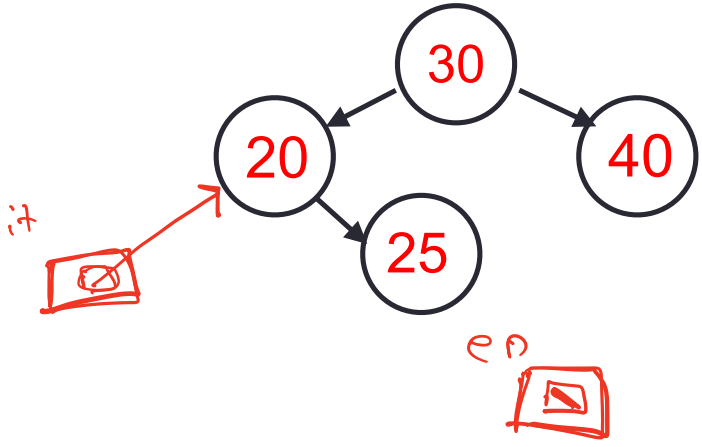


Which operators that must be overloaded for the iterator type?

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

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;
}
```



20 25 30 40

iterator it

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;
}
```

auto x = 5;

Compiler figures out the type of x based on the value used to initialize it. (In this case int)

Finally: unveiling the range based for-loop

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

PA02 Learning Goal

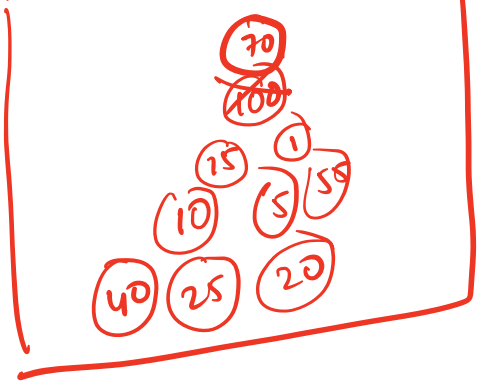
- Get familiarized with the STL documentation
- Select among available data structures

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/

priority-queue max: Heap



top() O(1)

Max-priority value 100

pop() O(log n)

top() O(1) 70