Member Access Operators

The dereference (`*`) and arrow (`->`) operators are often used in classes that present iterators and in smart pointer classes. We can logically add these operators to our own classes as well.

```
class Foo{
    public:
        string & operator*()const{
            auto p=check(curr);
            return (*p)[curr];// (*p) is the vector to which this object points
        }

        string *operator->()const{
            // delegate the real work to the deference operator
            // the dereference operator will do the check
            return & this->operator*();
        }
}
```

The dereference operator checks that `curr` is still in range and, if so, returns a reference to the element denoted by `curr`. The arrow operator avoids doing any work of its own by calling the dereference operator and returning the address of the element returned by that operator.

> Note: operator arrow must be a member. The dereference operator is not required to be a member but usually should be a member as well.

It is worth noting that we've defined these operators as `const` members. Unlike the increment and decrement operators, fetching an element doesn't change the state of objects.

We can use these operators the same way that we've used the corresponding operations on pointers or `vector` iterators: