PFI lecture classes in C++

- Object oriented programming leads to the notion of classes in C++.
- Classes allow us to define ADTs (abstract data types), which define the values and operations on these values.
- string type (C++ string) is in fact a class. Let us consider the string type in more details from the outside perspective (users of string perspective).

PFI lecture classes in C++

- The data value that the string class tries to capture is a sequence of characters. It should allow us to store a sequence of characters.
- Let us next consider some of the operations that we would like to perform on a sequence of characters.
  - How many characters are there in the string?
  - How do we initialize it?
  - How can we modify a portion of the string (adding some characters or removing some characters)?
  - How can we manipulate each character one by one?

PFI lecture classes in C++

- The designers of string class must provide “operations” (methods) to allows us to accomplished some of the tasks mentioned earlier.
- Examples:
  - string s=“Hi”; // s is an object of class string
  - string str=“Hey”; // str is an object of class string
  - int i = s.length(); // length() is a function or method
  - i = str.length(); // get str length
  - s.c_str() // c_str() is a function or method returns a cstring that the value is the same as s. A type conversion of a sort.

PFI lecture classes in C++

- Examples:
  - s.append(str); // value of s is “HiHey”
  - s.push_back(‘!’); // value of s is “HiHey!”
  - s.insert(2, “,”); // value of s is “Hi,Hey!”
  - str.assign(“Hello, world!”); // str gets new value
  - s.replace (s.find(","), 1, "! "); // value of s is “Hi! Hey!”
- The **member functions** or **methods** used in the examples are *append*, *push_back*, *insert*, *assign*, *find*, and *replace*

PFI lecture classes in C++

- Examples: working with each individual character

```
for (int i = 0; i< s.length(); i++)
    cout << s[i] << endl;
 // value of s is “Hi! Hey!”
s[2] = '?'; // value of s is “Hi? Hey!”
```

PFI lecture classes in C++

```
class Rectangle{
   public:
          Rectangle(); // constructor
          //~Rectangle(); not needed unless dynamic memory
   allocation is used
          int area() const; // method or member function
          void get(int& w, int& l) const;
          void set(int w, int l);
   private:
          int width; // attribute or member
          int length;
};
```

PFI lecture classes in C++

```
Rectangle::Rectangle(){
   width = 0;
   length = 0;
}

int Rectangle::area() const{
   return width*length;
}
```

PFI lecture classes in C++

```
void Rectangle::get(int& w, int& l) const{
    w = width;
    l = length;
}

void Rectangle::set(int w, int l){
    width = w;
    length = l;
}
```

PFI lecture classes in C++

```
Rectangle r1;
cout << "the area of r1 is " << r1.area() << endl;
int ww;
int ll;
r1.get(ww,ll);
cout << "the width and length of r1 are  " << ww << ", " << ll << endl;

r1.set(3,4);
cout << "the area of r1 is " << r1.area() << endl;

Rectangle r2;
r2.set(10,10);
cout << "the area of r2 is " << r2.area() << endl;
```