Comparing dates with a function

```
Date appt, today;
appt = 2010;
appt = 4;
appt = 10;
today.yr = 2010;
today.mon = 4;
today.day = 14;

if( before( appt, today ) )
   cout << "You missed it!" << endl;
```

- Function compares two objects: `appt` and `today`
  - The function takes `appt` and `today` as parameters
  - The function will return true if `appt` is before `today`

Comparing dates with a function

- Function definition
  - Takes 2 dates, `a` and `b`
  - Returns true if `a` is before `b`

```
bool before( Date a, Date b )
{
   if( (a.yr < b.yr) ||
        (a.yr == b.yr && a.mon < b.mon) ||
        (a.yr == b.yr && a.mon == b.mon && a.day < b.day) )
        return true;
   else
        return false;
}
```

Comparing dates with a method

```
Date appt, today;
appt = 2010;
appt = 4;
appt = 10;
today.yr = 2010;
today.mon = 4;
today.day = 14;

if( appt.Before( today ) )
   cout << "You missed it!" << endl;
```

- Method compares two objects: `appt` and `today`
  - The method is called on `appt`
  - The method takes `today` as a parameter
  - The method will return true if `appt` is before `today`

Comparing dates with a method

- Declare the method in the Date class
  - This method takes a single argument of type Date

```
class Date
{
public:
    int day, mon, yr;
    void Print();
    bool Before( Date other );
};
```

Comparing dates with a method

```
class Date
{
public:
    int day, mon, yr;
    void Print();
    bool Before( Date other );
};
```

- Method definition

```
bool Date::Before( Date other )
{
   if( (yr < other.yr) ||
        (yr == other.yr && mon < other.mon) ||
        (yr == other.yr && mon == other.mon && day < other.day) )
        return true;
   else
        return false;
}
```

Comparing dates with a method

- Comparing the object called on with the object passed in
  - `yr` is in the object called on
  - `other.yr` is in the object passed in

```
bool Date::Before( Date other )
{
   if( (yr < other.yr) ||
        (yr == other.yr && mon < other.mon) ||
        (yr == other.yr && mon == other.mon && day < other.day) )
        return true;
   else
        return false;
}
```

Exercise

1. Define a `Rectangle` class with:
   - Private data members `height` and `width` (doubles)
   - A public method `SetSize` that takes values for `height` and `width`
   - A public method `Area` that returns the area (double)
2. Define the `SetSize` and `Area` methods
3. Use the `Rectangle` class to:
   - Declare a `Rectangle` object
   - Call `SetSize` to set the size of the Rectangle to width 10 and height 3
   - Print the area of the `Rectangle`
   - Call `SetSize` to change the size of the Rectangle to width 15 and height 7
   - Print the area of the `Rectangle` again