Type Conversion

- Already can create user-defined types
- Already have default convert capabilities among built-in types.
- Need the ability to convert:
  - built-in type to user-defined type
  - user-defined type to built-in type
  - user-defined type to user-defined type

Built-in to User-defined

The conversion of a built-in type to a user-defined type can be accomplished by the use of an appropriate constructor for the targeted user-defined type.

This makes the conversion as simple as an explicit cast of one built-in type to another built-in type.
A Date Class

class Date {
private:
    int Month, Day, Year;
public:
    Date();
    Date(int M, int D, int Y);
    Date(int yyyymmd);  // conversion constructor
    void ShowDate();    // display function
};

Converts an int value into a Date object.

Date Class Implementation

Date::Date() {
    Month = 7;  Day = 4;  Year = 2001;
}
Date::Date(int M, int D, int Y) {
    Month = M;  Day = D;  Year = Y;
}
void Date::ShowDate() {
    cout << setfill('0')
    << setw(2) << Month << '/'
    << setw(2) << Day   << '/'
    << setw(4) << Year;
}
int to Date conversion

```cpp
Date::Date(int yyyymmdd) {
    Year = yyyymmdd / 10000;
    Month = (yyyymmdd - Year * 10000) / 100;
    Day = yyyymmdd - Year * 10000 - Month * 100;
}
```

Using the conversion

```cpp
void main() {
    Date a;
    cout << "Date a is:" << endl;
    a.ShowDate();
    cout << endl;
    a = Date(20020101);
    cout << "Date a is now: " << endl;
    a.ShowDate();
    cout << endl << endl;
}
```

Conversion of int value into a Date object.
Looks like standard explicit cast.

Output

Date a is:
07/04/2001

Date a is now:
01/01/2002
User-defined to Built-in

The conversion of a user-defined type to a built-in type can be accomplished by the use of an appropriate conversion operator function as a member of the user-defined type.

This also makes the conversion as simple as an explicit cast of one built-in type to another built-in type.

Revised Date Class

class Date {
private:
    int Month, Day, Year;
public:
    Date();
    Date(int M, int D, int Y);
    operator int();
    void ShowDate();
};

Converts a Date object into an int.
Date to int conversion

```cpp
Date::operator int() {
    int yyyymmdd;
    yyyymmdd = Year * 10000 + Month * 100 + Day;
    return yyyymmdd;
}
```

Using the conversion

```cpp
void main() {
    Date a(4, 1, 1999);
    int b;
    b = (Date) a;

    cout << "a's date is: ";
    a.ShowDate();
    cout << endl
         << "This date, as an int, is: "
         << b << endl;
}
```

Conversion of Date object into an int value. Looks like standard explicit cast.

Output

```
a's date is: 04/01/1999
This date, as an int, is: 19990401
```
**User-defined to User-defined**

The conversion of a user-defined type to a user-defined type is also accomplished by the use of a member conversion operator function.

This makes the conversion as simple as an explicit cast of one built-in type to another built-in type.

---

**Add an IntDate Class**

```cpp
// Dates.h
class IntDate; // forward declaration

class Date {
    private:
        int Month, Day, Year;
    public:
        Date(int M = 7, int D = 4, int Y = 2001);
        operator IntDate(); // conversion operator
        void ShowDate();
};
// continues...```

Converts a Date object into an IntDate object.
Add an IntDate Class

// ... class IntDate {
private:
    int yyyymmdd;
public:
    IntDate(int ymd = 0);
    operator Date(); // conversion operator
    void ShowIntDate();
};

Date to IntDate conversion

Date::operator IntDate() {
    int Temp;
    Temp = 10000 * Year + 100*Month + Day;
    return IntDate(Temp);
}
Date to IntDate conversion

```cpp
Date::operator IntDate() {
    int Temp;
    Temp = 10000 * Year + 100*Month + Day;
    return IntDate(Temp);
}
```

IntDate has an appropriate constructor.

IntDate Class Implementation

```cpp
IntDate::IntDate(int ymd) {
    yyyymmdd = ymd;
}

void IntDate::ShowIntDate() {
    cout << yyyymmdd;
}
```
**IntDate to Date conversion**

`IntDate::operator Date()` {
    int M, D, Y;

    Y = yyyymmdd / 10000;
    M = (yyyymmdd - Y*10000) / 100;
    D = yyyymmdd - Y*10000 - M*100;
    return Date(M, D, Y);
}

Assumes `IntDate` has an appropriate constructor.

**Using the conversion**

```cpp
void main() {
    Date a(4, 1, 1999), b;
    IntDate c(20011215), d;
    
    b = Date(c);
    d = IntDate(a);

    cout << "a's date is: ";
    a.ShowDate();

    cout << endl << "as an IntDate object this date is: ";
    d.ShowIntDate();
    // continues . . .
}
```

Conversions of `IntDate` object into a `Date` object and `Date` object into an `IntDate` object. Looks like standard explicit cast.
Using the conversion

// . . . continued
    cout << endl << "c's date is: ";
    c.ShowIntDate();

    cout << endl << "as a Date object this date is: ";
    b.ShowDate();
    cout << endl << endl;
    
} } } } } } } } } } }

Output

a's date is: 04/01/1999
as an IntDate object this date is: 19990401
c's date is: 20011215
as a Date object this date is: 12/15/2001