Declaring Objects

- Declared same as all variables
  - `<type> <variable_name>`
- Example:
  ```
  DayOfWeek today, birthday;
  ```
  - Declares two objects of class type DayOfWeek
- Objects include:
  - Data
    - Members month, day
  - Operations (member functions)
    - output()

Chapter 6
Classes

Class Member Access

- To access a member function use dot operator
  - `today.output()`
- Only public members can be accessed with dot operator
- Need functions to access private members

Class Definitions

- Example:
  ```
  class DayOfWeek {
    public:
      void output();
    private:
      int month;
      int day;
  };
  ```
- Notice only includes member function’s prototype
  - Function’s implementation is elsewhere
**Encapsulation**

- Any data type includes
  - Data (range of data)
  - Operations (that can be performed on data)
- Example:
  
  int data type has:
  Data: +-32,767
  Operations: +,-,*,/,%,logical,etc.

- Same with classes
  - But WE specify data, and the operations to be allowed on our data!

**Class Member Functions**

- Must define or "implement" class member functions
- Like other function definitions
  - Can be after main() definition
  - Must specify class:
    
    void DayOfYear::output() {...}

    - :: is scope resolution operator
    - Instructs compiler "what class" member is from
    - Item before :: called type qualifier
- Member functions can refer to data members

---

**Public and Private Members**

- Data in class is usually private
  - Upholds principles of OOP
  - Hide data from user
  - Allow manipulation only via operations
    - Which are member functions
- Public items (usually member functions) are "user-accessible"

**Class Type**

- A class is a full-fledged type
  - Just like data types int, double, etc.
- We can have variables of a class type
  - We simply call them "objects"
- We can have parameters of a class type
  - Pass-by-value
  - Pass-by-reference
- We can use class type like any other type
**Accessors & Mutators**

- Object needs to "do something" with its data
- Accessor member functions
  - Used to read the object’s data
  - Usually starts with "get"
- Mutator member functions
  - Used to change the object’s data
  - Usually starts with set

**Public and Private Example**

- Declare object: `DayOfYear today;`
- Object `today` can ONLY access public members
  - `cin >> today.month;` // NOT ALLOWED!
  - `cout << today.day;` // NOT ALLOWED!
  - Must instead call public operations:
    - `today.input();`
    - `today.output();`

**Use of Public & Private**

- Can mix & match public & private
- More typically place public first
  - Allows easy viewing of portions that can be USED by programmers using the class
  - Private data is "hidden", so irrelevant to users
- Outside of class definition, cannot change (or even access) private data