

Chapter 7

Enumeration Types and **typedef**

CPSC 601
Programming with C & Java
Hyunyoung Lee

Enumeration Types (1)

- Are user-defined types.
- Use keyword `enum`.
- Allow us to name a finite set and to declare identifiers, called *enumerators*, that are elements of the set.
- E.g. `enum day {sun, mon, tue, wed, thu, fri, sat};`
 - creates the user-defined type `enum day`.
 - The keyword `enum` is followed by the tag name `day`.
 - The enumerators are the identifiers `sun, mon, ..., sat`. They are constants of type `int` (by default, the first one is given the value 0, and each succeeding one has the next integer value).

Enumeration Types (2)

- To declare variables of type enum day:
enum day d1, d2;
 - declares d1 and d2 to be of type enum day.
 - The variables d1 and d2 can take on as values only the enumerators in the set.
 - d1 = mon; assigns the value mon to d1
 - if (d1 == d2) tests whether d1 is equal to d2.
- The enumerators can be initialized. Also, we can declare variables of the enumeration type right after the enumerator declaration:

```
enum suit {clubs = 1, diamonds, hearts, spades} a, b, c;
```

↑
type specifier

↑
variables of this type

- diamonds, hearts, and spades have the values 2, 3, and 4, respectively.

Enumeration Types (3)

- Another example of initialization:

```
enum fruit {apple = 7, pear, orange = 3, lemon} frt;  
enum veg {beet = 17, corn = 17} vege1, vege2;
```

- The tag name need not be present:

```
enum {fir, pine} tree;
```

- Since there is no tag name, no other variables of type `enum {fir, pine}` can be declared.
- In general, treat enumerators as programmer-specified constants and use them to aid program clarity.

The Use of typedef

- An identifier can be associated with a specific type: e.g.,

```
typedef int color;  
color red, blue, green;
```

- We can typedef our enumeration type:

```
/* Compute the next day. */  
enum day {sun, mon, tue, wed, thu, fri, sat};  
typedef enum day day;  
day find_next_day(day d)  
{  
    day next_day;  
    switch (d) {  
        case sun:  
            next_day = mon; break;  
        case mon:  
            next_day = tue; break;  
        .....        case sat:  
            next_day = sun; break;  
    } /* end of switch */  
    return next_day;  
}
```

- Another version of the find_next_day function:

```
/* Compute the next day with a cast. */  
enum day {sun, mon, tue, wed, thu, fri, sat};  
typedef enum day day;  
day find_next_day(day d)  
{  
    return ((day) (((int) d + 1) % 7));  
}
```

- Importance of enumeration types:
 - Self-documenting character (the enumerators are mnemonic).
 - Enforce the compiler to provide programmer-defined type checking so that one does not inadvertently mix apples and diamonds.

Things To Do

- Read Chapter 7.
- Test out all the example codes in Chapter 7 including those discussed in class.
- Always check out on the course home page:
<http://faculty.cs.tamu.edu/hlee/teaching/601/home.html>