Enumeration Type

- Enumeration allows you to define an ordered set of values
  - Each value is an identifier
  - Useful for dealing with a fixed set
  - More efficient than using strings, more informative than using numbers
- Examples:

```
enum phoneType { HOME, WORK, MOBILE, ADDITIONAL };
enum standing { FRESHMAN, SOPHOMORE, JUNIOR, SENIOR };
enum grade { A, B, C, D, F };
enum color { RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO,
   VIOLET };
```

Enumeration type

- Once you have defined a enumeration type, you can use it just like any other data type
- To declare a variable:

```
phoneType phone1Type, phone2Type;
int number1, number2;
```

- To assign it a value:

```
phone1Type = HOME;
number1 = 10;

phone2Type = phone1type;
number2 = number1;
```

Enumeration Type

- Enumeration values are identifiers
  - **Not strings or characters**
  - Must be valid identifiers
  - By convention typed **in all caps**
- The values in an enumeration must be unique
  - They can't appear in another enumeration in the same function

EXAMPLE 8-3

Consider the following statements:

```
enum grades {'A', 'B', 'C', 'D', 'F'}; //illegal enumeration type
enum places {1ST, 2ND, 3RD, 4TH};  //illegal enumeration type
```

These are illegal enumeration types because none of the values is an identifier. The following, however, are legal enumeration types:

```
enum grades {A, B, C, D, F};
enum places {FIRST, SECOND, THIRD, FOURTH};
```

EXAMPLE 8-4

Consider the following statements:

```
enum mathStudent {JOHN, BILL, CINDY, LISA, RON};
enum compStudent {SUSAN, CATHY, JOHN, WILLIAM}; //illegal
```

Suppose that these statements are in the same program in the same block. The second enumeration type, `compStudent`, is not allowed because the value `JOHN` was used in the previous enumeration type `mathStudent`.

Operations on Enumeration Types

- Arithmetic operators are not allowed:

```
phone1Type = phone2Type - 1;  // illegal
phone1Type++;                 // illegal
```

- Comparison operators are valid (since the values are ordered):

```
phone1Type == WORK
phone2Type < MOBILE
```