4.11 Enumerations

Some variables only need store a small set of named values. For example, a variable representing a traffic light need only store values named GREEN, YELLOW, or RED. An ***enumeration type*** declares a name for a new type and possible values for that type.

Construct 4.11.1: Enumeration type.

```
enum identifier {enumerator1, enumerator2,  ...};
```

Feedback?

The items within the braces ("enumerators") are integer constants automatically assigned an integer value, with the first item being 0, the second 1, and so on. An enumeration declares a new data type that can be used like the built-in types int, char, etc.

Figure 4.11.1: Enumeration example.

```
User commands: n (next), r
(red), q (quit).

Red light  n
Green light  n
Yellow light  n
Red light  n
Green light  r
Red light  n
Green light  n
Yellow light  n
Red light  q
Quit program.
```

```
#include <iostream>
using namespace std;

/* Manual controller for traffic light */
int main() {
   enum LightState {LS_RED, LS_GREEN, LS_YELLOW,
LS_DONE};
   LightState lightVal;
   char userCmd;

   lightVal = LS_RED;
   userCmd = '-';

   cout << "User commands: n (next), r (red), q
(quit)." << endl << endl;

   lightVal = LS_RED;
   while (lightVal != LS_DONE) {

      if (lightVal == LS_GREEN) {
         cout << "Green light  ";
         cin >> userCmd;
         if (userCmd == 'n') { // Next
            lightVal = LS_YELLOW;
         }
      }
      else if (lightVal == LS_YELLOW) {
         cout << "Yellow light  ";
         cin >> userCmd;
         if (userCmd == 'n') { // Next
            lightVal = LS_RED;
         }
      }
      else if (lightVal == LS_RED) {
         cout << "Red light  ";
         cin >> userCmd;
         if (userCmd == 'n') { // Next
            lightVal = LS_GREEN;
         }
      }

      if (userCmd == 'r') { // Force immediate red
         lightVal = LS_RED;
      }
      else if (userCmd == 'q') { // Quit
         lightVal = LS_DONE;
      }
   }

   cout << "Quit program." << endl;

   return 0;
}
```

Feedback?

The program declares a new enumeration type named LightState. The program then declares a new variable lightVal of that type. The loop updates lightVal based on the user's input.

The example illustrates the idea of a ***state machine*** that is sometimes used in programs, especially programs that interact with physical objects, wherein the program moves among particular situations ("states") depending on input; see What is: State machine.

Because different enumerated types might use some of the same names, e.g., `enum Colors {RED, PURPLE, BLUE, GREEN};` might also appear in the same program, the program above follows the practice of prepending a distinguishing prefix, in this case "LS" (for Light State).

One might ask why the light variable wasn't simply declared as a string, and then compared with strings "GREEN", "RED", and "YELLOW". Enumerations are safer. If using a string, an assignment like `light = "ORANGE"` would not yield a compiler error, even though ORANGE is not a valid light color. Likewise, `light == "YELOW"` would not yield a compiler error, even though YELLOW is misspelled.

One could instead declare constant strings like `const string LS_GREEN = "GREEN";` or even integer values like `const int LS_GREEN = 0;` and then use those constants in the code, but an enumeration is clearer, requires less code, and is less prone to error.

Note: Each enumerator by default is assigned an integer value of 0, 1, 2, etc. However, a programmer can assign a specific value to any enumerator. Ex:
`enum TvChannels {TC_CBS = 2, TC_NBC = 5, TC_ABC = 7};`

PARTICIPATION ACTIVITY 4.11.1: Enumeration syntax. ✓

1) Which of the following declares a new enumeration type named CarGear, with PARK, REVERSE, and DRIVE? ✓

- ○ `enum CarGear (PARK, REVERSE, DRIVE);`
- ○ `enum CarGear {PARK, REVERSE, DRIVE}`
- ○ `enum CarGear {PARK, REVERSE, DRIVE};`
- ○ `CarGear {PARK, REVERSE, DRIVE};`

Feedback?

PARTICIPATION ACTIVITY 4.11.2: Enumerations.

1) Declare a new enumeration type named HvacStatus with three

named values HVAC_OFF, AC_ON, FURNACE_ON, in that order.

Check Show answer

2) Declare a variable of the enumeration type HvacStatus named systemStatus.

Check Show answer

3) Assign AC_ON to the variable systemStatus.

Check Show answer

4) What is the integer value of systemStatus after the following?
`systemStatus = FURNACE_ON;`

Check Show answer

5) Given `enum TvChannels {TC_CBS = 2, TC_NBC = 5, TC_ABC = 7};`, what does `cout << TC_ABC;` output?

Check Show answer

Feedback?

CHALLENGE ACTIVITY 4.11.1: Enumerations: Grocery items.

Print either "Fruit", "Drink", or "Unknown" (followed by a newline) depending on the value of userItem. Print "Unknown" (followed by a newline) if the value of userItem does not match any of the defined options. For example, if userItem is GR_APPLES, output should be:

```
Fruit
```

```
#include <iostream>
using namespace std;

int main() {
   enum GroceryItem {GR_APPLES, GR_BANANAS, GR_JUICE, GR_WATER};
   GroceryItem userItem;

   userItem = GR_APPLES;

   /* Your solution goes here  */
   if(userItem == GR_APPLES || userItem == GR_BANANAS)
   {
      cout <<  "Fruit" << endl;
   } else if (userItem == GR_JUICE || userItem == GR_WATER){
      cout <<  "Drink" << endl;
   } else{
      cout <<  "Unknown" << endl;
   }

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with userItem = GR_APPLES

Your output
```
Fruit
```

✓ Testing with userItem = GR_JUICE

Your output
```
Drink
```

✓ Testing with userItem = (GroceryItem)5

Your output
```
Unknown
```

Feedback?

CHALLENGE ACTIVITY

4.11.2: Soda machine with enums.

Complete the code provided to add the appropriate amount to totalDeposit.

```
   int userInput;

   totalDeposit = 0;

   cout << "Add coin: 0 (add 25), 1 (add 10), 2 (add 5).  ";
   cin >> userInput;

   if (userInput == ADD_QUARTER) {
      totalDeposit = totalDeposit + 25;
   }

   /* Your solution goes here  */
   else if (userInput == ADD_DIME){
      totalDeposit = totalDeposit + 10;
   }else if(userInput == ADD_NICKEL){
      totalDeposit = totalDeposit + 5;
   }

   else {
      cout << "Invalid coin selection." << endl;
   }
```

Run ✓ All tests passed

✓ Testing for userInput = 0

Your value 25

✓ Testing for userInput = 1

Your value 10

✓ Testing for userInput = 2

Your value 5

✓ Testing for userInput = 5

Your value -1

Feedback?