You have used numeric type, `char` type, and `bool` type to declare variables. C++ enables you to declare your own type, known as enumerated type, using the `enum` keyword. For example,

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY};
```

declares an enumerated type named `Day` with possible values `MONDAY`, `TUESDAY`, `WEDNESDAY`, `THURSDAY`, and `FRIDAY` in this order.

An enumerated type defines a list of enumerated values. Each value is an identifier, not a string. The identifiers are known to the program once they are declared in the type.

By convention, an enumerated type is named with the first letter of each word capitalized and a value of an enumerated type is named like a constant with all uppercase letters.

Once a type is defined, you can declare a variable of that type:

```
Day day;
```

The variable `day` can hold one of the values defined in the enumerated type. For example, the following statement assigns enumerated value `MONDAY` to variable `day`:

```
day = MONDAY;
```

As with any other type, you can declare and initialize a variable in one statement:

```
Day day = MONDAY;
```

Furthermore, C++ allows you to declare an enumerated type and variable in one statement. For example,

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY} day = MONDAY;
```

**CAUTION**

An enumerated value can not be redeclared. For example, the following code would cause a syntax error.

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY};
```
const int MONDAY = 0; // Error: MONDAY already declared.

An enumerated variable holds a value. Often your program needs to perform a specific action depending on the value. For example, if the value is MONDAY, play soccer; if the value is TUESDAY, take piano lesson, and so on. You can use an if statement or a switch statement to test the value in the variable, as shown in (a) and (b)

```
if (day == MONDAY)
    // process Monday
else if (day == TUESDAY)
    // process Tuesday
else ...
```

Equivalent
```
switch (day)
{
    case MONDAY:
        // process Monday
        break;
    case TUESDAY:
        // process Tuesday
        break;
    ...
}
```

Enumerated values are stored as integers in memory. By default, the values correspond to 0, 1, 2, ..., in the order of their appearance in the list. So, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, and FRIDAY correspond to the integer values 0, 1, 2, 3, and 4. You can explicitly assign an enumerated value with any integer value. For example,

```
enum Color {RED = 20, GREEN = 30, BLUE = 40};
```

RED has an integer value 20, GREEN 30, and BLUE 40.

If you assign integer values for some values in the enumerated type declaration, the other values will receive default values. For example,

```
enum City {PARIS, LONDON, DALLAS = 30, HOUSTON};
```

PARIS will be assigned 0, LONDON 1, DALLAS 30, and HOUSTON 31.

You can assign an enumerated value to an integer variable. For example,

```
int i = PARIS;
```

This assigns 0 to i.

Enumerated values can be compared on their assigned integer values using the six comparison operators. For example, (PARIS < LONDON) yields true.

Listing 1 gives an example of using enumerated types.
Listing 1 TestEnumeratedType.cpp

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

int main()
{
    enum Day {MONDAY = 1, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY};

    cout << "Enter a day (1 for Monday, 2 for Tuesday, etc): ";
    int dayNumber;
    cin >> dayNumber;

    switch (dayNumber) {
        case MONDAY:
            cout << "Play soccer" << endl;
            break;
        case TUESDAY:
            cout << "Piano lesson" << endl;
            break;
        case WEDNESDAY:
            cout << "Math team" << endl;
            break;
        default:
            cout << "Go home" << endl;
    }

    return 0;
}
```

**Sample Output**

```
Enter a day (1 for Monday, 2 for Tuesday, etc): 1
Play soccer
```

**Sample Output**

```
Enter a day (1 for Monday, 2 for Tuesday, etc): 4
Go home
```

Line 6 declares an enumerated type `Day` and declares a variable named `day` in one statement. Line 10 reads an `int` value from the keyboard. The `switch` statement in lines 12-24 checks whether `day` is `MONDAY`, `TUESDAY`, `WEDNESDAY`, or others to display a message accordingly.