```
enum Season {spring,summer,autumn,winter};
```

Figure 1. Enumeration type *Season*.

enumeration-type-declaration:
    enum *enumeration-type-identifier* **{** *enumeration-constant-list* **};**

enumeration-type-identifier:
    id

enumeration-constant-list:
    enumeration-constant
    enumeration-constant-list **,** *enumeration-constant*

enumeration-constant:
    id

An *enumeration-type* specifies the set of values for that type. The set of values is given by the *enumeration-constant-list*. Each *enumeration-constant* is a name for an integer. By default, the first *enumeration-constant* in the list is assigned a value of zero (**0**). Subsequent *enumeration-constant*s in the list are given successive integer values. For example, the *enumeration-type Season* has *enumeration-constant*s *spring*, *summer*, *autumn*, and *winter*. *Enumeration-constant spring* has an integer value of zero (**0**). In the same way, *enumeration-constant*s *summer*, *autumn*, and *winter* have integer values 1, 2, and 3 respectively.

```
#include <iostream>
#include <iomanip>
using namespace std;
enum Season {spring,summer,autumn,winter};
int main()
{    cout << endl;
     cout << "enumeration constant spring=" << spring;
     cout << endl;
     cout << "enumeration constant summer=" << summer;
     cout << endl;
     cout << "enumeration constant autumn=" << autumn;
     cout << endl;
     cout << "enumeration constant winter=" << winter;
     cout << endl;
     return 0;
}
```

Figure 2. Program **p01**.

Program **p01** produces:

```
enumeration constant spring=0
enumeration constant summer=1
enumeration constant autumn=2
enumeration constant winter=3
```

```
#include <iostream>
#include <iomanip>
using namespace std;
enum Season {spring,summer,autumn,winter};
int main()
{    static char* SeasonSpelling[]={"spring","summer","autumn","winter"};
     for (Season s=spring;s<=winter;s=(Season)(s+1)) {
          cout << endl;
          cout << SeasonSpelling[s];
     }
     cout << endl;
     return 0;
}
```

Figure 3. Program **p02**.

Program **p02** output:

```
spring
summer
autumn
winter
```

Program **p02** notes:

1. The *for-statement* variable *s* has type *Season*. Type *Season* is an enumerated type having enumeration constants *spring*, *summer*, *autumn*, and *winter* with integer values **0**, **1**, **2**, and **3** respectively.
2. The *for-statement* test, *s*<=*winter*, is equivalent to the test *s*<=**3**.
3. In the for-statement increment expression **(***s*+**1)**, *s* is coerced to type integer and the sum **(***s*+**1)** has type integer.
4. The integer expression **(***s*+**1)** is coerced to type *Season* to be compatible with s in the assignment *s*=*Season***(***s*+**1)** by the type conversion properties of the type name *Season*.
5. Variable s indexes array *SeasonSpelling* to obtain a character string that matches the enumeration constant value of *s*.

```
#include <iostream>
#include <iomanip>
using namespace std;
enum Season {spring,summer,autumn,winter};
int main()
{    static char* SeasonSpelling[]={"spring","summer","autumn","winter"};
     for (Season s=spring;s<=winter;s=(Season)(s+1)) {
          cout << endl;
          switch (s) {
               case spring:      cout << "spring";     break;
               case summer:      cout << "summer";     break;
               case autumn:      cout << "autumn";     break;
               case winter:      cout << "winter";     break;
          }
     }
     cout << endl;
     return 0;
}
```

Figure 4. Program **p03**.

Program **p03** output:

```
spring
summer
autumn
winter
```

Program **p03** notes:

1. Program **p03** illustrates the use of *enumeration-constant*s as case labels.