

enum

Enumerated types allow the programmer to create their own data types. For example if the programmer is doing a calendar program it would be nice to have data types for days of the week and months of the year, instead of using integer values to represent these types.

```
enum DaysOfWeek {sun, mon, tue, wed, thu, fri, sat};  
enum MonthsOfYear {jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec};
```

In the type *DaysOfWeek* there are 7 objects to the data type, and the order is defined by the placement of the objects in the braces. Sunday is less than Monday, and Tuesday is less than Saturday. The compiler gives each type object an integer value starting with 0, starting with the first object in the braces (sun). Therefore Wednesday is 3, Saturday is 6. This is called it's ordinal value.

You use this data type like you would other data types.

```
int main()  
{  
    DaysOfWeek day;  
  
    for (day = mon; mon <= fri; day++)  
        // process week day info
```

This makes the code much clearer to understand.

One thing that cannot be done with enumerated type is I/O. We can do the following:

```
cin >> day;    // This will give a compiler error  
  
cout << day;   // This will output an integer (ordinal) value
```