Namespaces

- When a header file, such as `iostream`, is included in a program, the global identifiers in the header also become global identifiers in the program. If a program's global identifiers are the same as ones in the header the compiler generates a syntax error ("identifier redefined").
- The same problem can occur if a program uses 3rd party libraries. To overcome this problem 3rd party vendors began their global identifiers with an underscore (_). To avoid linking errors you shouldn't begin your identifiers with an underscore.
- C++ tries to solve the issue of overlapping global identifier names with the namespace mechanism, e.g.

```
namespace globalType
{
      const int N = 10;
      const double RATE = 7.5;
      void printResult();
}
```

- The syntax for accessing a namespace member is to use the scope resolution operator (`::`), so to access the `RATE` member in the above namespace you might use `globalType::RATE`.
- To simplify accessing of `namespace` members use the statement `using`, e.g.

```
using namespace globalType;
```

- This will allow you to access the `RATE` member directly without having to use `globalType::RATE`. You will not have to put the name and scope (`::`) before the member.