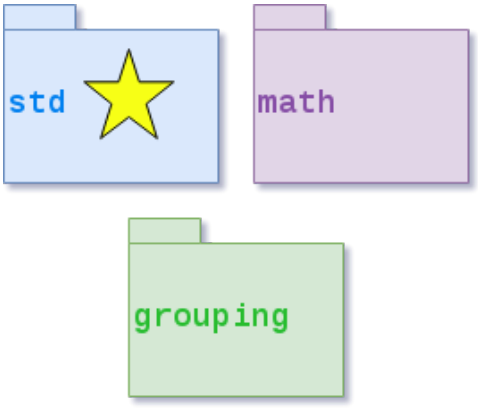
Standard Namespace



The namespace that is most common is the standard namespace. You have seen **using namespace std;** on the top of programs. This allows your program to use **cout**, **cin**, and **endl** among many other things, such as functions like **swap()** and classes like **ifstream**. Without **using namespace std;** you would have to prepend each standard namespace entity with **std::** such as **std::cout**. If that is too cumbersome, a second alternative is to replace **using namespace std;** with **using std::<ENTITY>;** at the top of your program, such as **using std::endl;** This would have to be done with each entity. This pulls only that single entity from the namespace and you can use it as you're used to.

So, why not always use **using namespace std;**? Using this is considered bad practice in the real world as the standard namespace encompasses several entities. For example, in CSII I made a template version of a swap function that swapped two variables (see the **Templates** page for more information on templates).

```
template <typename T>
void swap(T &x, T &y) {
    T temp = x;
    x = y;
    y = temp;
}
```

If you have this function in your program while having **using namespace std;** a compile-time error will arise and talk about ambiguity. This is due to the fact that the standard namespace has a swap function just like this one and the compiler doesn't know which one to use in the program.

For more complex programs, the **std::** route could prove to provide clarity while writing and reading code.

Examples

```
#include <iostream>
int main() {
    std::cout << "Hello World!" << std::endl;  }
```

```
#include <iostream>
using std::cout; using std::endl;
int main() {
    cout << "Hello World!" << endl;  }
```