Declaration vs. Definition

- A declaration tells the compiler the type of a variable, object or function.
- A definition allocates memory for a variable or object and is the implementation of a function.
- Multiple declarations are allowed, but only one definition.
- Some declarations are not definitions:

```
extern int i ;

int add_them (int, int) ;
```

Scope & Lifetime

- The **scope** of a declaration is the part of the program for which the declaration is in effect.
- C/C++ use lexical scoping.
- The **lifetime** of a variable or object is the time period in which the variable/object has valid memory.
- Lifetime is also called "allocation method" or "storage duration."

Lifetime

- **Static:** A static variable is stored in the data segment of the "object file" of a program. Its lifetime is the entire duration of the program's execution.
- **Automatic:** An automatic variable has a lifetime that begins when program execution enters the function or statement block or compound and ends when execution leaves the block. Automatic variables are stored in a "function call stack".
- **Dynamic:** The lifetime of a dynamic object begins when memory is allocated for the object (e.g., by a call to `malloc()` or using `new`) and ends when memory is deallocated (e.g., by a call to `free()` or using `delete`). Dynamic objects are stored in "the heap".

Scope

- **Local scope:** "visible" within function or statement block from point of declaration until the end of the block.

- **Class scope:** "seen" by class members.
- **Namespace scope:** visible within namespace block.
- **File scope:** visible within current text file.
- **Global scope:** visible everywhere unless "hidden".

Need for `namespace`

- Global namespace "pollution"
- Enclose functions, classes, etc by

```
namespace NS {
   // inside namespace block
   int i ;
   ...
}
```

- Hides identifiers within namespace block from others.
- Nested namespaces allowed.

The `::` operator

- Use `::` operator to refer to identifiers, (e.g, `NS::i`).
- The `using` declaration makes **one** identifier visible as local declaration:

```
using NS::i ;
```

- The `using namespace` directive makes **all** identifiers in namespace visible, as if `namespace NS {` did not exist.

```
using namespace NS ;
```