COSC1076 | Advanced Programming Techniques

Revision Questions | Answers | Week 02

1. Each term is:
   - A declaration creates a name within a given scope and provides all necessary typing information about the name.

     ```
     void foo();
     ```

   - A definition provides the implementation of a given name, and causes the program to set aside all necessary memory (where applicable)

     ```
     void foo() {
     }
     ```

   - An initialisation sets a variable for a value for the first time.

     ```
     int value = 7;
     ```

2. A namespace provides a new scope in which to encapsulate a set of named entities. This prevents the global scope from being polluted with unnecessary named entities and reduces the likelihood of errors from unforeseen name conflicts.
3. A global variable sits in the "global scope", outside of any function or class.
4. A function prototype is a function declaration, allowing any subsequent code to know about the function and call it.
5. At the low, CPU level, function parameters are always "passed-by-value", that is copied, (Even references!). However, by using pointers and references, at the abstract "high-level" of an algorithm, parameters can act as "pass-by-reference".
6. Arrays in Java are objects and "know" their own length. Arrays in C++ are primitive types and do not "know" their own length.
7. A string is an array of characters, for example:

   ```
   ichar string[10];
   ```

8. `\0`
9. This type of namespace import pollutes the global namespace with *all* named entities from *any* header file that is included. In general, a programmer may not be aware of all important entities, and this may result in unforeseen namespace conflicts.
10. The output is:

    ```
    example::foo
    ```

11. (a) The output is:

    ```
    zyxw
    ```

    (b) The value of each element of the array is:

```
string[0] = z
string[1] = y
string[2] = x
string[3] = w
string[4] = \0
string[5] = a
string[6] = b
string[7] = c
string[8] = d
string[9] = \0
```

12. A pointer is a memory address.
13. A reference is an alias to another named entity, such as a variable.
14. It is illegal to have a “reference to nothing”, that is, a reference to an unknown entity.
15.
```
void inc(double* value) {
    ++(*value);
}
```
16.
```
void inc(double& value) {
    ++value;
}
```
17. The program will “seg fault”. This happens because `ptr` is set to the value `NULL`, and is then dereferenced in the output statement on line 9.
18. This is so the `scanf` function can set the value of the variable(s) it is given to the values that are read from standard input.
19. The scoping elements are *public*, *protected*, and *private*.
20. The declaration is

```
class Example {
public:
   Example();
   void method();

private:
   int value;
}
```

The definition is

```
Example::Example() {
    value = 0;
}

void Example::method() {
    ++value;
}
```