Week 5

Ms. Kanwal Lodhi

Namespaces in C++

- **Namespace**
  - Namespace provide the space where we can define or declare identifier i.e. variable, method, classes.
  - Using namespace, you can define the space or context in which identifiers are defined i.e. variable, method, classes. In essence, a namespace defines a scope.
- **Advantage of Namespace to avoid name collision.**
  - Example, you might be writing some code that has a function called xyz() and there is another library available which is also having same function xyz(). Now the compiler has no way of knowing which version of xyz() function you are referring to within your code.
  - A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries.
  - The best example of namespace scope is the C++ standard library (std) where all the classes, methods and templates are declared. Hence while writing a C++ program we usually include the directive using namespace std;

namespace

- **Defining a Namespace:**
  - A namespace definition begins with the keyword namespace followed by the namespace name as follows:
  - namespace namespace_name

```
{
// code declarations
 }
```

The **using** directive (for namespaces)

- **The using directive:**
  - This directive tells the compiler that the subsequent code is making use of names in the specified namespace.
  - #include <iostream>
  - using namespace std;
  - // first name space
  - namespace first_space
  - {
  - void func()
  - {
  -     cout << "Inside first_space" << endl;
  - }
  - }
  - // second name space
  - namespace second_space
  - {
  - void func()
  - {
  -     cout << "Inside second_space" << endl;
  - }
  - }
  - using namespace first_space;
  - int main ()
  - {
  - // This calls function from first name space.
  - func();
  - return 0;
  - }
  -

using namespace std;

- Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries.
- The namespace keyword is used to declare a scope that contains **a set of related objects**. You can use a namespace to organize code elements and to create globally unique types.
- Syntax: using namespace std;
  - Here using is keyword for adding namespace to code.
  - namespace is also keyword.
  - std is **an abbreviation of "standard"**. std is the "standard namespace". cout , cin and a lot of other functions are defined within it. The reason for using this: When you don't use the std namespace, the compiler will try to call cout or cin as if they aren't defined in a namespace (like most functions in your codes).

Strings

- A collection of characters written in double quotations is called string or string constant.
- It consists of any alphabetic characters, digits and special symbols.
- String is a collection of characters. There are two types of strings commonly used in C++ programming language:
- Strings that are objects of string class (The Standard C++ Library string class)
- C-strings (C-style Strings)

C-Strings

- A string is stored as an array of characters.
- C-strings are arrays of type char terminated with null character, that is, \0.
- Null character indicates the end of string denoted by \0.
- Examples :
  1. “I’m student.”
  2. “Pakistan”
  3. “Ali khan”
- The length of the array should be adequate to store all the required characters plus 1 byte for null character.
- **Declaration and Initialization**
  – Syntax : char str[] = "C++";
  – In the above code, str is a string and it holds 4 characters.
  – Although, "C++" has 3 character, the null character \0 is added to the end of the string automatically.

C-Strings

- Other examples
  - char str[4] = "C++";
  - char str[] = {'C','+','+','\0'};
  - char str[4] = {'C','+','+','\0'};
- Example :
  - **C++ String to read a word**
  - char str[100];

```
cout << "Enter a string: ";
cin >> str;
cout << "You entered: " << str << endl;
```

  - If we add this code to our program, it can only works to show strings without spaces and if any space is entered as in sentences it will considers it as terminating character . Therefore we can’t input sentences in this type of code.

C-strings

- **C++ String to read a line of text**
  - To read the text containing blank space, cin.get function can be used. This function takes two arguments.
  - First argument is the name of the string (address of first element of string) and second argument is the maximum size of the array.
  - In the above program, str is the name of the string and 100 is the maximum size of the array.
  - Syntax :

```
char str[100];
cout << "Enter a string: ";
 cin.get(str, 100);
cout << "You entered: " << str << endl;
```

string object

- **string Object**
  - A C++ string is an object of the class string , which is defined in the header file <string> and which is in the standard namespace. The string class has several constructors that may be called (explicitly or implicitly) to create a string object.
  - In C++, you can also create a string object for holding strings.
  - Unlike using char arrays, string objects has no fixed length, and can be extended as per your requirement.

string class library

The string class library

- The library called string, which is part of the "Standard Template Library" in C++, contains a class called string
- Strings are declared as regular variables (not as arrays), and they support:
  - the assignment operator =
  - comparison operators ==, !=, etc
  - the + operator for concatenation
  - type conversions from c-strings to string objects
  - a variety of other member functions
- To use this library, make sure to #include it:

```
#include <string>
```

string object

Declaring and initializing

- Declare like a regular variable, using the word string as the type:
- string firstname;
- string lastname
- string address;
- string s1;
- To initialize, you can use the assignment operator, and assign a string object to another, or a string literal to a string object:
- firstname = "Joe";
- lastname = "Schmuckatelli";
- address = "123 Main St.";
- s1 = firstname;   // s1 is now "Joe" also
- And you can initialize on the same line as the declaration:
- string s2 = "How are you?";
- string s3 = "I am fine.";
- string s4 = "The quick brown duck jumped over the lazy octopus";

Different functions

- You can also initialize on the declaration statement in this format:
- string fname("Marvin");
- string lname("Dipwart");
- string s2(s1); // s2 is created as a copy of s1
- **Comparing string objects**
- You can compare the contents of string objects with the standard comparison operators:
- if (s1 == s2)
- cout << "The strings are the same";
- if (s1 < s2)
- cout << "s1 comes first lexicograpically";
- You can also mix and match with C-strings, as long as one of the operands is a string object:
- if (s1 == "Joe")
- cout << "The first student is Joe";
- if (s2 > "apple")
- cout << "s2 comes after apple in the dictionary";