# Programming in C/C++
## Basic Introduction to Strings

Kristina Shroyer

## Objectives

- We're going to go over the very basic concepts behind what a C++ string is
- We'll go over this in much more detail in Chapter 14
- For now all you need to know is what a string is and how to store it in a string type variable
- This will help you make your code for the planet program (HW #2) shorter and more efficient

## What is a String Constant?

- You have been using a lot of string constants/literals in your programs so far without a formal definition of what a string constant is
  - Character constants only hold one character
    - Example:  ‘A’
  - To store a series of characters a string constant needs to be used
    - Example: “Hello”
  - A string constant is enclosed in double quotation marks and a character constant is stored in single quotation marks

```
cout << ‘H’ << endl;
cout << “Hello” << endl;
```

## What is a String Constant?

- Strings allow a series of characters to be stored in consecutive memory locations

- String constants can be virtually any length
  - This means that there must be some way for the program to know how long the string constant is

  - In C++ this is done by appending an extra byte to the end of string constants
    - In this last byte of the consecutive memory locations that make up the string, the null terminator also known as the null character is stored
      - The null terminator is represented by the ‘\0’ character
      - The ASCII code for the null character is 0
      - Don’t confuse this with the character ‘0’ which is represented by the ASCII code 48

# What is a String Constant?

- Example of how a string constant is stored in memory

  `“Sebastian”`
- First notice that the quotation marks are not stored with the string. They are just a way of marking the beginning and end of the string in your source code
- Notice the very last byte of the string contains the null character
- The addition of the null character to the last byte of the string means that even though the string is 9 characters long it occupies 10 bytes of memory
- C++ automatically places the null terminator at the end of a string constant

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| S | e | b | a | s | t | i | a | n | \0 |

## Storage of a String Constant vs. a Character Constant

- Suppose you have the constants ‘A’ and “A” in a program

  ‘A’ is stored as: $\boxed{A}$ (A one byte element)

  “A” is stored as: $\boxed{A}\boxed{\backslash 0}$ (A two byte element)

- Since characters are really stored as ASCII codes this is what is actually being stored in memory

  ‘A’ is stored as: $\boxed{65}$

  “A” is stored as: $\boxed{65}\boxed{\backslash 0}$

- So even though some string constants look like character constants they aren’t
- There are also some characters that look like strings but aren’t
  - Example: ‘\n’ is a character constant represented by an ASCII code

## Storage of String Constant vs. a Character Constant

- Important points regarding character constants and string constants
  - Characters normally occupy a single byte of memory  (depends on whether ASCII or Unicode is being used)
  - Strings are consecutive sequences of characters that occupy consecutive bytes of memory
  - String constants have a null terminator at the end.  This marks the end of a string.
  - Character constants are enclosed in single quotation marks
  - String constants are enclosed in double quotation marks
  - Escape sequences such as ‘\n’ are stored as a single character

## Working with `strings` in C++

- So we want to know how to save strings into string type variables so we can use them more than once in our program
- We're going to do this using the C++ `string` class
  - We're talking class data type here (object oriented)
    - We will see we can work with strings in other ways though and that strings are ***NOT*** immutable in C++
- Later we'll see there is a lot more to the string class and we'll learn some more ways to work with strings
- There are three ways to work with strings of text in C++
  - The string class is just one way to work with strings in C++

# The C++ `string` Class

- Standard C++ provides a `string` class that allows a programmer to create a string type variable

- This `string` class wraps the representation of a string into an easier to use string type and attempts to standardize the representation of a string

- The `string` class provides many services to the outside code using the class
  - Some of the services include functions for declaring, creating, manipulating, and initializing a string
  - Advantage:  The string class takes care of any memory allocation you might need when manipulating the string (unlike C-Strings)

- The C++ `string` class does not **necessarily internally** represent a string by appending the `'/0'` character to the end of a string
  - This means string constants may be represented differently in memory than objects of the string class type
  - Therefore although the C++ string class may represent strings internally by terminating them with a null character there is no guarantee that is the way they are represented
  - The C++ string class handles the processing of the string internally within the class, allowing the programmer to not need to worry about exactly how the string is represented or how the operations done on the string are performed

## Basics of Using the C++ `string` Class

- The first step in using the string class is to **`#include`** the string header file in your program (as usual the include may differ depending on the compiler).
  - This is accomplished with the following preprocessor directive (don't forget **`using namespace std;`** ):
    - `#include <string>`

- The next step is to declare a string object and create a variable name to reference that string object in memory
    - `string movieTitle;`

- Now you can assign a string literal to the movieTitle with the assignment operator
    - `movieTitle = "The Matrix";`

- Now you can print the contents of movieTitle on the screen with the **`cout`** object.
    - `cout << "The movie title is " << movieTitle << endl;`

# Basics of Using the C++ `string` Class

**Example: firstStringEx.cpp**

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
   string movieTitle;
   movieTitle = "The Matrix";
   cout << "The movie title is " << movieTitle << "." << endl;
   system("PAUSE");
   return 0;
}
```

## The C++ `string` Class - Basics

- With the introduction of the new ANSI/ISO C++ standard, a **class** named string was provided as part of the standard C++ library.

- A `class` is a user defined data type (a class data type).
  - So the string class is the definition for a string data type
  - Just like a primitive data type, a class data type is defined by a set of valid data values and the operations that can be performed on those data values
  - A class data type and a built in (primitive) data type are constructed differently
    - A built in data type is provided as an integral part of the compiler
      - Provide standard operations like +,-,/,*…etc.
      - Storage areas are referred to as **variables**
    - A class data type is constructed using C++ code
      - Most of the operations are programmer defined functions
      - Storage areas are referred to as **objects**

# The C++ `string` Class - Basics

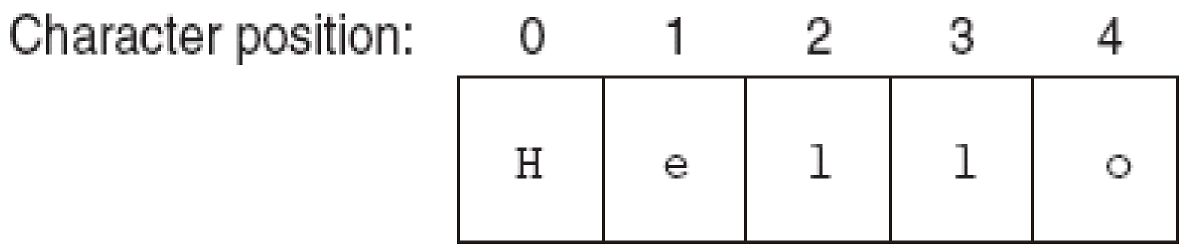
- This class provides a greatly expanded set of class functions/operations:
  - easy insertion and removal of characters from a string
  - automatic string expansion whenever a string's original capacity is exceeded
  - string contraction when characters are removed from the string
  - range checking to detect invalid character positions

- The values permitted by the string class are referred to as string literals
  - A **string literal**, as we have already seen, is any sequence of characters enclosed in double quotation marks
  - As has also been noted, a string literal is also referred to as a string value, a string constant, and more conventionally, simply as a string
  - Examples of strings are **`"This is a string"`**, **`"Hello World!"`** , and **`"xyz 123 *!#@&"`** . The double quotation marks are used to mark the beginning and ending points of the string and are never stored with the string.

# The C++ `string` Class - Basics

- The Figure below shows the programming representation of the string `"Hello"`
  - By convention, the first character in a string is always designated as position 0. This position value is also referred to as both the character's index value and its offset value.
  - Note the string in the diagram is not terminated by the null character.
    - This doesn't necessarily mean it isn't represented that way
    - Remember the representation of the C++ string is hidden from the programmer

**FIGURE 7.1** *The Storage of a* `string` *as a Sequence of Characters*



# Creating C++ `string` Class Instances

- Various methods are provided by the **`string`** class for declaring, creating, and initializing a string.

- When you create a new string a member function of the **`string`** class called a constructor is executed. The constructor creates and initializes your string type.

- We have seen one way to create a string object:

```
string movieTitle = "The Matrix";
```

- For now we'll use this simple way to save strings into string objects in our programs
  - We'll learn much more about the string class in Chapter 14