

## C++ Chapter 8: Strings

### Contents

C++ Chapter 8: Strings .....	1
C++ Strings Online Tutorials .....	1
String Member Functions.....	2
Tutorial 1: getline .....	2
String Methods .....	4
Concatenate .....	5
Length or Size .....	5
Operator and At .....	6
Substr .....	7
Change Case .....	8
Tutorial 2: String Methods with Joe Cool .....	9
Tutorial 3: Guess Password Length .....	11
Assignment 1: C++ String Practice .....	13
Assignment Submission.....	13

Time required: 90 minutes

### C++ Strings Online Tutorials

Please go through the following tutorials

- [https://www.w3schools.com/cpp/cpp\\_strings.asp](https://www.w3schools.com/cpp/cpp_strings.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_concat.asp](https://www.w3schools.com/cpp/cpp_strings_concat.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_numbers.asp](https://www.w3schools.com/cpp/cpp_strings_numbers.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_length.asp](https://www.w3schools.com/cpp/cpp_strings_length.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_access.asp](https://www.w3schools.com/cpp/cpp_strings_access.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_input.asp](https://www.w3schools.com/cpp/cpp_strings_input.asp)
- [https://www.w3schools.com/cpp/cpp\\_strings\\_namespace.asp](https://www.w3schools.com/cpp/cpp_strings_namespace.asp)

- <https://www.javatpoint.com/cpp-strings>

## String Member Functions

A string is a sequence of characters, most often used to represent words and names. The C++ standard library provides the string variable.

The string is part of the standard namespace, which means its full type name is **std::string**. You declare a string like any other variable.

```
std::string name;
```

You may assign a literal character sequence to a string via the familiar string quotation syntax:

```
/**
 * @name strings_joe.cpp
 */
#include <iostream>

int main()
{
    // Create and initialize string variable
    std::string name{"Joe"};
    std::cout << "The string variable contains: " << name << std::endl;
    // Assign new value to string variable
    name = "Jane";
    std::cout << "The string variable now contains: " << name << std::endl;
    return 0;
}
```

## Tutorial 1: getline

You can use cin with the >> operator to input strings. When cin reads data it passes over and ignores any leading whitespace characters (spaces, tabs, or line breaks ). Once it comes to the first nonblank character and starts reading, it stops reading when it gets to the next whitespace character. If we use the following statement

```
std::cin >> name1;
```

we can input "Mark" or "Twain" but not "Mark Twain" because cin cannot input strings that contain embedded spaces.

Let's create a simple program that gets a user's name from the console and displays it.

```

1  /**
2   * @name name_cin.cpp
3   */
4  #include <iostream>
5
6  int main()
7  {
8      // Create and initialize string variable
9      std::string name;
10     std::cout << "Enter your name: ";
11     // Get name from user, assign to string variable
12     std::cin >> name;
13     std::cout << "Your name is: " << name << std::endl;
14     return 0;
15 }

```

Example run:

```

PS Z:\_WNCC\Computer Science\Assignments CPP\08 Strings\Tutorials> .\name_cin.exe
Enter your name: Bill
Your name is: Bill
PS Z:\_WNCC\Computer Science\Assignments CPP\08 Strings\Tutorials> .\name_cin.exe
Enter your name: Bill Loring
Your name is: Bill

```

Notice that the first run worked just fine. The second run had an issue, it didn't get the full name. **cin** doesn't think that a white space or `\n` is a valid character for input. **cin** stops at the space.

B	L	A	H		B	L	A	H	\n
---	---	---	---	--	---	---	---	---	----

The **getline()** function solves that problem. It gets the entire line up to the `\n` character. The `\n` character is from the Enter/Return key.

```
getline(cin, variable)
```

The **getline()** function is part of the string library.

```

1  /**
2   * @name name_getline.cpp
3   */
4  #include <iostream>
5  #include <string>
6
7  int main()
8  {
9      // Create and initialize string variable
10     std::string name;
11     std::cout << "Enter your name: ";
12     // Get name from user, assign to string variable
13     getline(std::cin, name);
14     std::cout << "Your name is: " << name << std::endl;
15     return 0;
16 }

```

Example run:

```

Enter your name: Bill Arthur Loring
Your name is: Bill Arthur Loring

```

We now get everything up until the \n newline character.

## String Methods

The string class provides several methods. To use string object methods, you must provide the preprocessor directive.

```
#include <string>
```

Some string methods include:

- **[]** - provides access to the value stored at a given index within the string
- **=** - assigns one string to another
- **+=** - appends a string or single character to the end of a string object
- **.at()** - provides bounds-checking access to the character stored at a given index
- **.length()** - returns the number of characters that make up the string
- **.size()** - returns the number of characters that make up the string (same as length)

- **.find()** - locates the index of a substring within a string object
- **.substr()** - returns a new string object made of a substring of an existing string object
- **empty()** - returns true if the string contains no characters; returns false if the string contains one or more characters
- **clear()** - removes all the characters from a string

## Concatenate

We can concatenate strings (join strings) using + operator and append one string to another using the += operator.

```
#include <iostream>

int main()
{
    std::string st1("Hello ");
    std::string st2("World");
    std::cout << st1 + st2 << std::endl;
    st1 += "cpp";
    std::cout << st1 << std::endl;
    return 0;
}
```

Example run:

```
Hello World
Hello cpp
```

In the above example, the string variables s1 and s2 store the values "Hello " and "World".

**cout << st1 + st2 << endl;** - This is the same as writing **cout << st1 << st2 << endl;**  
This prints one string after the other.

**st1 += "cpp";** - We added the string "cpp" to the value of the string variable st1 making its value "Hello cpp".

## Length or Size

The **length()** or **size()** function is used to find the length of a string. These are pre-defined functions which belong to std::string. Let's see an example for the same.

```

/**
 * @name length_or_size.cpp
 * @brief C++ string length() and size()
 */

#include <iostream>

int main()
{
    std::string name;
    name = "I have 4 chocolates";
    std::cout << name << " has " << name.length()
                << " characters (length())" << std::endl;
    std::cout << name << " has " << name.size()
                << " characters (size())" << std::endl;
    return 0;
}

```

Example run:

```

I have 4 chocolates has 19 characters (length())
I have 4 chocolates has 19 characters (size())

```

The **length()** and **size()** function returned the length of the string including whitespace characters.

## Operator and At

[] and at() return the character at some specified position in a string. Let's see an example.

```

/**
 * @name string_operator_at.cpp
 * @brief C++ string [] operator and at()
 */

#include <iostream>
#include <string>

int main()
{
    std::string s = "I love C++";
    // Get character using [] operator
    std::cout << "Character at fifth position : " << s[5] << std::endl;
    // Get character using at() method
    std::cout << "Character at fifth position : " << s.at(5) << std::endl;
    return 0;
}

```

s represents the whole string and s[i] represents a character in a string at the ith position. Note that the position of the first character in a string is 0.

Example run:

```

Character at fifth position : e
Character at fifth position : e

```

## Substr

The **substr()** function returns a substring from a string by specifying its position.

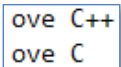
```

/**
 * @name substring.cpp
 * @brief C++ substring method
 */
#include <iostream>
#include <string>

int main()
{
    std::string str1("I love C++");
    // Return string from the 3 entry until the end of the string
    std::cout << str1.substr(3) << std::endl;
    // Return 5 characters starting at the 3rd position
    std::cout << str1.substr(3, 5) << std::endl;
    return 0;
}

```

Example run:



```

ove C++
ove C

```

## Change Case

C++ does not have a string change case method. We will use the char **tolower()** and **toupper()**.



```

/**
 * @name change_case.cpp
 * @brief C++ substring method
 */
#include <iostream>
#include <string>

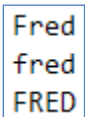
int main()
{
    // Declare a string object and initialize it
    std::string word{"Fred"};
    std::cout << word << std::endl;

    // Convert string to lower case
    for (int i = 0; i < word.length(); i++)
    {
        word[i] = tolower(word[i]);
    }
    std::cout << word << std::endl;

    // Convert string to upper case
    for (int i = 0; i < word.length(); i++)
    {
        word[i] = toupper(word[i]);
    }
    std::cout << word << std::endl;
    return 0;
}

```

Example run:



```

Fred
fred
FRED

```

## Tutorial 2: String Methods with Joe Cool

The following too cool program, **joe\_cool.cpp**, demonstrates several string methods.

```

1  /**
2   * @name joe_strings.cpp
3   * @brief Practice C++ string methods
4   */
5  #include <iostream>
6  // Needed for string methods only, standard string operations are
7  // included in the standard library
8  #include <string>
9  int main()
10 {
11     // Create and initialize string variables
12     std::string name{"Joe"};
13     std::string otherName{"Cool"};
14     std::string output;
15
16     // Concatenate the strings together with a space in the middle
17     std::cout << "Concatenate 2 string variables" << std::endl;
18     output = name + " " + otherName;
19     std::cout << output << std::endl;
20
21     // Concatenate another string
22     output = output + " rules!";
23     std::cout << output << std::endl;
24     std::cout << "Concatenate 2 string variables, find length of string"
25     |   |   |   << std::endl;
26
27     // Concatenate strings during cout, find the length of the string
28     std::cout << "The length of " + output + " is "
29     |   |   |   << output.length() << std::endl;

```

```

31 // Return character at a numbered position in the string
32 // using string [] operator. A string behaves like an array of characters
33 char oneChar{output[5]};
34 std::cout << "Character at 6th position : " << oneChar << std::endl;
35
36 // Return character at a numbered position in the string
37 // using at() operator. A string behaves like an array of characters.
38 oneChar = output.at(2);
39 std::cout << "Character at the 3rd position : " << oneChar << std::endl;
40
41 // Find first position of character in a string
42 std::cout << "C is at the " << output.find('C')
43 | << " position." << std::endl;
44
45 // Examples of substring
46 // All characters from 3 until the end of the string
47 std::cout << output.substr(3) << std::endl;
48 // 5 characters from position 3 were printed
49 std::cout << output.substr(3, 5) << std::endl;
50
51 return 0;
52 }

```

Example run:

```

Concatenate 2 string variables
Joe Cool
Joe Cool rules!
Concatenate 2 string variables, find length of string
The length of Joe Cool rules! is 15
Character at fifth position : o
Character at 2nd position : e
C is at the 4 position.
Cool rules!
Cool

```

## Tutorial 3: Guess Password Length

This program uses the `length()` function to determine if the users password is the same length as the hard coded password.

```

1  /**
2   * @name guess_password.cpp
3   */
4  #include <iostream>
5  #include <string>
6
7  int main()
8  {
9      // Create and initialize string variable
10     std::string input;
11     std::string password{"Password01"};
12     while (true)
13     {
14         std::cout << "Enter a password: ";
15         // Get name from user, assign to string variable
16         getline(std::cin, input);
17         if (input.length() < password.length())
18         {
19             std::cout << "Too short" << std::endl;
20         }
21         else if (input.length() > password.length())
22         {
23             std::cout << "Too long" << std::endl;
24         }
25         else
26         {
27             std::cout << "Just Right!" << std::endl;
28             break;
29         }
30     }
31     std::cout << "Press Enter to exit";
32     // Waits until it gets one character
33     std::cin.get();
34     return 0;
35 }

```

Example run:

```

Enter a password: pass
Too short
Enter a password: passwordkkkkd
Too long
Enter a password: password01
Just Right!

```

## Assignment 1: C++ String Practice

Create a C++ program named **StringPractice.cpp** with the following requirements.

1. Accept a string from the user.
2. Count the number of characters in the string.
3. Access and print out the last character.
4. Find and display the position of the first vowel.
5. Convert the string to UPPERCASE and lowercase.

---

### Assignment Submission

- Submit all C++ code files
- Insert a screenshot of each program showing that they work
- Submit in Blackboard