C++ Chapter 8: Strings

Contents

Time required: 90 minutes

C++ Strings Online Tutorials

Please go through the following short tutorials.

- https://www.w3schools.com/cpp/cpp_strings.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_length.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_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;
}
```

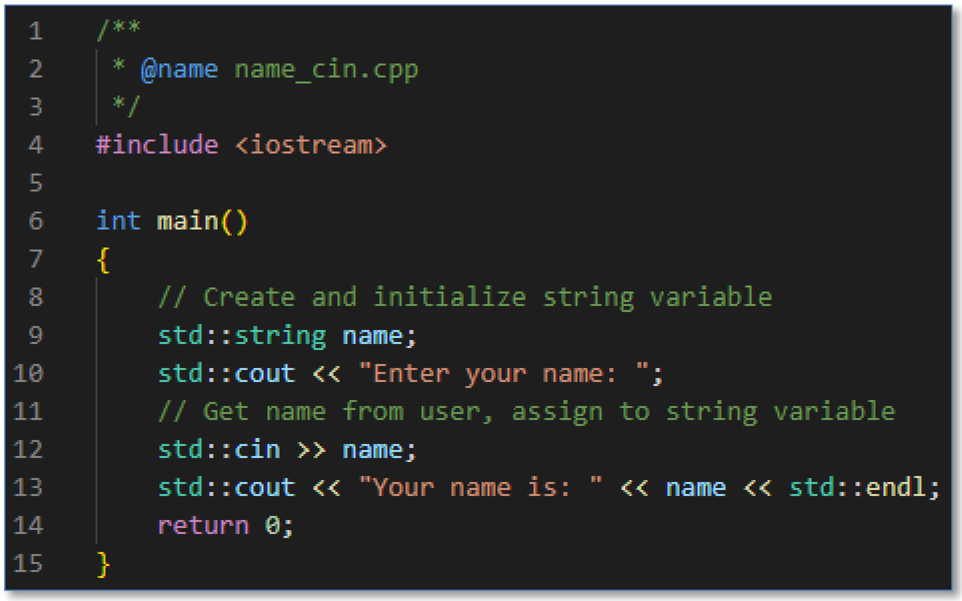
Tutorial 1: getline

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;
```

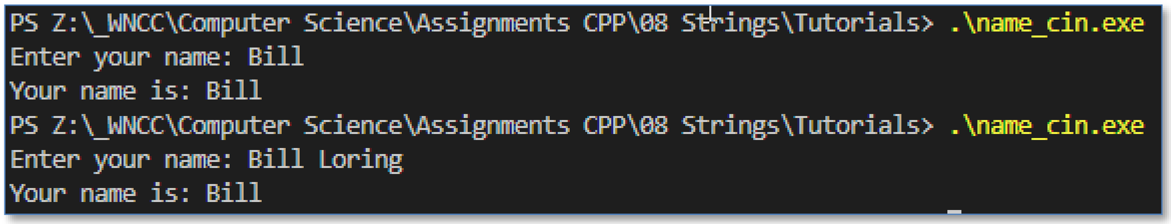
We can input "Mark" or "Twain" but not "Mark Twain" because **cin** cannot input strings that contain embedded spaces.

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



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

int main()
{
    // Create and initialize string variable
    std::string name;
    std::cout << "Enter your name: ";
    // Get name from user, assign to string variable
    std::cin >> name;
    std::cout << "Your name is: " << name << std::endl;
    return 0;
}
```

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
```

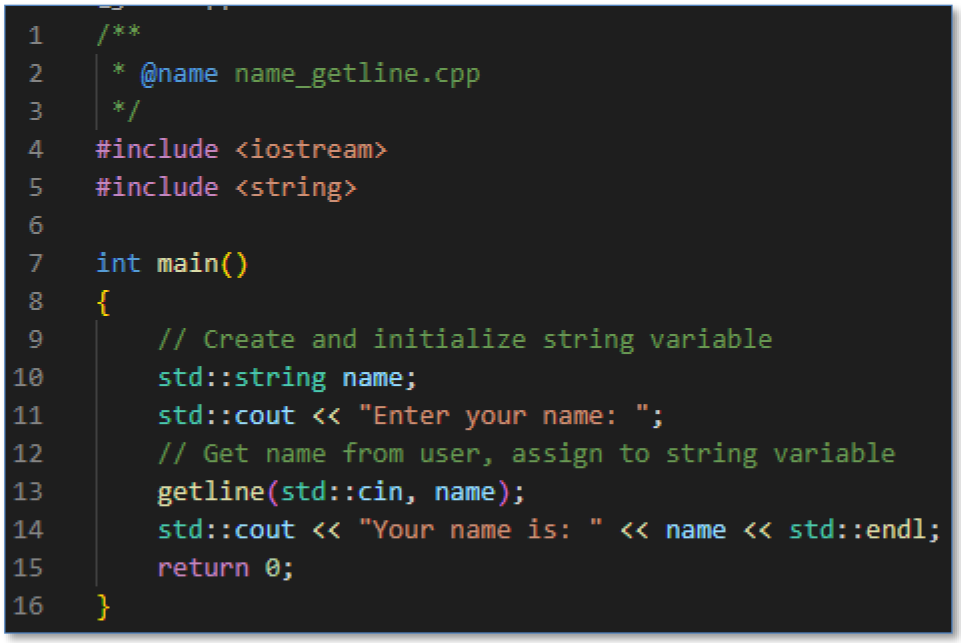
Notice that the first run worked just fine. The second run had an issue, it didn't get the full name. **cin** doesn't think that a white space or \n is a valid character for input. **cin** stops at the space.

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** new line character is input from the Enter/Return key.

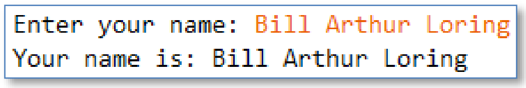
```
getline(cin, variable)
```

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



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

int main()
{
    // Create and initialize string variable
    std::string name;
    std::cout << "Enter your name: ";
    // Get name from user, assign to string variable
    getline(std::cin, name);
    std::cout << "Your name is: " << name << std::endl;
    return 0;
}
```

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)

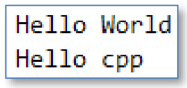
- **.find()** - locates the index of a substring within a string object
- **.substr()** - returns a new string object made of a substring of an existing string object
- **empty()** - returns true if the string contains no characters; returns false if the string contains one or more characters
- **clear()**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

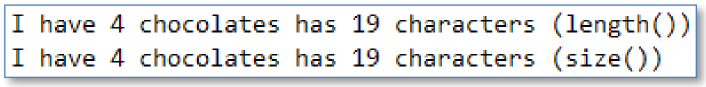
st1 += "cpp"; - We added the string "cpp" to the value of the string variable st1 making its value "Hello cpp".

Length or Size

The **length()** or **size()** function is used to find the length of a string. This are pre-defined functions which belongs 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.

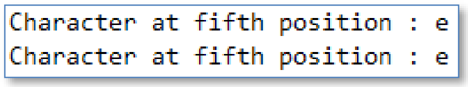
Bracket Operator and at() Function

[] 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:



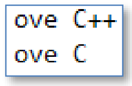
```
Character at fifth position : e
Character at fifth position : e
```

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:



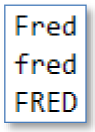
```
ove C++
ove C
```

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;
}
```

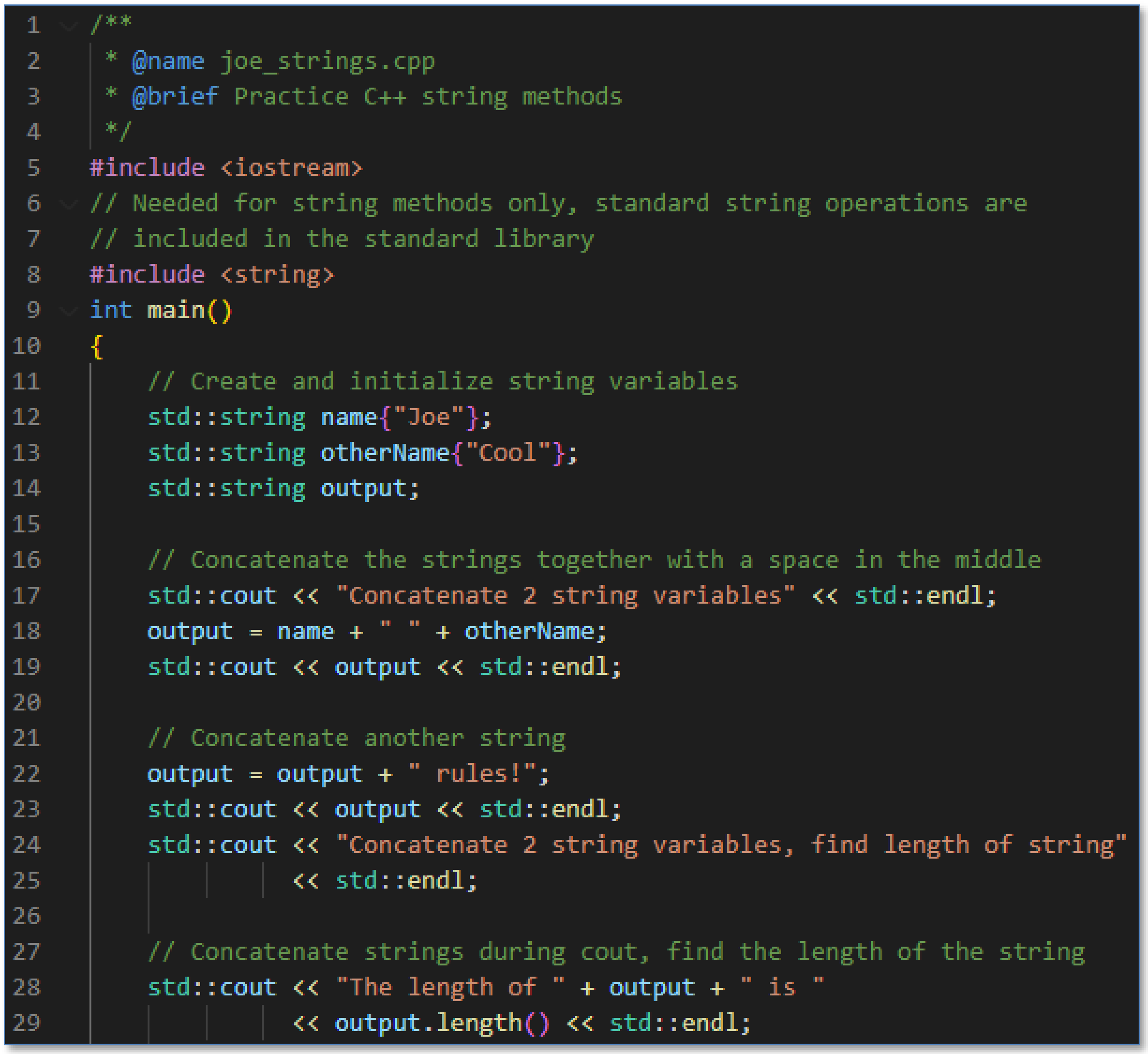
Example run:



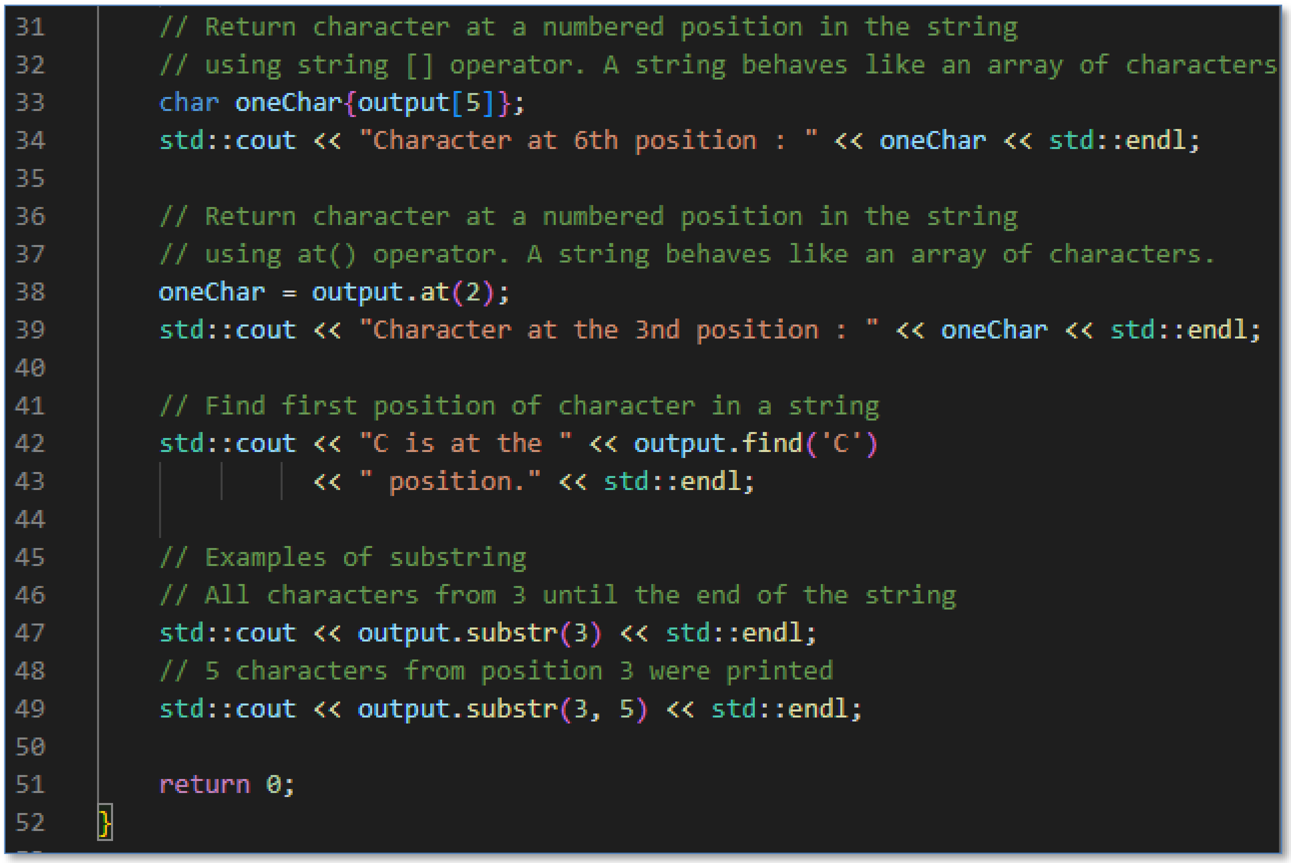
```
Fred
fred
FRED
```

Tutorial 2: String Methods with Joe Cool

The following too cool program, **joe_cool.cpp**, demonstrates several string methods.

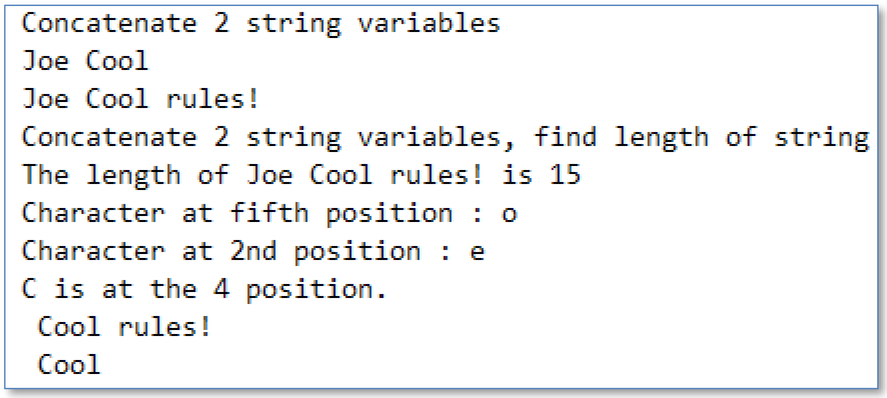


```cpp
/**
 * @name joe_strings.cpp
 * @brief Practice C++ string methods
 */
#include <iostream>
// Needed for string methods only, standard string operations are
// included in the standard library
#include <string>
int main()
{
    // Create and initialize string variables
    std::string name{"Joe"};
    std::string otherName{"Cool"};
    std::string output;

    // Concatenate the strings together with a space in the middle
    std::cout << "Concatenate 2 string variables" << std::endl;
    output = name + " " + otherName;
    std::cout << output << std::endl;

    // Concatenate another string
    output = output + " rules!";
    std::cout << output << std::endl;
    std::cout << "Concatenate 2 string variables, find length of string"
              << std::endl;

    // Concatenate strings during cout, find the length of the string
    std::cout << "The length of " + output + " is "
              << output.length() << std::endl;
```



```cpp
    // Return character at a numbered position in the string
    // using string [] operator. A string behaves like an array of characters
    char oneChar{output[5]};
    std::cout << "Character at 6th position : " << oneChar << std::endl;

    // Return character at a numbered position in the string
    // using at() operator. A string behaves like an array of characters.
    oneChar = output.at(2);
    std::cout << "Character at the 3nd position : " << oneChar << std::endl;

    // Find first position of character in a string
    std::cout << "C is at the " << output.find('C')
              << " position." << std::endl;

    // Examples of substring
    // All characters from 3 until the end of the string
    std::cout << output.substr(3) << std::endl;
    // 5 characters from position 3 were printed
    std::cout << output.substr(3, 5) << std::endl;

    return 0;
}
```

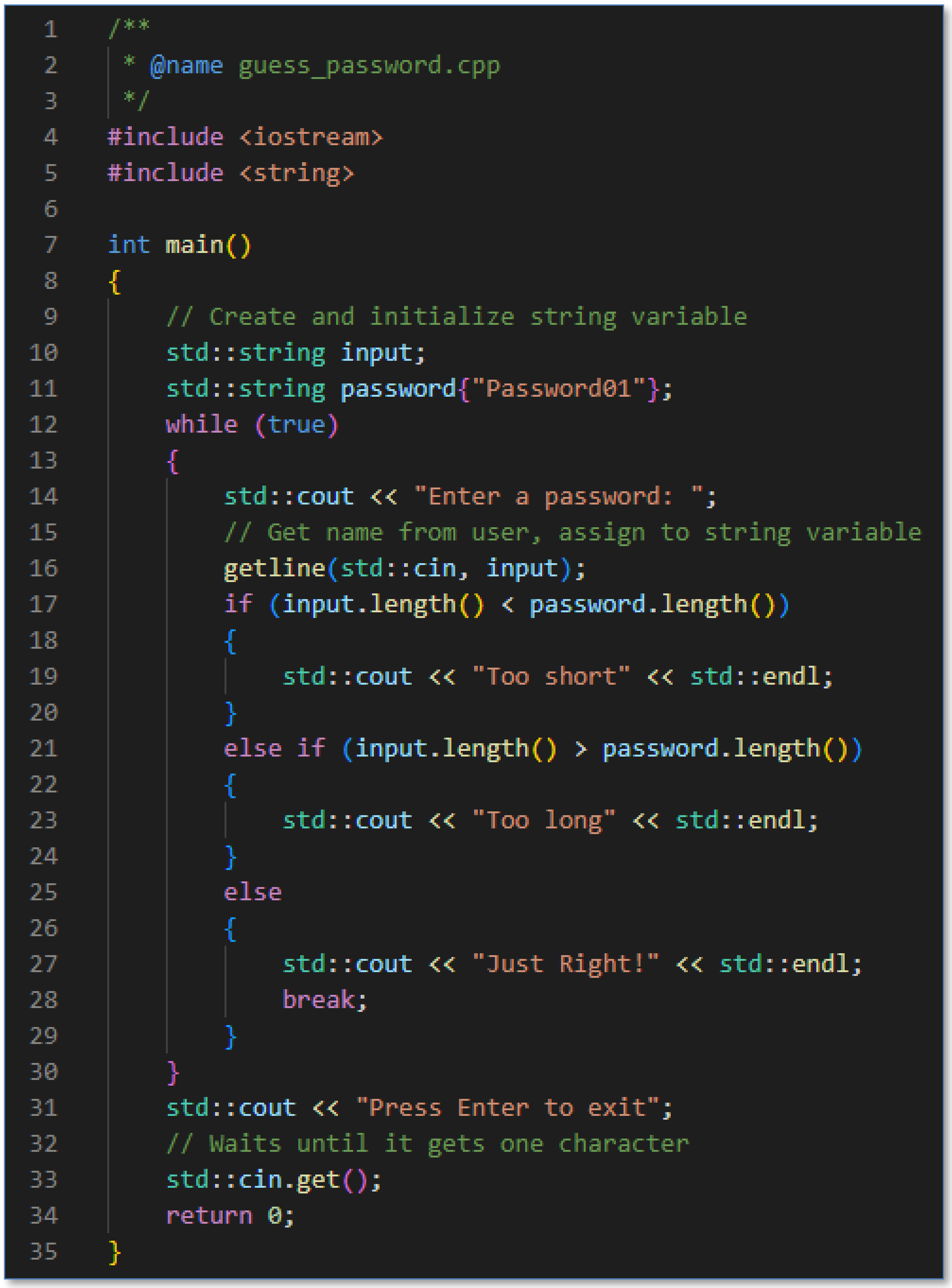
Example run:



```
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.



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

int main()
{
    // Create and initialize string variable
    std::string input;
    std::string password{"Password01"};
    while (true)
    {
        std::cout << "Enter a password: ";
        // Get name from user, assign to string variable
        getline(std::cin, input);
        if (input.length() < password.length())
        {
            std::cout << "Too short" << std::endl;
        }
        else if (input.length() > password.length())
        {
            std::cout << "Too long" << std::endl;
        }
        else
        {
            std::cout << "Just Right!" << std::endl;
            break;
        }
    }
    std::cout << "Press Enter to exit";
    // Waits until it gets one character
    std::cin.get();
    return 0;
}
```

Example run:

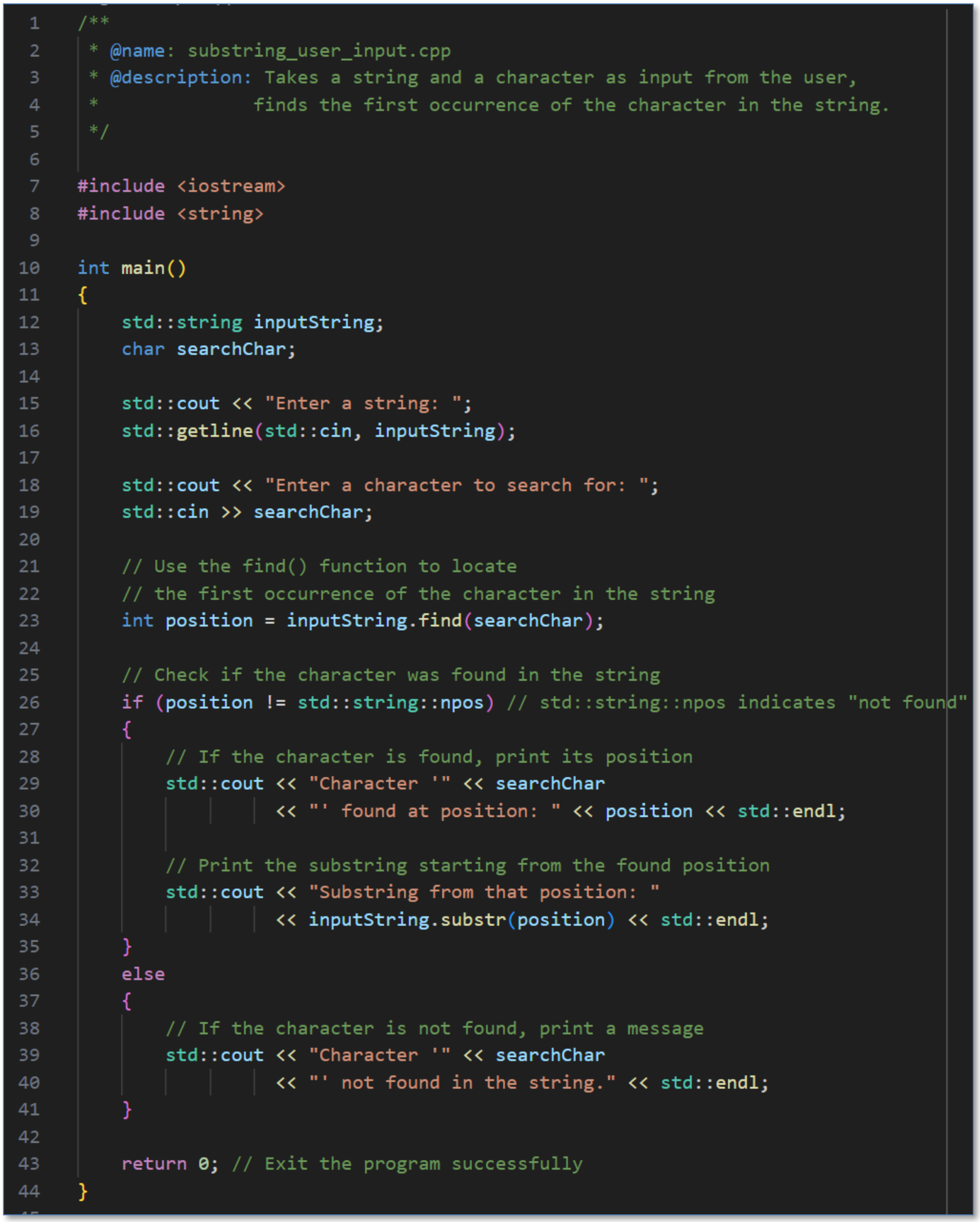
```
Enter a password: pass
Too short
Enter a password: passwordkkkkd
Too long
Enter a password: password01
Just Right!
```

Tutorial 4: Substring User Input

This program takes a string and a character as input from the user, finds the first occurrence of the character in the string, and performs specific actions based on whether the character is found or not.

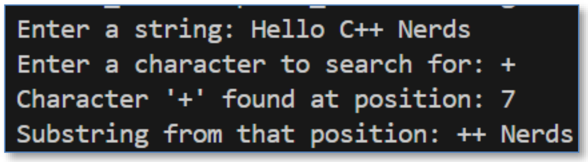
How It Works:

1. **Input**:
   - The user is prompted to enter a string (inputString).
   - The user is then asked to enter a single character (searchChar) to search for in the string.
2. **Finding the Character**:
   - The program uses the **find()** function of the std::string class to locate the first occurrence of searchChar in inputString.
   - If the character is found, **find()** returns its position (index). If not, it returns std::string::npos (indicating "not found").
3. **Output**:
   - If the character is found:
     - The program prints the position (index) of the character.
     - It also prints the substring starting from the found position to the end of the string using s**ubstr()**.
   - If the character is not found:
     - The program prints a message indicating that the character is not in the string.
4. **Program Termination**:
   - The program ends successfully with return 0.



```cpp
/**
 * @name: substring_user_input.cpp
 * @description: Takes a string and a character as input from the user,
 *               finds the first occurrence of the character in the string.
 */

#include <iostream>
#include <string>

int main()
{
    std::string inputString;
    char searchChar;

    std::cout << "Enter a string: ";
    std::getline(std::cin, inputString);

    std::cout << "Enter a character to search for: ";
    std::cin >> searchChar;

    // Use the find() function to locate
    // the first occurrence of the character in the string
    int position = inputString.find(searchChar);

    // Check if the character was found in the string
    if (position != std::string::npos) // std::string::npos indicates "not found"
    {
        // If the character is found, print its position
        std::cout << "Character '" << searchChar
                  << "' found at position: " << position << std::endl;

        // Print the substring starting from the found position
        std::cout << "Substring from that position: "
                  << inputString.substr(position) << std::endl;
    }
    else
    {
        // If the character is not found, print a message
        std::cout << "Character '" << searchChar
                  << "' not found in the string." << std::endl;
    }

    return 0; // Exit the program successfully
}
```

Example run:



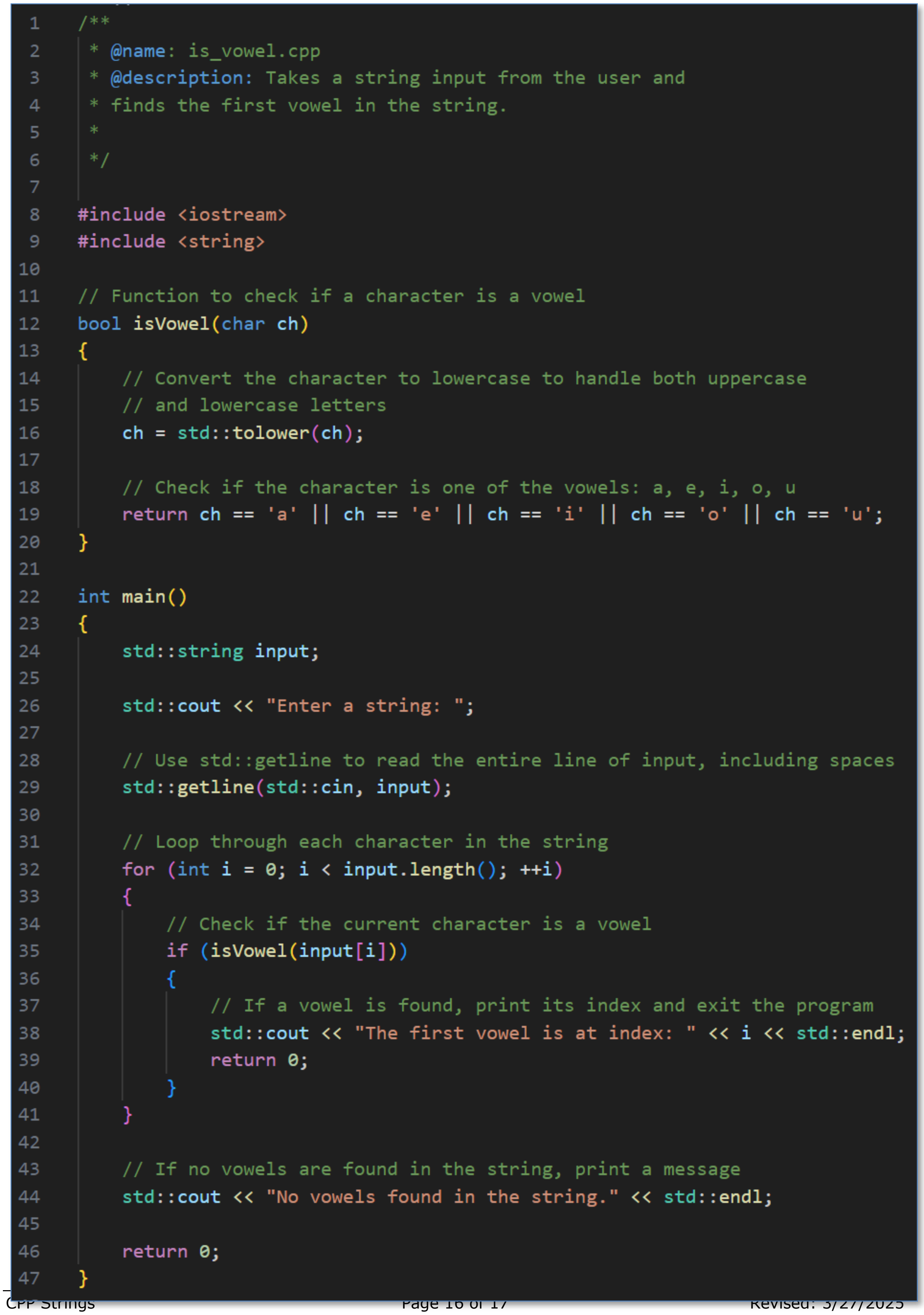
```
Enter a string: Hello C++ Nerds
Enter a character to search for: +
Character '+' found at position: 7
Substring from that position: ++ Nerds
```

Tutorial 5: Is Vowel

This program takes a string as input from the user and finds the first vowel in the string. If a vowel is found, it prints its index; otherwise, it informs the user that no vowels were found.

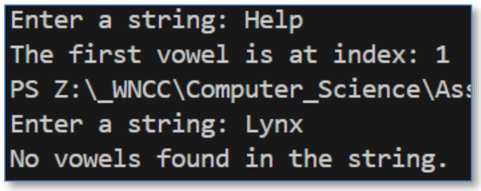
How It Works:

1. **isVowel() Function:**
    - This function checks if a given character is a vowel (a, e, i, o, u), regardless of case.
    - It converts the character to lowercase using **std::tolower()** to handle both uppercase and lowercase letters.
2. **User Input**:
    - The program prompts the user to enter a string.
    - It uses **std::getline** to read the entire input, including spaces.
3. **Finding the First Vowel**:
    - The program iterates through each character in the string using a for loop.
    - For each character, it calls the **isVowel** function to check if it is a vowel.
    - If a vowel is found, the program prints its index and exits immediately using return 0.
4. **No Vowels Case**:
    - If the loop completes without finding a vowel, the program prints a message indicating that no vowels were found.
5. **Program Termination**:
    - The program ends successfully with return 0.



```cpp
/**
 * @name: is_vowel.cpp
 * @description: Takes a string input from the user and
 * finds the first vowel in the string.
 *
 */

#include <iostream>
#include <string>

// Function to check if a character is a vowel
bool isVowel(char ch)
{
    // Convert the character to lowercase to handle both uppercase
    // and lowercase letters
    ch = std::tolower(ch);

    // Check if the character is one of the vowels: a, e, i, o, u
    return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
}

int main()
{
    std::string input;

    std::cout << "Enter a string: ";

    // Use std::getline to read the entire line of input, including spaces
    std::getline(std::cin, input);

    // Loop through each character in the string
    for (int i = 0; i < input.length(); ++i)
    {
        // Check if the current character is a vowel
        if (isVowel(input[i]))
        {
            // If a vowel is found, print its index and exit the program
            std::cout << "The first vowel is at index: " << i << std::endl;
            return 0;
        }
    }

    // If no vowels are found in the string, print a message
    std::cout << "No vowels found in the string." << std::endl;

    return 0;
}
```

Example run:



```
Enter a string: Help
The first vowel is at index: 1
PS Z:\_WNCC\Computer_Science\As
Enter a string: Lynx
No vowels found in the string.
```

Assignment 1: C++ String Practice

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

1. Accept a string from the user, even if it has a space.
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