CS12: C++ Strings

Fundamentals and Implementation

Mr. Gullo

Computer Science Department

March, 2025

Table of Contents

Learning Objectives

By the end of this lesson, you will be able to:

- Understand what strings are in C++ and how they are stored in memory
- Declare, initialize, and manipulate string objects
- Access and modify individual characters in strings
- Implement common string operations such as:
  - Counting specific characters
  - Converting case (uppercase/lowercase)
  - Checking for palindromes
  - Finding substrings
- Write functions that process and transform strings

Why Strings Matter

- Strings are fundamental for handling text data
- Used in almost every real-world application:
  - User interfaces
  - File I/O
  - Data processing
  - Network communication
- Understanding strings is essential for programming
- Many algorithms and problems involve string manipulation

String Declaration and Initialization

In C++, strings are objects of the `std::string` class:

```cpp
#include <string>  // Required for using strings
using namespace std;  // Avoid in header files!

// Different ways to declare strings:
string str1;              // Empty string
string str2 = "Hello";    // Initialize with value
string str3("World");     // Constructor initialization
string str4 = str2;       // Copy from another string
```

Important Notes

- Always include the string header
- In header files, use `std::string` instead of `using namespace std`
- Strings are mutable (can be changed)

Basic String Operations

```cpp
// Example 01: Basic string
string str1 = "Hello World\n";
cout << str1;  // Outputs: Hello World

// Example 02: Copying a string
string str2 = str1;
cout << str2;  // Outputs: Hello World

// Example 03: Concatenation
string str3 = str1 + ", happy Monday!" + "\n";
cout << str3;  // Outputs: Hello World, happy Monday!

// Example 04: String length
int str3Len = str3.size();  // or str3.length()
cout << "str3 length: " << str3Len << endl;
```

String Input

Two main ways to read strings:

```
// Using cin (stops at whitespace)
string name;
cout << "Enter your name: ";
cin >> name;  // Only reads until first space

// Using getline (reads entire line)
string fullText;
cout << "Enter a line of text: ";
getline(cin, fullText);  // Reads until newline

// Important: When mixing cin >> and getline()
cin >> name;
cin.ignore();  // Clear the newline character
getline(cin, fullText);
```

Common Pitfall

Always use `cin.ignore()` when switching from `cin >>` to `getline()`

Accessing Characters

Strings can be accessed character by character:

```
string text = "Hello";

// Access individual characters with indexing
char firstChar = text[0];  // 'H'
char lastChar = text[4];   // 'o'

// Modify characters
text[0] = 'J';  // Changes to "Jello"

// Iterate through all characters
for(int i = 0; i < text.size(); i++) {
    cout << text[i];  // Print each character
}
```

Remember

- Indexing starts at 0
- Be careful not to access beyond string length

Understanding ASCII Values

Characters in C++ are represented by ASCII values:

```
char c = 'A';
int value = c;  // 65

// Uppercase letters: 65-90 (A-Z)
// Lowercase letters: 97-122 (a-z)
// Digits: 48-57 (0-9)

// Difference between cases
int diff = 'a' - 'A';  // 32
```

Converting Case

- To uppercase: subtract 32 from lowercase
- To lowercase: add 32 to uppercase
- Or use library functions: `toupper()`, `tolower()`

Case Conversion Example

Implementation of case conversion functions:

```
// Convert to uppercase (returns new string)
string toUpper(string s) {
    int conversion = 'a' - 'A';
    for(int i = 0; i < s.length(); i++) {
        if(s[i] >= 'a' && s[i] <= 'z') {
            s[i] = s[i] - conversion;
        }    }
    return s;
}
// Convert to lowercase (modifies original)
void toLower(string &s) {
    int conversion = 'a' - 'A';
    for(int i = 0; i < s.length(); i++) {
        if(s[i] >= 'A' && s[i] <= 'Z') {
            s[i] = s[i] + conversion;
        }    }}
```

String Reversal

Implementing the reverseString function:

```
// Returns s, but in reverse order
string reverseString(string s) {
    string reversed = "";
    // Method 1: Build a new string from back to front
    for(int i = s.length() - 1; i >= 0; i--) {
        reversed += s[i];
    }
    /* Method 2: In-place reversal
    string reversed = s;
    int n = s.length();
    for(int i = 0; i < n/2; i++) {
        char temp = reversed[i];
        reversed[i] = reversed[n-1-i];
        reversed[n-1-i] = temp;
    }
    */
    return reversed;
}
```

Finding Substrings

Implementing the isSubstring function:

```
// Returns true if s_full contains the string s_sub
bool isSubstring(string s_full, string s_sub) {
    // Method 1: Using string's find method
    if(s_full.find(s_sub) != string::npos) {
        return true;
    }
    return false;
        /* Method 2: Manual implementation
    for(int i = 0; i <= s_full.length() - s_sub.length
        (); i++) {
        bool match = true;
        for(int j = 0; j < s_sub.length(); j++) {
            if(s_full[i+j] != s_sub[j]) {
                match = false;
                break;
            }           }
        if(match) return true;
    }
```

String Method Reference

Common string methods:

- `s.length()` / `s.size()` - Returns string length
- `s.empty()` - Checks if string is empty
- `s.clear()` - Clears the string
- `s.substr(pos, len)` - Returns substring
- `s.find(str)` - Finds position of substring
- `s.replace(pos, len, str)` - Replaces part of string
- `s.insert(pos, str)` - Inserts at position
- `s.erase(pos, len)` - Erases characters
- `s.append(str)` - Appends to end
- `s.compare(str)` - Compares strings

Note

The `string` class has many more methods. Check the C++ documentation for details.

Capitalizing Words

Implementing the capitalizeWords function:

```
// Capitalizes the first letter of each word
string capitalizeWords(string s) {
    string result = s;
    // Capitalize first character if it's a letter
    if(!result.empty() && isalpha(result[0])) {
        result[0] = toupper(result[0]);
    }
    // Check each character
    for(int i = 1; i < result.length(); i++) {
        // If previous character is a space, hyphen,
            or period
        if(result[i-1] == ' ' || result[i-1] == '-' ||
            result[i-1] == '.') {
            if(isalpha(result[i])) {
                result[i] = toupper(result[i]);
            }        }}
        return result;}
```

Summary

- C++ strings are objects that store sequences of characters
- String operations include:
  - Declaration and initialization
  - Input/output
  - Character access and manipulation
  - Finding and modifying substrings
- Implementing string functions requires:
  - Understanding character representation (ASCII)
  - Iteration through characters
  - String manipulation techniques
- Practice with the provided header file to strengthen your skills

Practice Assignment

Implement all functions in the provided header file:

- `myName()` - Returns your name
- `countChar()` - Counts occurrences of a character
- `countVowels()` - Counts vowels in a string
- `countNumbers()` - Counts numeric characters
- `longestWord()` - Finds length of longest word
- `capitalizeWords()` - Capitalizes first letter of each word
- `changeCase()` - Inverts case of each letter
- `reverseString()` - Reverses a string
- `isPalindrome()` - Checks if string is a palindrome
- `isSubstring()` - Checks if string contains substring

[Submit through schoology by the due date]