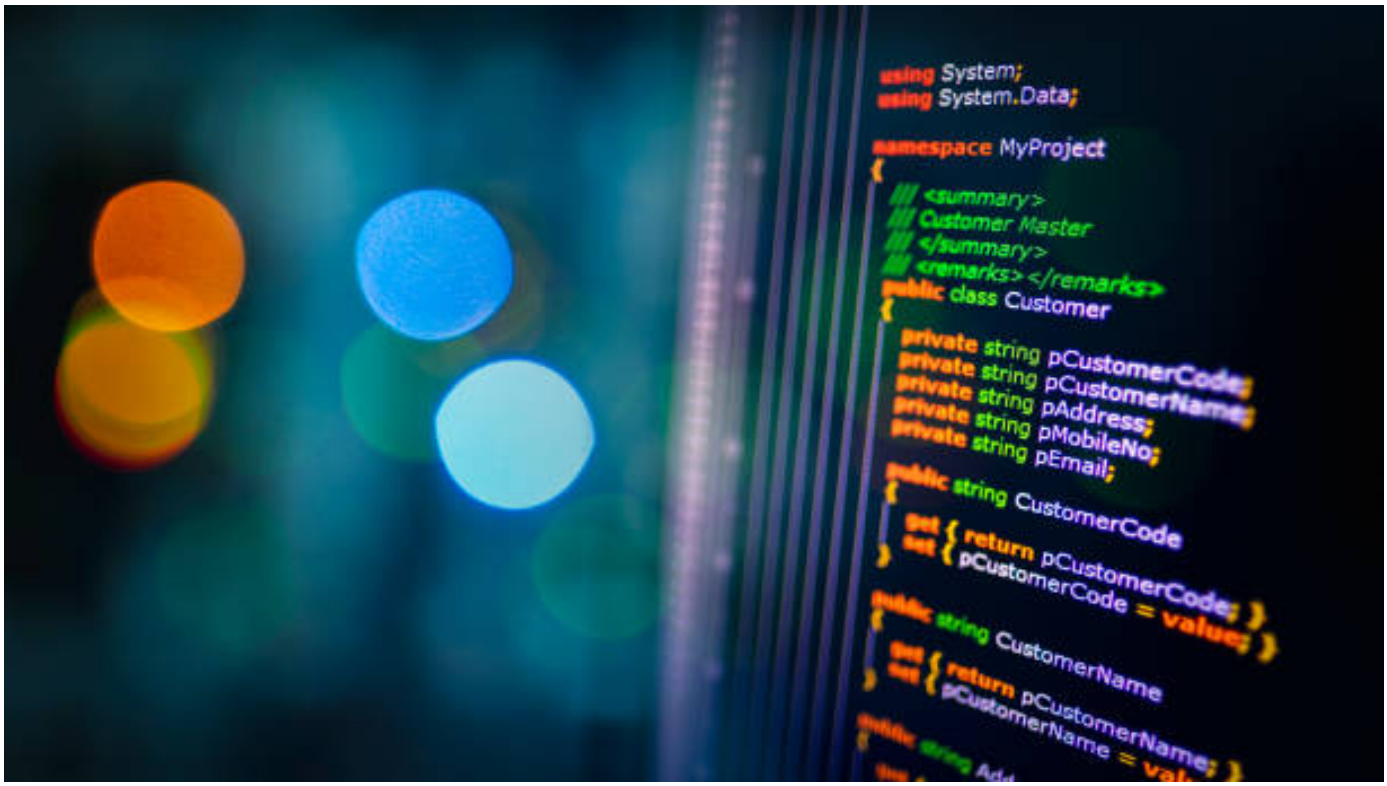
Object Oriented Programming



Assignment Title: Object-Oriented Custom String Class

Section: E

Fatima Ishtiaq, 23i-0696

Table of content

Introduction

The **Advanced String Manipulation Library** is a custom C++ library designed to provide a comprehensive set of tools for string operations. It introduces an enhanced `String` class that expands the capabilities of traditional string handling in C++ by incorporating features like operator overloading, custom constructors, and intuitive functionality for common string operations such as concatenation, substring removal, and character search.

This project aims to deliver a reusable, efficient, and user-friendly library for developers who require advanced string manipulation features in their applications.

Features

The library is designed with several key features:

1. **Dynamic Memory Allocation:** The `String` class manages memory dynamically to support variable-length strings.
2. **Operator Overloading:**
    - Addition (`+`) for concatenation.
    - Subtraction (`-`) for removing substrings or characters.
    - Brackets (`[]`) for accessing individual characters.
    - Parentheses (`()`) for finding character or substring positions.
3. **Equality Check:** Overloaded `==` operator to compare strings.
4. **Type Casting:** Enables conversion of `String` objects to integers to retrieve their length.
5. **Custom Constructors:**
    - Default constructor.
    - Constructor for initializing from `char*`.
    - Copy constructor.
    - Constructor for fixed-length empty strings.
6. **I/O Operations:** Supports `cin` and `cout` for user input and output.

Implementation Details

Header File: `String.h`

The header file declares the `String` class, including its attributes and methods. It also defines friend functions for I/O operations and operators. Key methods include:

- `char* getdata()`: Access the internal string data.
- Overloaded operators (`+`, `-`, `[]`, `==`, `!`, etc.).
- Constructor and destructor definitions.

Source File: `String.cpp`

The source file provides the implementation of all declared methods. It ensures:

- Safe handling of memory allocation and deallocation.
- Error handling for edge cases such as out-of-bound indices or null strings.
- Efficient concatenation and substring removal operations.

Unit Testing: `test.cpp`

The testing file ensures the correctness of the library's functionality using the **Google Test Framework**. Each test suite evaluates specific operations, such as:

1. Constructor functionality.
2. Addition and subtraction operators.
3. Bracket and parenthesis operators.
4. Input/output handling.
5. String equality and emptiness checks.

Applications

The library can be utilized in various domains, including:

- Text parsing and manipulation.
- Data cleaning and formatting in software applications.
- Enhancing built-in C++ string operations in competitive programming or academic projects.

Challenges Faced

1. **Memory Management:** Implementing efficient dynamic memory allocation to prevent memory leaks.
2. **Edge Cases:** Handling edge cases for null or empty strings in operations like subtraction and equality checks.
3. **Unit Testing:** Ensuring comprehensive test coverage for all functionalities.

Conclusion

The **Advanced String Manipulation Library** demonstrates the power of object-oriented programming in C++ by extending the functionality of strings through custom implementation. With its modular design and robust testing, the library is a valuable tool for developers needing enhanced string operations.

Future Improvements

1. Optimize performance for large-scale string operations.
2. Expand functionality with regex-based pattern matching and advanced formatting options.

References

- C++ Standard Library Documentation.
- Google Test Framework Documentation.