МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

**КУРСОВАЯ РАБОТА**

по дисциплине «Объектно-ориентированное программирование»

Тема: «Объект строка»

Выполнил: студент группы ИС-142

Григорьев Ю.В.

Проверил: доцент кафедры ПМиК

Ситняковская Е.И.

Новосибирск – 2022

**Содержание**

1. Постановка задачи
2. Технологии ООП
3. Структура классов
4. Программная реализация
5. Результат работы
6. Заключение
7. Используемые источники
8. Приложение. Листинг

**Постановка задачи**

В реализуемом задании требуется создать собственный класс для работы с типом данных «строка» (инкапсулирующий методы для работы со строками) с использованием объектно-ориентированных технологий.

Описания объектов и методов необходимо оформить в отдельном модуле.

Необходимый минимум содержания работы:

1. Инкапсуляция (все поля данных не доступны из внешних функций)
2. Наследование (минимум 3 класса, один из которых - абстрактный)
3. Полиморфизм
4. Конструкторы, Перегрузка конструкторов
5. Списки инициализации

Также желательно использование как минимум ещё 2 технологий ООП (статические элементы, дружественные функции, классы, виртуальные функции, шаблоны, множественное наследование, массивы указателей на объекты, конструкторы копирования, параметры по умолчанию, использование объектов в качестве аргументов или возвращаемых значений).

Отчет должен содержать:

1. Постановка задачи (вариант курсовой работы)
2. Описание алгоритма основной программы (желательно)
3. Текст программы и модуля (если классы оформлены в отдельном модуле)
4. Скриншоты работы программы (в нескольких экземплярах).

**Технологии ООП**

Работа реализована с помощью следующих технологий/методологий объектно-ориентированного программирования:

1. Инкапсуляция (все поля данных не доступны из внешних функций)
2. Наследование (3 класса, один из которых абстрактный)
3. Полиморфизм
4. Конструкторы, перегрузка конструкторов
5. Списки инициализации
6. Параметры по умолчанию
7. Статические элементы (функции)
8. Дружественные функции
9. Виртуальные функции
10. Использование объектов в качестве аргументов и возвращаемых значений функций

**Структура классов**

Структура классов и их назначение показаны на следующей диаграмме:

Класс **chars**

Статические функции для работы с массивами символов:

**scpy() –** копирование**, slen() –** нахождение длины массива символов**, scat() –** конкатенация двух массивов**, scmp() –** сравнение массивов символов лексикографически**, sdup() –** создание дубликата массива символов

Класс **abstract**

Виртуальные функции (прототипы):

**get(), set(), length()**

наследуется

Класс **string**

Поле для хранения массива символов **str**

Дружественные функции операторов

**(>>, <<, +, ==)**

Оператор присвоения (**=**)

Конструкторы с перегрузками и списками инициализации **string(),** **get(), set()**

Деструктор **~string()**

Функция **append()** для добавления в конец строки другую строку/массив символов

Функция **length()** для возвращения длины строки

применяется

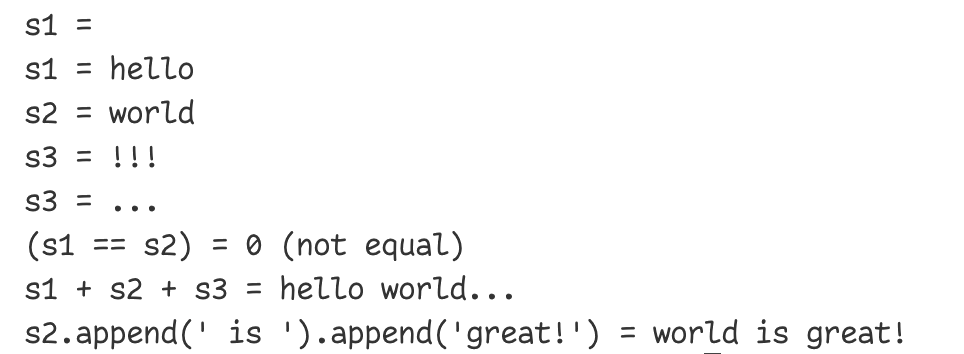
в функциях

**Программная реализация**

Для того, чтобы

**Результат работы**

Результаты работы можно увидеть на следующем скриншоте:



**Заключение**

Выполнив данную работу, я смог разработать собственную реализацию класса «строка» и методы для работы с ней, применив и закрепив знания из изученного курса объектно-ориентированного программирования.

**Используемые источники**

1. Лафоре Р. "Объектно-ориентированное программирование в С++"

2. Бьерн Страуструп. Язык программирования С++

3. Бертран Мейер «Почувствуй класс. Учимся программировать хорошо с объектами и контрактами»

4. Гради Буч «Объектно-ориентированный анализ и проектирование с примерами приложений»

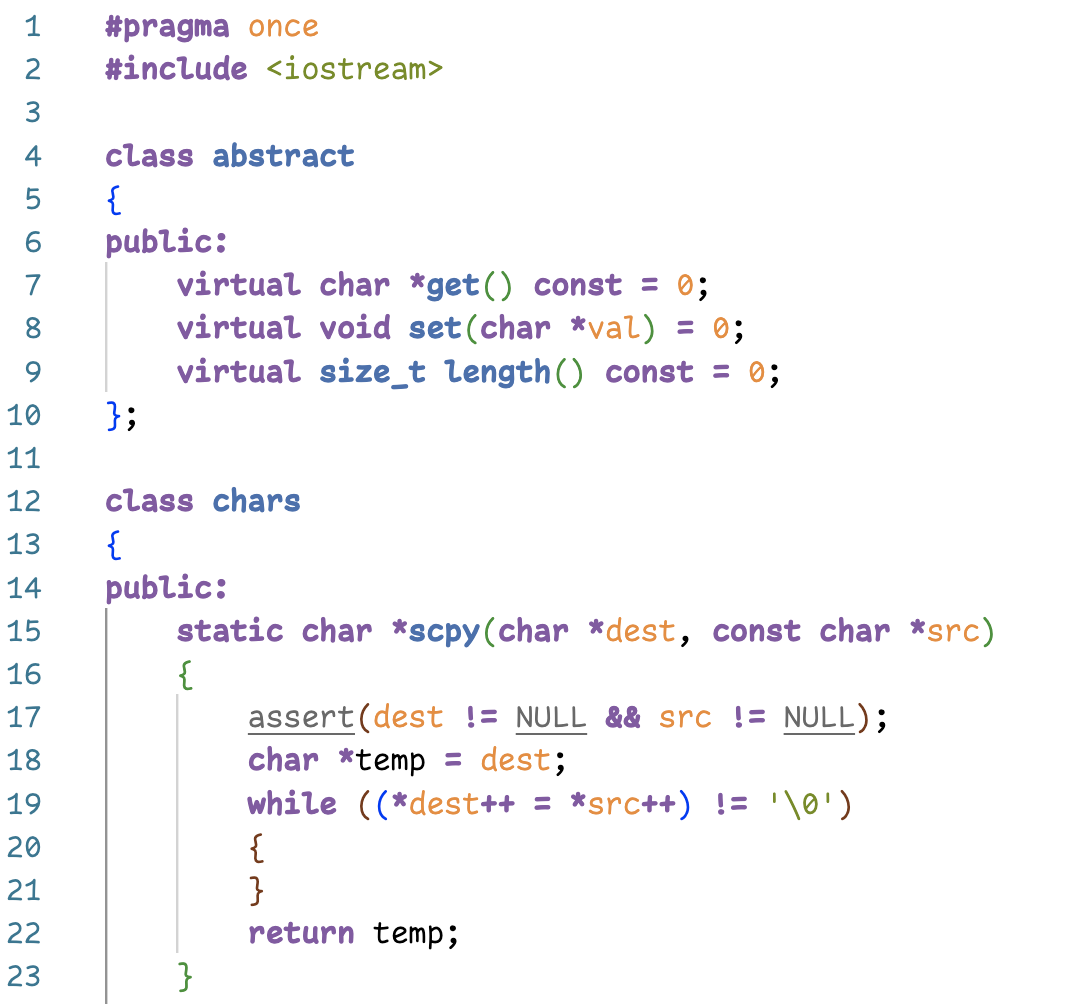
5. Мэтт Вайсфельд «Объектно-ориентированное мышление»

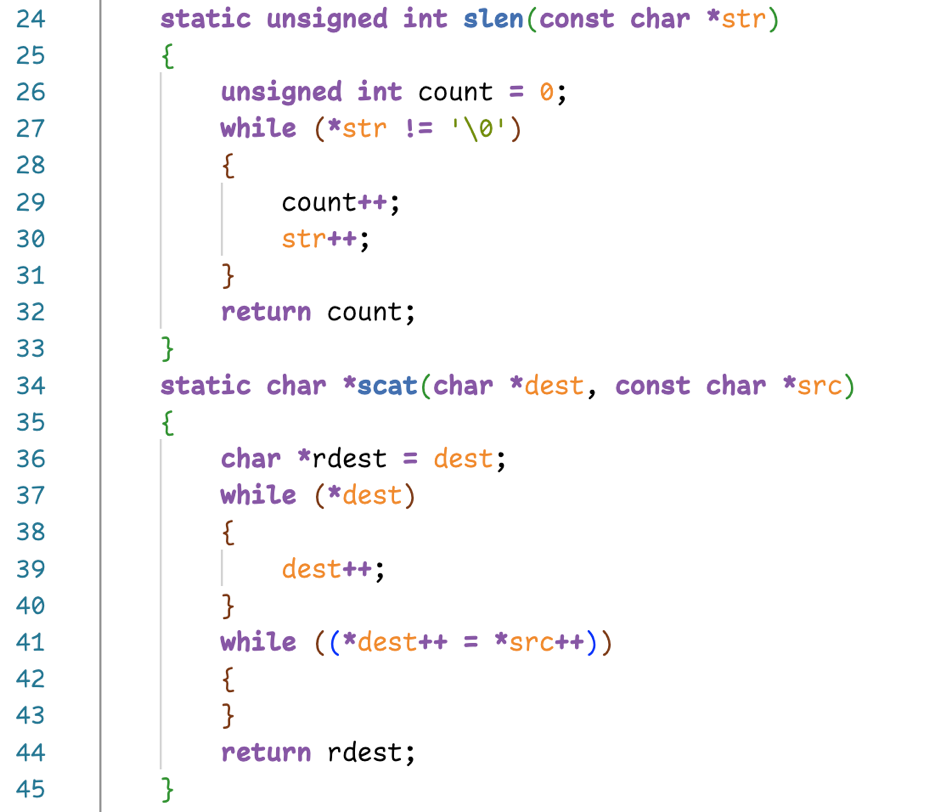
**Приложение. Листинг**

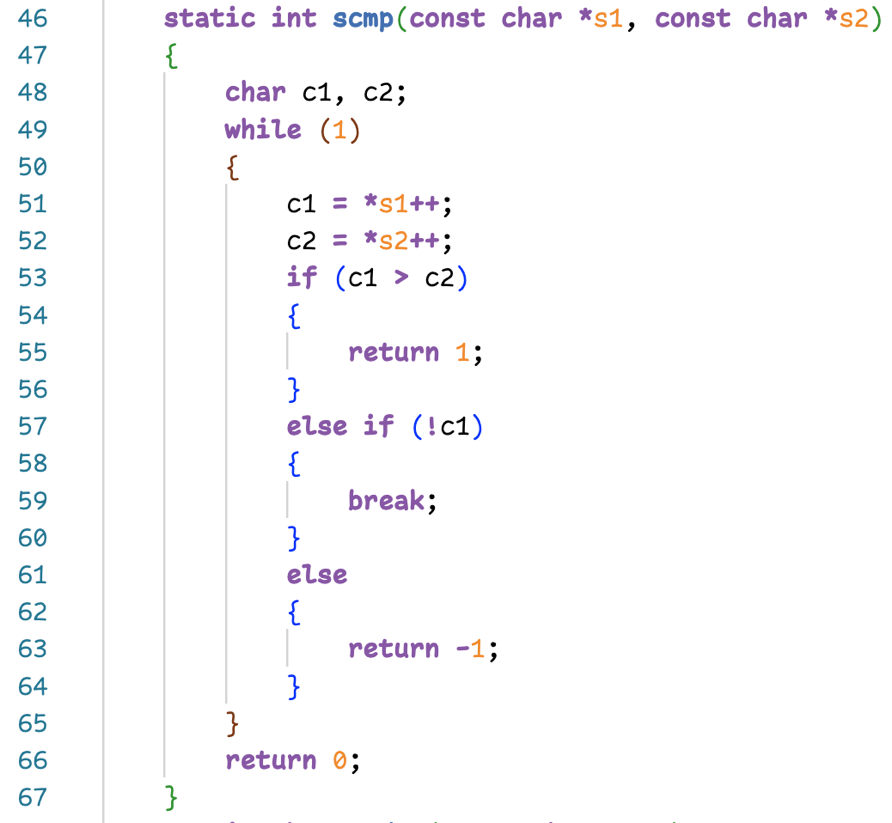
main.cpp

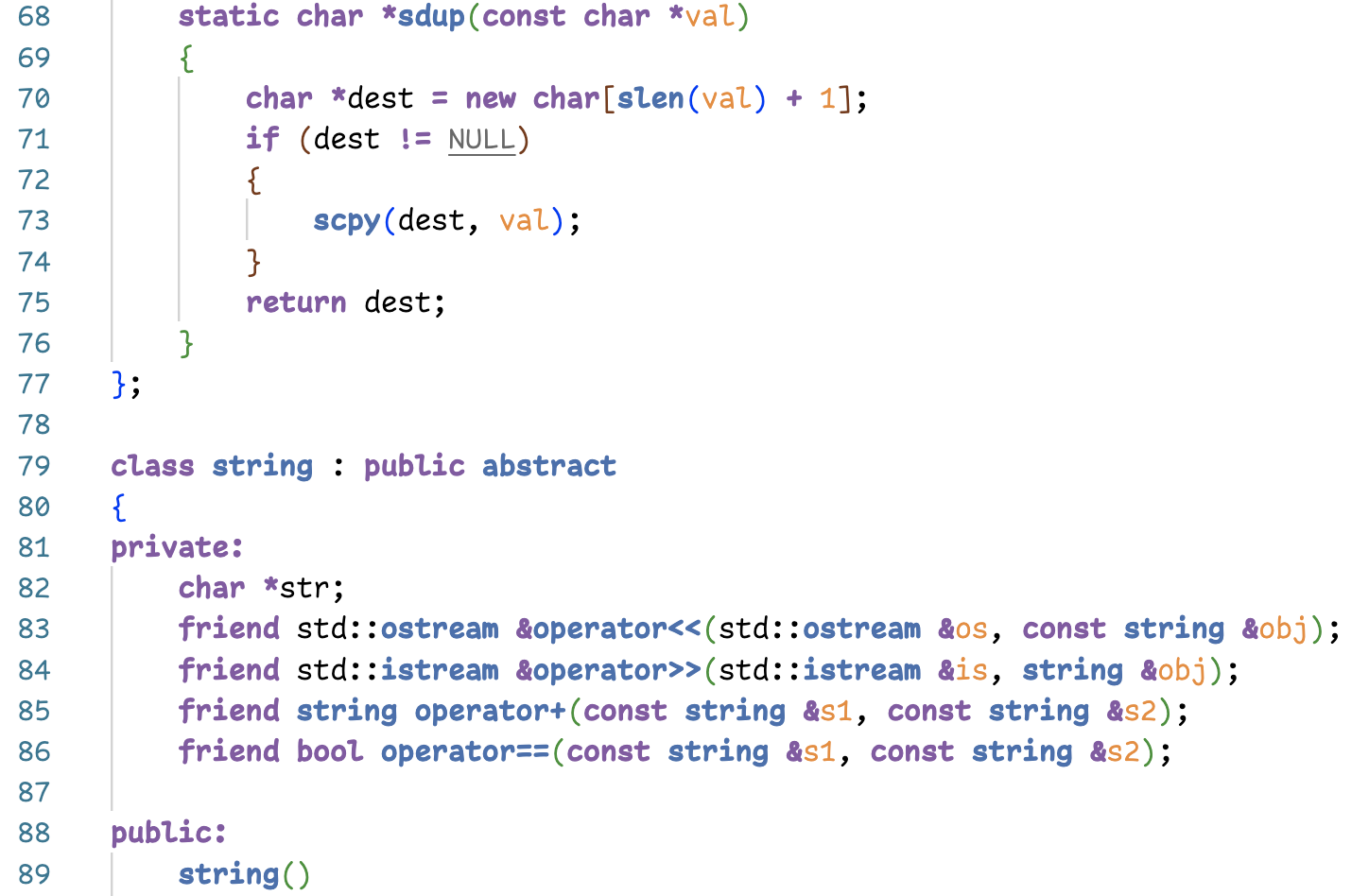


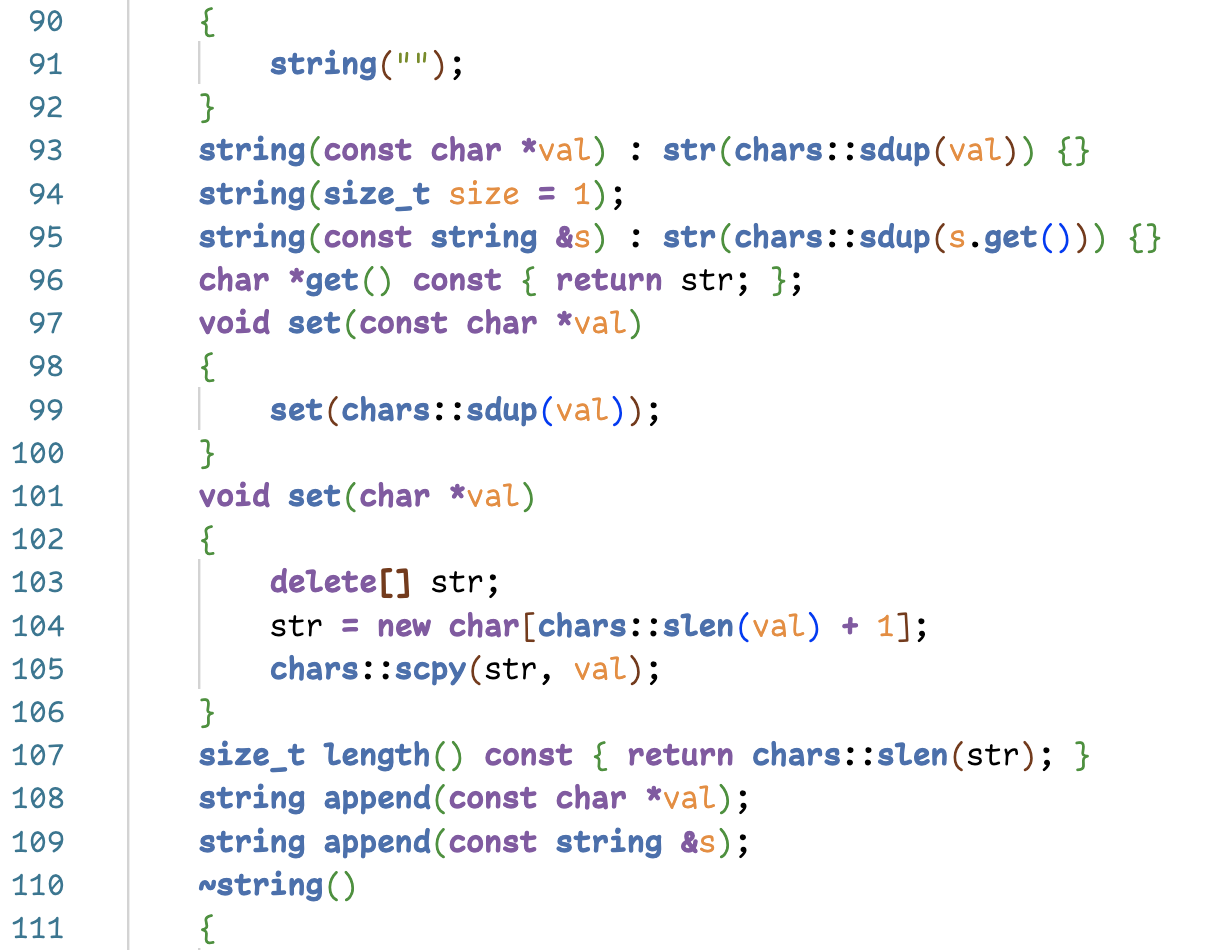
string.hpp

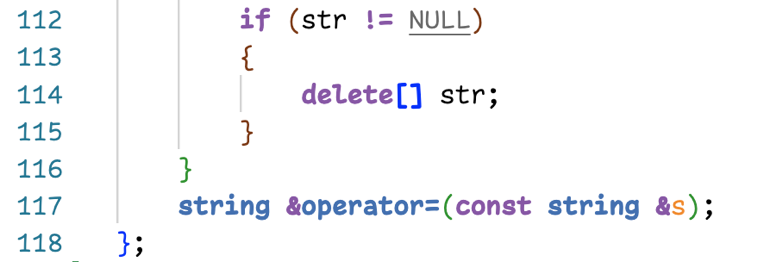




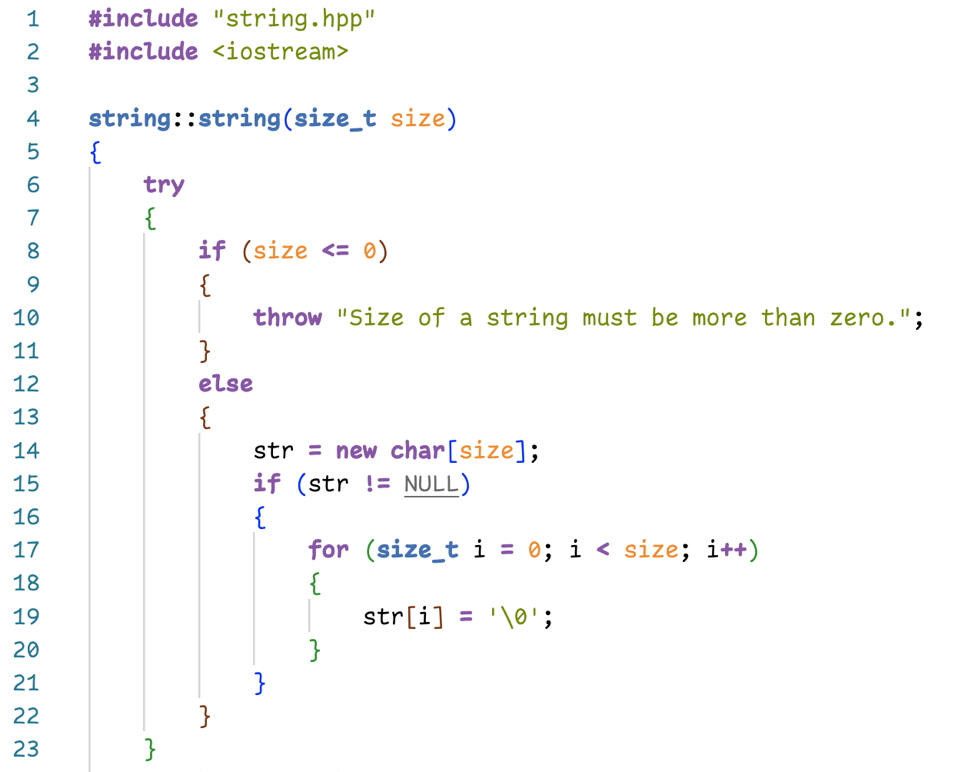


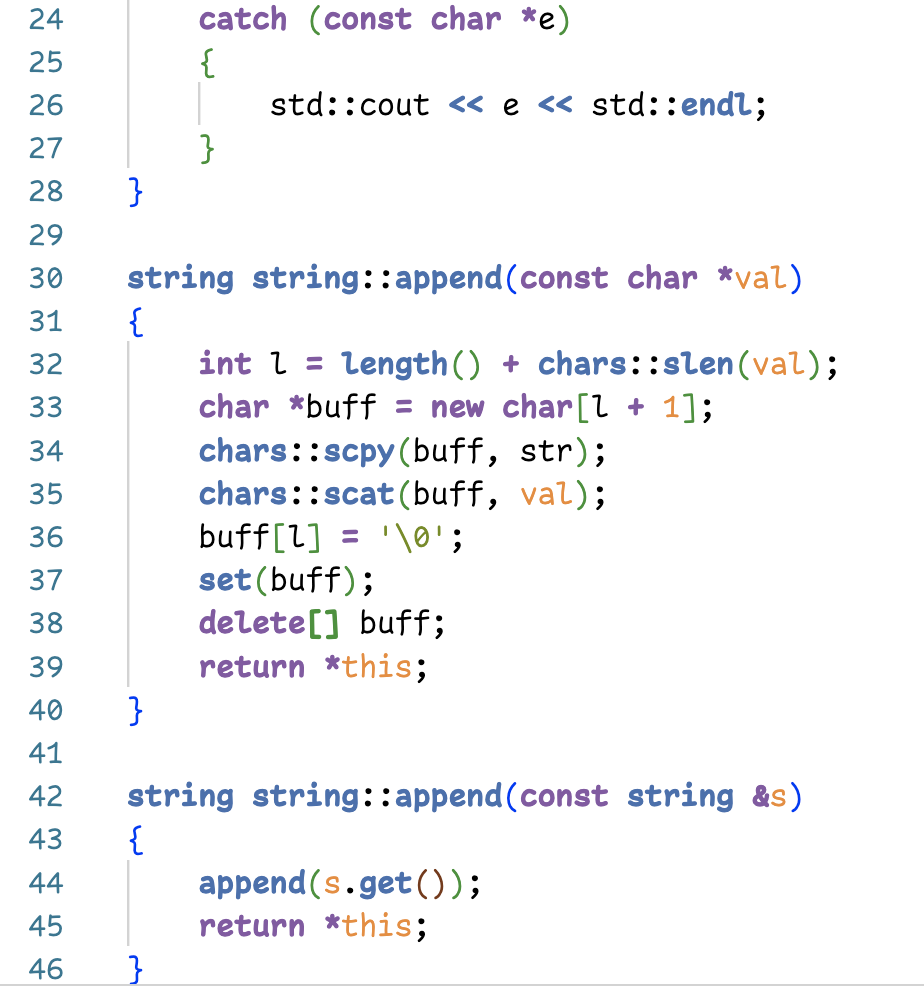


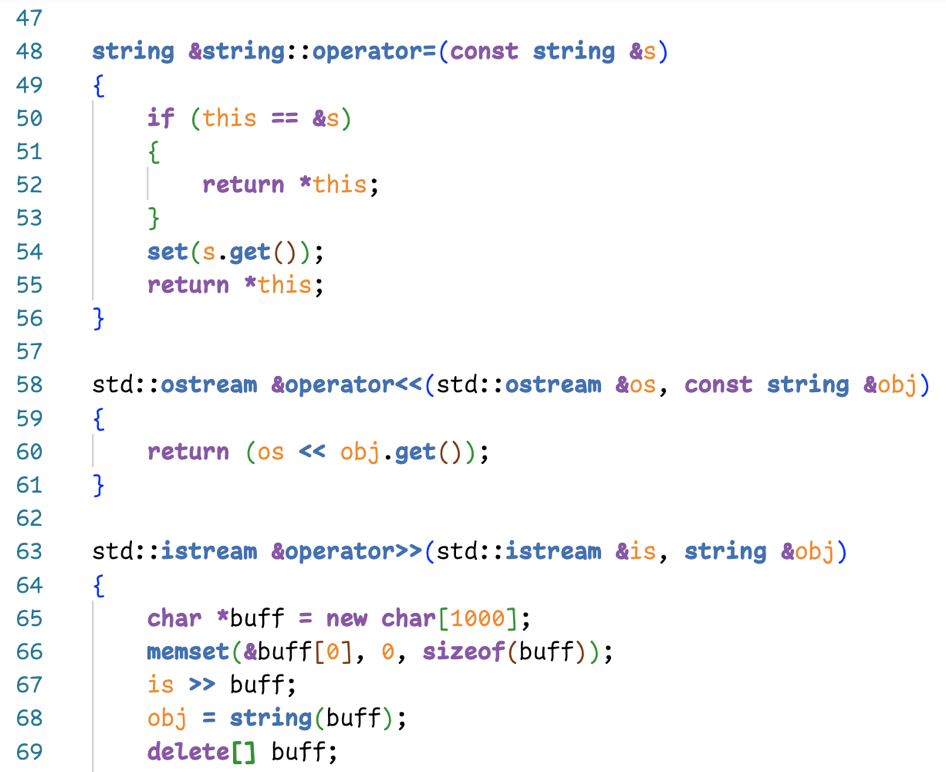


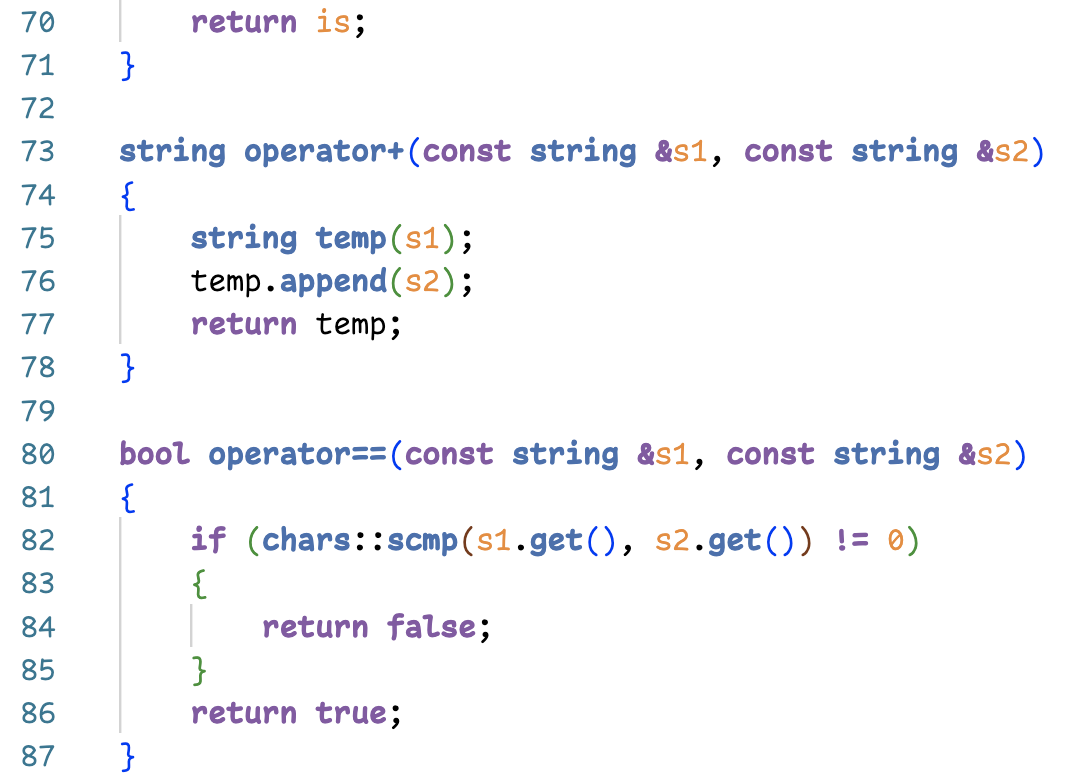


string.cpp









Makefile

