ЛАБОРАТОРНА РОБОТА №4

**Розробка класу рядка**

**Завдання:**

Розробити шаблон класу узагальненого рядка, що може містити символи довільного типу (тобто, String<T>). Передбачити можливість зберігання рядка довільної довжини. В загальному випадку передбачається, що рядок може включати нульові символи (тобто, значення T()).

Необхідно реалізувати наступні операції:

* Конструювання:
* за замовченням (порожній рядок); +
* копіювання; миша
* переміщення; +
* ініціалізація шляхом повторення символу заданої кількості разів;+
* ініціалізація вказівником на масив символів (при цьому, розмір рядка не передається, а кінцем рядка вважається перше включення нульового символу (значення T());
* ініціалізація вказівниками на початок (перший елемент) та кінець (наступний після останнього елементу) масиву символів (при цьому, допускається порожній масив (*початок* == *кінець*, а також масив може містити нульові елементи, але необхідно перевіряти, що *початок* розташовано не пізніше *кінця*));
* перетворення із рядка з іншим типом символів (перетворення символів може бути реалізоване за допомогою конструкції static\_cast). макс
* Присвоєння копіювання та переміщення.
* Отримання довжини рядка. миша
* Перевірка, чи є рядок порожнім. миша
* Очищення рядка (робить його порожнім). макс
* Доступ до символу за індексом (як для читання, так і для запису). Реалізувати за допомогою оператора []. богдан
* Отримання «підрядка»: дається початковий індекс і довжина та повертається новий рядок, що містить відповідну частину вихідного рядка. макс
* Конкатенація рядків та додавання одного символу до рядка (як зліва, так і справа). Реалізувати за допомогою перевантаження оператора. миша
* Повторення рядка задану кількість разів. Реалізувати як операцію множення рядка на ціле число (причому, як зліва, так і справа). Передбачити можливість множення на 0 (повертає порожній рядок). *Наприклад, результатом множення рядка “ABC” на 3 має бути “ABCABCABC”. богдан*
* Порівняння рядків (на (не)рівніть: ==, !=), а також за лексикографічним (як в словнику) порядком (<, >, <=, >=). Можлива реалізація одних операцій за допомогою інших. Реалізувати за допомогою перевантаження операторів. Передбачається, що для символів відповідні операції порівняння реалізовані. *Порівняння за лексикографічним порядком відбувається шляхом порівняння перших неоднакових символів. При цьому, кінець рядка вважається «меншим» за будь-який символ. богдан*
* Трансформації рядка за допомогою функціонального об’єкту:
* на місці, тобто, змінюючи символи вихідного рядка;
* а також зі створенням нового рядка та не змінюючи вихідний.

При цьому реалізувати дані операції за допомогою як динамічного, так і статичного поліморфізму (тобто, по дві версії).

* Функцію створення рядка відповідного типу із масиву символів (із автоматичним виведенням типу). Масив символів задається початком і кінцем, або лише початком (аналогічно до відповідних конструкторів). макс
* Для рядка звичайних символів (char) додати також операції вводу з клавіатури та виводу на екран. миша

Перетворення «на місці» означає, що змінюються символи вихідного рядка. При цьому, новий об’єкт не створюється. Інший варіант перетворення дозволяє зберегти вихідний рядок, а перетворений розмістити в новому об’єкті, що повертається з функції.

Для реалізації динамічного поліморфізму додати абстрактний базовий клас трансформації символів (у вигляді шаблону, наприклад, Transformer<T>), який містить операцію трансформації символу (наприклад, T operator()(const T&) const). Конкретні ж підкласи перетворень відповідним чином реалізують цю операцію. Тоді метод перетворення рядка може мати наступний вигляд:

void apply(const Transformer<T>& t)

В такому випадку конкретна операція, що буде застосована, може бути невідомою на етапі компіляції та визначатися деяким чином вже на етапі виконання.

Для випадку ж статичного поліморфізму базовий клас не є необхідним. Тип об’єкту-перетворювача задається параметром шаблону. Але для конкретної роботи він має містити відповідний метод для перетворення (наприклад, T operator()(const T&) const). Наприклад:

template <typename Trans>

void modify(const Trans& t)

Наявність відповідного методу в даному випадку не гарантується (але при його відсутності буде отримана помилка компіляції), проте, може досягатися більш висока швидкодія.

**Додаткові вимоги (обов’язкові):**

Реалізувати обробку помилкових ситуацій за допомогою механізму виключень (реалізувати можливість розрізняти різні помилки; бажано надавати максимум деталей про помилку). Уникати витоків пам’яті.

За необхідності дозволяється створення додаткових сутностей. Намагатися мінімізувати дублювання коду. Продумати інтерфейс та передбачити можливість роботи із константними об’єктами.

В тестовому коді продемонструвати використання всіх реалізованих можливостей (зокрема, застосування декількох різних трансформацій). Також продемонструвати обробку помилкових ситуацій (наприклад, із виводом відповідних повідомлень про помилки **у коді, що викликає**).

Також створити рядок із символів, для яких не визначені операції порівняння (для цього, наприклад, створити свій клас). Впевнитися, що інші (окрім операцій порівняння) операції для таких рядків залишаються доступними.

**Додаткові завдання (за бажанням):**

В якості додаткових завдань можуть бути реалізовані, зокрема, наступні:

* Реалізувати можливість трансформації (версія зі створенням нового рядка), що дозволяє отримати рядок із символами іншого типу (цей тип визначається типом значення, що повертає об’єкт-трансформатор).
* Реалізувати можливість визначення, чи може об’єкт заданого типу використовуватися як коректний трансформатор (тобто, чи містить він відповідну функцію(метод) трансформації). Тут мається на увазі випадок статичного поліморфізму. Реалізувати додаткові «безпечні» версії методів/функцій трансформації, що у випадку подачі некоректного «трансформатора» не призводять до помилок, а просто не змінюють рядок.

За бажанням робота може бути виконана в бригадах (до 3 учасників). В такому випадку максимальна оцінка за роботу – 10 балів. За самостійне виконання (не в бригаді) максимальна оцінка – 10 + 5 = 15 балів.

**Питання для самоперевірки:**

* Як реалізується обробка помилок за допомогою механізму виключень?
* Спільне та відмінне в статичному та динамічному поліморфізмі?
* Що таке спеціалізація шаблонів, для чого вона застосовується?
* Яка перевага створення екземплярів шаблонів класів за допомогою спеціальної функції (а не конструктора)?
* Що таке константні методи? Для чого вони застосовуються?
* В чому перевага механізму перевантаження операторів?
* В чому різниця між копіюванням та переміщенням?
* Які рекомендації є щодо реалізації бінарних операторів? Чому?
* Що таке абстрактні класи? Які вони мають особливості?
* Які є особливості інстанціювання методів та функцій?
* Яким чином можливо визначити наявність деякого члену в класу\*?