КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт
про виконання
лабораторної роботи №2/
навчального проєкту

Виконали
студенти групи ПІ-23
Петрик Юлія Олександрівна
Постернак Таїсія Вадимівна
Матвійчук Анастасія Миколаївна

Київ 2023

Тема: демонстрація роботи алгоритмів.

Ідея: розробити програму для демонстрації роботи певного класу алгоритмів, а саме: алгоритмів сортування (Bubble sort, Selection sort, Quick sort), роботи з деревами (Red-Black Tree, AVL-Tree), алгоритмів пошуку патернів за допомогою хеш-функцій (Boyer-Moore, Boyer-Moore-Horspool, Collins, Finite Automata, Gallagher-Syed-Gallagher, Knuth-Morris-Pratt, Rabin-Karp, Smith).

Було реалізовано:

- Відповідні алгоритми і структури даних.
- Механізм виміру продуктивності алгоритмів (час виконання, обсяг пам’яті).
- Механізм оцінки теоретичної складності алгоритмів.
- Візуалізація поведінки алгоритмів із інтерактивним режимом.
- Unit-tests
- Документація з використанням Doxygen

Використані патерни:

1. Патерн **Template Method**. Використовувався для реалізації варіантів алгоритмів, коли загальна структура лишається незмінною, але якісь аспекти реалізуються по-різному в підкласах.

   Зокрема, базовий клас Sort включає в себе віртуальний метод sort(), який має загальну структуру алгоритму сортування. Класи BubbleSort і SelectionSort успадковують базовий клас Sort і перевизначають метод sort(), щоб реалізувати конкретні алгоритми сортування.

   Також таким чином реалізовані алгоритми повороту піддерев для різних типів дерев.
2. Патерн **Decorator**.

Був використаний для вимірів часу виконання. Наприклад, клас TimingSortDecorator для алгоритмів сортування та TimedPatternSearchStrategy для алгоритмів пошуку патернів.

Для присвоєння властивостей «колір» і «висота» для вузлів відповідних дерев.

3. Патерн **Visitor**. Був використаний для розрахунку певних властивостей складних алгоритмів/структур даних. Зоврема, в алгоритмах сортування використовувався для визначення теоретичної складності.
4. Патерн **Command**. Використовувався для реалізації інтерактивного режиму роботи з алгоритмами.
5. Патерн **Memento**. Разом із патерном Command використовувався для реалізації інтерактивного режиму, зокрема для збереження та відновлення стану алгоритмів та структур даних. Наприклад, класи ListOfSteps і Step для алгоритмів сортування.
6. Патерн **Strategy**. Використовувався для того, щоб реалізувати можливість замінювати один алгоритм на інший прямо під час виконання програми. Клас PatternSearchStrategy є абстрактним базовим класом для всіх конкретних стратегій пошуку, таких як RabinKarp, KnuthMorrisPratt, GallagherSyedGallagher, FiniteAutomata, BoyerMooreHorspool та інші. Клас PatternSearchContext використовується для управління стратегією пошуку.
7. Патерн **Iterator**. Основна ідея полягає в тому, щоб винести поведінку обходу колекції з самої колекції в окремий об’єкт. Використаний для реалізації обходу дерев, який винесений в клас TreeIterator.
8. Патерн **Factory Method**. Застосований для створення нових об’єктів класів AVLTree і RedBlackTree.
9. Патерн **Mediator**. Застосований для вибору типу та власне алгоритму, візуалізацію якого користувач хоче побачити. У ролі Посередника вистувають реалізовані в QML класи.

10. Патерн **State**. Патерн дає змогу об’єктам змінювати поведінку в залежності від їхнього стану. В контексті алгоритмів PatternSearchStrategy, було створено клас StateBoyerMoore, що визначає який з підвидів алгоритмів Боєра-Мура (BoyerMoore чи BoyerMooreHorspool) доцільно застосувати і забезпечує виконання визначеного алгоритму.

Висновок

Виконання даної лабораторної роботи дозволило нам розробити програму для демонстрації роботи різних класів алгоритмів, таких як алгоритми сортування, роботи з деревами і алгоритми пошуку патернів за допомогою хеш-функцій.

У результаті були реалізовані відповідні алгоритми та структури даних, а також механізми виміру продуктивності алгоритмів, оцінки їх теоретичної складності та візуалізації їх поведінки з інтерактивним режимом.

Під час розробки програми ми використали різні патерни проєктування, такі як Template Method, Decorator, Visitor, Command, Memento, Strategy, Iterator, Factory Method, Mediator і State. Також було б доречно застосувати патерн Singleton для головного вікна програми, якби була можливість гарантувати єдиність екземпляру цього класу. Використання цих патернів дозволило нам зробити програму більш гнучкою, розширюваною та легко змінюваною.

Застосування патернів дало нам можливість реалізувати різні варіанти алгоритмів, вимірювати їх продуктивність, динамічно змінювати алгоритми під час виконання програми та відтворювати стан алгоритмів і структур даних.