**Документація до коду**У своєму варіанті виконання лабораторної , я вирішив узяти программу яка дмонструє деякі алгоритми , і структуру данних.  
А саме: SelectionSort , BubbleSortS , PolindromCheck та сама структура данних HashTable.

Під час роботи я вивчав такі аспекти:  
  
 Використання патернів проектування під час написання роботи , внесеня механізму підрахунку часу роботи алгоритмів , збереження структури коду ООП  
  
КОД  
-----------------------------------------------------------------------------------------------------------------------------------

**SortStrategy**  
Зображення, що містить текст, Шрифт, знімок екрана, ряд

Автоматично згенерований опис  
Описує абстрактний базовий клас SortStrategy, який визначає інтерфейс для всіх стратегій сортування. Він містить один чистий віртуальний метод sort, що дозволяє сортувати вектор цілих чисел data.  
-----------------------------------------------------------------------------------------------------------------------------------  
**''сортування бульбашкою''**   
Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис  
Цей класс описує конкретний метод **''сортування бульбашкою''**   
  
Алгоритм сортування бульбашкою працює, порівнюючи кожну пару сусідніх елементів у векторі і, якщо потрібно, обмінює їх місцями. Після кожного прогону найбільший елемент "спливає" на праву кінцеву позицію. Процес повторюється до тих пір, поки весь вектор не буде відсортованийю.

У внутрішньому циклі for проходиться по всіх елементах вектора data, порівнюючи кожну пару сусідніх елементів. Якщо поточний елемент data[j] більший за наступний елемент data[j + 1], то вони обмінюються місцями за допомогою функції swap(). Таким чином, більші елементи "спливають" на кінець вектора.  
-----------------------------------------------------------------------------------------------------------------------------------  
**''Сортування вибіркою''**  
Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований описАлгоритм сортування вибором працює таким чином: на кожній ітерації вибирається найменший елемент з невідсортованої частини вектора і обмінюється з першим елементом у невідсортованій частині. Таким чином, на кожній ітерації найменший елемент "вибирається" і поміщається на правильну позицію у відсортованій частині вектора.  
  
У зовнішньому циклі for проходиться по усіх елементах вектора data, крім останнього, оскільки на останній позиції залишається найбільший елемент після кожної ітерації. У внутрішньому циклі for проходиться по невідсортованій частині вектора, починаючи з наступного елемента після поточного. Кожен елемент порівнюється з поточним найменшим елементом data[min\_idx]. Якщо знайдений елемент менший за поточний найменший елемент, то індекс найменшого елемента оновлюється. Після закінчення внутрішнього циклу, найменший елемент з невідсортованої частини вектора обмінюється з першим елементом у невідсортованій частині.  
  
  
  
  
  
**SortContext**  
Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис  
  
Самец ей класс реалізує патерн проєктування **Strategy** Він дозволяє визначати різні алгоритми сортування, інкапсулюючи їх у відповідних класах стратегій. Клас SortContext використовує ці стратегії для виконання сортування в залежності від поточної обраної стратегії. Це забезпечує гнучкість і можливість легко змінювати алгоритм сортування без необхідності змінювати основний код програми.  
  
-----------------------------------------------------------------------------------------------------------------------------------  
**Класс PalindromeChecker**Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

У цьому класу було реалізовано метод перевірки числа на поліндром .Паліндром - це число, яке залишається незмінним при читанні зліва направо і справа наліво.  
  
Метод isPalindrome виконує перевірку, чи є число num паліндромом. Він створює змінну rev і початково присвоює їй значення 0. Потім виконується цикл do-while, який виконує наступні дії:

Знаходить останню цифру числа n шляхом взяття остачі від ділення n на 10 (digit = n % 10).

Оновлює змінну rev, додавши до неї останню цифру (rev = (rev \* 10) + digit).

Зменшує число n, видаляючи останню цифру (n = n / 10).

Цей процес триває до тих пір, поки n не стане рівним 0.

Після виходу з циклу перевіряється умова num == rev, щоб визначити, чи є число num паліндромом. Якщо вони рівні, то метод повертає значення true, що означає, що число є паліндромом. В іншому випадку метод повертає значення false, що означає, що число не є паліндромом.  
-----------------------------------------------------------------------------------------------------------------------------------

**HashTable**

Цей клас реалізує структуру данних Хеш-Таблиця  
  
Ця структура данних яки дозволяе вносити у таблицю данні , які потім можна відшукати за індексом масива в який вони вносяться. Користувач вводить стрінгову змінну . Вона проходить через так звану хеш-функцію яка опреділяє індекс під яким буде заноситися данні в масив  
  
Цей клас має такі методи   
  
**static HashTable\* getInstance():** Цей статичний метод використовує патерн Одинак (Singleton). Він повертає єдиний екземпляр класу HashTable. Якщо екземпляр ще не існує (тобто instance є нульовим), метод створює новий екземпляр за допомогою оператора new і присвоює його instance. В результаті виклику getInstance(), буде повернуто посилання на єдиний екземпляр.

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис  
  
**bool isEmpty() const:** Цей метод перевіряє, чи є хеш-таблиця порожньою. Він проходиться по всіх групах хеш-таблиці (table) та сумує кількість елементів у кожній групі. Якщо сума дорівнює нулю, то хеш-таблиця вважається порожньою і метод повертає true.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
  
**int hashFunc(int key):** Цей метод обчислює хеш для заданого ключа. Використовується простий алгоритм модулярного хешування, де ключ ділиться на кількість груп хеш-таблиці (hashGroups), і залишок від ділення використовується як хеш.  
Зображення, що містить текст, Шрифт, знімок екрана, Графіка

Автоматично згенерований опис  
  
**void insertItem(int key, string value):** Цей метод додає нову пару ключ-значення до хеш-таблиці. Використовуючи хеш-функцію, він визначає групу (hashValue) для вставки пари. Далі перевіряється, чи ключ вже існує в цій групі. Якщо так, значення оновлюється. Якщо ні, пара додається до групи.  
Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
 **void removeItem(int key):** Цей метод видаляє пару з хеш-таблиці за заданим ключем. Використовуючи хеш-функцію, він знаходить групу, де можливо знаходиться пара з заданим ключем. Потім він шукає пару з таким ключем у групі. Якщо знайдено, пара видаляється з групи. Якщо не знайдено, виводиться повідомлення про відсутність ключа.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
  
**void printTable():** Цей метод виводить всю хеш-таблицю на екран. Він проходиться по кожній групі хеш-таблиці та виводить усі пари ключ-значення, які знаходяться в групі.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
  
**string searchTable(int key):** Цей метод шукає значення за заданим ключем в хеш-таблиці. Він проходиться по кожній групі хеш-таблиці та перевіряє кожну пару ключ-значення. Якщо знайдено пару з заданим ключем, метод повертає відповідне значення. Якщо ключ не знайдено, повертається порожній рядок.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис  
  
-----------------------------------------------------------------------------------------------------------------------------------  
У цьому фрагменті коду було використанно патерн Singleton  
  
Клас HashTable використовує патерн Одинак, оскільки має приватний конструктор і статичний метод getInstance(), який повертає єдиний екземпляр цього класу.

Приватний статичний член instance забезпечує зберігання єдиного екземпляру класу HashTable.

Конструктор HashTable() оголошений як приватний, щоб заборонити створення нових екземплярів класу ззовні.  
  
Цей патерн дозволяє гарантувати, що у програмі буде наявний лише один екземпляр класу HashTable і надає глобальний доступ до цього екземпляру через статичний метод getInstance().  
-----------------------------------------------------------------------------------------------------------------------------------  
  
ЗАУВАЖЕННЯ   
При виконанні роботи було використано патерни проектування , збережено вигляд об'єктно орієнтованого програмування , додано механізм обчислення часу роботи деяких алгоритмів   
  
Повністю роботу можна подивитися за посиланням на репозиторій   
https://github.com/burch1302/OOP\_lab\_2