Лабораторна робота №1

Реалізувати у вигляді об’єктно-орієнтованої системи структури даних та алгоритми з інших предметів (**Список 1**). Реалізовані структури даних мають підтримувати роботу з даними різних типів. Продемонструвати роботу системи на примітивних типах (int, double), бібліотечних типах (string, vector), а також реалізованих класах (**Список 2**). Для всіх структур даних реалізувати методи отримання текстового подання (наприклад, з метою виводу на екран), а також генерації відповідних структур даних, заповнених випадковими даними. Можна додатково реалізувати запис відповідних структур у файл та зчитування з файлу (і отримати ***додаткові бали***)

Ця лабораторна робота передбачає використання інтерфейсу командного рядка, проте за бажанням можна реалізувати графічний інтерфейс та/або засоби візуалізації і отримати ***додаткові бали***.

Під час реалізації можна використовувати бібліотеки чи фреймворки, як стандартну бібліотеку С++ (чи іншої обраної мови програмування), так і зовнішні бібліотеки.

Обов’язковою частиною цієї лабораторної є проектування правильного ОО дизайну для поставлених завдань (як зі списку 1, так і зі списку 2). Зокрема, треба дотримуватись основних принципів проектування, уникати дублювання коду. Реалізовані структури даних мають бути зручними для використання – зокрема, клієнт має можливість обирати, яку з реалізацій використовувати, і зміна реалізації не має викликати зміни коду клієнта. Додатково можна реалізувати патерни проектування (і отримати ***додаткові бали***).

Всі реалізовані структури даних та алгоритми мають бути організовані у вигляді однієї програми (один виконуваний файл, одна функція main(), довільна кількість файлів з кодом). За бажання можна розбити на бібліотеку та виконуваний файл, що її використовує. Бажано реалізувати unit tests (і отримати ***додаткові бали***). Для студентів, які претендують на оцінку «відмінно» за семестр, реалізація unit tests є обов’язковою. Якщо тести демонструють всю необхідну поведінку, можна не реалізовувати додаткові режими роботи для перевірки реалізацій (зокрема, інтерактивний режим чи демонстраційний режим). Можна використати micro-benchmarking frameworks/libraries для виміру часу виконання різних алгоритмів/операцій для різних реалізацій, (і отримати ***додаткові бали***).

В цьому семестрі не обов’язково створювати документацію (наприклад, з використанням Doxygen) для реалізованого коду (в наступному семестрі це буде обов’язковою умовою). Проте це бажано зробити, і за це будуть виставлені ***додаткові бали.***

Обов’язковим для цього завдання (а також для інших завдань починаючи з цього семестру) є використання репозиторію source control (revision control, version control, …). Репозиторій треба створювати на початку роботи з кодом, а не перед здачею.

Варіантом цієї лабораторної роботи є номери пунктів зі списків 1 та 2. Різні варіанти мають різний рівень складності, умовно позначений зірочками: \* – більш прості завдання, \*\*\* – завдання середньої складності, \*\*\*\*\* – більш складні завдання. Для отримання кращої оцінки в семестрі варто обирати більш складні варіанти. Зокрема, для отримання максимальної кількості балів треба набрати не менше 8\* **в сумі за список 1 та список 2**.

Для деяких варіантів запропоновані можливості розвитку завдання, які дозволяють підвищити складність і відповідно отримати кращий результат. Це лише рекомендації, можна зробити якісь інші покращення (хоча бажано погодити з викладачем).

Варіанти (тобто комбінація номеру зі списку 1 та списку 2) має бути унікальною серед усіх студентів. Тобто можна, наприклад, одному студенту обрати варіант 7 зі списку 1 та варіант 5 зі списку 2, а іншому студенту варіант 7 зі списку 1 та варіант 9 зі списку 2. Проте не можна двом студентам обрати однакові варіанти з обох списків (наприклад, два студента обирають варіант 1 зі списку 1 та варіант 3 зі списку 2).

Деякі з комбінацій варіантів зі списку 1 та списку 2 можуть бути не дуже «осмисленими» – наприклад, операції зі списку 1 не дуже застосовані до сутностей зі списку 2. Такі комбінації можна не обирати (кількість можливих варіантів достатня, щоб всі студенти могли обрати собі унікальні комбінації). Також можна придумати свої інтерпретації операцій, щоб зробити комбінацію більш зрозумілою.

Необхідно вказати в репозиторії обрані варіанти (з обох списків) та реалізовані покращення. Наприклад, в README файлі. Бажано також вказати інші моменти реалізації, за які можуть бути виставлені додаткові бали. Бажано вказати кількість зірочок \*, яка була реалізована.

**Список 1**: структури даних та алгоритми.

1. (\*\*\*\*) Графи на основі списку суміжності, матриці суміжності (збереження даних у вершинах та ребрах графів). Додавання та видалення вершин/ребер. Перевірка на зв’язність графу. Визначення відстані між двома вершинами графу.
   1. +\* за інші алгоритми на графах
   2. +\* за реалізацію загального механізму, що дозволить підключати нові алгоритми
   3. +\* за візуалізацію графів

**Список 2**: класи для опису даних.

1. (\*\*)Інформація про дату (рік, місяць, день) та час (години, хвилини, секунди). Перевірка правильності дати та часу відповідно до григоріанського календаря. Арифметика моментів часу: різниця між двома моментами часу у заданих одиницях, додавання чи віднімання такої різниці до заданого моменту часу. Обчислення дня тижня для заданої дати.
   1. +\* за обчислення додаткових параметрів, наприклад, номер тижня в місяці та в році
   2. +\* за альтернативні варіанти побудови дати та часу (наприклад, «перший вівторок листопада»)
   3. +\* за підтримку часових зон
   4. +\* за обчислення статистики, наприклад, на який день тижня найчастіше припадає 13 число (за даним діапазоном дат чи взагалі в календарі).
   5. +\* за підтримку альтернативних календарів (юліанського, …)