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. (\*) Список, циклічний список. Реалізації на основі зв’язних списків, масивів змінної довжини, бібліотечних засобів. Пошук елемента за значенням, за індексом, першого елемента за заданою умовою.
   1. +\* за інші реалізації списку, наприклад, Skip List, XOR Linked List, Unrolled Linked List, VList, Hashed Array Tree, …
2. (\*\*) Список. Реалізації на основі зв’язних списків, масивів, бібліотечних засобів. Алгоритми сортування: insertion sort, quicksort, merge sort.
   1. +\* за інші алгоритми сортування;
   2. +\* за реалізацію швидких алгоритмів, не заснованих на порівнянні (non-comparison sorts, наприклад counting, radix, bucket sort);
   3. +\* за реалізацію загального механізму, що дозволить підключати нові алгоритми сортування;
   4. +\* за підтримку різних способів сортування (ключів сортування, зростання/спадання) для одного об’єкта.
   5. +\* за підтримку сортування за кількома критеріями (наприклад, спершу за одним полем, якщо значення однакові – за іншим і т.д.)
3. (\*\*) Stack, queue, deque (double-ended queue). Реалізації на основі зв’язних списків, масивів, бібліотечних засобів. Стандартні операції для кожної структури. Deque має реалізовувати інтерфейси stack та queue.
4. (\*\*\*) Розріджені списки та матриці. Доступ за індексом, пошук за значенням, пошук першого елемента за заданою умовою.
   1. +\* за операції з матрицями – додавання, множення, множення матриці на вектор, транспонування…
   2. +\* за реалізацію різних механізмів зберігання розріджених матриць та порівняння їх ефективності для різних прикладів матриць
5. (\*\*\*) Черга з пріоритетом на основі списку (реалізації на основі зв’язного списку та масивів змінної довжини) та дерева. Додавання елементу, забирання елементу (за правилами черги), подивитись наступний елемент, не забираючи його.
   1. +\*\* за реалізацію на основі самобалансованого дерева – AVL, Red-Black, …
   2. +\* за реалізацію на основі не-бінарного дерева
   3. +\* за реалізацію на основі heap
6. (\*\*) Дерева з довільною кількістю дітей (на основі зв’язного списку та бібліотечних засобів для списку), бінарні дерева, дерева бінарного пошуку. Додавання елементів, видалення елементів (за батьківським вузлом та індексом, за значенням), пошук за значенням.
   1. +\* за різні варіанти обходу дерев, пошук за шляхом в дереві;
   2. +\* за підтримку різних механізмів видалення, що якось обробляють елементи з видалених під-дерев
7. (\*\*\*\*) 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
8. (\*\*\*\*) Графи, дерева. Додавання та видалення елементів/вершин/ребер. Побудова кістякового дерева для заданого графу.
   1. +\* за побудову мінімального кістякового дерева;
   2. +\* за кілька різних алгоритмів побудови кістякового дерева і механізм додавання нових алгоритмів;
   3. +\* за більш складні алгоритми побудови кістякового дерева.

**Список 2**: класи для опису даних.

1. (\*) Фігури на площині: трикутники, чотирикутники, п’ятикутники. Обчислення периметра та площі фігури. Перевірка на спеціальні фігури – прямокутні, рівнобедрені, рівносторонні трикутники; трапеції, паралелограми, ромби, прямокутники, квадрати; правильні п’ятикутники.
   1. +\* за інші спеціальні типи фігур;
   2. +\* за побудову спеціальних точок для даної фігури та фігур на їх основі – наприклад, точки перетину медіан/бісектрис/висот, центр описаного кола, точка перетину діагоналей, теорема Наполеона, … –та перевірку відповідних властивостей та теорем геометрії;
   3. +\* за підтримку n-кутників для довільного n;
   4. +\* за перевірку опуклості фігур.
2. (\*\*\*) Фігури на площині: прямі та кола. Обчислення точок перетину двох фігур. Операції симетричного відображення відносно заданої прямої та інверсії відносно заданого кола (для точок та цих фігур).
   1. +\* обчислення кутів між фігурами, перевірка збереження кутів під час перетворень
3. (\*\*\*\*) Фігури на площині, задаються довільною невід’ємною функцією (фігура від осі x до графіку функції) або двома функціями (фігура між графіками функцій). Обчислення площі та периметру фігур. Обчислення фігури-перетину двох заданих фігур.
   1. +\* обчислення фігури – об’єднання двох заданих фігур)
   2. +\* за можливість вводити складні функції з клавіатури/через інтерфейс користувача
   3. +\* за підтримку різних способів задання функції (символьний, табличний, графічний, код, …)
4. (\*\*\*\*\*) Тіла в просторі, задаються довільною невід’ємною функцією двох аргументів (тіло від площини xy до графіку функції) або тіла обертання – задаються довільною невід’ємною функцією одного аргументу. Обчислення об’єму та площі поверхні тіла.
   1. +\* тіла обертання навколо різних осей
   2. +\* за можливість вводити складні функції з клавіатури/через інтерфейс користувача
   3. +\* за підтримку різних способів задання функції (символьний, табличний, графічний, код, …)
5. (\*\*) Функції багатьох змінних (підтримуються стандартні функції – арифметичні операції, піднесення до степеня, логарифми, тригонометричні функції). Обчислення в заданій точці. Диференціювання за заданою змінною.
   1. +\* спрощення отриманих виразів
   2. +\*\* символьне інтегрування за заданою змінною
6. (\*\*\*) «Нечесні» гральні кості з кількістю граней N=2, 4, 6, 8, 10, 12, 20. Грані помічено числами від 1 до N. Для кожної грані задається ймовірність випадіння (число від 0 до 1). Обчислення всіх можливих сум для заданого набору костей (набір може містити довільну кількість костей з однаковими чи різними гранями та однаковими чи різними ймовірностями) та ймовірність випадіння кожної з них.
   1. +\* порівняння двох наборів за очікуваною сумою значень граней.
   2. +\* підтримка інших схем нумерування граней (не лише від 1 до N)
7. (\*\*) Адреси IPv4 (a1.a2.a3.a4, 0<=ai<=255) та IPv6 (b1:b2:b3:b4:b5:b6:b7:b8, кожна з bi – набір від 1 до 4 шістнадцяткових цифр 0..9a..f). Адреси підмереж у форматі CIDR: address/subnet\_bits, де address – адреса в одному з форматів, subnet\_bits – кількість бітів маски підмережі. Перевірка належності адреси до підмережі.
   1. +\* за реалізацію пошуку вільного діапазону адрес заданого розміру (підмережі);
   2. +\* за підтримку вкладених підмереж;
   3. +\* за підтримку інших варіантів запису IPv6 адрес, наприклад скорочень :: та IPv4 адрес в нотації IPv6;
   4. +\* за побудову адрес з текстового подання (parsing);
   5. +\* за додаткову підтримку MAC адрес та побудову IPv6 адреси на основі MAC адреси
   6. +\* за пошук мінімальної кількості вільних підмереж, які в сумі мають не менше заданого розміру (розвиток a).
8. (\*) Інформація про об’єкти файлової системи – файли та каталоги (назва, розмір, час створення, час модифікації, тип файлу). Розрахунок повного шляху до файлу/каталогу. Пошук файлів за заданими критеріями в дереві підкаталогу.
   1. +\* за підтримку критеріїв за кількома параметрами одночасно;
   2. +\* за підтримку складних критеріїв – довільних виразів з логічними операторами;
   3. +\* за пошук імен та шляхів з використанням регулярних виразів (regular expressions);
   4. +\* за підтримку symlinks/shortcuts;
   5. +\* за реалізацію системи прав доступу – можна задати права доступу до певного файлу чи каталогу, права доступу з каталогу за замовчуванням застосовуються до всіх дітей;
   6. +\* за імпорт з реальної файлової системи.
9. (\*\*)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, …)
10. (\*\*\*) Інформація про книги (назва, автор(и), дата виходу, кількість сторінок, коротка анотація) та персонажів книг (список імен/псевдонімів, в яких книгах згадується, для кожної книги рівень участі – головний, чи другорядний, чи епізодичний персонаж). Розбиття книг на серії (дві книги належать до однієї серії, якщо у них є спільний головний чи другорядний персонаж; в рамках серії книги впорядковано за датою виходу).
    1. +\* за реалізацію «бази даних» – з можливістю додавати, редагувати, видаляти, шукати книги та авторів (аналогічно лабораторній роботі №1 з минулого семестру, реалізація на основі файлів);
    2. +\* за реалізацію бази даних на основі існуючої SQL або NoSql СКБД;
    3. +\* за підтримку «альтернативних реальностей» – наприклад, канонічні серії книг, книги на основі фільмів, неофіційні книги, …
11. Самостійно обрані сутності для реалізації (**обов’язково** погодити з викладачем!) Наприклад, можна подумати про якісь застосунки/сервіси, які використовуються досить часто, та реалізувати певні сутності з цих застосунків та операції з ними.