Лабараторна робота 1

Варіант 2c

Реалізувати «навчальну базу даних» – систему для роботи з даними певного типу. Необхідно реалізувати підтримку наступних операцій:

1. Додавання елементів;
2. Зберігання даних (запис даних у файл);
3. Відновлення даних (зчитування даних з файлу);
4. Вивід всіх збережених даних;
5. Пошук за заданими критеріями (див. підваріанти a-c);
6. Модифікація елементів (додаткові бали);
7. Видалення елементів (додаткові бали).

Для зберігання елементів треба реалізувати наступні механізми:

1. Зберігання в пам'яті, без збереження на диск (можна використати довільну структуру даних, зокрема бібліотечні структури на зразок std::vector);
2. Зберігання у вигляді текстового файлу;
3. Зберігання у вигляді бінарного файлу

Для кожного елементу, окрім описаних у відповідному варіанті даних, треба також зберігати унікальний ID – ціле число, яке буде унікальним для кожного елементу даних.

Необхідно реалізувати зберігання даних, які вводить користувач, а також генерацію випадкових даних для зберігання та пошуку. У випадку зберігання даних на диск, ці дані мають залишатись доступними після повного закриття та перезапуску програми. Тобто користувач може повністю закрити програму, і навіть перезавантажити ОС, а після нового запуску програми всі раніше збережені дані залишаються доступними і програма може їх прочитати.

Необхідно реалізувати наступні режими роботи для реалізованої програми:

1. Інтерактивний діалоговий режим – коли користувач може вибирати, які операції виконувати та задавати необхідні параметри.
2. Демонстраційний режим – задаються фіксовані значення параметрів та послідовності виконання операцій, що демонструють правильність роботи операцій в різних ситуаціях. В цьому режимі користувач не має нічого вводити.
3. Автоматичний режим "benchmark" з вимірами ефективності роботи різних механізмів зберігання.

В автоматичному режимі "benchmark" задається певне число елементів N; генерується N випадкових елементів, вони додаються та зберігаються, потім відновлюються і виконується пошук. При цьому вимірюється час виконання операцій (з точністю до мілісекунд) та

розмір збережених даних (в байтах). Бажано окремо виміряти час виконання кожної з операцій. Ця процедура виконується для всіх реалізованих механізмів зберігання. Потім значення N збільшується, і виміри повторюються для нового значення N. Така процедура повторюється, доки час виконання операцій не стане більшим за 10 секунд. Результати вимірів для всіх розглянутих значень N треба зберегти і надати викладачу під час здачі лабораторної (наприклад, додавши відповідні файли в репозиторій з кодом виконаних завдань).

Варіант 2

Інформація про товари в магазині. Зберігаються наступні дані про кожен товар: назва товару; одиниці виміру – кілограми, літри, поштучно, пакунки; кількість товару (у відповідних одиницях, дійсне число); дата та час виробництва; термін зберігання (кількість діб з дати виробництва, не більше 10 років). Критерії пошуку:

Пункт с

Товари з назвою, що завершується на заданий фрагмент тексту; товари із заданою одиницею вимірювання та терміном зберігання не менше заданого; товари з датою виробництва у заданому діапазоні.

Лабараторна робота 2

Варіант 2

Підваріант 2

В цій лабораторній необхідно реалізувати задану структуру даних, використовуючи три різні підходи:

1. На основі масиву фіксованого розміру. При виході за межі масиву видавати повідомлення про помилку.
2. На основі масиву змінного розміру (ArrayList, наприклад, std::vector).
3. На основі зв’язного списку. Самостійно обрати однозв’язний чи двозв’язний список та інші деталі реалізації.

Необхідно реалізувати наступні режими роботи для реалізованої програми:

1. Інтерактивний діалоговий режим – коли користувач може вибирати, які операції виконувати та задавати необхідні параметри.
2. Демонстраційний режим – задаються фіксовані значення параметрів та послідовності виконання операцій, що демонструють правильність роботи структур даних та операцій в різних ситуаціях. В цьому режимі користувач не має нічого вводити.
3. Автоматичний режим “benchmark” з вимірами ефективності роботи різних версій структур даних.

В автоматичному режимі необхідно виміряти та порівняти час виконання кожної операції окремо для всіх трьох підходів реалізації – на основі масиву фіксованого розміру, масиву змінного розміру та зв'язних списків. Можна додатково виміряти розмір пам’яті, який займають структури даних (і отримати додаткові бали). Варто виміряти час виконання для різних розмірів структур даних та різної кількості операцій. Результати вимірів для всіх розглянутих розмірів, операцій та сценаріїв треба зберегти та надати викладачу під час здачі лабораторної.

Всі версії структур даних, режими роботи, операції та сценарії мають бути реалізовані в рамках однієї програми (тобто один виконуваний файл, одна функція main(), довільна кількість файлів з кодом).

Варіант 2

Реалізувати структуру даних «стек» з наступними операціями: 1) create_empty – створення пустого стеку; 2) push – додавання елементу до стеку; 3) pop – вилучення елементу зі стеку; 4) peek – значення верхнього елементу стеку (без видалення); 5) is_empty – перевірка на пустоту.

Підваріант 2

В структурі зберігаються дати (день, місяць, рік)

Лабараторна робота 3a

Реалізувати алгоритми сортування для заданої структури даних(список1). Реалізувати наступні алгоритми:

1) простий алгоритм (список 2);
2) більш ефективний алгоритм quicksort (список 3);
3) більш ефективний алгоритм merge sort (список 4);
4) комбінований алгоритм, який використовує більш ефективний алгоритм (один з quicksort чи merge sort) для великих масивів і простий алгоритм для маленький масивів;
5) сортування з використанням бібліотечних функцій.

Список 1:

7. Текстові рядки, спершу за довжиною, за зростанням (“A”<”B”<”AA”<”AB”)

Список 2:

2. Insertion sort

Список 3:

9. Lomuto partition scheme, random pivot

Список 4:

3. bottom-up

Лабараторна робота 3b

В цій лабораторній роботі треба розширити можливості програми, реалізованої в лабораторній роботі № 1. Необхідно додати можливість сортування даних за довільним полем чи за довільною комбінацією полів (тобто спершу за полем1, якщо там однакові значення – за полем2). Для реалізації сортування використати один з ефективних алгоритмів сортування (comparison sort зі складністю в середньому O(n log n), наприклад, quicksort, merge sort, heapsort, …). Можна також використати бібліотечну реалізацію сортування.

Крім цього, необхідно реалізувати два алгоритми сортування, які не є comparison sort:

1. Counting sort
2. Radix sort

Треба реалізувати стабільні версії цих алгоритмів (тобто не переставляються елементи з однаковим значенням ключа сортування).

Ця лабораторна робота передбачає сортування в пам’яті.

Лабараторна робота 3c

Задача 13(**)

Реалізувати алгоритм сортування bucket sort

Задача 21(***)

Реалізувати алгоритм сортування cubesort

Лабараторна робота 4

Дерева

Завдання цієї лабораторної організовано у вигляді дерева. Окремі завдання є вузлами дерева. Варіант лабораторної відповідає шляху від кореня до листових вузлів. Дерево будується, починаючи з кореня (блок 0) і послідовно додаючи вузли з наступних блоків.

Блок 0

1) Реалізувати структуру даних для опису дерева з довільною кількістю «дітей». У вузлах дерева зберігаються цілі числа. Core.

Блок 1 – додавання елементів до дерева

6)Реалізувати функцію для додавання елемента до довільного дерева. Передається значення нового елемента, а також мінімальна та максимальна кількість дітей. Елемент додається за такими правилами: 1) якщо кількість дітей поточного елемента менше мінімальної – додаємо до поточного елемента; 2) інакше вибираємо дитину поточного елемента, у якої кількість дітей мінімальна; 3) якщо кількість дітей у цієї дитини не більше максимальної – додаємо елемент за тою ж процедурою до цієї дитини; 4) інакше, якщо кількість дітей поточного вузла не більше максимальної – додаємо до поточного вузла; 5) інакше шукаємо дитину з найменшим розміром під-дерева і додаємо до цієї дитини за тою ж процедурою. Core. Provides: ADD_PROC

Блок 2 – вивід дерева

8) Реалізувати функцію для виводу дерева з використанням відступів для виводу дітей різних рівнів. Core.

Блок 3 – видалення елементів

15) Реалізувати функцію для видалення елементу з довільного дерева. Передається значення елементу (якщо є кілька елементів з таким значенням – видаляються всі вузли). Звільнити пам’ять видалених елементів. Core.

Блок 4 – бінарні дерева (у всіх завданнях реалізувати також вивід дерева, на основі відповідної функції з блоку 2)

18) Реалізувати структуру даних для опису бінарного дерева. Реалізувати функцію для додавання елементів відповідно до значення (менші значення наліво, всі інші направо). Core.

Блок 5 – подання бінарних дерев

21) Побудувати з бінарного дерева послідовне подання на основі прямого порядку.

Блок 6 – використання дерев

24) Реалізувати дерево для подання логічних виразів. Підтримуються константи (0 та 1) та змінні, а також операції AND, OR, NOT, XOR, імплікації, еквівалентності. Реалізувати спрощення виразів, обчислення константних підвиразів. Реалізувати обчислення значення виразу для заданих значень змінних. Реалізувати перевірку, що вираз є завжди істинним (тавтологія) чи завжди хибним (протиріччя).

Лабораторна робота 5

Графи

Завдання цієї лабораторної організовано у вигляді блоків. Необхідно реалізувати мінімум по одному завданню з кожного блоку – окрім блоку 0, де треба реалізувати мінімум два завдання.

Блок 0 - Вибрати не менше 2-х різних завдань з цього блоку. У всіх випадках реалізувати додавання ребер в граф, вивід графу, а також створення випадкових графів із заданою кількістю вершин та ребер. Реалізувати перетворення між реалізованими поданнями графів. Реалізовані структури даних мають підтримувати орієнтовані та неорієнтовані графи, а також вагові коефіцієнти ребер.

1) Реалізувати структуру даних граф на основі матриці суміжності.

4) Реалізувати структуру даних граф на основі бітового вектору суміжності (довільна кількість вершин у графі)

Блок 1 – прості алгоритми на графах

10) Реалізувати алгоритм побудови транзитивного замикання графу.

Блок 2 – обхід графів. В цьому блоці реалізувати дві версії алгоритмів – сусіди заданого вузла обходяться в довільному порядку (наприклад, в порядку номерів вершин), та сусіди обходяться в порядку від найменшої до найбільшої ваги відповідних ребер.

13) Реалізувати алгоритм обходу графу в ширину.

Блок 3 – пошук шляхів мінімальної довжини. В цьому блоці реалізувати версії алгоритмів для пошуку шляху між двома заданими вершинами, від заданої вершини до всіх інших, між усіма вершинами графу. Розглянути випадки орієнтованого та неорієнтованого графів.

16) Реалізувати алгоритм Белмана-Форда.

Блок 4 – топологічне сортування

17) Реалізувати алгоритм Кана.

Блок 5 – кістякові дерева. Реалізувати на основі обох версій обходу (див. блок 2). Розрахувати сумарну вагу дерева.

20) Реалізувати алгоритм побудови кістякового дерева на основі пошуку в ширину.

Блок 6 – мінімальні кістякові дерева.Розрахувати сумарну вагу дерева.

22) Реалізувати алгоритм побудови мінімального кістякового дерева (Dijkstra-Janik-Prim).

Лабораторна робота 6

Реалізувати структуру даних **«впорядкований список»** на основі:

1. Зв’язного списку (Linked List)
2. Списку на основі масиву (Array List)
3. Бінарного дерева пошуку (без балансування)
4. АВЛ-дерева
5. 2-3 дерева

У впорядкованому списку зберігаються структури даних відповідно до варіантів. (Кожен варіант має містити всі 5 реалізацій)

Реалізувати операції:

- створення пустого списку;
- додавання елементу;
- вилучення елементу;
- пошук елементів за значенням та за діапазоном значень;
- вивід елементів у правильному порядку;
- виконання певних дій над усіма елементами в правильному порядку.

Також реалізувати заповнення списку заданою кількістю випадкових елементів.

Варіант (структура даних та порядок зберігання):

17) Комплексні числа z=x+y*i (з дійсними компонентами x,y), спершу за модулем величини f(z)=z^2-20*z+22, за зростанням