Лабораторна робота №1.
Лінійні масиви. Пошук. Сортування

Мета роботи: Навчитись розробляти програми, що використовують сортування та пошук, та аналізувати їхню ефективність

Завдання

1. Сформувати масив з 10^5 випадкових цілих чисел з діапазону [0, 10^5+N*1000) (N – номер варіанту за списком групи).
2. Порахувати кількість чисел у заданому масиві, що потрапляють у діапазон [N,2*N]
3. Визначити найменший індекс елементу масиву, що дорівнює 1000+N. Якщо масив не містить такого елементу, вивести -1.
4. Згідно варіанту, розробити програму, що виконує сортування створеного масиву. Метод сортування обрати згідно таблиці. Виміряти час роботи алгоритму сортування.
5. Згенерувати масив аналогічно до п. 1 збільшивши кількість елементів масиву у 2 рази. Відсортувати новий масив методом з п.4. Порівняти час роботи сортування для двох масивів.
6. Повторити завдання п.3 для відсортованого масиву, використовуючи алгоритм бінарного пошуку. Порівняти кількість операцій порівнянь, які необхідно виконати для пошуку елементу у впорядкованому та невпорядкованому масиві.

Варіанти	**Метод сортування**
1,4,7,10,13,16,19,22,25,28	Метод вставки
2,5,8,11,14,17,20,23,26,29	Метод обмінів
3,6,9,12,15,18,21,24,27,30	Метод вибору

Пояснення до виконання

Якщо завдання виконується мовою Java, для генерування масиву випадкових чисел можна скористатись методами генерації випадкових чисел класу Random:

`Random rnd = new Random();` - створення генератору випадкових чисел

`rnd.nextInt(bound);` – випадкове ціле число з діапазону [0, bound)

`rnd.ints(N, origin, bound).toArray();` – масив з N випадкових цілих чисел з діапазону [origin, bound)

Для вимірювання часу роботи програми можна скористатися методом currentTimeMillis() класу System. Це можна зробити за такою схемою:

```
long start = System.currentTimeMillis();
// обчислювальний процес, час роботи якого треба виміряти
long finish = System.currentTimeMillis();
System.out.println("Time = " + (finish-start));
```

Якщо завдання виконується мовою C++, для генерування випадкових чисел доцільно використовувати описані в бібліотеці <random> механізми. Наприклад, так:

```
// створюємо Генератор Псевдовипадкових Чисел та ініціалізуємо його значенням  time(0)
std::mt19937 gen(time(0));
// створюємо розподіл uid та ініціалізуємо його початковими значеннями
std::uniform_int_distribution<int> uid(0, 100_000);
// наша змінна отримує значення випадкового числа з указаного розподілу
int randomNumber = uid(gen)
```

Використання функцій rand(), srand() не рекомендується, оскільки вони дають погану різноманітність даних та невеликий діапазон значень.

Якщо завдання виконується мовою Kotlin, можна скористатись наведеним у цьому проекті шаблоном програми

Для вимірювання часу роботи програми можна скористатися методом currentTimeMillis() класу System. Це можна зробити за такою схемою:

```
val start = System.currentTimeMillis()
// обчислювальний процес, час роботи якого треба виміряти
val finish = System.currentTimeMillis()
println("Time = ${finish - start}")
```