# Міністерство освіти і науки України Львівський національний університет імені Івана Франка

**Факультет прикладної математики та інформатики**

Кафедра програмування

Лабораторна робота №1

# АЛГОРИТМИ СОРТУВАННЯ

з курсу “Алгоритми та структури даних”

Виконав: студент групи ПМІ-16 Процай Іван Сергійович

Львів – 2024

# Сортування підрахунком

**Складність роботи алгоритму:** 𝑂(𝑁 + 𝐾) **Просторова складність:** 𝑂(𝑁 + 𝐾) **Стабільність алгоритму:** стабільний

# Принцип роботи алгоритму:

1. Створюємо допоміжний масив, який міститиме всі значення від нуля до максимального елемента масиву,який потрібно посортувати. Заповнюємо цей масив нулями.
2. Рахуємо кількість однакових елементів та записуємо кількість входжень у допоміжний масив. Місце знаходимо за індексами (значення головного масиву є індексом у допоміжному).
3. Проходимося по основному масиву і вписуємо у нього значення з допоміжного, стільки разів, скільки це значення зустрічається в допоміжному, робимо це поки значення у допоміжному не стане нулем, тоді переходимо до наступного елементу з допоміжного масиву.

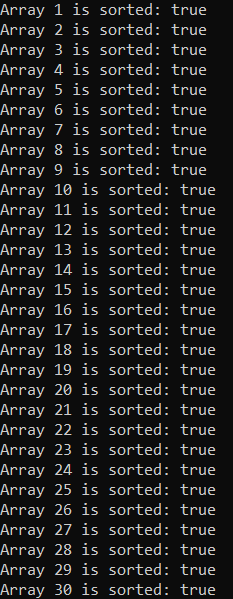
# Приклад №1

Дано: випадково згенерований масив чисел довжиною 20, спершу виводиться початковий масив, потім виводиться повідомлення, що масив посортовано, після цього виконується перевірка чи посортовано масив(для цього я написав окрему функцію)



# Приклад №2

Дано: 30 випадкових масивів розміром 20 елементів, перевірка правильності сортування здійснюється окремою функцією



**Висновок:** алгоритм сортування підрахунком доцільно використовувати при малій кількості різних елементів у масиві даних. Діапазон значень (максимальне число) має бути достатньо малим порівняно із загальною кількістю елементів.

# Сортування злиттям

**Найгірший час роботи алгоритму: O(n\*log(n))**

**Найкращий: O(n\*log(n))**

**Стабільність алгоритму:** стабільний

# Принцип роботи алгоритму:

Принцип роботи алгоритму MergeSort полягає в рекурсивному поділі масиву на менші підмасиви, сортуванні їх окремо і потім злитті відсортованих підмасивів у відсортований вихідний масив.

1. Функція MergeSort приймає вхідний масив arr, ліву та праву границі left і right (це важливо оскільки пізніше ми будемо викликати її рекурсивно).
2. Якщо ліва границя менша за праву(тобто дані введено коректно), обчислюється середина масиву.
3. Якщо масив складається з двох елементів, то вони порівнюються та, якщо потрібно, обмінюються(тобто якщо елементи два елементи стоять не в порядку зростання).
4. Інакше(масив не більший ніж 2 елементи) рекурсивно викликається MergeSort для лівої та правої половин масиву.
5. Після чого викликається функція Merge, яка зливає відсортовані під масиви.

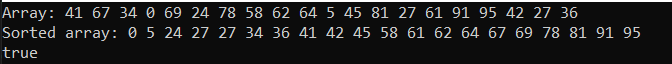
Функція Merge злиття двох відсортованих підмасивів у відсортований масив:

1. Створюється допоміжний масив storage.
2. Поки індекси лівого та правого підмасиву не дорівнюють їх кінцевим індексам, порівнюються елементи та записуються у storage.
3. Якщо один з підмасивів закінчується раніше, залишки копіюються у storage.
4. На останньому кроці, елементи з storage записуються назад у вихідний масив починаючи з початкового індексу.

Цей процес повторюється рекурсивно для кожного підмасиву, доки не буде досягнуто базового випадку (коли лише один елемент у підмасиві), що гарантує, що всі підмасиви будуть відсортовані, а потім злиттям їх у вихідний впорядкований масив отримуємо остаточний результат сортування.

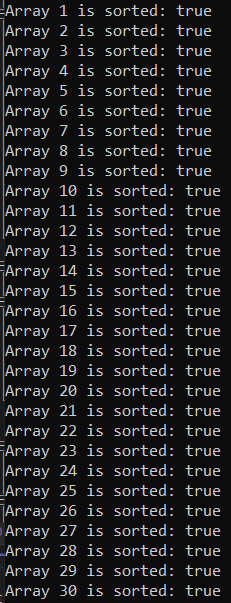
# Приклад №1

Дано: випадково згенерований масив чисел довжиною 20, спершу виводиться початковий масив, потім виводиться повідомлення, що масив посортовано, після цього виконується перевірка чи посортовано масив(для цього я написав окрему функцію)



# Приклад №2

Дано: 30 випадкових масивів розміром 20 елементів, перевірка правильності сортування здійснюється окремою функцією



Масив сортування MergeSort є ефективним алгоритмом сортування, який базується на принципі рекурсивного поділу та злиття підмасивів. Він гарантує стабільність сортування та має часову складність O(n log n), що робить його одним з найефективніших алгоритмів для великих масивів даних. MergeSort також добре підходить для сортування великих об'ємів даних через свою рекурсивну природу, яка дозволяє розподілити сортування на менші частини.

Однак, важливо враховувати, що MergeSort вимагає додаткового місця пам'яті для збереження проміжних результатів підчас злиття підмасивів. Також, у випадку великого об'єму даних, може бути витрачено багато часу на копіювання та злиття підмасивів.

Наостанок, вважаю необхідним довести коректність роботи функції, що перевіряє чи масив посортовано.

**Приклад 1**

Вхідні дані: {1, 2, 3, 4, 5} (посортований масив)

Результат



**Приклад 2**

Вхідні дані: { 2, 3, 1, 5, 4} (непосортований масив)

Результат

