НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп’ютерних систем

***КУРСОВА РОБОТА***

***з дисципліни "Структури даних і алгоритми"***

Виконала: Данелія С.І.

Група: КB-41

Номер залікової книжки:

КВ14843484

Допущений до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 семестр 2024/2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп’ютерних систем

Узгоджено ЗАХИЩЕНА "\_\_"\_\_\_\_\_\_\_\_\_2025р.

Керівник роботи з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_/Марченко О.І./ \_\_\_\_\_\_\_\_ /Марченко О.І./

***Дослідження ефективності методів сортування (Алгоритм сортування №7 методу прямого вибору,сортування №8 методу прямого вибору, сортування №2 Шелла)на багатовимірних масивах***

Виконавець роботи:

Данелія Софіко Ісідорівна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 р.

***ТЕХНІЧНЕ ЗАВДАННЯ***

***на курсову роботу з дисципліни***

***“Структури даних і алгоритми”***

**ТЕХНІЧНЕ ЗАВДАННЯ НА КУРСОВУ РОБОТУ**

**I.** Описати теоретичні положення, від яких відштовхується дослідження, тобто принцип та схему роботи кожного із досліджуваних алгоритмів сортування для одновимірного масиву, навести загальновідомі властивості цих алгоритмів та оцінки кількості операцій порівняння та присвоєння для них.

**II.** Скласти алгоритми рішення задачі сортування в багато-вимірному масиві заданими за варіантом методами та написати на мові програмування за цими алгоритмами програму, яка відповідає вимогам розділу «Вимоги до програми курсової роботи».

**III.** Виконати налагодження та тестування коректності роботи написаної програми.

**IV.** Провести практичні дослідження швидкодії складених алгоритмів, тобто виміри часу роботи цих алгоритмів для різних випадків та геометричних розмірів багатовимірних масивів.

**V.** За результатами досліджень скласти порівняльні таблиці за різними ознаками.

Одна таблиця результатів (вимірів часу сортування впорядкованого, випадкового і обернено-впорядкованого масиву) для масиву з заданими геометричними розмірами повинна бути такою:

Таблиця № для масиву A[P,M,N], де P= ; M= ; N= ;

|  |  |  |  |
| --- | --- | --- | --- |
|  | Впорядко-ваний | Невпорядко-ваний | Обернено впорядкований |
| Назва алгоритму 1 |  |  |  |
| Назва алгоритму 2 |  |  |  |
| Назва алгоритму 3 |  |  |  |

Для варіантів курсової роботи, де крім алгоритмів порівнюються також способи обходу, в назвах рядків таблиць потрібно вказати як назви алгоритмів, так і номери способів обходу.

Для виконання ґрунтовного аналізу алгоритмів потрібно зробити виміри часу та побудувати таблиці для декількох масивів з різними геометричними розмірами.

Зробити виміри часу для стандартного випадку одномірного масиву, довжина якого вибирається такою, щоб можна було виконати коректний порівняльний аналіз з рішенням цієї ж задачі для багатовимірного масиву.

Кількість необхідних таблиць для масивів з різними геометричними розмірами залежить від задачі конкретного варіанту курсової роботи і вибираються так, щоб виконати всебічний та ґрунтовний порівняльний аналіз заданих алгоритмів.

Рекомендації випадків дослідження з різними геометричними розмірами масивів наведені у розділі «Випадки дослідження».

**VI.** Для наочності подання інформації за отриманими результатами рекомендується також будувати стовпчикові діаграми та графіки.

**VII.** Виконати порівняльний аналіз поведінки заданих алгоритмів за отриманими результатами (вимірами часу):

1. для одномірного масиву відносно загальновідомої теорії;
2. для багатовимірних масивів відносно результатів для одномірного масиву;
3. для заданих алгоритмів на багатовимірних масивах між собою;
4. дослідити вплив різних геометричних розмірів багатовимірних масивів на поведінку алгоритмів та їх взаємовідношення між собою;
5. **для всіх вищезазначених пунктів порівняльного аналізу пояснити, ЧОМУ алгоритми в розглянутих ситуаціях поводять себе саме так, а не інакше**.

**VIII.** Зробити висновки за зробленим порівняльним аналізом.

**IX.** Програму курсової роботи під час її захисту ОБОВ’ЯЗКОВО мати при собі на електронному носії інформації.

**Варіант №104**

**Задача**

Впорядкувати окремо кожен переріз тривимірного масива Аrr3D [P,M,N] таким чином: переставити стовпчики перерізу за незменшенням сум їх елементів.

**Досліджувані методи та алгоритми**

* Алгоритм сортування №7 методу прямого вибору
* Алгоритм сортування № 8 методу прямого вибору
* Алгоритм №2 методу сортування Шелла (спосіб реалізації на основі гібридного алгоритму «вставка-обмін»)

**Способи обходу**

В якості першого етапу сортування сформувати додатковий вектор Sum, довжина якого дорівнює кількості стовпчиків і значеннями якого є суми елементів відповідних стовпчиків. Використовуючи елементи вектора Sum як ключі сортування, переставляти відповідні стовпчики кожен раз, коли треба переставляти ключі.

**Випадки дослідження**

1. Залежність часу роботи алгоритмів від довжини стовпчиків масива.
2. Залежність часу роботи алгоритмів від форми перерізів масива
3. Залежність часу роботи алгоритмів від кількості ключів у кожному перерізі масива при однаковій загальній кількості ключів у всьому масиві

**Опис теоретичних положень**

***Алгоритм сортування №7 методу прямого вибору.***

**Принцип роботи:**

Алгоритм працює за принципом того, що сортування відбувається одночасно з двох сторін масиву. Тобто він знаходить мінімальний і максимальний елемент в невідсортованій частині масиву та переміщує їх на коректні місця.

**Під час роботи:**

1)Спочатку визначається ліва і права границя (L=0, R=n-1).Потім йде цикл поки ліва границя менша за праву(тобто поки вони не зустрінуться).

2)Ми визначаємо 4 змінні- imax, imin-в яких буде індекс нашого максимального або мінімального елемента,max,min-в яких вже буде саме число, яке відповідає одному з цих критеріїв.

3)Присвоюємо imax, imin=L,max,min=L[i]. Далі починаємо новий цикл, який буде визначати нашу невідсортовану частину, і в ньому записуємо: якщо наш елемент L+1 менший за поточний мінімальний, то він тепер найменший і запам’ятовуємо його індекс, якщо це неправда, робимо такі ж дії, але для знаходження максимального числа.

4)Далі у нас іде зміна того де буде починатися наша ліва і права границя, виходить :якщо індекс мінімального не дорівнює індексу лівої границі, то цей мінімальний тепер буде виконувати функцію лівої границі, майже теж саме для максимального значення та правої границі, але там ще додається умова якщо imax==L, тобто максимальний елемент був спочатку на місці, де знаходився мінімальний, то після першої заміни (A[L]=Min) цей елемент уже перенесений. Тому треба оновити A[imin] (де зараз знаходиться Min) значенням A[R], а потім записати Max у A[R].

5)Встановлюємо, що наша границя тепер L+1 і R-1, і повторюємо всі вищезазначені дії доки наш масив повністю не буде відсортований.

**Схема роботи:**

Початковий стан:

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.Зображення, що містить текст, ряд, схема, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Відсортований масив:

Зображення, що містить текст, знімок екрана, ряд, число

Вміст, створений ШІ, може бути неправильним.

**Характеристика:**

Цей метод сортування відрізняється від звичайного прямого вибору №1 тим, що замість знаходження мінімального елемента за один прохід, він шукає одразу і мінімальний і максимальний елемент, також тут додається права і ліва границя, які поступово зсуваються до центру. На кожному кроці алгоритм шукає мінімальний елемент з лівої частини масиву і максимальний – з правої частини, а потім ставить їх на відповідні місця. Якщо мінімальний елемент не знаходиться на лівій межі, він замінюється з елементом на позиції лівої межі, а якщо максимальний елемент не знаходиться на правій межі — він замінюється з елементом на правій межі. Цей алгоритм значно зменшує кількість проходів по масиву і цим ефективніше порівняно з класичними видами сортування прямого вибору. Але якщо зрівнювати його з такими сортуваннями як- швидке сортування цей метод не має кращих показників, коли масиви великі й сильно випадкові. Загалом можна рекомендувати використовувати це сортування для обробки невеликих масивів, або майже впорядкованих.

**Кількість порівнянь C=O(n2), а кількість присвоєнь має до квадратичну залежність R=O(nln(n))**

**Код на С:**

clock\_t Select7**(**int **\***A**,** int N**)**

**{**

int Min**,** Max**;**

int L**,** R**,** imin**,** imax**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

L**=**0**;**

R**=**N**-**1**;**

**while** **(**L**<**R**)**

**{**

Min**=**A**[**L**];**

imin**=**L**;**

Max**=**A**[**L**];**

imax**=**L**;**

**for(**int i**=**L**+**1**;** i**<**R**+**1**;** i**++)**

**{**

**if** **(**A**[**i**]** **<** Min**)**

**{**

Min**=**A**[**i**];**

imin**=**i**;**

**}**

**else** **if** **(**A**[**i**]** **>** Max**)**

**{**

Max**=**A**[**i**];**

imax**=**i**;**

**}**

**}**

**if** **(**imin**!=**L**)**

**{**

A**[**imin**]=**A**[**L**];**

A**[**L**]=**Min**;**

**}**

**if** **(**imax**!=**R**)**

**{**

**if** **(**imax**==**L**)** A**[**imin**]=**A**[**R**];**

**else** A**[**imax**]=**A**[**R**];**

A**[**R**]=**Max**;**

**}**

L**=**L**+**1**;**

R**=**R**-**1**;**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

***Алгоритм сортування №8 методу прямого вибору.***

**Принцип роботи:**

Алгоритм працює за принципом знаходження мінімального і максимального значення у невідсортованій частині масиву та переміщує їх на коректні місця.

**Під час роботи:**

1)Спочатку визначається ліва і права границя (L=0, R=n-1).Потім йде цикл поки ліва границя менша за праву(тобто поки вони не зустрінуться).

2)Ми визначаємо 2 змінні- imax, imin-в яких буде індекс нашого максимального або мінімального елемента, та комірку tmp- яка буде допомагати переміщувати елементи один з одним.

3)Присвоюємо imax, imin=L. Далі починаємо новий цикл, який буде визначати нашу невідсортовану частину, і в ньому записуємо: якщо наш елемент L+1 менший за поточний мінімальний(L), то він тепер найменший і запам’ятовуємо його індекс, якщо це неправда, робимо такі ж дії, але для знаходження максимального числа.

4)Потім, якщо індекс мінімального не дорівнює лівій границі, то ми кладемо в нашу комірку цей новий мінімальний, далі стару ліву границю кладемо на місце поточного мінімального, а на місце лівої границі перейде значення з комірки. Повторюємо ті ж дії для максимального, з одним нюансом, що тепер у нас ще додається перевірка на те, чи не дорівнює наш максимальний лівій границі, якщо це так, то переставляємо на місце правої границі.

5)Встановлюємо, що наша границя тепер L+1 і R-1, і повторюємо всі вищезазначені дії доки наш масив повністю не буде відсортований.

**Схема роботи:**

Початковий стан:

**Зображення, що містить текст, Шрифт, число, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить схема, текст, ряд, План

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить схема, текст, ряд

Вміст, створений ШІ, може бути неправильним.**

Зображення, що містить схема, текст

Вміст, створений ШІ, може бути неправильним.

Відсортований масив:

Зображення, що містить текст, Шрифт, знімок екрана, ряд

Вміст, створений ШІ, може бути неправильним.

**Характеристика:**

Цей метод сортування відрізняється від звичайного прямого вибору №1 тим, що замість знаходження мінімального елемента за один прохід, він шукає одразу і мінімальний і максимальний елемент, також тут додається права і ліва границя, які поступово зсуваються до центру. Але на відміну від алгоритму №7 вибору, тут є одна комірка пам’яті, яка допомагає пересувати елементи по масиву, замість двох, тому для великих масивів в деяких випадках це буде економити пам’ять, але не дуже сильно, тож ці алгоритми можна сказати рівносильні.

**Кількість порівнянь C=O(n2), а кількість присвоєнь має до квадратичну залежність R=O(nln(n))**

**Код на С:**

clock\_t Select8**(**int **\***A**,** int N**)**

**{**

int L**,** R**,** imin**,** imax**,** tmp**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

L**=**0**;**

R**=**N**-**1**;**

**while** **(**L**<**R**)**

**{**

imin**=**L**;**

imax**=**L**;**

**for(**int i**=**L**+**1**;** i**<**R**+**1**;** i**++)**

**if** **(**A**[**i**]<**A**[**imin**])** imin**=**i**;**

**else** **if** **(**A**[**i**]>**A**[**imax**])** imax**=**i**;**

**if** **(**imin**!=**L**)**

**{**

tmp**=**A**[**imin**];**

A**[**imin**]=**A**[**L**];**

A**[**L**]=**tmp**;**

**}**

**if** **(**imax**!=**R**)**

**if** **(**imax**==**L**)**

**{**

tmp**=**A**[**imin**];**

A**[**imin**]=**A**[**R**];**

A**[**R**]=**tmp**;**

**}**

**else**

**{**

tmp**=**A**[**imax**];**

A**[**imax**]=**A**[**R**];**

A**[**R**]=**tmp**;**

**}**

L**=**L**+**1**;**

R**=**R**-**1**;**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

***Алгоритм №2 методу сортування Шелла (спосіб реалізації на основі гібридного алгоритму «вставка-обмін»)***

**Принцип роботи:**

В кожен момент часу масив вважається поділеним на групи, серед яких спостерігається відносна відсортованість.

К-ть етапів та оптимальних відстаней розраховуються за формулою Хіббарда: k=2^p – 1 (1, 3, 7, 15, 31 … у зворотньому порядку).

**Етап 1:**

Методом обміну-вставки відбувається сортування елементів, які розташовані на відстані в позицій, інші елементи не сортуються. Таких груп елеметів буде , при цьому кожна група сортується окремо

**Етап 2:**

Методом обміну-вставки відбувається сортування елементів, які розташовані на відстані в позицій, інші елементи не сортуються. Таких груп елеметів буде , при цьому кожна група сортується окремо

**Етап 3:**

Методом обміну-вставки відбувається сортування елементів, які розташовані на відстані в позицій, інші елементи не сортуються. Таких груп елеметів буде , при цьому кожна група сортується окремо

**Етап 4:**

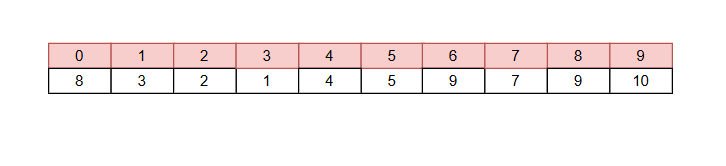
Методом обміну-вставки відбувається сортування елементів, які розташовані на відстані в 1 позицію.

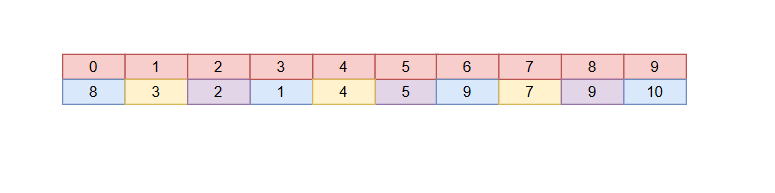
1. Порівняння чергового елемента з елементом на відстані в -k позицій,
2. Якщо даний елемент є меншим за попередній з групи, запам’ятовуємо його в комірку пам’яті, на його місце ставимо попередній, вставляємо взятий елемент на місце попереднього (тобто відбувається обмін та вставка меншого елемента на лівішу позицію)
3. Переставляємо лічильник на –k позицію лівіше.
4. Повторюємо пункти 1-3 допоки не досягнемо відсортованості по даних групах
5. Виконуємо проходи, що залишились, за аналогією

Спосіб сортування: вставка-обмін для 1-ого елемента 1-ої групи, 1-ого елемента 2-ої і тд. Далі аналогічно для 2-ого елемента 1-ої групи, 2-ого елемента 2-ої і так далі.

**Схема роботи:**

Початковий масив:

**Розділений масив на частини k=3:**

****

**Зображення, що містить схема, ряд, текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить схема, ряд, текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить ряд, текст, схема, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

Зображення, що містить схема, ряд, текст

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.Почався новий етап k=1

Зображення, що містить текст, ряд, схема, знімок екрана

Вміст, створений ШІ, може бути неправильним.

**Зображення, що містить текст, схема, ряд, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить ряд, текст, лінійка

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, ескіз, малюнок, ряд

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, ряд, схема, Паралель

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить схема, ряд, текст, План

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить лінійка, ряд, ескіз, мультфільм

Вміст, створений ШІ, може бути неправильним.**

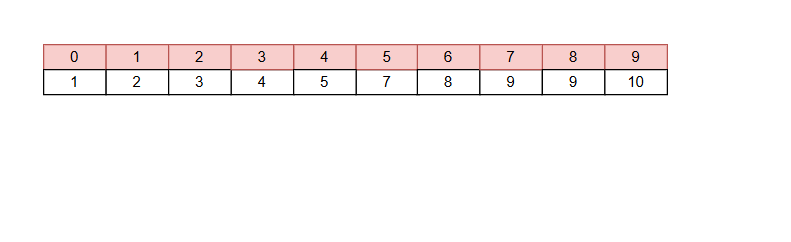
**Зображення, що містить текст, схема, Шрифт, ряд

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить ряд, схема

Вміст, створений ШІ, може бути неправильним.**

**Відсортований масив**

****

**Характеристика :**

Переваги даного алгоритму:

- Алгоритм Хіббарда дозволяє швидко здійснювати пошук та перестановку елементів, які знаходяться на значній відстані один від одного, у правильному порядку відносно сортування. Це дозволяє досягти часткової відсортованості навіть після першого етапу сортування. Крім того, імовірно, що елементи між групами також будуть перебувати у відносній відсортованості.

-У порівнянні з класичним сортуванням Шелла, яке використовує зсув елементів з подальшою вставкою найменшого елемента на свою позицію, гібридний підхід алгоритму Хіббарда полягає у використанні вставка-обміну елементів з використанням додаткової комірки пам'яті. Це дозволяє ефективніше виконувати операції сортування на великих відстанях і зменшує кількість операцій переміщення елементів

Цей підхід може бути особливо корисним у випадках, коли необхідно сортувати великі обсяги даних з великими відстанями між елементами, таких як великі масиви або великі файли даних.

Код на С

clock\_t Shell\_2**(**int **\***A**,** int N**)**

**{**

int tmp**,** t**,** j**,** k**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

**if** **(**N**<**4**)** t**=**1**;**

**else** t**=(**int**)**log2f**((**float**)**N**)-**1**;**

Stages**[**t**-**1**]=**1**;**

**for** **(**int i**=**t**-**2**;** i**>=**0**;** i**--)**

Stages**[**i**]=**2**\***Stages**[**i**+**1**]+**1**;**

**for** **(**int p**=**0**;** p**<**t**;** p**++)**

**{**

k**=**Stages**[**p**];**

**for** **(**int i**=**k**;** i**<**N**;** i**++)**

**{**

j**=**i**;**

**while** **(**j**>=**k **&&** A**[**j**]<**A**[**j**-**k**])**

**{**

tmp**=**A**[**j**];**

A**[**j**]=**A**[**j**-**k**];**

A**[**j**-**k**]=**tmp**;**

j**=**j**-**k**;**

**}**

**}**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

**Схема імпорту/експорту модулів**

**Зображення, що містить текст, схема, Паралель, ряд

Вміст, створений ШІ, може бути неправильним.**

**Структурна схема взаємовикликів процедур/функцій**

**Зображення, що містить текст, схема, План, Паралель

Вміст, створений ШІ, може бути неправильним.**

**Опис призначення процедур та функцій**

1. ***Common*** – модуль спільної інформації
2. оголошення розмірів багатовимірного масиву та вектору
3. application\_RAM () – виділення пам’яті для динамічного масиву
4. free\_RAM () – вивільнення пам’яті від динамічного масиву
5. ***Main*** – головний модуль програми.
6. ***Menu*** – модуль меню
7. Викликає допоміжні меню для кожного випадку
8. Викликає процеси сортування а також функцію table(таблиця)
9. ***Measurement*** – модуль усереднення вимірів часу для вектору та тривимірного масиву
10. MeasurementProcessing() – функція обробки та усереднення значень вимірів часу роботи заданого алгоритму
11. Select7\_measurement\_vec() – функція для заповнення масиву вимірів часу алгоритмом “метод прямого вибору №7” (для вектору)
12. Select8\_measurement\_vec() – функція для заповнення масиву вимірів часу алгоритмом “метод прямого вибору №8” (для вектору)
13. Shell\_2\_measurement\_vec() – функція для заповнення масиву вимірів часу алгоритмом №2 методу сортування Шелла (для вектору)
14. Select7\_measurement\_arr() – функція для заповнення масиву вимірів часу алгоритмом “метод прямого вибору №7” (для масиву)
15. Select8\_measurement\_arr() – функція для заповнення масиву вимірів часу алгоритмом “метод прямого вибору №8” (для масиву)
16. Shell\_2\_measurement\_arr() – функція для заповнення масиву вимірів часу алгоритмом №2 методу сортування Шелла (для масиву)
17. ***filling\_vector*** – модуль заповнення векторів обраним варіантом відсортованості
18. filling\_ordered () – функція для впорядкованого заповнення вектору
19. filling\_random () – функція для заповнення вектору рандомними значеннями
20. filling\_back\_ordered () – функція для обернено впорядкованого заповнення вектору
21. ***filling\_arr*** – модуль заповнення масивів обраним варіантом відсортованості
22. Filling\_ordered () – функція для впорядкованого заповнення масиву
23. Filling\_random () – функція для заповнення масиву рандомними значеннями
24. Filling\_back\_ordered () – функція для обернено впорядкованого заповнення масиву
25. ***sort\_vector*** – модуль сортування векторів обраним алгоритмом
26. Select7\_vec () – функція для сортування вектору алгоритмом “метод прямого вибору №7”
27. Select8\_vec () – функція для сортування вектору алгоритмом “метод прямого вибору №8”
28. Shell\_2\_vec () – функція для сортування вектору №2 методу сортування Шелла
29. ***sort\_arr*** – модуль сортування тривимірного масиву обраним алгоритмом

1)Select7\_arr () – функція для сортування масиву алгоритмом “метод прямого вибору №7”

2)Select8\_arr () – функція для сортування масиву алгоритмом “метод прямого вибору №8”

3)Shell\_2\_arr () – функція для сортування масиву алгоритмом №2 методу сортування Шелла

1. *table* – модуль побудови таблиці для всіх алгоритмів та випадків відсортованості як для вектору так і для багатовимірного масиву

* Table () – функція для побудови таблиці та пакетного запуску всіх алгоритмів для всіх випадків відсортованості як для вектору так і для багатовимірного масиву

**КОД ПРОГРАМИ**

**Common.h**

#ifndef COMMON\_H\_INCLUDED

#define COMMON\_H\_INCLUDED

// змінення вимірів для масива та вектору

#define N 200000

#define M 10

#define P 20

#define VECTOR\_SIZE 5000 // визначення розміру вектору

int**\*\*\*** Arr3D**;** // оголощення тривимірного масиву

int Vector **[**VECTOR\_SIZE**];** //оголошення вектору

void application\_RAM **();**

void free\_RAM**();**

int Sum**[**N**];**

#endif // COMMON\_H\_INCLUDED

**Common.c**

#include <stdio.h>

#include <stdlib.h>

// виклик модулів

#include "common.h"

void application\_RAM**()** // виділення пам'яті

**{**

Arr3D **=** **(**int**\*\*\*)** malloc**(**P**\*sizeof(**int**\*\*));**

**for** **(**int k**=**0**;** k**<**P**;** k**++)**

**{**

Arr3D**[**k**]** **=** **(**int**\*\*)** malloc**(**M**\*sizeof(**int**\*));**

**for** **(**int i**=**0**;** i**<**M**;** i**++)**

Arr3D**[**k**][**i**]** **=** **(**int**\*)** malloc**(**N**\*sizeof(**int**));**

**}**

**}**

void free\_RAM**()** // звільнення пам'яті

**{**

**for** **(**int k**=**0**;** k**<**P**;** k**++)**

**{**

**for** **(**int i**=**0**;** i**<**M**;** i**++)**

free**(**Arr3D**[**k**][**i**]);**

free**(**Arr3D**[**k**]);**

**}**

free**(**Arr3D**);**

**}**

**Filling.arr.h**

#ifndef FILLING\_ARR\_H\_INCLUDED

#define FILLING\_ARR\_H\_INCLUDED

#include <time.h>

void fill\_ordered\_arr**();**

void fill\_random\_arr**();**

void fill\_back\_arr**();**

#endif // FILLING\_ARR\_H\_INCLUDED

**Filling.arr.c**

#include <stdio.h>

#include <stdlib.h>

// виклик модулів

#include "filling.arr.h"

#include "common.h"

void fill\_ordered\_arr**()** // заповнення масива "впорядкований"

**{**

int number**=**0**;**

**for** **(**int k**=**0**;** k**<**P**;** k**++)**

**{**

**for** **(**int j**=**0**;** j**<**N**;** j**++)**

**{**

**for** **(**int i**=**0**;** i**<**M**;** i**++)**

**{**

Arr3D**[**k**][**i**][**j**]** **=** number**++;**

**}**

**}**

**}**

**}**

void fill\_random\_arr**()** // заповнення масива "випадкові значення"

**{**

**for** **(**int k**=**0**;** k**<**P**;** k**++)**

**{**

**for** **(**int j**=**0**;** j**<**N**;** j**++)**

**{**

**for** **(**int i**=**0**;** i**<**M**;** i**++)**

**{**

Arr3D**[**k**][**i**][**j**]** **=** rand**()** **%** **(**P**\***M**\***N**);**

**}**

**}**

**}**

**}**

void fill\_back\_arr**()** // заповнення масива "обернено впорядкований"

**{**

int number **=** P**\***M**\***N**;**

**for** **(**int k**=**0**;** k**<**P**;** k**++)**

**{**

**for** **(**int j**=**0**;** j**<**N**;** j**++)**

**{**

**for** **(**int i**=**0**;** i**<**M**;** i**++)**

**{**

Arr3D**[**k**][**i**][**j**]** **=** number**--;**

**}**

**}**

**}**

**}**

**Filling.vector.h**

#ifndef FILLING\_VECTOR\_H\_INCLUDED

#define FILLING\_VECTOR\_H\_INCLUDED

#include <time.h>

void fillVector\_ordered**();**

void fillVector\_random**();**

void fillVector\_back**();**

#endif // FILLING\_VECTOR\_H\_INCLUDED

**Filling.vector.c**

#include <stdio.h>

#include <stdlib.h>

// виклик модулів

#include "filling.vector.h"

#include "common.h"

void fillVector\_ordered**()** // заповнення вектору "впорядкований"

**{**

**for** **(**int i **=** 0**;** i **<** VECTOR\_SIZE**;** i**++)**

**{**

Vector**[**i**]** **=** i **+** 1**;**

**}**

**}**

void fillVector\_random**()** // заповнення вектору "випадкові значення"

**{**

**for** **(**int i **=** 0**;** i **<** VECTOR\_SIZE**;** i**++)**

**{**

Vector**[**i**]** **=** rand**()%**VECTOR\_SIZE**;**

**}**

**}**

void fillVector\_back**()** // заповнення вектору "обернено впорядкований"

**{**

**for** **(**int i **=** 0**;** i **<** VECTOR\_SIZE**;** i**++)**

**{**

Vector**[**i**]** **=** VECTOR\_SIZE **-** i**;**

**}**

**}**

**Measurement.h**

#ifndef MEASUREMENT\_H\_INCLUDED

#define MEASUREMENT\_H\_INCLUDED

// Загальна кількість вимірів часу роботи алгоритма

#define measurements\_number 28

// Кількість відкинутих початкових вимірів

#define rejected\_number 2

// Кількість відкинутих вимірів з мінімільними значеннями.

// Вона ж дорівнює кількості відкинутих вимірів

// з максимальними значеннями.

#define min\_max\_number 3

#include <time.h>

// Масив значень часу роботи алгоритма

extern clock\_t Res**[**measurements\_number**];**

// Функція обробки і усереднення значень вимірів

// часу роботи алгоритма.

// Повертає усереднене значення часу роботи алгоритма

float MeasurementProcessing**();**

void Select7\_measurement\_vec**(**char fil**);**

void Select8\_measurement\_vec**(**char fil**);**

void Shell\_2\_measurement\_vec**(**char fil**);**

void Select7\_measurement\_arr**(**char fil**);**

void Select8\_measurement\_arr**(**char fil**);**

void Shell\_2\_measurement\_arr**(**char fil**);**

#endif // MEASUREMENT\_H\_INCLUDED

**Measurement.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// виклик модулів

#include "common.h"

#include "measurement.h"

#include "filling.arr.h"

#include "filling.vector.h"

#include "sort.arr.h"

#include "sort.vector.h"

clock\_t Res**[**measurements\_number**];**

// функція обробки і усереднення значень вимірів часу роботи алгоритма

float MeasurementProcessing**()**

**{**

long int Sum**;**

float AverageValue**;**

clock\_t buf**;**

int L **=** rejected\_number**,** R **=** measurements\_number **-** 1**;**

int k **=** rejected\_number**;**

**for** **(**int j**=**0**;** j **<** min\_max\_number**;** j**++)**

**{**

**for** **(**int i **=** L**;** i **<** R**;** i**++)**

**{**

**if** **(**Res**[**i**]** **>** Res**[**i **+** 1**])**

**{**

buf **=** Res**[**i**];**

Res**[**i**]** **=** Res**[**i **+** 1**];**

Res**[**i **+** 1**]** **=** buf**;**

k **=** i**;**

**}**

**}**

R **=** k**;**

**for** **(**int i **=** R **-** 1**;** i **>=** L**;** i**--)**

**{**

**if** **(**Res**[**i**]** **>** Res**[**i **+** 1**])**

**{**

buf **=** Res**[**i**];**

Res**[**i**]** **=** Res**[**i **+** 1**];**

Res**[**i **+** 1**]** **=** buf**;**

k **=** i**;**

**}**

**}**

L **=** k **+** 1**;**

**}**

Sum**=**0**;**

**for** **(**int i **=** rejected\_number **+** min\_max\_number**;** i **<** measurements\_number **-** min\_max\_number**;** i**++)**

Sum **=** Sum **+** Res**[**i**];**

AverageValue **=** **(**float**)**Sum**/(**float**)(**measurements\_number **-** 2**\***min\_max\_number **-** rejected\_number**);**

printf**(**""**);**

**return** AverageValue**;**

**}**

// Для Масива Arr3D

void Select7\_measurement\_arr**(**char fil**)** // функція для заповнення масиву вимірів часу (Select7, масив)

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

**if** **(**fil **==** 'o'**)**

**{**

application\_RAM**();**

fill\_ordered\_arr**();**

Res**[**i**]** **=** Select7\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

application\_RAM**();**

fill\_random\_arr**();**

Res**[**i**]** **=** Select7\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

application\_RAM**();**

fill\_back\_arr**();**

Res**[**i**]** **=** Select7\_arr**();**

free\_RAM**();**

**}**

**}**

**}**

void Select7\_measurement\_vec**(**char fil**)** // функція для заповнення масиву вимірів часу (Select7, вектор)

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

**if** **(**fil **==** 'o'**)**

**{**

fillVector\_ordered**();**

Res**[**i**]** **=** Select7\_vec**();**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

fillVector\_random**();**

Res**[**i**]** **=** Select7\_vec**();**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

fillVector\_back**();**

Res**[**i**]** **=** Select7\_vec**();**

**}**

**}**

**}**

void Select8\_measurement\_arr**(**char fil**)** // функція для заповнення масиву вимірів часу (Select8, масив)

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

**if** **(**fil **==** 'o'**)**

**{**

application\_RAM**();**

fill\_ordered\_arr**();**

Res**[**i**]** **=** Select8\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

application\_RAM**();**

fill\_random\_arr**();**

Res**[**i**]** **=** Select8\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

application\_RAM**();**

fill\_back\_arr**();**

Res**[**i**]** **=** Select8\_arr**();**

free\_RAM**();**

**}**

**}**

**}**

void Select8\_measurement\_vec**(**char fil**)** // функція для заповнення масиву вимірів часу (Select8, вектор)

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

**if** **(**fil **==** 'o'**)**

**{**

application\_RAM**();**

fillVector\_ordered**();**

Res**[**i**]** **=** Select8\_vec**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

application\_RAM**();**

fillVector\_random**();**

Res**[**i**]** **=** Select8\_vec**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

application\_RAM**();**

fillVector\_back**();**

Res**[**i**]** **=** Select8\_vec**();**

free\_RAM**();**

**}**

**}**

**}**

void Shell\_2\_measurement\_arr**(**char fil**)** //функція для заповнення масиву вимірів часу (для №2 методу сортування Шелла, масив)

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

**if** **(**fil **==** 'o'**)**

**{**

application\_RAM**();**

fill\_ordered\_arr**();**

Res**[**i**]** **=** Shell\_2\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

application\_RAM**();**

fill\_random\_arr**();**

Res**[**i**]** **=** Shell\_2\_arr**();**

free\_RAM**();**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

application\_RAM**();**

fill\_back\_arr**();**

Res**[**i**]** **=** Shell\_2\_arr**();**

free\_RAM**();**

**}**

**}**

**}**

void Shell\_2\_measurement\_vec**(**char fil**)** //функція для заповнення масиву вимірів часу (для №2 методу сортування Шелла, вектор)

**{**

**if** **(**fil **==** 'o'**)**

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

fillVector\_ordered**();**

Res**[**i**]** **=** Shell\_2\_vec**();**

**}**

**}**

**else** **if** **(**fil **==** 'r'**)**

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

fillVector\_random**();**

Res**[**i**]** **=** Shell\_2\_vec**();**

**}**

**}**

**else** **if** **(**fil **==** 'b'**)**

**{**

**for** **(**int i **=** 0**;** i **<** measurements\_number**;** i**++)**

**{**

fillVector\_back**();**

Res**[**i**]** **=** Shell\_2\_vec**();**

**}**

**}**

**}**

**Menu.h**

#ifndef MENU\_H\_INCLUDED

#define MENU\_H\_INCLUDED

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void menu**();**

#endif // MENU\_H\_INCLUDED

**Menu.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#include <windows.h>

#include <conio.h>

// виклик модулів

#include "common.h"

#include "table.h"

#include "measurement.h"

#include "menu.h"

void GotoXY**(**int X**,** int Y**)** //для переміщення курсора текстового вікна на задані координати

**{**

HANDLE hStdOut **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

COORD coord **=** **{** X**,** Y **};**

SetConsoleCursorPosition**(**hStdOut**,** coord**);**

**}**

void ConsoleCursorVisible**(**int show**,** short size**)** // Функція для керування видимістю курсора у консолі

**{**

HANDLE hStdOut **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

CONSOLE\_CURSOR\_INFO structCursorInfo**;**

GetConsoleCursorInfo**(**hStdOut**,** **&**structCursorInfo**);**

structCursorInfo**.**bVisible **=** show**;**

structCursorInfo**.**dwSize **=** size**;**

SetConsoleCursorInfo**(**hStdOut**,** **&**structCursorInfo**);**

**}**

void colour**(**int i**)** // зміна кольору

**{**

HANDLE hStdOut **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

SetConsoleTextAttribute**(**hStdOut**,** i**);**

**}**

// MENU for ARR3d

void Menu\_choose\_sort\_arr**()** // меню для вибору сортування для масива

**{**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Select 7"**,** "Select 8"**,** "Shell 2"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"ARR MENU"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**//

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

Menu\_choose\_Select7\_arr**();** //виклик меню для сортування select7

system**(**"cls"**);**

**break;**

**case** 1**:**

Menu\_choose\_Select8\_arr**();** //виклик меню для сортування select8

system**(**"cls"**);**

**break;**

**case** 2**:**

Menu\_choose\_Shell2\_arr**();** //виклик меню для сортування shell2

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Select7\_arr**()** // меню для сортування Select 7 для масива

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"SELECT 7 for ARR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**// Зчитуємо натиснуту клавішу без виводу на екран

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:** // Клавіша Escape

exit**(**0**);**

**case** 72**:**// Стрілка вгору

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**// Переміщення курсору меню вгору

**break;**

**case** 80**:**// Стрілка вниз

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**// Переміщення курсору меню вниз

**break;**

**case** 13**:**// Клавіша Enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

system**(**"cls"**);**

Select7\_measurement\_arr**(**'o'**);**// заповнення масива "впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nOrdered array: P = %d, M = %d, N = %d was sorted by Select 7 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**// Очищення екрану

**break;**

**case** 1**:**

system**(**"cls"**);**

Select7\_measurement\_arr**(**'r'**);** // заповнення масива "випадкові значення"

Time **=** MeasurementProcessing**();**

printf **(**"\nRandom array: P = %d, M = %d, N = %d was sorted by Select 7 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Select7\_measurement\_arr**(**'b'**);** // заповнення масива "обернено впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nBack-ordered array: P = %d, M = %d, N = %d was sorted by Select 7 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Select8\_arr**()** // меню для сортування Select8 для масива

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"SELECT 8 for ARR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**//escape

exit**(**0**);**//loophole

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)** // вибір заповнення

**{**

**case** 0**:**

system**(**"cls"**);**

Select8\_measurement\_arr**(**'o'**);** // заповнення масива "впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nOrdered array: P = %d, M = %d, N = %d was sorted by Select 8 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 1**:**

system**(**"cls"**);**

Select8\_measurement\_arr**(**'r'**);** // заповнення масива "випадкові значення"

Time **=** MeasurementProcessing**();**

printf **(**"\nRandom array: P = %d, M = %d, N = %d was sorted by Select 8 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Select8\_measurement\_arr**(**'b'**);** // заповнення масива "обернено впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nBack ordered array: P = %d, M = %d, N = %d was sorted by Select 8 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Shell2\_arr**()** // меню для сортування Shell\_2 для масива

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"SHELL 2 for ARR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**

**switch** **(**active\_menu**)** // вибір заповнення

**{**

**case** 0**:**

system**(**"cls"**);**

Shell\_2\_measurement\_arr**(**'o'**);** // заповнення масива "впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nOrdered array: P = %d, M = %d, N = %d was sorted by SHELL 2 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 1**:**

system**(**"cls"**);**

Shell\_2\_measurement\_arr**(**'r'**);** // заповнення масива "випадкові значення"

Time **=** MeasurementProcessing**();**

printf **(**"\nRandom array: P = %d, M = %d, N = %d was sorted by SHELL 2 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Shell\_2\_measurement\_arr**(**'b'**);** // заповнення масива "обернено впорядкований"

Time **=** MeasurementProcessing**();**

printf **(**"\nBack ordered array: P = %d, M = %d, N = %d was sorted by SHELL 2 algorithm. Time = %7.2f"**,** P**,**M**,**N**,** Time**);**

printf**(**"\nPress Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

// VECTOR MENU//

void Menu\_choose\_sort\_vec**()** // меню для вибору сортування у векторі

**{**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Select 7"**,** "Select 8"**,** "Shell 2"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"VECTOR MENU"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

Menu\_choose\_Select7\_vec**();** // виклик меню для сортування Select 7

system**(**"cls"**);**

**break;**

**case** 1**:**

Menu\_choose\_Select8\_vec**();** // виклик меню для сортування Select\_8

system**(**"cls"**);**

**break;**

**case** 2**:**

Menu\_choose\_Shell2\_vec**();** // виклик меню для сортування Shell\_2

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Select7\_vec**()** //Select 7 for VECTOR//

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"Select 7 for VECTOR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

system**(**"cls"**);**

Select7\_measurement\_vec**(**'o'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nOrder array: Vector = %d was sorted by Select 7 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 1**:**

system**(**"cls"**);**

Select7\_measurement\_vec**(**'r'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nRandom array: Vector = %d was sorted by Select 7 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Select7\_measurement\_vec**(**'b'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nBack order array: Vector = %d was sorted by Select 7 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Select8\_vec**()** //SELECT 8 for VECTOR//

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"Select 8 for VECTOR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

system**(**"cls"**);**

printf**(**"Please, wait"**);**

Select8\_measurement\_vec**(**'o'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nOrder array: Vector = %d was sorted by Select 8 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 1**:**

system**(**"cls"**);**

Select8\_measurement\_vec**(**'r'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nRandom array: Vector = %d was sorted by Select 8 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Select8\_measurement\_vec**(**'b'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nBack order array: Vector = %d was sorted by Select 8 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

void Menu\_choose\_Shell2\_vec**()** //SHELL 2 for VECTOR//

**{**

float Time**;**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** MenuOptions**[]** **=** **{**"Ordered"**,** "Random"**,** "Back ordered"**,** "Back"**};**

int numMenuOptions **=** **sizeof(**MenuOptions**)** **/** **sizeof(**MenuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"SHELL 2 for VECTOR"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numMenuOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** MenuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** MenuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**

exit**(**0**);**

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numMenuOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

system**(**"cls"**);**

Shell\_2\_measurement\_vec**(**'o'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nOrder array: Vector = %d was sorted by SHELL 2 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 1**:**

system**(**"cls"**);**

Shell\_2\_measurement\_vec**(**'r'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nRandom array: Vector = %d was sorted by SHELL 2 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 2**:**

system**(**"cls"**);**

Shell\_2\_measurement\_vec**(**'b'**);**

Time **=** MeasurementProcessing**();**

printf**(**"\nBack order array: Vector = %d was sorted by SHELL 2 algorithm. Time = %7.2f\n"**,** VECTOR\_SIZE**,** Time**);**

printf**(**"Press Enter to return..."**);**

getchar**();**

system**(**"cls"**);**

**break;**

**case** 3**:**

**return;**

**break;**

**}**

**break;**

**}**

**}**

**}**

//MAIN MENU//

void menu**()** // головне меню

**{**

system**(**"cls"**);**

ConsoleCursorVisible**(**0**,** 100**);**

const char**\*** menuOptions**[]** **=** **{**"Arr3D"**,**"Vector"**,**"Table"**,**"Exit"**};**

int numOptions **=** **sizeof(**menuOptions**)** **/** **sizeof(**menuOptions**[**0**]);**

int active\_menu **=** 0**;**

char key**;**

**while** **(**1**)**

**{**

colour**(**13**);**

GotoXY**(**45**,** 7**);**

printf**(**"MAIN MENU"**);**

int x **=** 45**,** y **=** 9**;**

GotoXY**(**x**,** y**);**

**for** **(**int i **=** 0**;** i **<** numOptions**;** i**++)**

**{**

GotoXY**(**x**,** y**++);**

**if** **(**i **==** active\_menu**)**

**{**

colour**(**5**);**

printf**(**"\* %s \* "**,** menuOptions**[**i**]);**

**}**

**else**

printf**(**" %s "**,** menuOptions**[**i**]);**

colour**(**13**);**

**}**

key **=** \_getch**();**

**if** **(**key **==** **-**32**)** key **=** \_getch**();**

**switch** **(**key**)**

**{**

**case** 27**:**//escape

exit**(**0**);**//loophole

**case** 72**:**

**if** **(**active\_menu **>** 0**)**

active\_menu**--;**

**break;**

**case** 80**:**

**if** **(**active\_menu **<** numOptions **-** 1**)**

active\_menu**++;**

**break;**

**case** 13**:**//enter

**switch** **(**active\_menu**)**

**{**

**case** 0**:**

colour**(**13**);**

system**(**"CLS"**);**

GotoXY**(**x**,** y**);**

Menu\_choose\_sort\_arr**();** // перехід до меню для масива

system**(**"cls"**);**

**break;**

**case** 1**:**

colour**(**13**);**

system**(**"CLS"**);**

Menu\_choose\_sort\_vec**();** // перехід до меню для вектора

\_getch**();**

system**(**"CLS"**);**

**break;**

**case** 2**:**

colour**(**13**);**

system**(**"CLS"**);**

table**();** // виклик таблиці

\_getch**();**

system**(**"CLS"**);**

**break;**

**case** 3**:**

colour**(**13**);**

GotoXY**(**45**,** 16**);**

exit**(**0**);**

**break;**

**}**

**break;**

**}**

**}**

**}**

**Sort.arr.h**

#ifndef SORT\_ARR\_H\_INCLUDED

#define SORT\_ARR\_H\_INCLUDED

#include <time.h>

clock\_t Select7\_arr**();**

clock\_t Select8\_arr**();**

clock\_t Shell\_2\_arr**();**

#endif // SORT\_ARR\_H\_INCLUDED

**Sort.arr.c**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

// виклик модулів

#include "sort.arr.h"

#include "common.h"

#include <stdio.h>

clock\_t Select7\_arr**()** // Алгоритм сортування №7 методу прямого вибору

**{**

int Min**,** Max**;**

int L**,** R**,** imin**,** imax**,** A**,** summa**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

**for** **(**int k **=** 0**;** k **<** P**;** k**++)**

**{**

**for** **(**int j **=** 0**;** j **<** N**;** j**++)**

**{**

summa **=** 0**;**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)**

**{**

summa **+=** Arr3D**[**k**][**i**][**j**];**

**}**

Sum**[**j**]** **=** summa**;**

**}**

L **=** 0**;**

R **=** N **-** 1**;**

**while** **(**L **<** R**)**

**{**

Min **=** Sum**[**L**];**

imin **=** L**;**

Max **=** Sum**[**L**];**

imax **=** L**;**

**for** **(**int i **=** L **+** 1**;** i **<=** R**;** i**++)**

**{**

**if** **(**Sum**[**i**]** **<** Min**)**

**{**

Min **=** Sum**[**i**];**

imin **=** i**;**

**}**

**else** **if** **(**Sum**[**i**]** **>** Max**)**

**{**

Max **=** Sum**[**i**];**

imax **=** i**;**

**}**

**}**

**if** **(**imin **!=** L**)**

**{**

Sum**[**imin**]** **=** Sum**[**L**];**

Sum**[**L**]** **=** Min**;**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imin**];**

Arr3D**[**k**][**a**][**imin**]=**Arr3D**[**k**][**a**][**L**];**

Arr3D**[**k**][**a**][**L**]=**A**;**

**}**

**}**

**if** **(**imax **!=** R**)**

**{**

**if** **(**imax **==** L**)**

**{**

Sum**[**imin**]** **=** Sum**[**R**];**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imin**];**

Arr3D**[**k**][**a**][**imin**]=**Arr3D**[**k**][**a**][**R**];**

Arr3D**[**k**][**a**][**R**]=**A**;**

**}**

**}**

**else**

**{**

Sum**[**imax**]** **=** Sum**[**R**];**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imax**];**

Arr3D**[**k**][**a**][**imax**]=**Arr3D**[**k**][**a**][**R**];**

Arr3D**[**k**][**a**][**R**]=**A**;**

**}**

**}**

Sum**[**R**]** **=** Max**;**

**}**

L**++;**

R**--;**

**}**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

clock\_t Select8\_arr**()** // Алгоритм сортування №8 методу прямого вибору

**{**

int L**,** R**,** imin**,** imax**,** tmp**,** summa**,** A**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

**for** **(**int k **=** 0**;** k **<** P**;** k**++)**

**{**

**for** **(**int j **=** 0**;** j **<** N**;** j**++)**

**{**

summa **=** 0**;**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)**

**{**

summa **+=** Arr3D**[**k**][**i**][**j**];**

**}**

Sum**[**j**]** **=** summa**;**

**}**

L **=** 0**;**

R **=** N **-** 1**;**

**while** **(**L **<** R**)**

**{**

imin **=** L**;**

imax **=** L**;**

**for** **(**int i **=** L **+** 1**;** i **<=** R**;** i**++)**

**{**

**if** **(**Sum**[**i**]** **<** Sum**[**imin**])**

imin **=** i**;**

**else** **if** **(**Sum**[**i**]** **>** Sum**[**imax**])**

imax **=** i**;**

**}**

**if** **(**imin **!=** L**)**

**{**

tmp **=** Sum**[**imin**];**

Sum**[**imin**]** **=** Sum**[**L**];**

Sum**[**L**]** **=** tmp**;**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imin**];**

Arr3D**[**k**][**a**][**imin**]=**Arr3D**[**k**][**a**][**L**];**

Arr3D**[**k**][**a**][**L**]=**A**;**

**}**

**}**

**if** **(**imax **!=** R**)**

**{**

**if** **(**imax **==** L**)**

**{**

tmp **=** Sum**[**imin**];**

Sum**[**imin**]** **=** Sum**[**R**];**

Sum**[**R**]** **=** tmp**;**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imin**];**

Arr3D**[**k**][**a**][**imin**]=**Arr3D**[**k**][**a**][**R**];**

Arr3D**[**k**][**a**][**R**]=**A**;**

**}**

**}**

**else**

**{**

tmp **=** Sum**[**imax**];**

Sum**[**imax**]** **=** Sum**[**R**];**

Sum**[**R**]** **=** tmp**;**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**imax**];**

Arr3D**[**k**][**a**][**imax**]=**Arr3D**[**k**][**a**][**R**];**

Arr3D**[**k**][**a**][**R**]=**A**;**

**}**

**}**

**}**

L**++;**

R**--;**

**}**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

clock\_t Shell\_2\_arr**()** // Алгоритм №2 методу сортування Шелла

**{**

int tmp**,** t**,** y**,** r**,** A**,** summa**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

**for** **(**int k **=** 0**;** k **<** P**;** k**++)**

**{**

**for** **(**int j **=** 0**;** j **<** N**;** j**++)**

**{**

summa **=** 0**;**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)**

**{**

summa **+=** Arr3D**[**k**][**i**][**j**];**

**}**

Sum**[**j**]** **=** summa**;**

**}**

**if** **(**N **<** 4**)**

t **=** 1**;**

**else**

t **=** **(**int**)**log2f**((**float**)**N**)** **-** 1**;**

int Stages**[**t**];**

Stages**[**t **-** 1**]** **=** 1**;**

**for** **(**int h **=** t **-** 2**;** h **>=** 0**;** h**--)**

Stages**[**h**]** **=** 2 **\*** Stages**[**h **+** 1**]** **+** 1**;**

**for** **(**int p **=** 0**;** p **<** t**;** p**++)**

**{**

r **=** Stages**[**p**];**

**for** **(**int h **=** r**;** h **<** N**;** h**++)**

**{**

y **=** h**;**

**while** **(**y **>=** r **&&** Sum**[**y**]** **<** Sum**[**y **-** r**])**

**{**

tmp **=** Sum**[**y**];**

Sum**[**y**]** **=** Sum**[**y **-** r**];**

Sum**[**y **-** r**]** **=** tmp**;**

**for(**int a**=**0**;** a**<**M**;** a**++)**

**{**

A**=**Arr3D**[**k**][**a**][**y**];**

Arr3D**[**k**][**a**][**y**]=**Arr3D**[**k**][**a**][**y **-** r**];**

Arr3D**[**k**][**a**][**y **-** r**]=**A**;**

**}**

y **=** y **-** r**;**

**}**

**}**

**}**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

**Sort.vector.h**

#ifndef SORT\_VECTOR\_H\_INCLUDED

#define SORT\_VECTOR\_H\_INCLUDED

#include <time.h>

clock\_t Select7\_vec**();**

clock\_t Select8\_vec**();**

clock\_t Shell\_2\_vec**();**

#endif // SORT\_VECTOR\_H\_INCLUDED

**Sort.vector.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

// виклик модулів

#include "sort.vector.h"

#include "common.h"

clock\_t Select7\_vec**()**

**{**

int Min**,** Max**;**

int L**,** R**,** imin**,** imax**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

L **=** 0**;**

R **=** VECTOR\_SIZE **-** 1**;**

**while** **(**L **<** R**)**

**{**

Min **=** Vector**[**L**];**

imin **=** L**;**

Max **=** Vector**[**L**];**

imax **=** L**;**

**for** **(**int i **=** L **+** 1**;** i **<=** R**;** i**++)**

**{**

**if** **(**Vector**[**i**]** **<** Min**)**

**{**

Min **=** Vector**[**i**];**

imin **=** i**;**

**}**

**else** **if** **(**Vector**[**i**]** **>** Max**)**

**{**

Max **=** Vector**[**i**];**

imax **=** i**;**

**}**

**}**

**if** **(**imin **!=** L**)**

**{**

Vector**[**imin**]** **=** Vector**[**L**];**

Vector**[**L**]** **=** Min**;**

**}**

**if** **(**imax **!=** R**)**

**{**

**if** **(**imax **==** L**)**

**{**

Vector**[**imin**]** **=** Vector**[**R**];**

**}**

**else**

**{**

Vector**[**imax**]** **=** Vector**[**R**];**

**}**

Vector**[**R**]** **=** Max**;**

**}**

L**++;**

R**--;**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

clock\_t Select8\_vec**()**

**{**

int L**,** R**,** imin**,** imax**,** tmp**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

L **=** 0**;**

R **=** VECTOR\_SIZE **-** 1**;**

**while** **(**L **<** R**)**

**{**

imin **=** L**;**

imax **=** L**;**

**for** **(**int i **=** L **+** 1**;** i **<=** R**;** i**++)**

**{**

**if** **(**Vector**[**i**]** **<** Vector**[**imin**])**

**{**

imin **=** i**;**

**}**

**else** **if** **(**Vector**[**i**]** **>** Vector**[**imax**])**

**{**

imax **=** i**;**

**}**

**}**

**if** **(**imin **!=** L**)**

**{**

tmp **=** Vector**[**imin**];**

Vector**[**imin**]** **=** Vector**[**L**];**

Vector**[**L**]** **=** tmp**;**

**}**

**if** **(**imax **!=** R**)**

**{**

**if** **(**imax **==** L**)**

**{**

tmp **=** Vector**[**imin**];**

Vector**[**imin**]** **=** Vector**[**R**];**

Vector**[**R**]** **=** tmp**;**

**}**

**else**

**{**

tmp **=** Vector**[**imax**];**

Vector**[**imax**]** **=** Vector**[**R**];**

Vector**[**R**]** **=** tmp**;**

**}**

**}**

L**++;**

R**--;**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

clock\_t Shell\_2\_vec**()**

**{**

int tmp**,** t**,** j**,** k**;**

clock\_t time\_start**,** time\_stop**;**

time\_start **=** clock**();**

**if** **(**VECTOR\_SIZE **<** 4**)**

t **=** 1**;**

**else**

t **=** **(**int**)**log2f**((**float**)**VECTOR\_SIZE**)** **-** 1**;**

int Stages**[**t**];**

Stages**[**t **-** 1**]** **=** 1**;**

**for** **(**int i **=** t **-** 2**;** i **>=** 0**;** i**--)**

Stages**[**i**]** **=** 2 **\*** Stages**[**i **+** 1**]** **+** 1**;**

**for** **(**int p **=** 0**;** p **<** t**;** p**++)**

**{**

k **=** Stages**[**p**];**

**for** **(**int i **=** k**;** i **<** VECTOR\_SIZE**;** i**++)**

**{**

j **=** i**;**

**while** **(**j **>=** k **&&** Vector**[**j**]** **<** Vector**[**j **-** k**])**

**{**

tmp **=** Vector**[**j**];**

Vector**[**j**]** **=** Vector**[**j **-** k**];**

Vector**[**j **-** k**]** **=** tmp**;**

j **-=** k**;**

**}**

**}**

**}**

time\_stop **=** clock**();**

**return** time\_stop **-** time\_start**;**

**}**

**Table.h**

#ifndef TABLE\_H\_INCLUDED

#define TABLE\_H\_INCLUDED

void table**();**

#endif // TABLE\_H\_INCLUDED

**Table.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// виклик модулів

#include "common.h"

#include "measurement.h"

#include "filling.arr.h"

#include "filling.vector.h"

#include "sort.arr.h"

#include "sort.vector.h"

void table**()**

**{**

float Ordered**,** Random**,** Back\_Ordered**;**

printf**(**"\t\t\t Table for the array: P = %d, M = %d, N = %d\n\n"**,** P**,** M**,** N**);**

printf**(**"\t\t\tOrdered\t\t\tRandom Ordered\t\tBack Ordered\n\n"**);**

// Shell\_2 (Масив)

Shell\_2\_measurement\_arr**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Shell\_2\_measurement\_arr**(**'r'**);**

Random **=** MeasurementProcessing**();**

Shell\_2\_measurement\_arr**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Shell\_2\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

// Select7 (Масив)

Select7\_measurement\_arr**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Select7\_measurement\_arr**(**'r'**);**

Random **=** MeasurementProcessing**();**

Select7\_measurement\_arr**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Select7\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

// Select8 (Масив)

Select8\_measurement\_arr**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Select8\_measurement\_arr**(**'r'**);**

Random **=** MeasurementProcessing**();**

Select8\_measurement\_arr**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Select8\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

// Таблиця для вектора

printf**(**"\t\t\t\t Table for the vector: N = %d\n\n"**,** VECTOR\_SIZE**);**

printf**(**"\t\t\tOrdered\t\t\tRandom Ordered\t\tBack Ordered\n\n"**);**

// Shell\_2 (Вектор)

Shell\_2\_measurement\_vec**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Shell\_2\_measurement\_vec**(**'r'**);**

Random **=** MeasurementProcessing**();**

Shell\_2\_measurement\_vec**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Shell\_2\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

// Select7 (Вектор)

Select7\_measurement\_vec**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Select7\_measurement\_vec**(**'r'**);**

Random **=** MeasurementProcessing**();**

Select7\_measurement\_vec**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Select7\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

// Select8 (Вектор)

Select8\_measurement\_vec**(**'o'**);**

Ordered **=** MeasurementProcessing**();**

Select8\_measurement\_vec**(**'r'**);**

Random **=** MeasurementProcessing**();**

Select8\_measurement\_vec**(**'b'**);**

Back\_Ordered **=** MeasurementProcessing**();**

printf**(**"Select8\t\t\t%f\t\t%f\t\t%f\n\n"**,** Ordered**,** Random**,** Back\_Ordered**);**

**}**

**Main.c**

#include <stdio.h>

#include <stdlib.h>

// виклик модулів

#include "menu.h"

int main**()**

**{**

srand**(**time**(NULL));**

menu**();**

// виклик меню

**return** 0**;**

**}**

**ТЕСТИ**

Зображення, що містить текст, знімок екрана, Шрифт, документ

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, Шрифт, алгебра

Вміст, створений ШІ, може бути неправильним.

Алгоритми показали коректну роботу при сортуванні багатовимірних масивів та векторів, для усіх випадків відсортованості:

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, дизайн

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, дизайн

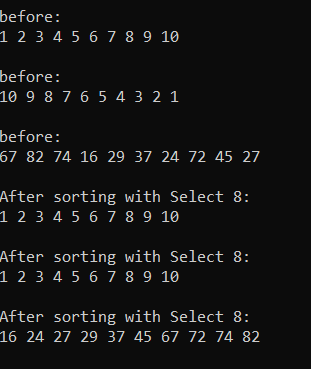
Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, дизайн

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.



Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Вміст, створений ШІ, може бути неправильним.

**РЕЗУЛЬТАТИ**

***Випадок 1.*** Залежність часу роботи алгоритмів від довжини стовпчиків масива.

**P = 3 , M =1 N = 5000**

**Зображення, що містить знімок екрана, текст

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 2 , N = 5000**

**Зображення, що містить знімок екрана, текст, фіолетовий, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 4 , N = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 8 N = 5000**

**Зображення, що містить знімок екрана, текст, фіолетовий

Вміст, створений ШІ, може бути неправильним.**

**P = 3, M = 16 , N = 5000**

**Зображення, що містить знімок екрана, текст

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 32 , N = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 64 , N = 5000**

**Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 128 , N = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 256 , N = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 512 , N = 5000**

**Зображення, що містить текст, знімок екрана, монітор

Вміст, створений ШІ, може бути неправильним.**

**P = 3 , M = 1024, N = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт, фіолетовий

Вміст, створений ШІ, може бути неправильним.**

**Vector = 5000**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, Графік, схема

Вміст, створений ШІ, може бути неправильним.**

Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, схема, Графік

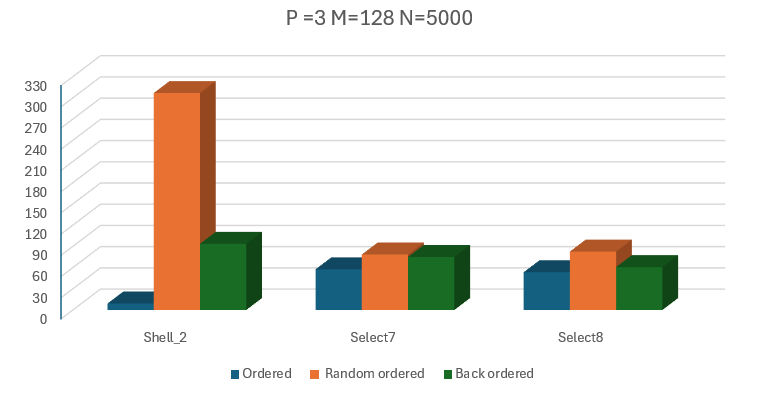
Вміст, створений ШІ, може бути неправильним.

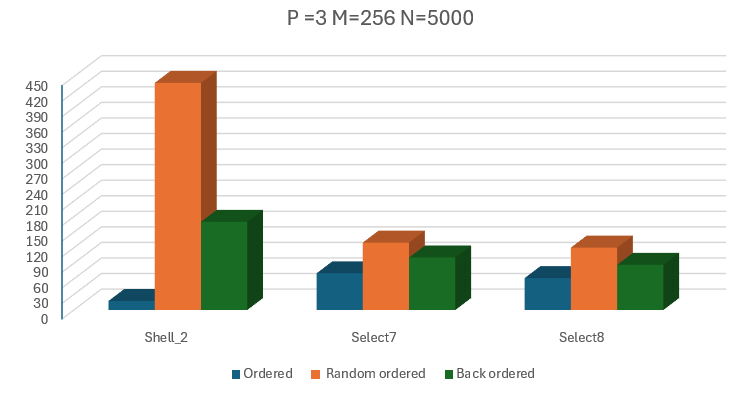
Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, схема, Графік

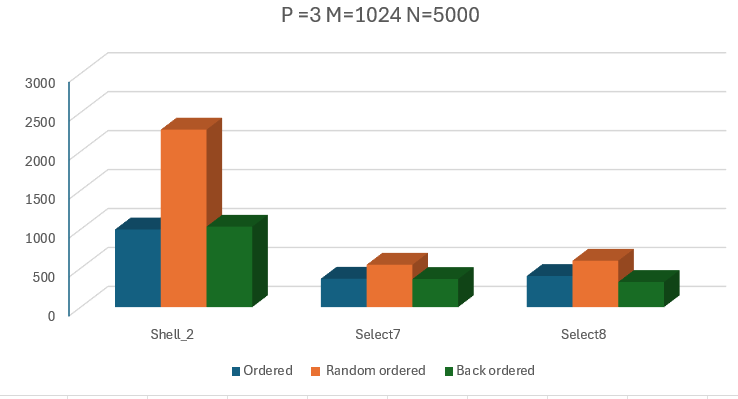
Вміст, створений ШІ, може бути неправильним.





Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.



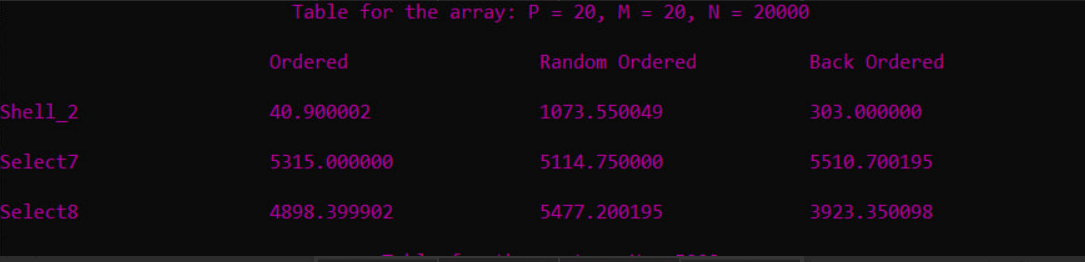
Для вектору N =5000 (значення помножені на три , для порівняння з масивом )

Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.

***Випадок 2.*** Залежність часу роботи алгоритмів від форми перерізів масива

**P = 20, M = 20 , N = 20000**

****

**P = 20, M = 50 , N = 8000**

**Зображення, що містить текст, знімок екрана, фіолетовий, Шрифт

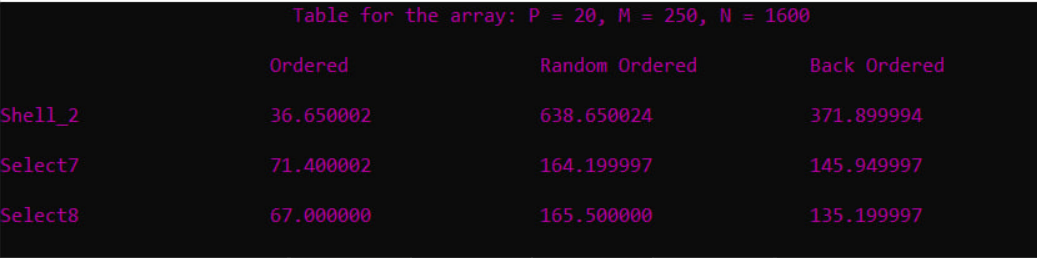
Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 200 , N = 2000**

**Зображення, що містить знімок екрана, текст, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 300 , N = 1600**

****

**P = 20, M = 400 , N = 1000**

**Зображення, що містить текст, знімок екрана, Шрифт, фіолетовий

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 1000 , N = 400**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 1600 , N = 250**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 2000 , N = 200**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 8000 , N = 50**

**Зображення, що містить знімок екрана, текст

Вміст, створений ШІ, може бути неправильним.**

**P = 20, M = 20000 , N = 20**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

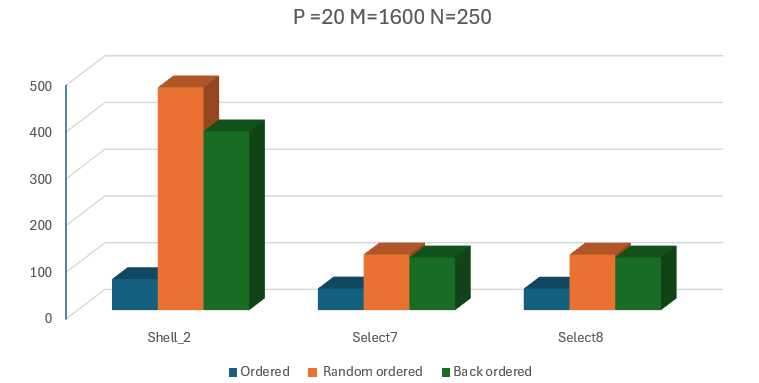
Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

****

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

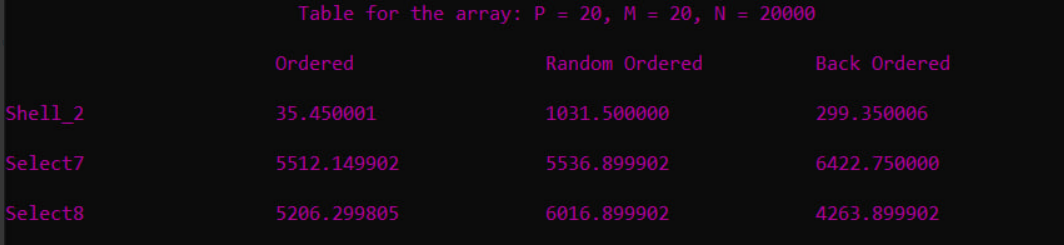
Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, схема, Графік, текст

Вміст, створений ШІ, може бути неправильним.**

***Випадок 3.*** Залежність часу роботи алгоритмів від кількості ключів у кожному перерізі масива при однаковій загальній кількості ключів у всьому масиві

**P = 20, M = 20 , N = 20000**

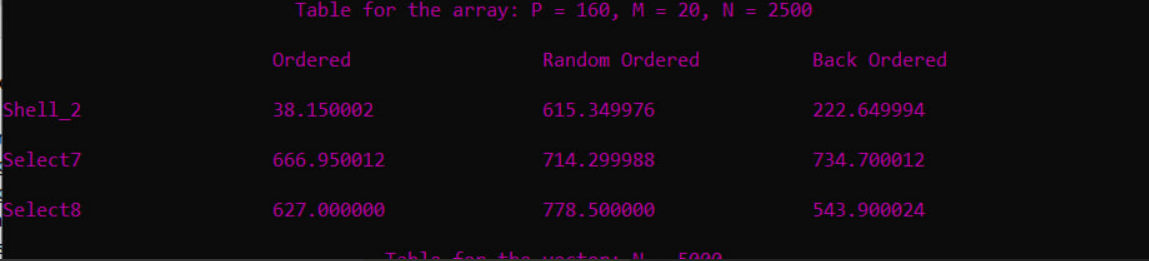
****

**P = 50, M = 20 , N = 8000**

**Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

**P = 160, M = 20 , N = 2500**

****

**P = 200, M = 20 , N = 2000**

**Зображення, що містить знімок екрана, текст, фіолетовий, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 400, M = 20 , N = 1000**

**Зображення, що містить текст, знімок екрана, Шрифт, фіолетовий

Вміст, створений ШІ, може бути неправильним.**

**P = 1000, M = 20 , N = 400**

**Зображення, що містить знімок екрана, текст

Вміст, створений ШІ, може бути неправильним.**

**P = 2000, M = 20 , N = 200**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

**P = 2500, M = 20 , N = 160**

**Зображення, що містить знімок екрана, текст

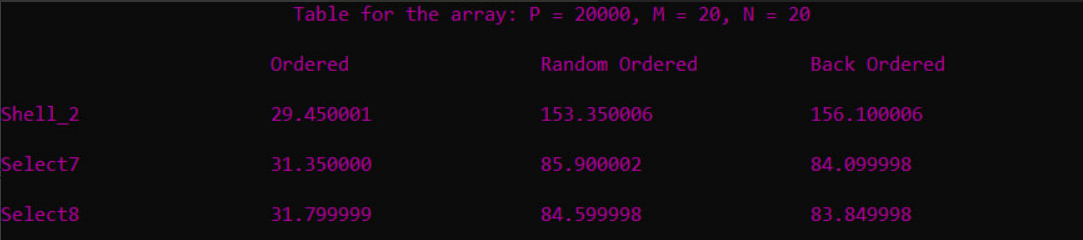
Вміст, створений ШІ, може бути неправильним.**

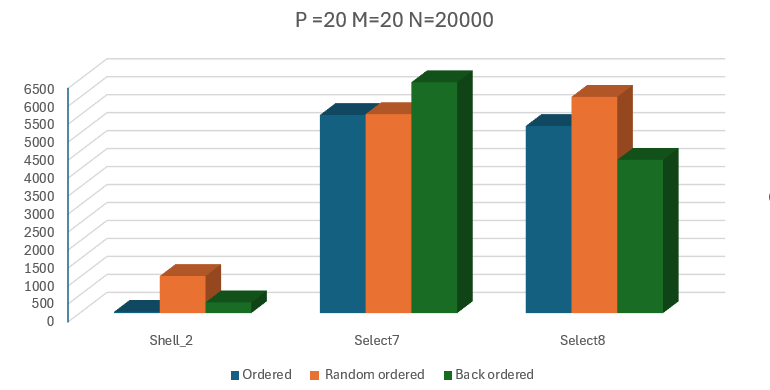
**P = 8000, M = 20 , N = 50**

**Зображення, що містить знімок екрана, текст

Вміст, створений ШІ, може бути неправильним.**

**P = 20000, M = 20 , N = 20**

****

****

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, схема, Графік, текст

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

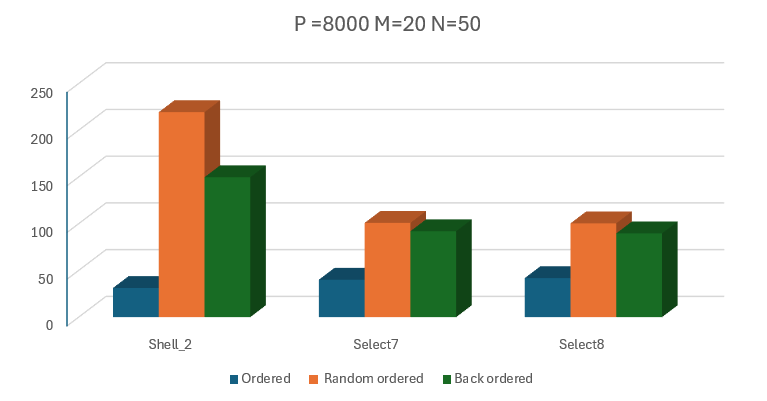
Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**Зображення, що містить знімок екрана, текст, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

****

**Зображення, що містить текст, знімок екрана, схема, Графік

Вміст, створений ШІ, може бути неправильним.**

**ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ**

В ході виконання курсової роботи було реалізовано 3 алгоритми:

- Алгоритм сортування №7 методу прямого вибору

- Алгоритм сортування №8 методу прямого вибору

- Алгоритм №2 методу сортування Шелла (спосіб реалізації на основі гібридного алгоритму «вставка-обмін»)

Ці алгоритми були використані при сортуванні одновимірних та багатовимірних масивів. За умовою завдання, сортування здійснювалось окремо для кожного перерізу, наскрізно по стовпчиках за незменшенням.

З метою подальшого аналізу були проведені тести для 3-ох випадків дослідження:

* Залежність часу роботи алгоритмів від довжини стовпчиків масиву
* Залежність часу роботи алгоритмів від форми перерізів масиву
* Залежність часу роботи алгоритмів від кількості ключів у кожному перерізі масиву при однаковій загальній кількості ключів у всьому масиві

**1. Алгоритм сортування №7 методу прямого вибору.**

Цей алгоритм є модифікацією алгоритму сортування вибором, який одночасно знаходить як мінімальний, так і максимальний елементи у кожній ітерації, що призводить до двостороннього сортування.

Перевагою цього алгоритму є оптимізація стандартного сортування вибором шляхом одночасного розміщення найменшого елемента на початок масиву та найбільшого елемента в кінець масиву. Це дозволяє скоротити кількість проходів через масив майже вдвічі порівняно з класичним сортуванням вибором, оскільки за один прохід упорядковуються два елементи.

Особливістю цього алгоритму є додаткова перевірка ситуації, коли максимальний елемент знаходиться на початку поточного діапазону (imax==L), що вимагає коригування після переміщення мінімального елемента. Це збільшує кількість умовних операцій порівняно з простим сортуванням вибором. В більшості випадків він показав роботу, яка повинна була бути “в теорії”. Цей алгоритм треба використовувати для невеликих масивів, коли не треба високої швидкості сортування. А от використовувати для великих обсягах даних цей алгоритм не рекомендується, перевага зберігання мінімального та максимального значень у змінних не компенсує загальну неефективність алгоритму на великих обсягах даних, тому треба використовувати більш швидке сортування, таке як наприклад Shell sort,яке буде розглядатися трохи пізніше.

2**. Алгоритм сортування №8 методу прямого вибору.**

Цей алгоритм є варіацією двостороннього сортування вибором із використанням тимчасової змінної для обміну значеннями. Замість зберігання мінімального та максимального значень як у Select7, цей алгоритм зберігає лише індекси цих елементів, а потім виконує обмін через тимчасову змінну. Перевагою такого підходу є спрощення логіки визначення мінімального та максимального елементів у масиві. Алгоритм не зберігає значення елементів у окремих змінних, а працює безпосередньо з індексами, що зменшує кількість операцій присвоєння під час пошуку.

Проте, використання тимчасової змінної для обміну значеннями збільшує кількість операцій присвоєння при власне обміні. Кожен обмін вимагає трьох операцій присвоєння, що може погіршити продуктивність алгоритму при роботі з великими масивами даних.

Алгоритм також містить додаткову перевірку випадку, коли максимальний елемент знаходиться на позиції L, що може спричинити додаткові обчислення. Ця обробка особливого випадку збільшує кількість умовних операцій. Порівняно з Select7, цей алгоритм має подібну загальну структуру та продуктивність, але розбіжності в реалізації обміну елементами можуть призвести до незначних відмінностей у швидкодії залежно від характеристик вхідних даних.

3.Алгоритм №2 методу сортування Шелла (спосіб реалізації на основі гібридного алгоритму «вставка-обмін»)

Алгоритм Шелла виявився найефективнішим у порівнянні з двома попередніми алгоритмами сортування, що не є несподіваним результатом. Цей алгоритм базується на техніці вставки-обміну, яка також використовується в одному з попередніх алгоритмів. На відміну від двостороннього сортування вибором (Select7, Select8), алгоритм Shell\_2 демонструє кращу ефективність на практично невпорядкованих масивах та значно кращу продуктивність при збільшенні розміру вхідних даних. Завдяки використанню проміжків, алгоритм може "перестрибувати" через елементи, що дозволяє швидше переміщувати елементи на великі відстані в масиві, особливо на початкових етапах сортування.

**Загальний аналіз:**

-Одновимірний масив:

У порівнянні з теорією , алгоритм Шелла №2 повинен був показати найкращі результати, з чим він успішно впорався. Алгоритми №7,№8 прямого вибору мають дуже схожі результати, і обидва поступаються цьому алгоритму.

-Багатовимірні масиви:

Знову спираючись на теорію, Шелла №2 повинен бути найкрашим, але є декілька тестів, на яких можна побачити ефективнішим алгоритм сортування №7,№8 методу прямого вибору.

-Вплив різних геометричних розмірів багатовимірних масивів:

Дослідження впливу різних розмірів багатовимірних масивів показує, що збільшення кількості стовпців може призводити до збільшення часу виконання алгоритмів сортування. Проте, відносна ефективність алгоритмів залишається подібною на різних розмірах масиву.

Залежність часу роботи алгоритмів від довжини стовпчиків масива







Можна спостерігати, що при збільшенні довжини стовпчика час роботи алгоритму зростає. Це пов’язано з тим, що при збільшення довжини стовпчика кількість елементів яких треба переставити збільшується, а отже і час роботи алгоритмів збільшується.

Залежність часу роботи алгоритмів від форми перерізів масива

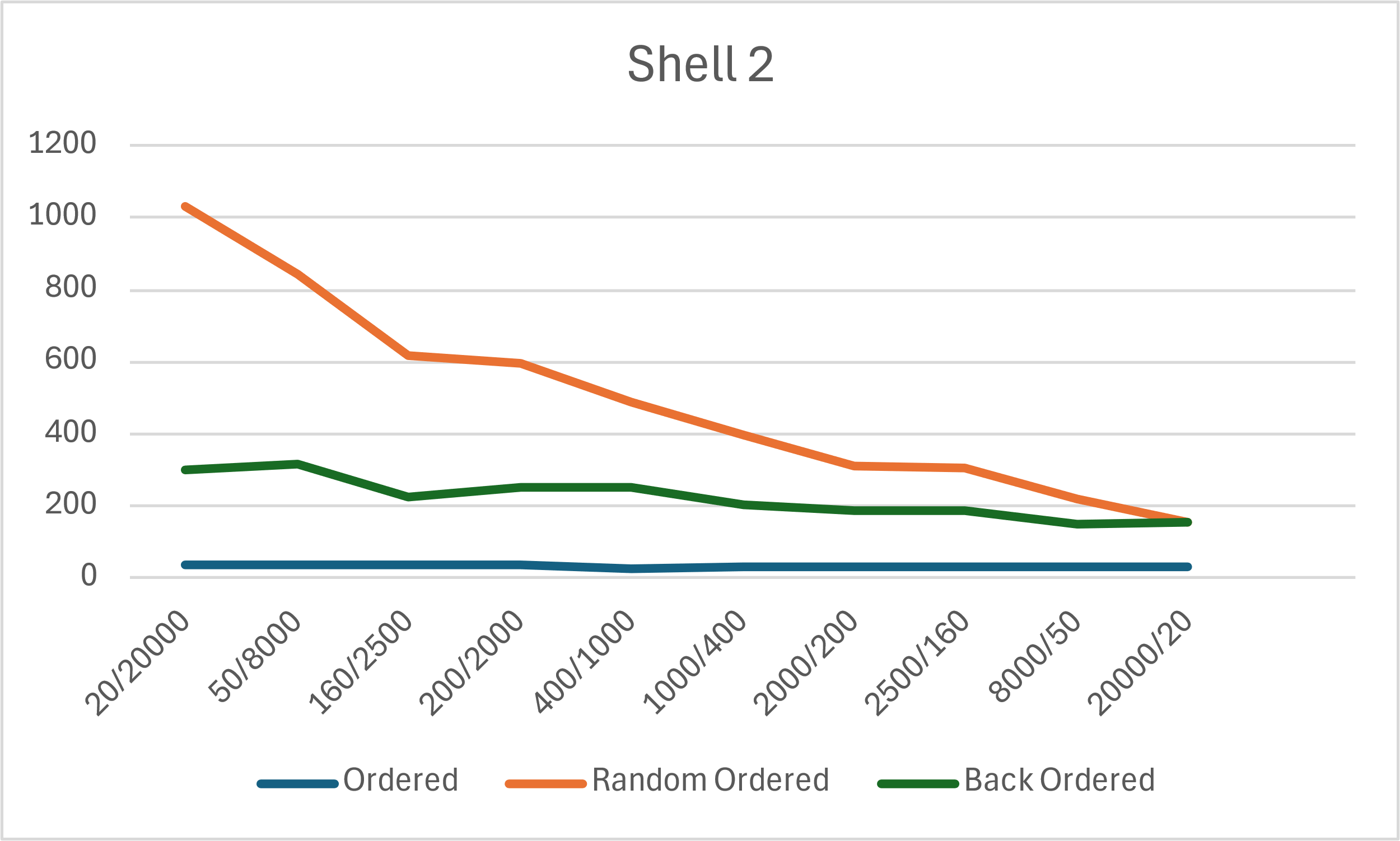


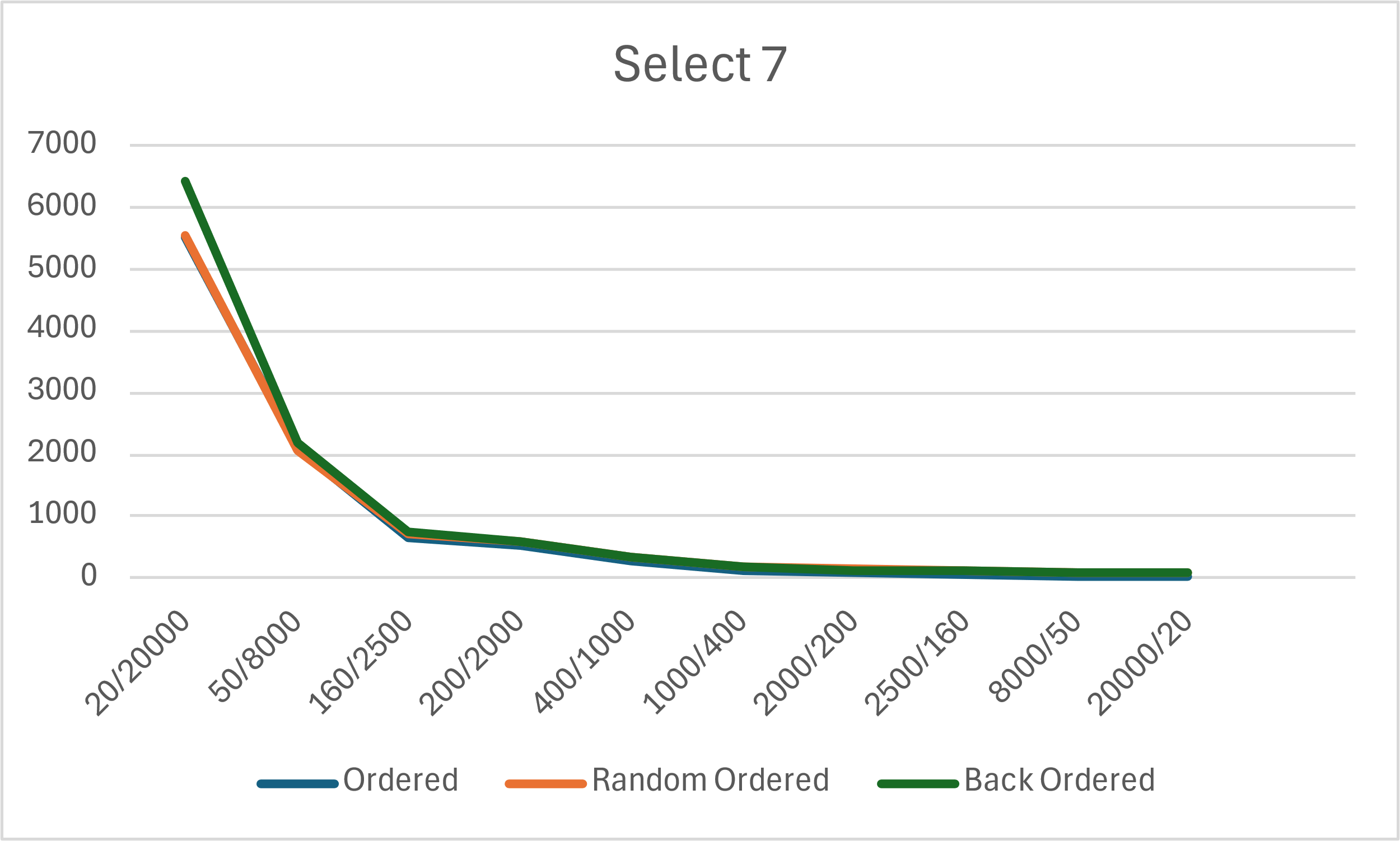


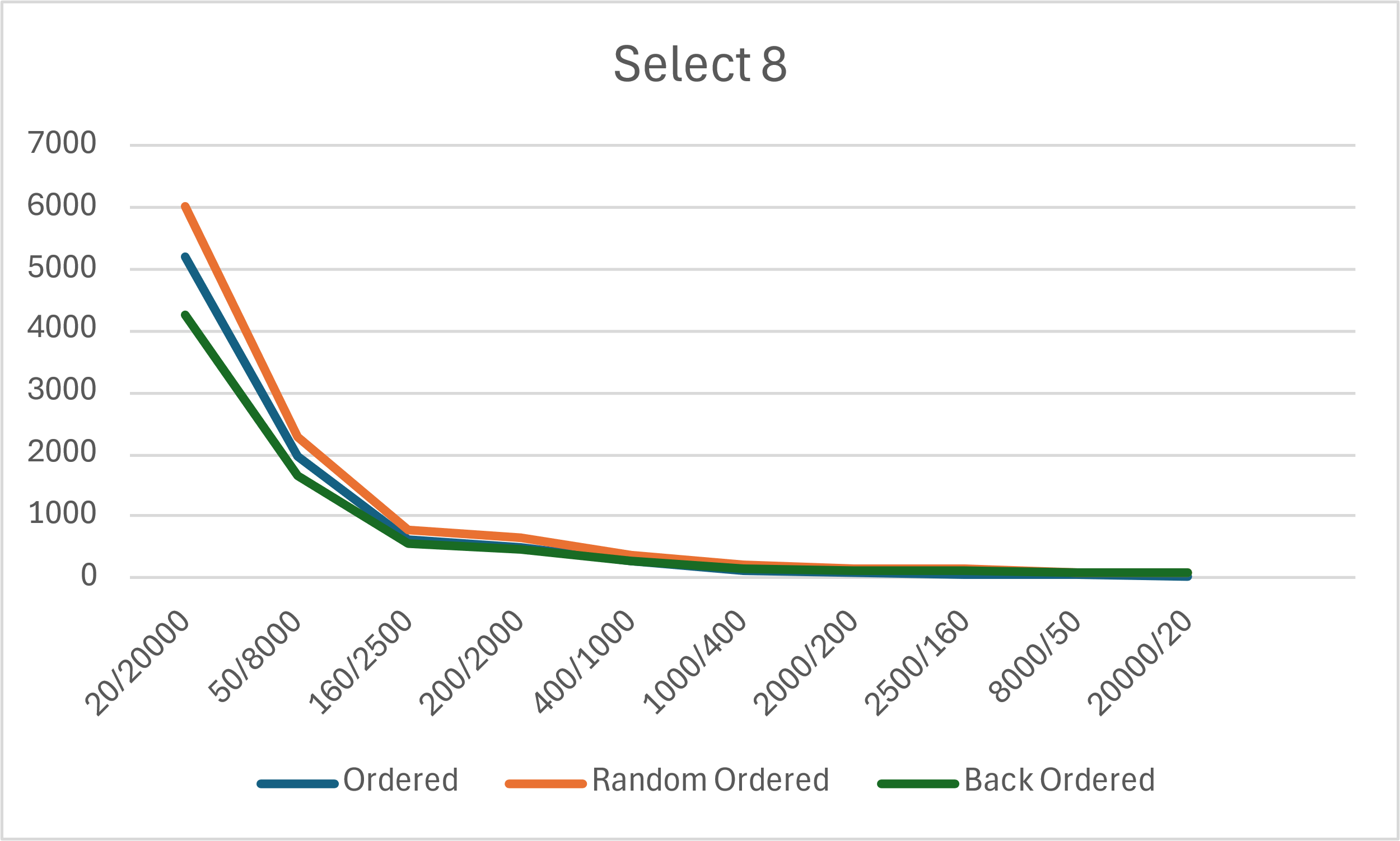


Можна спостерігати, що при зменшенні кількості стовпчиків час роботи алгоритмів зменшується. Це пов’язано з тим, що кількість елементів яких треба відсортувати зменшується.

Залежність часу роботи алгоритмів від кількості ключів у кожному перерізі масиву при однаковій загальній кількості ключів у всьому масиві







Можна спостерігати, що при зменшенні кількості стовпчиків час роботи алгоритмів зменшується. Це пов’язано з тим, що кількість елементів яких треба відсортувати зменшується.

**ВИСНОВКИ ПО ОТРИМАНИХ РЕЗУЛЬТАТАХ**

Ця курсова робота спрямована на дослідження ефективності трьох методів сортування на багатовимірних масивах:

1. Алгоритм сортування №7 методу прямого вибору.

2. Алгоритм сортування №8 методу прямого вибору.

3. Алгоритм №2 методу сортування Шелла (спосіб реалізації на основі гібридного алгоритму «вставка-обмін»).

Дослідження проведено для вивчення залежності часу роботи алгоритмів від форми та розміру перерізів масиву.

З погляду ефективності методів сортування, найкращими виявилися метод сортування Шелла , а №7,№8 методу прямого вибору можна прирівняти до майже однакової ефективності, як для одновимірних, так і для багатовимірних масивів. Порівнюючи результати для багатовимірних масивів із результатами для одновимірних, можна побачити, що час виконання зазвичай збільшується разом із збільшенням розмірів масиву. Однак ефективність методів сортування залишається приблизно на тому ж рівні, як і для одновимірного масиву.

Ураховуючи обсяги даних, усі методи можуть показувати прийняті результати. Для малих масивів краще вибирати метод сортування №7,№8 методу прямого вибору або Шелла, а для великих масивів — метод Шелла.

Отже, при виборі методу сортування важливо враховувати як розмір даних, так і їхню структуру, щоб забезпечити оптимальну ефективність сортування.

**СПИСОК ЛІТЕРАТУРИ**

1. YouTube канал “Марченко Олександр Іванович”.
2. Підручник “Структури даних і алгоритми”