МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра Інформаційної Безпеки

Практикум з Алгоритмів та структур даних

Лабораторна робота №3

Структури даних: масиви, зв’язні списки

Мета роботи:

обробити текст без використання стандартних бібліотек для роботи з даними символьного типу; порівняти ефективність обробки тексту з використанням символьних масивів і динамічних списків, відпрацювати навики роботи зі списками.

Виконав:
студент II курсу
групи ФБ-01
Сахній Н.Р.

Київ 2022

Виконання лабораторної роботи

Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; текст закінчується крапкою. Написати програму введення такого тексту з клавіатури та його обробки, використовуючи:

а) Динамічний масив

```
"D:\KPI\АСД\ASD_Sakhnii Nazar FB-01\venv\Scripts\python.exe" "D:/KPI/АСД/ASD_Sakhnii Nazar FB-01/using_Dynamic_array.py

◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> Okay @I - want to type any text 777 !
Йой, введений тип даних не відповідає заданому формату! Спробуйте використовувати лише символи [a-z]


◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> perhaps the text like that would be better
Оу, щось не видно крапки в кінці рядка! Спробуйте ще раз, але не забудьте про крапку в кінці


◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> well maybe it needed type like that .
Файний текст!

* Щоб продовжити, натисність |Enter| *


                          ◄◎► Меню можливих операцій ◄◎►
        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
```

```
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 1
☼ Переглад динамічного масиву:  ['well', 'maybe', 'it', 'needed', 'type', 'like', 'that', '.']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який масив виводиться на екран: 1.719989813864231e-05

* Щоб продовжити, натисність |Enter| *


                              ◄◙► Меню можливих операцій ◄◙►
       ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 2
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
       ▼ Яке слово необхідно добавити?
    >>> azure
```

```
☼ Переглад динамічного масиву, у який було додано 'azure' на його початок: ['azure', 'well', 'maybe', 'it', 'needed',
'type', 'like', 'that', '.']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбувається вставка елемента в масив: 3.00002284348011e-06

* Щоб продовжити, натисність |Enter| *


                                ◄◙► Меню можливих операцій ◄◙►
        ---------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 3
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
        ▼ Яке слово необхідно добавити?
    >>> abcdefg
☼ Переглад динамічного масиву, у який було додано 'abcdefg' в його кінці: ['azure', 'well', 'maybe', 'it', 'needed',
'type', 'like', 'that', '.', 'abcdefg']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбувається вставка елемента в масив: 3.700144588947296e-06

* Щоб продовжити, натисність |Enter| *


                                ◄◙► Меню можливих операцій ◄◙►
        ---------------------------------------------------------------
```

```
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 4
◘ Номер позиції, у яку потрібно вставити елемент, це натуральне число:
        ▼ Який номер позиції масиву?
    >>> 2
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
        ▼ Яке слово необхідно добавити?
    >>> seem
☼ Переглад динамічного масиву, у який було додано 'seem' за заданим індексом '2': ['azure', 'seem', 'well', 'maybe', 'it',
'needed', 'type', 'like', 'that', '.', 'abcdefg']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбується вставка елемента в масив: 3.6999117583036423e-06

* Щоб продовжити, натисність |Enter| *


                                ◄○► Меню можливих операцій ◄○►
        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
```

```
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 5
☼ Переглад динамічного масиву, із початку якого було видалено перший елемент: ['seem', 'well', 'maybe', 'it', 'needed',
'type', 'like', 'that', '.', 'abcdefg']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбується видалення елемента із масиву: 3.6999117583036423e-06

* Щоб продовжити, натисність |Enter| *


                                ◄○► Меню можливих операцій ◄○►
        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 6
☼ Переглад динамічного масиву, із кінця якого було видалено останній елемент: ['seem', 'well', 'maybe', 'it', 'needed',
'type', 'like', 'that', '.']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбується видалення елемента із масиву: 1.800013706088066e-06
```

```
* Щоб продовжити, натисність |Enter| *


                                ◀◎▶ Меню можливих операцій ◀◎▶
        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 7
◘ Номер позиції, елемент якої необхідно видалити, це натуральне число:
        ▼ Який номер позиції масиву?
    >>> 3
☼ Переглад динамічного масиву, із якого було видалено елемент за заданим індексом '3': ['seem', 'well', 'it', 'needed',
'type', 'like', 'that', '.']
size ► Об'єм памяті, який займає цей динамічний масив: 152 Bytes
time ► Час, за який відбується видалення елемента із масиву: 3.200024366378784e-06

* Щоб продовжити, натисність |Enter| *


Process finished with exit code -1
```

Варіант №2

Надрукувати всі слова, які відрізняються від першого слова і співпадають з початковим відрізком алфавіту (a, ab, abc, abcd, abcde тощо). Видалити першу літеру в цих словах. До кожного слова дописати крапку.

```
"D:\KPI\АСД\ASD_Sakhnii Nazar FB-01\venv\Scripts\python.exe" "D:/KPI/АСД/ASD_Sakhnii Nazar FB-01/using_Dynamic_array.py"

◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> for them abc abcdea wait abcdsead abcdef .
Файний текст!

* Щоб продовжити, натисність |Enter| *

                                  ◄○► Меню можливих операцій ◄○►
        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 8
☼ Переглянемо всі такі елементи: ['bc.', 'bcdef.']
abc_size ► Об'єм памяті, який займає цей масив із 'abc...': 88 Bytes
full_size ► Об'єм памяті, який займає масив з усіма словами: 152 Bytes
time ► Час, за який відбулась ця взаємодія з текстом: 5.1900045946240425e-05

* Щоб продовжити, натисність |Enter| *
```

```
                                ◀◎▶ Меню можливих операцій ◀◎▶

        ------------------------------------------------------------------
        | 1. Переглянути динамічний масив, що складається із слів тексту
        |
        | 2. Добавити елемент на початок динамічного масиву
        | 3. Добавити елемент в кінець динамічного масиву
        | 4. Добавити елемент у динамічний масив за заданим індексом
        |
        | 5. Видалити перший елемент динамічного масиву
        | 6. Видалити останній елемент динамічного масиву
        | 7. Видалити елемент динамічного масиву за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 0

Process finished with exit code 0
```

b) Двозв'язний список

```
"D:\KPI\АСД\ASD_Sakhnii Nazar FB-01\venv\Scripts\python.exe" "D:/KPI/АСД/ASD_Sakhnii Nazar FB-01/using_DL_list.py"

◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> %^&*() type and format with 4884 */ .
Йой, введений тип даних не відповідає заданому формату! Спробуйте використовувати лише символи [a-z]


◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> okay without any comments
Оу, щось не видно крапки в кінці рядка! Спробуйте ще раз, але не забудьте про крапку в кінці
```

```
◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> soo with point now .
Файний текст!

* Щоб продовжити, натисність |Enter| *


                              ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
       | 1. Переглянути двозв'язний список, що складається із слів тексту
       |
       | 2. Добавити елемент на початок двозв'язного списку
       | 3. Добавити елемент в кінець двозв'язного списку
       | 4. Добавити елемент у двозв'язний список за заданим індексом
       |
       | 5. Видалити перший елемент двозв'язного списку
       | 6. Видалити останній елемент двозв'язного списку
       | 7. Видалити елемент двозв'язного списку за заданим індексом
       |
       | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
       |
       └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 1
☼ Переглад двозв'язного списку:
soo <-> with <-> point <-> now <-> . <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який список виводиться на екран: 9.320001117885113e-05

* Щоб продовжити, натисність |Enter| *


                              ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
```

```
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 2
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
        ▼ Яке слово необхідно добавити?
    >>> addword
☼ Переглад двозв'язного списку, у який було додано 'addword' на його початок:
addword <-> soo <-> with <-> point <-> now <-> . <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбується вставка елемента у список: 1.3299984857439995e-05

* Щоб продовжити, натисність |Enter| *


                              ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
```

```
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 3
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
        ▼ Яке слово необхідно добавити?
    >>> stanford
☼ Перегляд двозв'язного списку, у який було додано 'stanford' в його кінці:
addword <-> soo <-> with <-> point <-> now <-> . <-> stanford <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбувається вставка елемента у список: 9.100185707211494e-06

* Щоб продовжити, натисність |Enter| *


                              ◄○► Меню можливих операцій ◄○►
        ----------------------------------------------------------------
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 4
◘ Номер позиції, у яку потрібно вставити елемент, це натуральне число:
        ▼ Який номер позиції списку?
```

```
    >>> 5
◘ Новий елемент має складатися лише з малих латинських літер [a-z]:
       ▼ Яке слово необхідно добавити?
    >>> six
☼ Переглад динамічного списку, у який було додано 'six' за заданим індексом '5':
addword <-> soo <-> with <-> point <-> six <-> now <-> . <-> stanford <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбувається вставка елемента у список: 1.5099998563528061e-05

* Щоб продовжити, натисність |Enter| *

                                  ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
       | 1. Переглянути двозв'язний список, що складається із слів тексту
       |
       | 2. Добавити елемент на початок двозв'язного списку
       | 3. Добавити елемент в кінець двозв'язного списку
       | 4. Добавити елемент у двозв'язний список за заданим індексом
       |
       | 5. Видалити перший елемент двозв'язного списку
       | 6. Видалити останній елемент двозв'язного списку
       | 7. Видалити елемент двозв'язного списку за заданим індексом
       |
       | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
       |
       ∟ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 5
☼ Переглад двозв'язного списку, із початку якого було видалено перший елемент:
soo <-> with <-> point <-> six <-> now <-> . <-> stanford <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбувається видалення елемента із списку: 6.400048732757568e-06

* Щоб продовжити, натисність |Enter| *

                                  ◄○► Меню можливих операцій ◄○►
```

```
        ------------------------------------------------------------------
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        └ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 6
☼ Переглад двозв'язного списку, із кінця якого було видалено останній елемент:
soo <-> with <-> point <-> six <-> now <-> . <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбується видалення елемента із списку: 1.9799917936325073e-05

* Щоб продовжити, натисність |Enter| *

                                ◀◎▶ Меню можливих операцій ◀◎▶
        ------------------------------------------------------------------
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
```

```
       ∟ 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 7
◘ Номер позиції, елемент якої необхідно видалити, це натуральне число:
       ▼ Який номер позиції списку?
    >>> 4
☼ Переглад двозв'язного списку, із якого було видалено елемент за заданим індексом '4':
soo <-> with <-> point <-> now <-> . <-> None

size ► Об'єм памяті, який займає цей двозв'язний список: 48 Bytes
time ► Час, за який відбується видалення елемента із чписку: 1.3199867680668831e-05

* Щоб продовжити, натисність |Enter| *


Process finished with exit code -1
```

Варіант №2

Надрукувати всі слова, які відрізняються від першого слова і співпадають з початковим відрізком алфавіту (a, ab, abc, abcd, abcde тощо). Видалити першу літеру в цих словах. До кожного слова дописати крапку.

```
"D:\KPI\АСД\ASD_Sakhnii Nazar FB-01\venv\Scripts\python.exe" "D:/KPI/АСД/ASD_Sakhnii Nazar FB-01/using_DL_list.py"

◘ Напишіть довільний текст наступного типу та формату:
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою .
    >>> oh abcdedcba not abcde abc yes ab .
Файний текст!

* Щоб продовжити, натисність |Enter| *

                                ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
       | 1. Переглянути двозв'язний список, що складається із слів тексту
       |
       | 2. Добавити елемент на початок двозв'язного списку
```

```
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
    >>> 8
☼ Переглянемо всі такі елементи:
abc_size ► Об'єм памяті, який займає цей список із 'abc...': 48 Bytes
bcde. <-> bc. <-> b. <-> None

full_size ► Об'єм памяті, який займає список з усіма словами: 48 Bytes
time ► Час, за який відбулась ця взаємодія з текстом: 0.00010469998233020306

* Щоб продовжити, натисність |Enter| *

                              ◄○► Меню можливих операцій ◄○►
       ------------------------------------------------------------------
        | 1. Переглянути двозв'язний список, що складається із слів тексту
        |
        | 2. Добавити елемент на початок двозв'язного списку
        | 3. Добавити елемент в кінець двозв'язного списку
        | 4. Добавити елемент у двозв'язний список за заданим індексом
        |
        | 5. Видалити перший елемент двозв'язного списку
        | 6. Видалити останній елемент двозв'язного списку
        | 7. Видалити елемент двозв'язного списку за заданим індексом
        |
        | 8. Надрукувати всі слова, які відповідають умові 2-го варіанту
        |
        L 0. Закінчити виконання програми

* Щоб виконати необхідну операцію меню, введіть її номер:
```

```
    >>> 0

Process finished with exit code 0
```

Програмні коди:

- Файл `DoubleLinked_list.py` містить реалізацію двозв’язного списка (Атрибути та методи):

```
class Node:
    # Допоміжний клас - вузол двобічно зв'язаного списку """

    def __init__(self, item):
        """ Кноструктор вузла

        :param item: Елемент списку
        """
        self.mItem = item  # Дані
        self.mNext = None  # Наступний вузол
        self.mPrev = None  # Попередній вузол


class DoublyLinkedList:
    # Двобічно зв'язаний список

    def __init__(self):
        """ Конструктор списку - створює порожній список """
        self.mFirst = None  # Перший вузол списку
        self.mLast = None   # Останній бузол списку
        self.mCurr = None   # ПОточний бузол списку

    def empty(self):
        """ Перевіряє чи список порожній
        :return: True, якщо список порожній
        """
        return self.mFirst is None
```

```
    def setFirst(self):
        """ Зробити поточним перший елемент списку """
        self.mCurr = self.mFirst

    def setLast(self):
        """ Зробити поточним останній елемент списку """
        self.mCurr = self.mLast

    def next(self):
        """ Перейти до наступного елемента """
        if self.mCurr != self.mLast:
            self.mCurr = self.mCurr.mNext
            return True
        else:
            return False

    def prev(self):
        """ Перейти до попереднього елемента """
        if self.mCurr != self.mFirst:
            self.mCurr = self.mCurr.mPrev
            return True
        else:
            return False

    def current(self):
        """ Отримати поточний елемент
        :return: Навантаження поточного вузла
        """
        if self.mCurr is not None:
            return self.mCurr.mItem
        else:
            return None

    def insertAtBegin(self, item):
        """ Можливість вставки елемента елемента на початок списку """
        if self.empty():
            node = Node(item)
            self.mFirst = node
            self.mLast = node
```

```
            return
        node = Node(item)
        node.mNext = self.mFirst
        self.mFirst.mPrev = node
        self.mFirst = node

    def insertAtEnd(self, item):
        """ Можливість вставки елемента в кінець списку """
        if self.empty():
            node = Node(item)
            self.mFirst = node
            return
        self.mCurr = self.mFirst
        while self.mCurr.mNext is not None:
            self.mCurr = self.mCurr.mNext
        node = Node(item)
        self.mCurr.mNext = node
        self.mLast = node
        node.mPrev = self.mCurr

    def insertByIndex(self, index, item):
        """ Можливість вставки елемента у список за заданим індексом """
        node = Node(item)
        if self.empty():
            self.mFirst = self.mLast = self.mCurr = node
        else:
            if index <= 0:
                print("Не існує такої позиції у списку \n")
            else:
                self.mCurr = self.mFirst
                n = 1
                while n != index and self.mCurr is not None:
                    self.mCurr = self.mCurr.mNext
                    n += 1
                if self.mCurr is None:
                    print(f"Не можливо вставити елемент, так як позиції під номером '{index}' не існує у списку \n")
                else:
                    node.mPrev = self.mCurr
                    node.mNext = self.mCurr.mNext
```

```
                if self.mCurr.mNext is not None:
                    self.mCurr.mNext.mPrev = node
                else:
                    self.mLast = node
                self.mCurr.mNext = node


    def deleteAtBegin(self):
        """ Видалення початкового елементу із списку """
        if self.empty():
            print("Не можливо виконати, так як двозв'язний список не містить жодного елемента \n")
            return
        if self.mFirst.mNext is None:
            self.mFirst = None
            print("Єдиний елемент двозв'язного списку, що залишився, видалено \n")
            return
        self.mFirst = self.mFirst.mNext
        self.mFirst.mPrev = None

    def deleteAtEnd(self):
        """ Видалення кінцевого елементу зі списку """
        if self.empty():
            print("Не можливо виконати, так як двозв'язний список не містить жодного елемента \n")
            return
        if self.mFirst.mNext is None:
            self.mFirst = None
            print("Єдиний елемент двозв'язного списку, що залишився, видалено \n")
            return
        self.mCurr = self.mFirst
        while self.mCurr.mNext is not None:
            self.mCurr = self.mCurr.mNext
        self.mCurr.mPrev.mNext = None
        self.mCurr = self.mCurr.mPrev
        self.mLast = self.mCurr

    def deleteByIndex(self, index):
        """ Видалення елемента зі списку за заданим індексом """
        if self.empty():
            print("Не можливо виконати, так як двозв'язний список не містить жодного елемента \n")
```

```
        else:
            if index <= 0:
                print("Задано неіснуєчий елемент списку \n")
            else:
                self.mCurr = self.mFirst
                n = 1
                while n != index and self.mCurr is not None:
                    self.mCurr = self.mCurr.mNext
                    n += 1
                if self.mCurr is None:
                    print(f"Не можливо видалити елемент, так як елемент під номером '{index}' відсутній у списку \n")
                else:
                    if self.mCurr.mPrev is None:
                        self.mFirst = self.mFirst.mNext
                        self.mFirst.mPrev = None
                        return
                    if self.mCurr.mNext is not None:
                        self.mCurr.mPrev.mNext = self.mCurr.mNext
                        self.mCurr.mNext.mPrev = self.mCurr.mPrev
                        self.mCurr = None
                        return
                    self.mLast = self.mCurr.mPrev
                    self.mCurr.mPrev.mNext = None

    def outputList(self):
        """ Обхід списку та виведення його на екран """
        if self.mFirst is None:
            print("Не можливо виконати, так як двозв'язний список не містить жодного елемента \n")
        else:
            self.mCurr = self.mFirst
            while self.mCurr is not None:
                print(self.mCurr.mItem, end=" <-> ")
                self.mCurr = self.mCurr.mNext
```

- Файл `using_DL_list.py` містить реалізацію програми через двозв'язний список:

```
import string
import time
import sys

from DoubleLinked_list import *


def character_check(text, operation):
    # функція для перевірки нових елементів що додаються до списку

    characters = string.ascii_lowercase + " " + "."
    # Перевірка на наявність крапки в кінці тексту
    # Перевірка введених даних, на наявність лише зазначених в змінній chars символів
    if all((symbol in characters) for symbol in text):
        if operation == "create":
            if text.endswith("."):
                print("Файний текст! \n")
                return True
            else:
                print("Оу, щось не видно крапки в кінці рядка! Спробуйте ще раз, але не забудьте про крапку в кінці \n")
                return False
        else:
            return True

    else:
        print("Йой, введений тип даних не відповідає заданому формату! Спробуйте використовувати лише символи [a-z] \n")
        return False


def interaction_with_text(ls):
    # функція для виконання завдання за варіантом
    ls.setFirst()
    first_item = ls.current()

    abc_list = DoublyLinkedList()
    while ls.next():
```

```
        if (ls.current() != first_item) and (ls.current() in string.ascii_lowercase) and ls.current().startswith('a'):
            new_item = ls.current()[1:] + "."
            abc_list.insertAtEnd(new_item)
    if abc_list.mFirst:
        print(f"☼ Переглянемо всі такі елементи: ")
        print(f"abc_size ► Об'єм памяті, який займає цей список із 'abc...': {sys.getsizeof(abc_list)} Bytes")
    print(abc_list.outputList(), "\n")



while True:
    my_text = input("\n◘ Напишіть довільний текст наступного типу та формату: \n\
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою . \n\
    >>> ")
    if character_check(my_text, "create"):
        array = my_text.split()

        # Заповнення зв'язного списку даними
        dl_list = DoublyLinkedList()
        for item in array:
            dl_list.insertAtEnd(item)
        break

while True:
    input("* Щоб продовжити, натисність |Enter| * \n")
    print("\t\t\t\t    ◄○► Меню можливих операцій ◄○►    \n"
          "\t-------------------------------------------------------------------\n"
          "\t| 1. Переглянути двозв'язний список, що складається із слів тексту\n\t|\n"
          "\t| 2. Добавити елемент на початок двозв'язного списку\n"
          "\t| 3. Добавити елемент в кінець двозв'язного списку\n"
          "\t| 4. Добавити елемент у двозв'язний список за заданим індексом\n\t|\n"
          "\t| 5. Видалити перший елемент двозв'язного списку\n"
          "\t| 6. Видалити останній елемент двозв'язного списку\n"
          "\t| 7. Видалити елемент двозв'язного списку за заданим індексом\n\t|\n"
          "\t| 8. Надрукувати всі слова, які відповідають умові 2-го варіанту\n\t|\n"
          "\t└ 0. Закінчити виконання програми\n")

    option = input("* Щоб виконати необхідну операцію меню, введіть її номер: \n    >>> ")
    if option == "1":
```

```
        print("☼ Переглад двозв'язного списку: ")
        start = time.perf_counter()
        print(dl_list.outputList(), "\n")
        stop = time.perf_counter()
        print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
        print(f"time ► Час, за який список виводиться на екран: {stop - start} \n")
    elif option == "2":
        new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                    \r\t▼ Яке слово необхідно добавити? \n    >>> ")
        if character_check(new_word, "insert"):
            start = time.perf_counter()
            dl_list.insertAtBegin(new_word)
            stop = time.perf_counter()
            print(f"☼ Переглад двозв'язного списку, у який було додано '{new_word}' на його початок: ")
            print(dl_list.outputList(), "\n")
            print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
            print(f"time ► Час, за який відбувається вставка елемента у список: {stop - start} \n")
        else:
            print("Повторіть операцію вставки елемента на початок списку")
    elif option == "3":
        new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                    \r\t▼ Яке слово необхідно добавити? \n    >>> ")
        if character_check(new_word, "insert"):
            start = time.perf_counter()
            dl_list.insertAtEnd(new_word)
            stop = time.perf_counter()
            print(f"☼ Переглад двозв'язного списку, у який було додано '{new_word}' в його кінці: ")
            print(dl_list.outputList(), "\n")
            print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
            print(f"time ► Час, за який відбувається вставка елемента у список: {stop - start} \n")
        else:
            print("Повторіть операцію вставки елемента в кінець списку")
    elif option == "4":
        index = int(input("◘ Номер позиції, у яку потрібно вставити елемент, це натуральне число: \n\
                    \r\t▼ Який номер позиції списку? \n    >>> "))
        new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                    \r\t▼ Яке слово необхідно добавити? \n    >>> ")
        if character_check(new_word, "insert"):
            start = time.perf_counter()
```

```
            dl_list.insertByIndex(index - 1, new_word)
            stop = time.perf_counter()
            if dl_list.mCurr is not None:
                print(f"☼ Переглад динамічного списку, у який було додано '{new_word}' за заданим індексом '{index}': ")
                print(dl_list.outputList(), "\n")
                print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
                print(f"time ► Час, за який відбується вставка елемента у список: {stop - start} \n")
        else:
            print('Повторіть операцію вставки елемента в список за заданим індексом \n')

    elif option == "5":
        start = time.perf_counter()
        dl_list.deleteAtBegin()
        stop = time.perf_counter()
        if not dl_list.empty():
            print(f"☼ Переглад двозв'язного списку, із початку якого було видалено перший елемент: ")
            print(dl_list.outputList(), "\n")
            print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
            print(f"time ► Час, за який відбується видалення елемента із списку: {stop - start} \n")
    elif option == "6":
        start = time.perf_counter()
        dl_list.deleteAtEnd()
        stop = time.perf_counter()
        if not dl_list.empty():
            print(f"☼ Переглад двозв'язного списку, із кінця якого було видалено останній елемент: ")
            print(dl_list.outputList(), "\n")
            print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
            print(f"time ► Час, за який відбується видалення елемента із списку: {stop - start} \n")
    elif option == "7":
        index = int(input("◘ Номер позиції, елемент якої необхідно видалити, це натуральне число: \n\
                          \r\t▼ Який номер позиції списку? \n    >>> "))
        start = time.perf_counter()
        dl_list.deleteByIndex(index)
        stop = time.perf_counter()
        if not dl_list.empty() and dl_list.mCurr is None:
            print(f"☼ Переглад двозв'язного списку, із якого було видалено елемент за заданим індексом '{index}': ")
            print(dl_list.outputList(), "\n")
            print(f"size ► Об'єм памяті, який займає цей двозв'язний список: {sys.getsizeof(dl_list)} Bytes")
            print(f"time ► Час, за який відбується видалення елемента із чписку: {stop - start} \n")
```

```
    elif option == "8":
        start = time.perf_counter()
        interaction_with_text(dl_list)
        stop = time.perf_counter()
        print(f"full_size ▶ Об'єм памяті, який займає список з усіма словами: {sys.getsizeof(dl_list)} Bytes")
        print(f"time ▶ Час, за який відбулась ця взаємодія з текстом: {stop - start} \n")
    elif option == "0":
        break
    else:
        print("Неа, такого номера опцерації в меню не існує \n")
```

- Файл `using_Dynamic_array.py` містить реалізацію програми через динамічний масив:

```
import string
import time
import sys


def character_check(text, operation):
    # функція для перевірки нових елементів що додаються до списку

    characters = string.ascii_lowercase + " " + "."
    # Перевірка на наявність крапки в кінці тексту
    # Перевірка введених даних, на наявність лише зазначених в змінній chars символів
    if all((symbol in characters) for symbol in text):
        if operation == "create":
            if text.endswith("."):
                print("Файний текст! \n")
                return True
            else:
                print("Оу, щось не видно крапки в кінці рядка! Спробуйте ще раз, але не забудьте про крапку в кінці \n")
                return False
        else:
            return True
```

```
    else:
        print("Йой, введений тип даних не відповідає заданому формату! Спробуйте використовувати лише символи [a-z] \n")
        return False


def interaction_with_text(arr):
    # функція для виконання завдання за варіантом
    first_item = arr[0]

    abc_array = []
    for current_item in arr:
        if (current_item != first_item) and (current_item in string.ascii_lowercase) and current_item.startswith('a'):
            new_item = current_item[1:] + "."
            abc_array.append(new_item)
    print(f"☼ Переглянемо всі такі елементи: {abc_array}")
    print(f"abc_size ► Об'єм памяті, який займає цей масив із 'abc...': {sys.getsizeof(abc_array)} Bytes")


while True:
    my_text = input("\n◘ Напишіть довільний текст наступного типу та формату: \n\
    ▼ Слова тексту із малих латинських літер записані не менше, ніж через один пробіл; Текст закінчується крапкою . \n\
    >>> ")
    if character_check(my_text, "create"):
        dyn_array = my_text.split()
        break

while True:
    input("* Щоб продовжити, натисність |Enter| * \n")
    print("\t\t\t\t    ◄○► Меню можливих операцій ◄○►    \n"
          "\t-----------------------------------------------------------------\n"
          "\t| 1. Переглянути динамічний масив, що складається із слів тексту\n\t|\n"
          "\t| 2. Добавити елемент на початок динамічного масиву\n"
          "\t| 3. Добавити елемент в кінець динамічного масиву\n"
          "\t| 4. Добавити елемент у динамічний масив за заданим індексом\n\t|\n"
          "\t| 5. Видалити перший елемент динамічного масиву\n"
          "\t| 6. Видалити останній елемент динамічного масиву\n"
          "\t| 7. Видалити елемент динамічного масиву за заданим індексом\n\t|\n"
          "\t| 8. Надрукувати всі слова, які відповідають умові 2-го варіанту\n\t|\n"
          "\t└ 0. Закінчити виконання програми\n")
```

```
option = input("* Щоб виконати необхідну операцію меню, введіть її номер: \n    >>> ")
if option == "1":
    print("☼ Переглад динамічного масиву: ", end=" ")
    start = time.perf_counter()
    print(dyn_array)
    stop = time.perf_counter()
    print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
    print(f"time ► Час, за який масив виводиться на екран: {stop - start} \n")
elif option == "2":
    new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                \r\t▼ Яке слово необхідно добавити? \n    >>> ")
    if character_check(new_word, "insert"):
        start = time.perf_counter()
        dyn_array.insert(0, new_word)
        stop = time.perf_counter()
        print(f"☼ Переглад динамічного масиву, у який було додано '{new_word}' на його початок: {dyn_array}")
        print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
        print(f"time ► Час, за який відбувається вставка елемента в масив: {stop - start} \n")
    else:
        print("Повторіть операцію вставки елемента на початок масиву")
elif option == "3":
    new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                \r\t▼ Яке слово необхідно добавити? \n    >>> ")
    if character_check(new_word, "insert"):
        start = time.perf_counter()
        dyn_array.insert(len(dyn_array), new_word)
        stop = time.perf_counter()
        print(f"☼ Переглад динамічного масиву, у який було додано '{new_word}' в його кінці: {dyn_array}")
        print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
        print(f"time ► Час, за який відбувається вставка елемента в масив: {stop - start} \n")
    else:
        print("Повторіть операцію вставки елемента в кінець масиву")
elif option == "4":
    index = int(input("◘ Номер позиції, у яку потрібно вставити елемент, це натуральне число: \n\
                \r\t▼ Який номер позиції масиву? \n    >>> "))
    new_word = input("◘ Новий елемент має складатися лише з малих латинських літер [a-z]: \n\
                \r\t▼ Яке слово необхідно добавити? \n    >>> ")
    if 0 < index < len(dyn_array):
```

```
            if character_check(new_word, "insert"):
                start = time.perf_counter()
                dyn_array.insert(index - 1, new_word)
                stop = time.perf_counter()
                print(f"☼ Переглад динамічного масиву, у який було додано '{new_word}' за заданим індексом '{index}': "
                      f"{dyn_array}")
                print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
                print(f"time ► Час, за який відбується вставка елемента в масив: {stop - start} \n")
            else:
                print('Повторіть операцію вставки елемента в масив за заданим індексом \n')
        else:
            print("Схоже, що така позиція масиву, в яку потрібно вставити елемент, не існує \n")
    elif option == "5":
        if not dyn_array:
            print("Не можливо виконати, так як динамічний масив не містить жодного елемента \n")
        else:
            start = time.perf_counter()
            dyn_array.pop(0)
            stop = time.perf_counter()
            print(f"☼ Переглад динамічного масиву, із початку якого було видалено перший елемент: {dyn_array}")
            print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
            print(f"time ► Час, за який відбується видалення елемента із масиву: {stop - start} \n")
    elif option == "6":
        if not dyn_array:
            print("Не можливо виконати, так як динамічний масив не містить жодного елемента \n")
        else:
            start = time.perf_counter()
            dyn_array.pop()
            stop = time.perf_counter()
            print(f"☼ Переглад динамічного масиву, із кінця якого було видалено останній елемент: {dyn_array}")
            print(f"size ► Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
            print(f"time ► Час, за який відбується видалення елемента із масиву: {stop - start} \n")
    elif option == "7":
        index = int(input("◘ Номер позиції, елемент якої необхідно видалити, це натуральне число: \n\
                          \r\t▼ Який номер позиції масиву? \n    >>> "))
        if 0 < index < len(dyn_array):
            if not dyn_array:
                print("Не можливо виконати, так як динамічний масив не містить жодного елемента \n")
            else:
```

```
            start = time.perf_counter()
            dyn_array.pop(index - 1)
            stop = time.perf_counter()
            print(f"⛭ Переглад динамічного масиву, із якого було видалено елемент за заданим індексом '{index}': "
                  f"{dyn_array}")
            print(f"size ▶ Об'єм памяті, який займає цей динамічний масив: {sys.getsizeof(dyn_array)} Bytes")
            print(f"time ▶ Час, за який відбується видалення елемента із масиву: {stop - start} \n")
    else:
        print("Схоже, що така позиція масиву, елемент якої необхідно видалити, не існує \n")
elif option == "8":
    if not dyn_array:
        print("Не можливо виконати, так як динамічний масив не містить жодного елемента \n")
    else:
        start = time.perf_counter()
        interaction_with_text(dyn_array)
        stop = time.perf_counter()
        print(f"full_size ▶ Об'єм памяті, який займає масив з усіма словами: {sys.getsizeof(dyn_array)} Bytes")
        print(f"time ▶ Час, за який відбулась ця взаємодія з текстом: {stop - start} \n")
elif option == "0":
    break
else:
    print("Неа, такого номера опцерації в меню не існує")
```