

**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №2**  
з дисципліни  
«Алгоритми і структури даних»

Виконав:

студент групи ІМ-44  
Мундурс Нікіта Юрійович  
номер у списку групи: 16

Перевірила:

Сергієнко М. А.

## Завдання

1. Створити список з  $n$  ( $n > 0$ ) елементів ( $n$  вводиться з клавіатури), якщо інша кількість елементів не вказана у конкретному завданні за варіантом.
2. Тип ключів (інформаційних полів) задано за варіантом.
3. Вид списку (черга, стек, дек, прямий однозв'язний лінійний список, обернений однозв'язний лінійний список, двозв'язний лінійний список, однозв'язний кільцевий список, двозв'язний кільцевий список) вибрати самостійно з метою найбільш доцільного розв'язку поставленої за варіантом задачі.
4. Створити функції (або процедури) для роботи зі списком (для створення, обробки, додавання чи видалення елементів, виводу даних зі списку в консоль, звільнення пам'яті тощо).
5. Значення елементів списку взяти самостійно такими, щоб можна було продемонструвати коректність роботи алгоритму програми. Введення значень елементів списку можна виконати довільним способом (випадкові числа, формування значень за формулою, введення з файлу чи з клавіатури).
6. Виконати над створеним списком дії, вказані за варіантом, та коректне звільнення пам'яті списку.
7. **При виконанні заданих дій, виводі значень елементів та звільненні пам'яті списку вважати, що довжина списку (кількість елементів) невідома на момент виконання цих дій.** Тобто, не дозволяється зберігати довжину списку як константу, змінну чи додаткове поле.

При проектуванні програм **слід врахувати наступне:**

- 1) при виконанні завдання кількість операцій (зокрема, операцій читання й запису) має бути мінімізованою, а також максимально мають використовуватися властивості списків;
- 2) повторювані частини алгоритму необхідно оформити у вигляді процедур або функцій (для створення, обробки, виведення та звільнення пам'яті списків) з передачею списку за допомогою параметра(iv).
- 3) у таких видів списків, як черга, стек, дек функції для роботи зі списком мають забезпечувати роботу зі списком, відповідну тому чи іншому виду списку (наприклад, не можна додавати нові елементи всередину черги);
- 4) програми мають бути написані мовою програмування C.

### Варіант № 16

Ключами елементів списку є дійсні числа. Розширити список, дописавши в його кінець свої ж елементи, але у оберненому порядку, не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

## Текст програми

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    double data;
    struct Node* prev;
    struct Node* next;
};

struct Node* createNode(double data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;

    return newNode;
}

void addElements(struct Node** headRef, double data) {
    struct Node* newNode = createNode(data);

    if (*headRef == NULL) {
        *headRef = newNode;
    } else {
        struct Node* current = *headRef;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
        newNode->prev = current;
    }
}

void extendInReverse(struct Node** headRef) {
    struct Node* last = *headRef;

    while (last->next != NULL) {
        last = last->next;
    }
}
```

```

struct Node* current = last;

while (current != NULL) {
    addElements(headRef, current->data);
    current = current->prev;
}
}

void printList(struct Node* head) {
    struct Node* current = head;

    while (current != NULL) {
        printf("%.2f ", current->data);
        current = current->next;
    }
    printf("\n");
}

void freeList(struct Node** headRef) {
    struct Node* current = *headRef;
    struct Node* next;

    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
    *headRef = NULL;
}

int main() {
    int i, n;
    double data;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    struct Node* head = NULL;

    for (i = 0; i < n; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%lf", &data);
        addElements(&head, data);
    }
}

```

```
extendInReverse(&head);

printf("Result: ");
printList(head);

freeList(&head);

return 0;
}
```

### Результати тестування програми

Кількість початкових елементів списку: 4

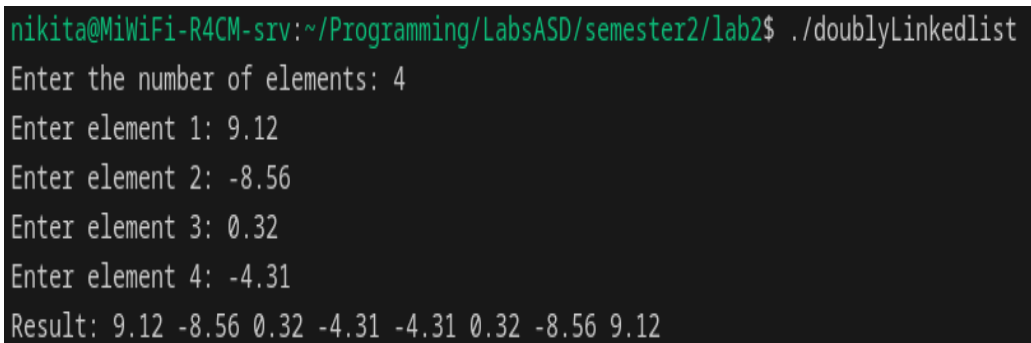
1-ий елемент: 9,12

2-ий елемент: -8,56

3-ий елемент: 0,32

4-ий елемент: -4,31

**Очікуваний результат: 9,12 -8,56 0,32 -4,31 -4,31 0,32 -8,56 9,12**



```
nikita@MiWiFi-R4CM-srv:~/Programming/LabsASD/semester2/lab2$ ./doublyLinkedList
Enter the number of elements: 4
Enter element 1: 9.12
Enter element 2: -8.56
Enter element 3: 0.32
Enter element 4: -4.31
Result: 9.12 -8.56 0.32 -4.31 -4.31 0.32 -8.56 9.12
```

Очікуваний та отриманий результати збігаються.

**Висновок:** на цій лабораторній роботі я навчився складати алгоритми з використанням динамічних структур даних у вигляді списків. Для виконання завдання я обрав двозв'язний лінійний список, оскільки корисно мати вказівник на попередній елемент, коли потрібно розширити список елементами в оберненому порядку. Списки краще підходять для цієї задачі, ніж, наприклад, масиви, адже списки використовують пам'ять ефективно, скорочуючись та розширюючись у залежності від кількості елементів, тоді як у масивів увесь розмір повинен бути визначений заздалегідь, навіть якщо багато елементів можуть бути непотрібними.