**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: Перевірила:

студент групи ІМ-44 Сергієнко М. А.  
Мундурс Нікіта Юрійович  
номер у списку групи: 16

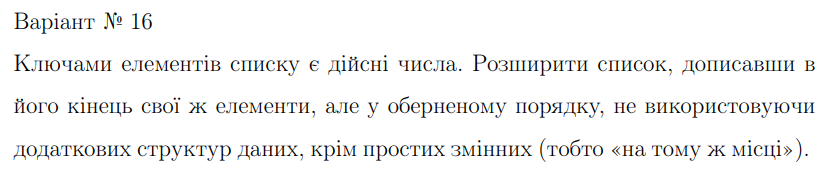
Київ 2025

**Завдання**

1. Створити список з ***n*** (***n*** > 0) елементiв (***n*** вводиться з клавiатури), якщо iнша кiлькiсть елементiв не вказана у конкретному завданнi за варiантом.
2. Тип ключiв (iнформацiйних полiв) задано за варiантом.
3. Вид списку (черга, стек, дек, прямий однозв’язний лiнiйний список, обернений однозв’язний лiнiйний список, двозв’язний лiнiйний список, однозв’язний кiльцевий список, двозв’язний кiльцевий список) вибрати самостiйно з метою найбiльш доцiльного розв’язку поставленої за варiантом задачi.
4. Створити функцiї (або процедури) для роботи зi списком (для створення, обробки, додавання чи видалення елементiв, виводу даних зi списку в консоль, звiльнення пам’ятi тощо).
5. Значення елементiв списку взяти самостiйно такими, щоб можна було продемонструвати коректнiсть роботи алгоритму програми. Введення значень елементiв списку можна виконати довiльним способом (випадковi числа, формування значень за формулою, введення з файлу чи з клавiатури).
6. Виконати над створеним списком дiї, вказанi за варiантом, та коректне звiльнення пам’ятi списку.
7. **При виконаннi заданих дiй, виводi значень елементiв та звiльненнi пам’ятi списку вважати, що довжина списку (кiлькiсть елементiв) *невiдома* на момент виконання цих дiй.** Тобто, не дозволяється зберiгати довжину списку як константу, змiнну чи додаткове поле.

При проєктуваннi програм ***слiд врахувати наступне***:

1. при виконаннi завдання кiлькiсть операцiй (зокрема, операцiй читання й запису) має бути мiнiмiзованою, а також максимально мають використовуватися властивостi спискiв;
2. повторюванi частини алгоритму необхiдно оформити у виглядi процедур або функцiй (для створення, обробки, виведення та звiльнення пам’ятi спискiв) з передачею списку за допомогою параметра(iв).
3. у таких видiв спискiв, як черга, стек, дек функцiї для роботи зi списком мають забезпечувати роботу зi списком, вiдповiдну тому чи iншому виду списку (наприклад, не можна додавати новi елементи всередину черги);
4. програми мають бути написанi мовою програмування С.



**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

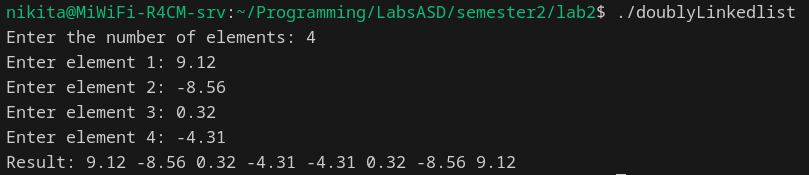
1-ий елемент: 9,12

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**



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.