**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: Перевірив:

студент групи IM-43 Сергієнко А. М.

Олексійчук Станіслав Юрійович

номер у списку групи: 22

Київ 2025

***Постановка задачі***

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.

***Варіант 22:***

Ключами елементiв списку є цiлi ненульовi числа, якi розташовуються в на- ступному порядку: 5 вiд’ємних, 5 додатних i т. д. Кiлькiсть елементiв списку n повинна бути кратною 20-ти. Перекомпонувати елементи списку так, щоб розташування елементiв було наступним: 10 вiд’ємних, 10 додатних i т. д., не використовуючи додаткових структур даних, крiм простих змiнних (тобто «на тому ж мiсцi»).

***Текст програми***

Для цього завдання було використано два методи вводу даних: власноруч та автоматично.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

typedef struct Node {

int info;

struct Node\* next;

} Node;

Node\* init(int element) {

Node\* new\_node = malloc(sizeof(Node));

if (!new\_node) {

printf("Unable to allocate memory.\n");

exit(1);

}

new\_node->info = element;

new\_node->next = NULL;

return new\_node;

}

void add\_element(Node\*\* head, int value) {

Node\* new\_node = init(value);

if (\*head == NULL) {

\*head = new\_node;

} else {

Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = new\_node;

}

}

void print\_list(Node\* head) {

while (head != NULL) {

printf("%d ", head->info);

head = head->next;

}

printf("\n");

}

void free\_list(Node\* head) {

while (head != NULL) {

Node\* temp = head;

head = head->next;

free(temp);

}

}

void move\_elements(Node\* head) {

while (head != NULL) {

Node\* group = head;

Node\* current = group;

for (int count = 10; count > 0 && current != NULL; current = current->next) {

if (current->info >= 0) {

Node\* search = current->next;

while (search != NULL && search->info >= 0)

search = search->next;

if (search) {

int temp = current->info;

current->info = search->info;

search->info = temp;

count--;

}

} else {

count--;

}

}

for (int i = 0; i < 20 && head != NULL; i++) {

head = head->next;

}

}

}

void manual\_input(Node\*\* head, int n) {

int count = 0, num;

int toggle = 0;

while (count < n) {

printf("Enter %s number %d: ", toggle ? "positive" : "negative", count + 1);

scanf("%d", &num);

if ((toggle == 0 && num < 0) || (toggle == 1 && num > 0)) {

add\_element(head, num);

if (++count % 5 == 0) toggle = !toggle;

} else {

printf("Number must be %s.\n", toggle ? "positive" : "negative");

}

}

}

void auto\_generate(Node\*\* head, int n) {

int count = 0;

int toggle = 0;

while (count < n) {

int num = (rand() % 50 + 1) \* (toggle ? 1 : -1);

add\_element(head, num);

count++;

if (count % 5 == 0) toggle = !toggle;

}

}

int main() {

Node\* head = NULL;

int n, mode;

srand(time(NULL));

printf("Enter n of elements (must be positive and multiple of 20): ");

scanf("%d", &n);

if (n <= 0 || n % 20 != 0) {

printf("Number must be positive and multiple of 20!\n");

return 1;

}

printf("Choose input mode: (1 - Manual, 2 - Auto): ");

scanf("%d", &mode);

if (mode == 1) {

manual\_input(&head, n);

} else if (mode == 2) {

auto\_generate(&head, n);

} else {

printf("Invalid mode.\n");

return 1;

}

printf("\nInitial list:\n");

print\_list(head);

move\_elements(head);

printf("\nList after changes:\n");

print\_list(head);

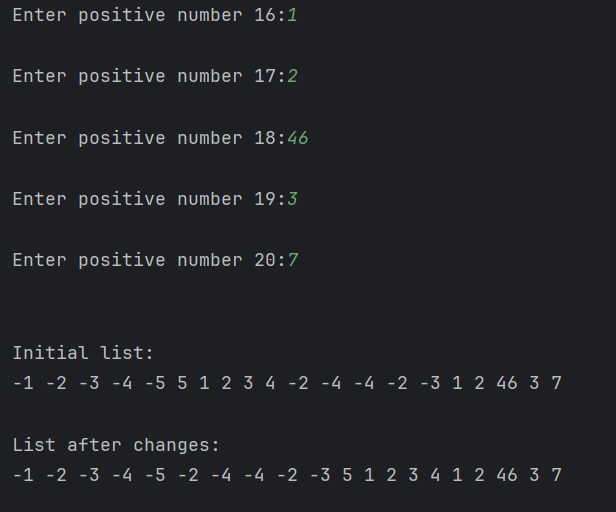
free\_list(head);

return 0;

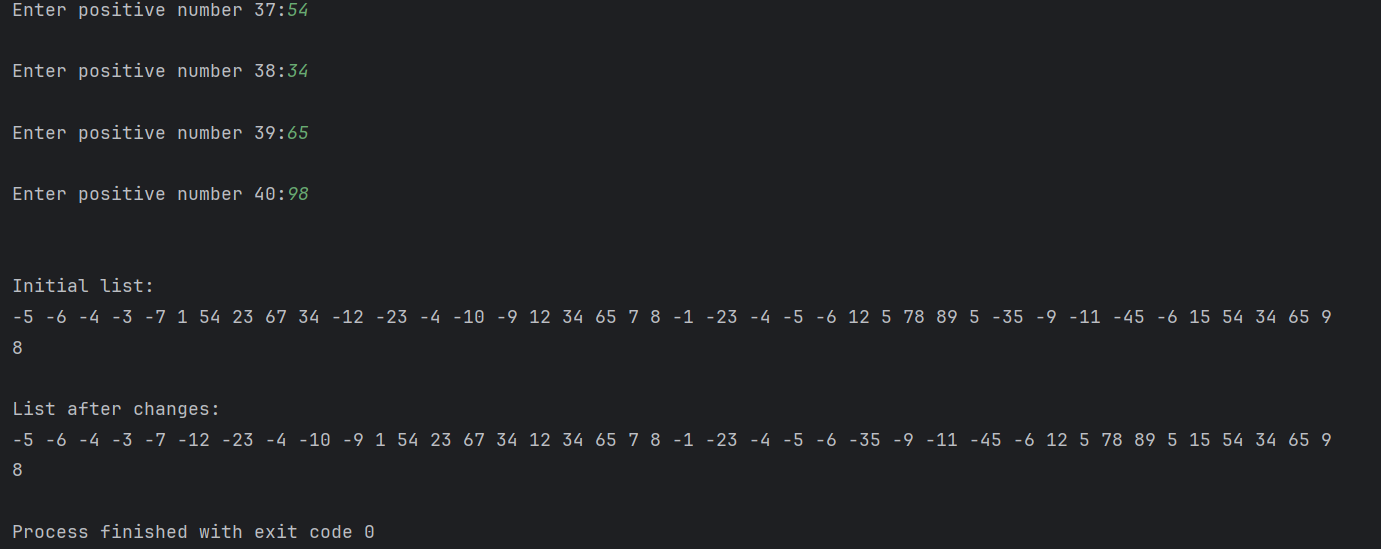
}

***Результати тестування програми***

1. Manual input
2. 20 елементів

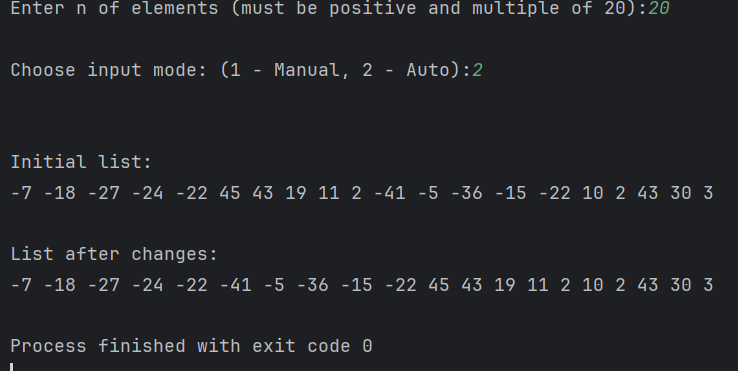


б) 40 елементів

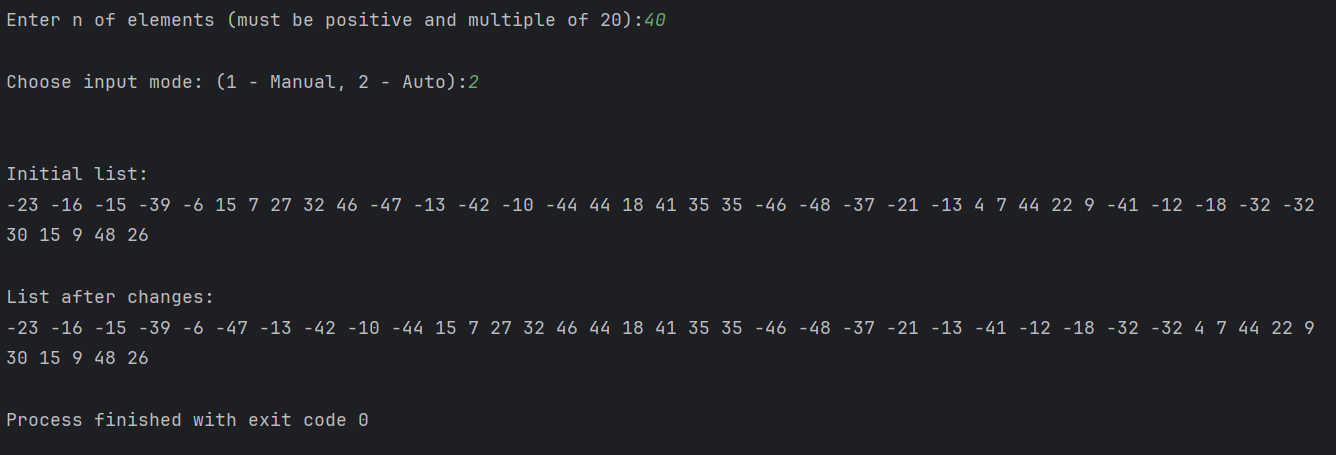


1. Auto input

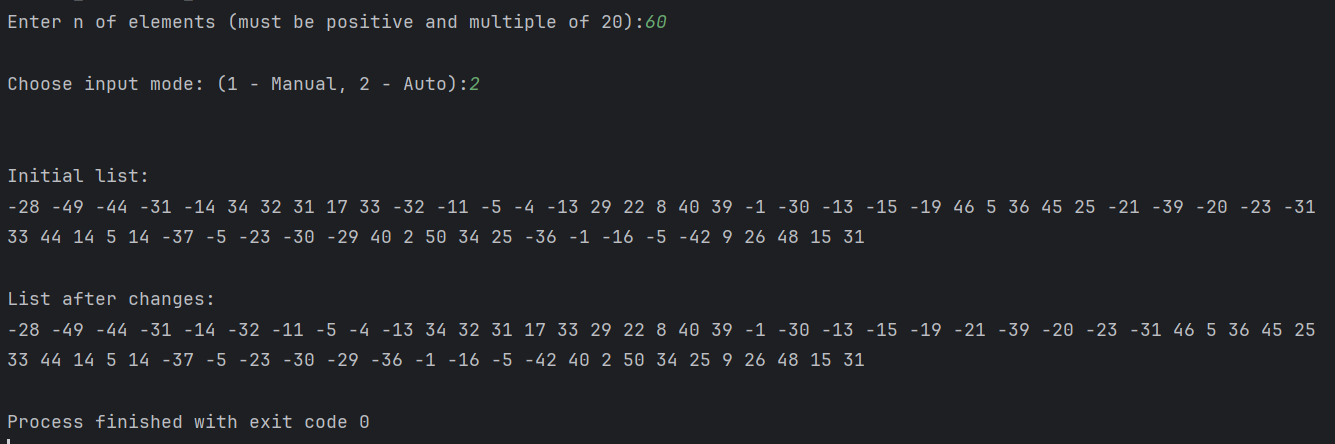
а) 20 елементів



б) 40 елементів

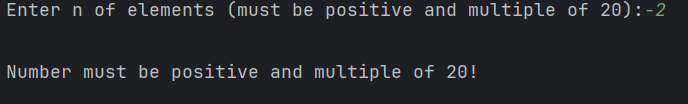
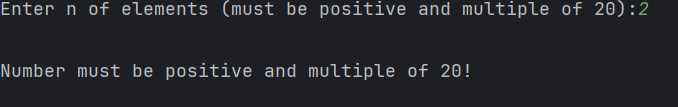


в) 60 елементів

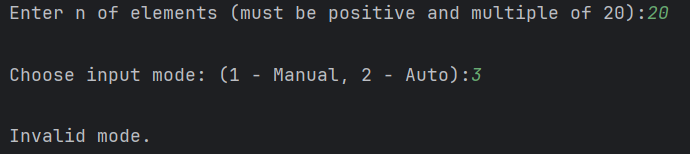
******

1. Error handling

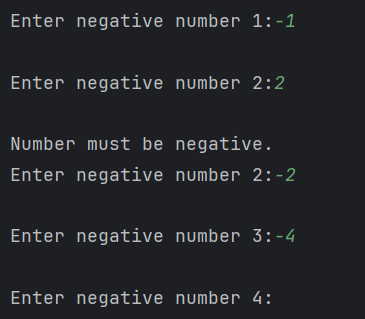
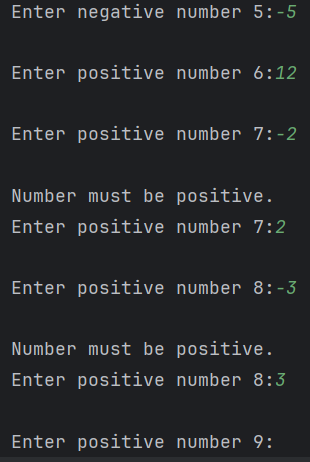
а) Кількість елементів

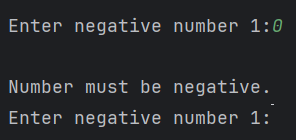
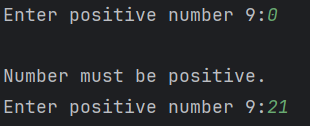


б) Метод вводу



в) Ввід даних



***Висновки***

У ході виконання лабораторної роботи було реалізовано прямий однозв’язний лінійний список, що дозволяє додавати, переглядати, змінювати та очищувати елементи. Було розроблено два режими введення: ручний (із валідацією знаків чисел) та автоматичний (із генерацією значень у заданому шаблоні). Також реалізовано функцію move\_elements(), яка здійснює обмін елементів, забезпечуючи правильне чергування груп по 10 від’ємних і 10 додатних чисел.

У результаті виконання роботи було отримано практичні навички роботи з динамічною пам’яттю та списками на мові C, опрацьовано навички переструктурування елементів у списку без використання додаткових структур, а також закріплено розуміння принципів маніпуляції з вузлами в однозв’язному списку. Програма була протестована для різної кількості елементів і працює коректно за будь-якого коректного вхідного значення, що кратне 20.