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-43
Олексійчук Станіслав Юрійович
номер у списку групи: 22

Перевірив:
Сергієнко А. М.

Київ 2025

Постановка задачі

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.

Варіант 22:

Ключами елементів списку є цілі ненульові числа, які розташовуються в наступному порядку: 5 від'ємних, 5 додатних і т. д. Кількість елементів списку n повинна бути кратною 20-ти. Перекомпонувати елементи списку так, щоб розташування елементів було наступним: 10 від'ємних, 10 додатних і т. д., не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

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

   a) 20 елементів

```
Enter positive number 16:1

Enter positive number 17:2

Enter positive number 18:46

Enter positive number 19:3

Enter positive number 20:7


Initial list:
-1 -2 -3 -4 -5 5 1 2 3 4 -2 -4 -4 -2 -3 1 2 46 3 7

List after changes:
-1 -2 -3 -4 -5 -2 -4 -4 -2 -3 5 1 2 3 4 1 2 46 3 7
```

   б) 40 елементів

```
Enter positive number 37:54

Enter positive number 38:34

Enter positive number 39:65

Enter positive number 40:98


Initial list:
-5 -6 -4 -3 -7 1 54 23 67 34 -12 -23 -4 -10 -9 12 34 65 7 8 -1 -23 -4 -5 -6 12 5 78 89 5 -35 -9 -11 -45 -6 15 54 34 65 9
8

List after changes:
-5 -6 -4 -3 -7 -12 -23 -4 -10 -9 1 54 23 67 34 12 34 65 7 8 -1 -23 -4 -5 -6 -35 -9 -11 -45 -6 12 5 78 89 5 15 54 34 65 9
8

Process finished with exit code 0
```

2) Auto input

   a) 20 елементів

```
Enter n of elements (must be positive and multiple of 20):20

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


Initial list:
-7 -18 -27 -24 -22 45 43 19 11 2 -41 -5 -36 -15 -22 10 2 43 30 3

List after changes:
-7 -18 -27 -24 -22 -41 -5 -36 -15 -22 45 43 19 11 2 10 2 43 30 3

Process finished with exit code 0
```

б) 40 елементів

```
Enter n of elements (must be positive and multiple of 20):40

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


Initial list:
-23 -16 -15 -39 -6 15 7 27 32 46 -47 -13 -42 -10 -44 44 18 41 35 35 -46 -48 -37 -21 -13 4 7 44 22 9 -41 -12 -18 -32 -32
30 15 9 48 26

List after changes:
-23 -16 -15 -39 -6 -47 -13 -42 -10 -44 15 7 27 32 46 44 18 41 35 35 -46 -48 -37 -21 -13 -41 -12 -18 -32 -32 4 7 44 22 9
30 15 9 48 26

Process finished with exit code 0
```

в) 60 елементів

```
Enter n of elements (must be positive and multiple of 20):60

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


Initial list:
-28 -49 -44 -31 -14 34 32 31 17 33 -32 -11 -5 -4 -13 29 22 8 40 39 -1 -30 -13 -15 -19 46 5 36 45 25 -21 -39 -20 -23 -31
33 44 14 5 14 -37 -5 -23 -30 -29 40 2 50 34 25 -36 -1 -16 -5 -42 9 26 48 15 31

List after changes:
-28 -49 -44 -31 -14 -32 -11 -5 -4 -13 34 32 31 17 33 29 22 8 40 39 -1 -30 -13 -15 -19 -21 -39 -20 -23 -31 46 5 36 45 25
33 44 14 5 14 -37 -5 -23 -30 -29 -36 -1 -16 -5 -42 40 2 50 34 25 9 26 48 15 31

Process finished with exit code 0
```

3) Error handling

а) Кількість елементів

```
Enter n of elements (must be positive and multiple of 20):2

Number must be positive and multiple of 20!
```

```
Enter n of elements (must be positive and multiple of 20):-2

Number must be positive and multiple of 20!
```

б) Метод вводу

```
Enter n of elements (must be positive and multiple of 20):20

Choose input mode: (1 - Manual, 2 - Auto):3

Invalid mode.
```

в) Ввід даних

```
Enter negative number 5:-5

Enter positive number 6:12

Enter positive number 7:-2

Number must be positive.
Enter positive number 7:2

Enter positive number 8:-3

Number must be positive.
Enter positive number 8:3

Enter positive number 9:
```

```
Enter negative number 1:-1

Enter negative number 2:2

Number must be negative.
Enter negative number 2:-2

Enter negative number 3:-4

Enter negative number 4:
```

```
Enter negative number 1:0

Number must be negative.
Enter negative number 1:
```

```
Enter positive number 9:0

Number must be positive.
Enter positive number 9:21
```

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 С, 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.