**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-42
Федоренко Іван Русланович
номер у списку групи: 29

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.

   При проєктуванні 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(ів).

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 С.

Варіант № 29

Ключами елементів списку є рядки довжиною не більше 25-ти символів. Кількість елементів списку $\boldsymbol{n}$ повинна бути кратною 20-ти. Перекомпонувати список всередині кожних 20-ти елементів, розташувавши їх у наступному порядку: $a_1, a_{11}, a_2, a_{12}, \ldots, a_{21}, a_{31}, a_{22}, a_{32}, \ldots$, де $\boldsymbol{a_i}$ — $\boldsymbol{i}$ елемент списку. При необхідності дозволяється використати ще один список, інші структури даних, крім простих змінних, використовувати не дозволяється.

## Код програми:

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

#define MAX_STRING_LENGTH 26
#define BLOCK 20
#define HALF  (BLOCK/2)

typedef struct Node {
    char data[MAX_STRING_LENGTH];
    struct Node* next;
} Node;

Node* create_node(const char* str) {
    Node* new_node = malloc(sizeof(Node));
    if (!new_node) {
        perror("malloc");
        exit(EXIT_FAILURE);
    }
    strncpy(new_node->data, str, MAX_STRING_LENGTH - 1);
    new_node->data[MAX_STRING_LENGTH - 1] = '\0';
    new_node->next = NULL;
    return new_node;
}

void append(Node** head, const char* str) {
    Node* nd = create_node(str);
    if (*head == NULL) {
        *head = nd;
    } else {
        Node* t = *head;
        while (t->next) t = t->next;
        t->next = nd;
    }
}

void print_list(Node* head) {
    for (; head; head = head->next) {
        printf("%s ", head->data);
    }
    printf("\n");
}

void free_list(Node* head) {
    while (head) {
        Node* t = head;
        head = head->next;
```

```
        free(t);
    }
}

Node* rearrange(Node* head) {
    Node dummy;
    Node* tail = &dummy;
    dummy.next = NULL;

    Node* rest = head;

    while (rest) {
        Node* first_half = rest;
        Node* mid = rest;

        for (int i = 0; i < HALF; i++) {
            if (!mid) return dummy.next;
            mid = mid->next;
        }

        Node* block_tail = mid;
        for (int i = 0; i < HALF - 1; i++) {
            if (!block_tail) return dummy.next;
            block_tail = block_tail->next;
        }
        Node* next_block = block_tail ? block_tail->next : NULL;
        if (block_tail) block_tail->next = NULL;

        while (first_half && mid) {
            Node* first = first_half;
            first_half = first_half->next;
            first->next = NULL;
            tail->next = first;
            tail = first;

            Node* second = mid;
            mid = mid->next;
            second->next = NULL;
            tail->next = second;
            tail = second;
        }

        rest = next_block;
    }

    return dummy.next;
}

int main(void) {
```

```c
    int n;
    printf("Enter number of elements (positive multiple of %d): ", BLOCK);
    if (scanf("%d", &n) != 1 || n <= 0 || n % BLOCK != 0) {
        fprintf(stderr, "Bad input: n must be positive multiple of %d\n", BLOCK);
        return EXIT_FAILURE;
    }

    Node* list = NULL;
    char buf[MAX_STRING_LENGTH];

    printf("Enter %d strings (up to %d chars each):\n", n, MAX_STRING_LENGTH - 1);
    for (int i = 0; i < n; i++) {
        if (scanf("%25s", buf) != 1) {
            fprintf(stderr, "Failed to read string\n");
            free_list(list);
            return EXIT_FAILURE;
        }
        append(&list, buf);
    }

    printf("\nOriginal list:\n");
    print_list(list);

    Node* new_list = rearrange(list);

    printf("\nRearranged list:\n");
    print_list(new_list);

    free_list(new_list);
    return EXIT_SUCCESS;
}
```

## Результати тестування:

```
Enter the number of elements (multiple of 20): 20
Enter the elements (strings up to 25 characters each):
one

two

three

four

five

six

seven

eight

nine

ten

eleven

twelve

thirteen

fourteen

fifteen

sixteen

seventeen

eighteen

nineteen

twenty
```

```
seventeen

eighteen

nineteen

twenty

Original list:
one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twenty
Rearranged list:
one eleven two twelve three thirteen four fourteen five fifteen six sixteen seven seventeen eight eighteen nine nineteen ten twenty
```

```
Enter the number of elements (multiple of 20): 40
Enter the elements (strings up to 25 characters each):
one
```

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
sixteen
seventeen
eighteen
nineteen
twenty
twenty-one
twenty-two
twenty-three
twenty-four
```

```
twenty-two
twenty-three
twenty-four
twenty-five
twenty-six
twenty-seven
twenty-eight
twenty-nine
thirty
thirty-one
thirty-two
thirty-three
thirty-four
thirty-five
thirty-six
thirty-seven
thirty-eight
thirty-nine
forty
```

```
Original list:
one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twent
y twenty-one twenty-two twenty-three twenty-four twenty-five twenty-six twenty-seven twenty-eight twenty-nine thirty thirty-one th
irty-two thirty-three thirty-four thirty-five thirty-six thirty-seven thirty-eight thirty-nine forty
Rearranged list:
one eleven two twelve three thirteen four fourteen five fifteen six sixteen seven seventeen eight eighteen nine nineteen ten twent
y twenty-one thirty-one twenty-two thirty-two twenty-three thirty-three twenty-four thirty-four twenty-five thirty-five twenty-six
 thirty-six twenty-seven thirty-seven twenty-eight thirty-eight twenty-nine thirty-nine thirty forty
```

```
Enter the number of elements (multiple of 20): 60
Enter the elements (strings up to 25 characters each):
one
```

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
sixteen
seventeen
eighteen
nineteen
twenty
twenty-one
twenty-two
twenty-three
twenty-four
twenty-five
```

```
twenty-six
twenty-seven
twenty-eight
twenty-nine
thirty
thirty-one
thirty-two
thirty-three
thirty-four
thirty-five
thirty-six
thirty-seven
thirty-eight
thirty-nine
forty
forty-one
forty-two
forty-three
forty-four
forty-five
forty-six
forty-seven
forty-eight
forty-nine
fifty
```

```
fifty
fifty-one
fifty-two
fifty-three
fifty-four
fifty-five
fifty-six
fifty-seven
fifty-eight
fifty-nine
sixty
```

```
sixty

Original list:
one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twent
y twenty-one twenty-two twenty-three twenty-four twenty-five twenty-six twenty-seven twenty-eight twenty-nine thirty thirty-one th
irty-two thirty-three thirty-four thirty-five thirty-six thirty-seven thirty-eight thirty-nine forty forty-one forty-two forty-thr
ee forty-four forty-five forty-six forty-seven forty-eight forty-nine fifty fifty-one fifty-two fifty-three fifty-four fifty-five
fifty-six fifty-seven fifty-eight fifty-nine sixty
Rearranged list:
one eleven two twelve three thirteen four fourteen five fifteen six sixteen seven seventeen eight eighteen nine nineteen ten twent
y twenty-one thirty-one twenty-two thirty-two twenty-three thirty-three twenty-four thirty-four twenty-five thirty-five twenty-six
 thirty-six twenty-seven thirty-seven twenty-eight thirty-eight twenty-nine thirty-nine thirty forty forty-one fifty-one forty-two
 fifty-two forty-three fifty-three forty-four fifty-four forty-five fifty-five forty-six fifty-six forty-seven fifty-seven forty-e
ight fifty-eight forty-nine fifty-nine fifty sixty
```

```
Enter number of elements (positive multiple of 20): 32
Bad input: n must be positive multiple of 20
```

## Висновки:

Я навчився працювати з однонаправленим зв'язним списком рядків, маніпулювати списком відповідно до заданого шаблону та ефективно управляти пам'яттю. Також закріпив знання з перерозподілення списку та виведення результату, звільняючи пам'ять в кінці для уникнення витоків.