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»

Виконала:

студентка групи ІМ - 43
Хоанг Чан Кам Лі
номер у списку групи: 29

Перевірив:

Сергієнко А. М.

Київ 2025

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.

Варіант 29:

Ключами елементів списку є рядки довжиною не більше 25-ти символів. Кількість елементів списку 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 a_i — i елемент списку. При необхідності дозволяється використати ще один список, інші структури даних, крім простих змінних, використовувати не дозволяється.

Текст програми

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

typedef struct Node {
    char str[26];
    struct Node* next;
} Node;
Node* createNode(const char* str) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(1);
    }
    strcpy(newNode->str, str);
    newNode->next = NULL;
    return newNode;
}
void appendNode(Node** head, Node** tail, const char*
str) {
    Node* newNode = createNode(str);
    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        (*tail)->next = newNode;
```

```
        *tail = newNode;
    }
}
Node* getNodeAt(Node* head, int pos) {
    Node* current = head;
    for (int i = 0; i < pos && current != NULL; i++) {
        current = current->next;
    }
    return current;
}
void addNodeToNewList(Node** newHead, Node** newTail,
Node* nodeToAdd) {
    if (nodeToAdd == NULL) return;

    nodeToAdd->next = NULL;

    if (*newHead == NULL) {
        *newHead = *newTail = nodeToAdd;
    } else {
        (*newTail)->next = nodeToAdd;
        *newTail = nodeToAdd;
    }
}
void interleaveList(Node** head) {
    if (*head == NULL) return;

    Node* groupStart = *head;
    Node* newHead = NULL;
    Node* newTail = NULL;

    while (groupStart != NULL) {
        Node* X = groupStart;
        Node* Y = getNodeAt(groupStart, 10);
        Node* nextGroupStart = getNodeAt(groupStart,
20);
```

```
        for (int i = 0; i < 10; i++) {
            if (X != NULL) {
                Node* nextX = X->next;
                addNodeToNewList(&newHead, &newTail, X);
                X = nextX;
            }
            if (Y != NULL) {
                Node* nextY = Y->next;
                addNodeToNewList(&newHead, &newTail, Y);
                Y = nextY;
            }
        }
        groupStart = nextGroupStart;
    }

    *head = newHead;
}
void printList(Node* head) {
    Node* current = head;
    while (current != NULL) {
        printf("%s\n", current->str);
        current = current->next;
    }
}
void freeList(Node* head) {
    Node* current = head;
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }
}
int main() {
    int N;
```

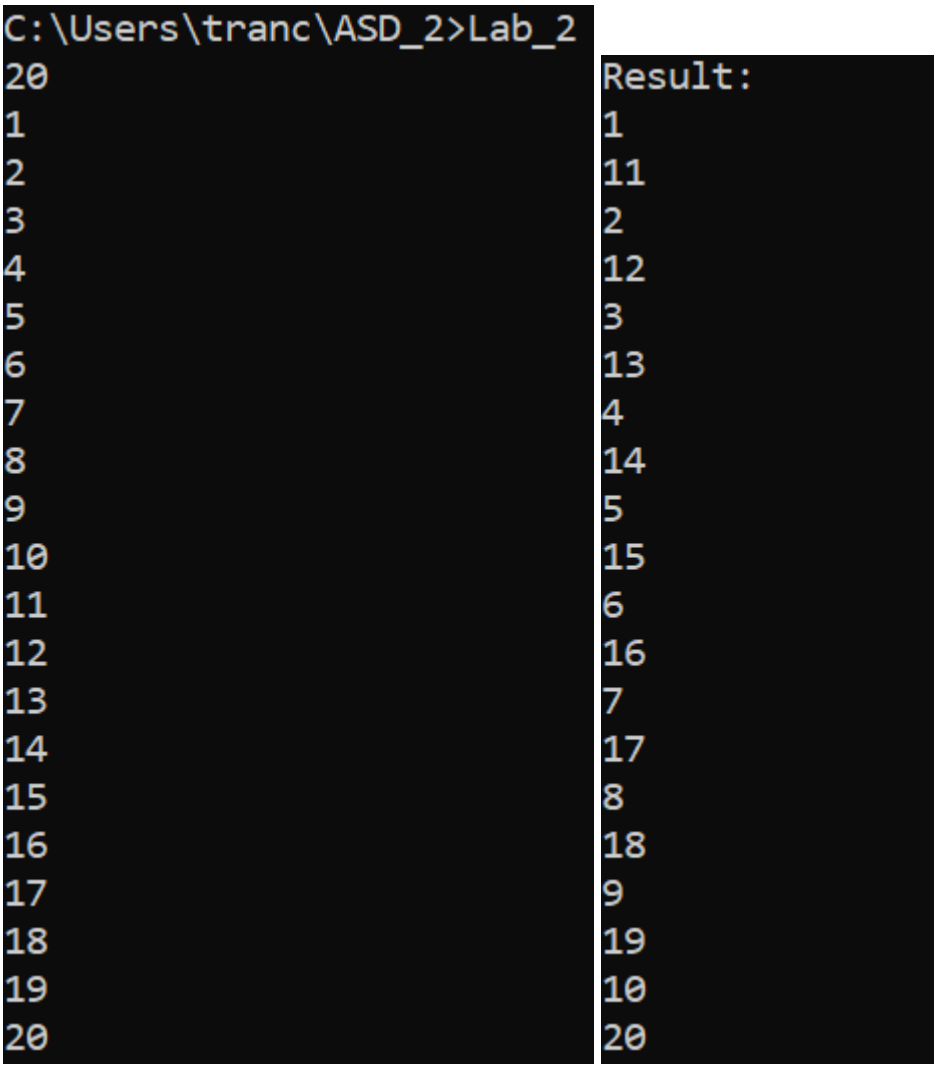
```
    scanf("%d", &N);
    getchar();

    Node* head = NULL;
    Node* tail = NULL;
    for (int i = 0; i < N; i++) {
        char buffer[26];
        if (fgets(buffer, sizeof(buffer), stdin) !=
NULL) {
            buffer[strcspn(buffer, "\n")] = '\0';
            appendNode(&head, &tail, buffer);
        }
    }
    interleaveList(&head);

    printf("Result:\n");
    printList(head);
    freeList(head);

    return 0;
}
```

Результати тестування програми та перевірки



Висновки

У результаті виконання лабораторної роботи було реалізовано алгоритм обробки динамічної структури даних - однозв’язного лінійного списку, що містить символьні рядки. Основна мета полягала у здійсненні групової модифікації списку шляхом формування блоків по 20 елементів з наступним чергуванням перших і других 10 елементів у кожному блоці.

Особливістю поставленого завдання є необхідність чіткого контролю над структурними зв’язками між елементами списку. Це, у свою чергу, вимагає уважної роботи з вказівниками та коректної реалізації функцій, що оперують із динамічною пам’яттю. Створення допоміжних функцій для додавання, виводу, переструктурування та очищення списку забезпечило модульність коду, полегшило налагодження й зробило програму гнучкою до можливих змін у структурі задачі.