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
Костеніч Степан Станіславович
номер у списку групи: 17

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 №17

Ключами елементів списку є дійсні числа. Обчислити значення виразу:

$a_1 \cdot a_n + a_2 \cdot a_{n-1} + \ldots + a_n \cdot a_1$, де $\boldsymbol{a_i}$ — $\boldsymbol{i}$-й елемент списку.

Текст програми

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct List {
    double data;
    struct List* next;
    struct List* prev;
} List;

void InputNumberOfElements(int* n) {
    printf("Enter the number of elements:");
    if (scanf("%d", n) != 1 || *n <= 0) {
        printf("\nInvalid input\n");
        exit(EXIT_FAILURE);
    }
}

void Append(List** head, const double value) {
    List* newNode = malloc(sizeof(List));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }

    newNode->data = value;

    if (!*head) {
        newNode->next = newNode->prev = newNode;
        *head = newNode;
    } else {
        List* tail = (*head)->prev;
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}

double RandomRange(const double min, const double max) {
    return min + rand() / (RAND_MAX / (max - min));
}
```

```
void FillListWithRandomValues(List** head, const int n) {
    for (int i = 0; i < n; i++) {
        double randValue = RandomRange(-10.0, 10.0);
        double roundedValue = round(randValue * 100) / 100;
        Append(head, roundedValue);
    }
}

void PrintList(List* head) {
    if (!head) return;

    List* temp = head;

    printf("\nRandomly filled list:\n");
    do {
        printf("%.2lf ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}

void ComputeExpression(List* head, const int n) {
    if (!head) return;

    List* left = head;
    List* right = head->prev;
    double sum = 0;

    for (int i = 0; i < n / 2; i++) {
        sum += left->data * right->data * 2;
        left = left->next;
        right = right->prev;
    }

    if ((n & 1) != 0) {
        sum += left->data * left->data;
    }

    printf("\nResult: %.2lf\n", sum);
}

void FreeList(List** head) {
    if (!*head) return;

    List* current = *head;
```

```
    printf("\nFreeing memory:\n");
    do {
        List *temp = current;
        current = current->next;
        printf("Freeing up: %.2lf\n", temp->data);
        free(temp);
    } while (current != *head);

    *head = NULL;
    printf("Memory freeing completed\n");
}

int main() {
    srand(time(NULL));

    int n;
    List* head = NULL;

    InputNumberOfElements(&n);

    FillListWithRandomValues(&head, n);
    PrintList(head);
    ComputeExpression(head, n);

    FreeList(&head);

    return 0;
}
```

Результати тестування програми

```
Enter the number of elements: 0

Invalid input

Process finished with exit code 1
```

```
Enter the number of elements: 1

Randomly filled list:
-9.20

Result: 84.64

Freeing memory:
Freeing up: -9.20
Memory freeing completed

Process finished with exit code 0
```

```
Enter the number of elements: 2

Randomly filled list:
-9.12 3.75

Result: -68.40

Freeing memory:
Freeing up: -9.12
Freeing up: 3.75
Memory freeing completed

Process finished with exit code 0
```

```
Enter the number of elements: 3

Randomly filled list:
-9.08 8.40 7.22

Result: -60.56

Freeing memory:
Freeing up: -9.08
Freeing up: 8.40
Freeing up: 7.22
Memory freeing completed

Process finished with exit code 0
```

```
Enter the number of elements: 6

Randomly filled list:
-9.02 -8.23 5.23 4.73 -5.40 5.88

Result: 32.28

Freeing memory:
Freeing up: -9.02
Freeing up: -8.23
Freeing up: 5.23
Freeing up: 4.73
Freeing up: -5.40
Freeing up: 5.88
Memory freeing completed

Process finished with exit code 0
```

```
Enter the number of elements: 9

Randomly filled list:
-8.96 1.70 -5.86 -5.29 -3.95 2.57 2.15 3.81 -5.79

Result: 79.92

Freeing memory:
Freeing up: -8.96
Freeing up: 1.70
Freeing up: -5.86
Freeing up: -5.29
Freeing up: -3.95
Freeing up: 2.57
Freeing up: 2.15
Freeing up: 3.81
Freeing up: -5.79
Memory freeing completed

Process finished with exit code 0
```

```
Enter the number of elements: 10

Randomly filled list:
-8.77 4.93 9.96 9.99 -5.87 -9.74 2.62 -5.92 -8.39 0.61

Result: -44.66

Freeing memory:
Freeing up: -8.77
Freeing up: 4.93
Freeing up: 9.96
Freeing up: 9.99
Freeing up: -5.87
Freeing up: -9.74
Freeing up: 2.62
Freeing up: -5.92
Freeing up: -8.39
Freeing up: 0.61
Memory freeing completed

Process finished with exit code 0
```

Висновок

Під час виконання лабораторної роботи № 2 я засвоїв теоретичний матеріал та набув практичного досвіду використання зв’язаних динамічних структур даних у вигляді одно- та однозв’язних списків при складанні різних алгоритмів.

Я на практиці закріпив знання щодо зв’язаних динамічних структур даних, зокрема двозв’язних циклічних списків. Я отримав практичний досвід реалізації базових операцій із цим видом структури: створення, виділення пам’яті, додавання елементів, обхід, обчислення виразів на основі його елементів та коректне звільнення пам’яті.

Я краще зрозумів та розібрався у різних видах списків, показаних на лекціях. Зокрема, проаналізувавши завдання свого варіанту, зупинився на двозв’язному циклічному списку. Він поєднує переваги двохзв'язного та циклічного списків. Можна мати доступ через голову та хвіст, що чудово задовольняло потреби мого виразу. Циклічність полегшила пересування до кінця списку, хоч і мала свою ціну. За деяких обставин циклічність списку може призвести до додаткових витрат. Також було трохи важче визначити, чи обхід списку повністю пройшов коло чи ні, порівняно зі звичайним двозв’язним списком. Я це відчув під час звільнення пам’яті, зробивши декілька помилок, але вчасно помітив це при тестуванні та виправив проблему.

Під час виконання роботи я також покращив свої навички роботи з мовою С, зокрема у використанні покажчиків, динамічного виділення пам’яті та структур даних. Я зрозумів важливість правильного управління пам’яттю та необхідність уважного підходу до логіки обчислень у випадках, коли довжина списку невідома наперед.

Отже, виконання лабораторної роботи № 2 було корисним, дозволило закріпити теоретичні знання та набути практичних навичок в області програмування мовою С.