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-32
Король Олександр Володимирович
номер у списку групи: 14

Перевірила:

Молчанова А. А.

Київ 2024

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 № 15

Ключами елементів списку є дійсні числа. Перекомпонувати елементи списку таким чином, щоб його елементи розташовувались у такому порядку: $a_1, a_n, a_2, a_{n-1}, ..., a_{[\frac{n+1}{2}]}$, де $\boldsymbol{a_i}$ — $\boldsymbol{i}$-й елемент списку. При необхідності дозволяється використати ще один список, інші структури даних, крім простих змінних, використовувати не дозволяється.

Текст програми

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

typedef struct Node {
   float data;
   struct Node* next;
} Node; // лінійний однозв'язний список

// Функція для створення нового вузла
Node* createNode(float data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

// Функція для вставки нового вузла у початок списку
Node* insertNode(Node* currentNode, float data) {
   if (!currentNode) {
       currentNode = createNode(data);
   } else {
```

```
        Node* newNode = createNode(data);
        newNode->next = currentNode;
        currentNode = newNode;
    }
    return currentNode;
}

// Функція для виведення списку на екран
void printList(Node* currentNode) {
    Node* pointer = currentNode;
    while (pointer) {
        printf("%f ", pointer->data);
        pointer = pointer->next;
    }
    printf("\n");
}

// Функція для вивільнення пам'яті, звільнення списку
void freeList(Node* currentNode) {
    Node* temp = NULL;
    while (currentNode != NULL) {
        temp = currentNode;
        currentNode = currentNode->next;
        free(temp);
    }
}

// Функція для перестановки вузлів у списку
void swapList(Node** headRef) {
```

```
    Node* head = *headRef;
    if (!head || !head->next)
        return;

    // знайти середину листа
    Node* slow = head;
    Node* fast = head->next;
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }

    // відокремити листи на 2 частини
    Node* secondHalf = slow->next;
    slow->next = NULL;

    Node* currentOfSecondHalf = secondHalf;
    Node* nextNode = currentOfSecondHalf->next;
    currentOfSecondHalf->next = NULL;
    while (nextNode) {
        currentOfSecondHalf = insertNode(currentOfSecondHalf,
nextNode->data);
        nextNode = nextNode->next;
    }

    // об'єднати два листа в один
    Node* currentOfFirstHalf = head;
    while (currentOfSecondHalf) {
        // зберігаємо наступні елементи кожної половини
       Node* nextOfFirstHalf = currentOfFirstHalf->next;
```

```
        Node* nextOfSecondHalf = currentOfSecondHalf->next;
        // об'єднуємо листи
        currentOfFirstHalf->next = currentOfSecondHalf;
        currentOfSecondHalf->next = nextOfFirstHalf;
        // переходимо до наступних елементів
        currentOfSecondHalf = nextOfSecondHalf;
        currentOfFirstHalf = nextOfFirstHalf;
    }
}

int main() {
    float data;
    Node* head = NULL;
    int n;
    printf("enter n =");
    scanf("%d", &n);
    // Зчитуємо дані для створення списку з консолі
    printf("Enter the values for the nodes\n");
    for (int i = 0; i < n; i++) {
        printf("Enter a value:");
        scanf("%f", &data);
        head = insertNode(head, data);
    }
    // Виводимо початковий список на екран
    printf("Original list: ");
    printList(head);

    // Переставляємо вузли у списку
    swapList(&head);
```

```
    // Виводимо переставлений список на екран
    printf("Rearranged list: ");
    printList(head);

    // Звільняємо пам'ять, вивільняємо список
    freeList(head);

    return 0;
}
```

Результати тестування програми

```
D:\programming\university\labs\DSA-LABS\second_semester\Lab_2\cmake-build-debug\Lab_2.exe
enter n =1
Enter the values for the nodes
Enter a value:1
Original list: 1.000000
Rearranged list: 1.000000

Process finished with exit code 0
```

```
D:\programming\university\labs\DSA-LABS\second_semester\Lab_2\cmake-build-debug\Lab_2.exe
enter n =5
Enter the values for the nodes
Enter a value:5
Enter a value:4
Enter a value:3
Enter a value:2
Enter a value:1
Original list: 1.000000 2.000000 3.000000 4.000000
 5.000000
Rearranged list: 1.000000 5.000000 2.000000 4.000000 3.000000

Process finished with exit code 0
```

```
D:\programming\university\labs\DSA-LABS\second_semester\Lab_2\cmake-build-debug\Lab_2.exe
enter n =6
Enter the values for the nodes
Enter a value:6
Enter a value:5
Enter a value:4
Enter a value:3
Enter a value:2
Enter a value:1
Original list: 1.000000 2.000000 3.0
00000 4.000000 5.000000 6.000000
Rearranged list: 1.000000 6.000000 2.000000 5.000000 3.000000 4.000000

Process finished with exit code 0
```

Висновки

Під час виконання цієї лабораторної роботи я отримав цінний практичний досвід зі створення динамічних структур даних, зокрема однозв’язних лінійних списків. Розділивши список на дві частини та використовуючи той факт, що елементи додаються в початок списку, я зумів ефективно та легко перекомпонувати елемети списку так, щоб вони йшли в порядку: перший з початку, перший з кінця, другий з початку, другий з кінця..., що виявилось легким та зручним способом

У процесі вивчення теоретичного матеріалу я ознайомився з різними аспектами роботи з однозв’язними лінійними списками, включаючи додавання елементів у список, вивільнення пам’яті під кожен елемент та після використання динамічної структури даних. Це дозволило мені отримати глибше розуміння принципів функціонування цих структур.

Результати дослідження підтвердили, що однозв'язні лінійні списки виявляються корисними та ефективними інструментами у різних областях. Зокрема, вони демонструють свою силу в ситуаціях, де потрібно оперативно маніпулювати даними.

Отже, ця лабораторна робота дала мені не лише практичний досвід, але й поглиблене розуміння принципів роботи з динамічними структурами даних, що буде корисним у подальших проектах та розвитку моїх навичок програмування.