Національний технічний університет України “Київський політехнічний інститут”

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №2.6

з дисципліни

“Структури даних і алгоритми”

ТЕМА: “ЗВ’ЯЗАНІ ДИНАМІЧНІ СТРУКТУРИ ДАНИХ. СПИСКИ”

Група: КВ-34

Виконав

: Фесенко Денис

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.
2. Виконати над створеним списком дії, задані за варіантом, та коректне звільнення пам'яті списку.
3. При виконанні заданих дій, виводі значень елементів та звільненні пам'яті списку вважати, що довжина списку (кількість елементів створеного списку) невідома на момент виконання цих дій.
Тобто, ідентифікатор змінної n НЕ повинен використовуватися при реалізації алгоритмів обробки списку та виведенні результуючого списку, а перевірка досягнення кінців списку допускається тільки перевіркою вказівників на значення NULL.
4. Тип ключів (інформаційних полів) задано за варіантом. 91
5. Введення значень елементів списку можна виконати довільним способом.
6. 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 рішення поставленої за варіантом задачі.
7. Повторювані частини алгоритму необхідно оформити у вигляді функцій (для створення списку, обробки списку, виведення та звільнення пам'яті списків) з передачею списку та інших необхідних даних за допомогою параметра(ів).

Завдання за варіантом 20:

Варіант 20

Ключами елементів списку є цілі числа. Кількість елементів списку повинна дорівнювати ***2n***. Перекомпонувати елементи списку так, розташування елементів було наступним:

$$a_1, a_{n+1}, a_2, a_{n+2}, a_3, \dots, a_n, a_{2n},$$

де a_i – ***i***-й компонент файлу.

При необхідності дозволяється використати ще один список, інші структури даних, крім простих змінних, використовувати не дозволяється.

Код програми

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

typedef struct doublyLinkedList
{
    float info;
    struct doublyLinkedList *next;
    struct doublyLinkedList *previous;
} LinkedList;

LinkedList *ListCreation(int SIZE)
{
    printf("\n-- Doubly Linked List Create with 2*n elements  --
\n");

    LinkedList *Beg2L = NULL;
    LinkedList *End2L = NULL;
    LinkedList *P;

    P = (LinkedList*)malloc(sizeof(LinkedList));

    P->next = NULL;
    P->info = 1;
    P->previous = NULL;

    Beg2L = P;
    End2L = P;

    for(int i = 2; i <= SIZE; i++)
    {
         P = (LinkedList*)malloc(sizeof(LinkedList));
         P->next = NULL;
         P->info = i;
         P->previous = End2L;

         End2L->next = P;
         End2L = P;
    }

    return Beg2L;
}

void printList(LinkedList *Beg2L)
{
    if(!Beg2L) return;

    LinkedList *current = Beg2L;
    while(current)
    {
         double info = current->info;
         LinkedList *next = current->next;
         LinkedList *previous = current->previous;
```

```
        printf("info = %lf pointer = %p previous = %p next =
%p\n", info, current, previous, next);

        current = next;
    }
}

void listModification(LinkedList *Beg2L)
{
    float tempInfo;
    int SIZE = 0;

    //count of elements
    LinkedList *current = Beg2L;
    while(current != NULL)
    {
        current = current->next;
        SIZE++;
    }

    printf("count of el = %d", SIZE);

    //look for a mid of list + 1
    LinkedList *mid = Beg2L;
    for(int i = 0; i < SIZE/2; i++)
    {
        mid = mid->next;
    }

    int i, j;
    current = Beg2L;
    for (i = 0; i < SIZE / 2; i++) {
        tempInfo = mid->info;

        LinkedList *moving = mid;
        for (int j = i + SIZE / 2; j > i * 2 + 1; j--) {
            moving->info = moving->previous->info;
            moving = moving->previous;
        }

        current->next->info = tempInfo;
        current = current->next->next;

        mid = mid->next;
    }
}

void ListRemoving(LinkedList *Beg2L)
{
    while (Beg2L != NULL) {
        LinkedList *temp = Beg2L;
        Beg2L = Beg2L->next;
```

```
        free(temp);
    }
}

int main(void){
    int n;

    do{
         printf("\nEnter n > 0 -->");
         scanf("%d", &n);
    }
    while(n <= 0);

    LinkedList *Beg2L = ListCreation(2*n);

    printf("\nYour Doubly Linked List before modification\n");
    printList(Beg2L);

    listModification(Beg2L);

    printf("\nYour Doubly Linked List after modification\n");
    printList(Beg2L);

    ListRemoving(Beg2L);

    return 0;
}
```

Тестування

```
Enter n > 0 -->0

Enter n > 0 -->1

-- Doubly Linked List Create with 2*n elements  --

Your Doubly Linked List before modification
info = 1.000000 pointer = 0x5af92bccaac0 previous = (nil) next = 0x5af92bccaae0
info = 2.000000 pointer = 0x5af92bccaae0 previous = 0x5af92bccaac0 next = (nil)
count of el = 2
Your Doubly Linked List after modification
info = 1.000000 pointer = 0x5af92bccaac0 previous = (nil) next = 0x5af92bccaae0
info = 2.000000 pointer = 0x5af92bccaae0 previous = 0x5af92bccaac0 next = (nil)
```

```
Enter n > 0 -->3

-- Doubly Linked List Create with 2*n elements  --

Your Doubly Linked List before modification
info = 1.000000 pointer = 0x592ef5981ac0 previous = (nil) next = 0x592ef5981ae0
info = 2.000000 pointer = 0x592ef5981ae0 previous = 0x592ef5981ac0 next = 0x592ef5981b00
info = 3.000000 pointer = 0x592ef5981b00 previous = 0x592ef5981ae0 next = 0x592ef5981b20
info = 4.000000 pointer = 0x592ef5981b20 previous = 0x592ef5981b00 next = 0x592ef5981b40
info = 5.000000 pointer = 0x592ef5981b40 previous = 0x592ef5981b20 next = 0x592ef5981b60
info = 6.000000 pointer = 0x592ef5981b60 previous = 0x592ef5981b40 next = (nil)
count of el = 6
Your Doubly Linked List after modification
info = 1.000000 pointer = 0x592ef5981ac0 previous = (nil) next = 0x592ef5981ae0
info = 4.000000 pointer = 0x592ef5981ae0 previous = 0x592ef5981ac0 next = 0x592ef5981b00
info = 2.000000 pointer = 0x592ef5981b00 previous = 0x592ef5981ae0 next = 0x592ef5981b20
info = 5.000000 pointer = 0x592ef5981b20 previous = 0x592ef5981b00 next = 0x592ef5981b40
info = 3.000000 pointer = 0x592ef5981b40 previous = 0x592ef5981b20 next = 0x592ef5981b60
info = 6.000000 pointer = 0x592ef5981b60 previous = 0x592ef5981b40 next = (nil)
```

```
count of el = 24
Your Doubly Linked List after modification
info = 1.000000 pointer = 0x5c90c3dd8ac0 previous = (nil) next = 0x5c90c3dd8ae0
info = 13.000000 pointer = 0x5c90c3dd8ae0 previous = 0x5c90c3dd8ac0 next = 0x5c90c3dd8b00
info = 2.000000 pointer = 0x5c90c3dd8b00 previous = 0x5c90c3dd8ae0 next = 0x5c90c3dd8b20
info = 14.000000 pointer = 0x5c90c3dd8b20 previous = 0x5c90c3dd8b00 next = 0x5c90c3dd8b40
info = 3.000000 pointer = 0x5c90c3dd8b40 previous = 0x5c90c3dd8b20 next = 0x5c90c3dd8b60
info = 15.000000 pointer = 0x5c90c3dd8b60 previous = 0x5c90c3dd8b40 next = 0x5c90c3dd8b80
info = 4.000000 pointer = 0x5c90c3dd8b80 previous = 0x5c90c3dd8b60 next = 0x5c90c3dd8ba0
info = 16.000000 pointer = 0x5c90c3dd8ba0 previous = 0x5c90c3dd8b80 next = 0x5c90c3dd8bc0
info = 5.000000 pointer = 0x5c90c3dd8bc0 previous = 0x5c90c3dd8ba0 next = 0x5c90c3dd8be0
info = 17.000000 pointer = 0x5c90c3dd8be0 previous = 0x5c90c3dd8bc0 next = 0x5c90c3dd8c00
info = 6.000000 pointer = 0x5c90c3dd8c00 previous = 0x5c90c3dd8be0 next = 0x5c90c3dd8c20
info = 18.000000 pointer = 0x5c90c3dd8c20 previous = 0x5c90c3dd8c00 next = 0x5c90c3dd8c40
info = 7.000000 pointer = 0x5c90c3dd8c40 previous = 0x5c90c3dd8c20 next = 0x5c90c3dd8c60
info = 19.000000 pointer = 0x5c90c3dd8c60 previous = 0x5c90c3dd8c40 next = 0x5c90c3dd8c80
info = 8.000000 pointer = 0x5c90c3dd8c80 previous = 0x5c90c3dd8c60 next = 0x5c90c3dd8ca0
info = 20.000000 pointer = 0x5c90c3dd8ca0 previous = 0x5c90c3dd8c80 next = 0x5c90c3dd8cc0
info = 9.000000 pointer = 0x5c90c3dd8cc0 previous = 0x5c90c3dd8ca0 next = 0x5c90c3dd8ce0
info = 21.000000 pointer = 0x5c90c3dd8ce0 previous = 0x5c90c3dd8cc0 next = 0x5c90c3dd8d00
info = 10.000000 pointer = 0x5c90c3dd8d00 previous = 0x5c90c3dd8ce0 next = 0x5c90c3dd8d20
info = 22.000000 pointer = 0x5c90c3dd8d20 previous = 0x5c90c3dd8d00 next = 0x5c90c3dd8d40
info = 11.000000 pointer = 0x5c90c3dd8d40 previous = 0x5c90c3dd8d20 next = 0x5c90c3dd8d60
info = 23.000000 pointer = 0x5c90c3dd8d60 previous = 0x5c90c3dd8d40 next = 0x5c90c3dd8d80
info = 12.000000 pointer = 0x5c90c3dd8d80 previous = 0x5c90c3dd8d60 next = 0x5c90c3dd8da0
info = 24.000000 pointer = 0x5c90c3dd8da0 previous = 0x5c90c3dd8d80 next = (nil)
```