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

студента групи ІМ-42
Ватолкін Михайло Андрійович
номер у списку групи: 8

Перевірила:

Сергієнко А. М.

Київ 2024

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 8:

Ключами елементів списку є цілі ненульові числа, причому кількість від'ємних чисел дорівнює кількості додатних. Перекомпонувати список так, щоб отримати послідовність чисел із чергуванням знаків, не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

Текст програми:

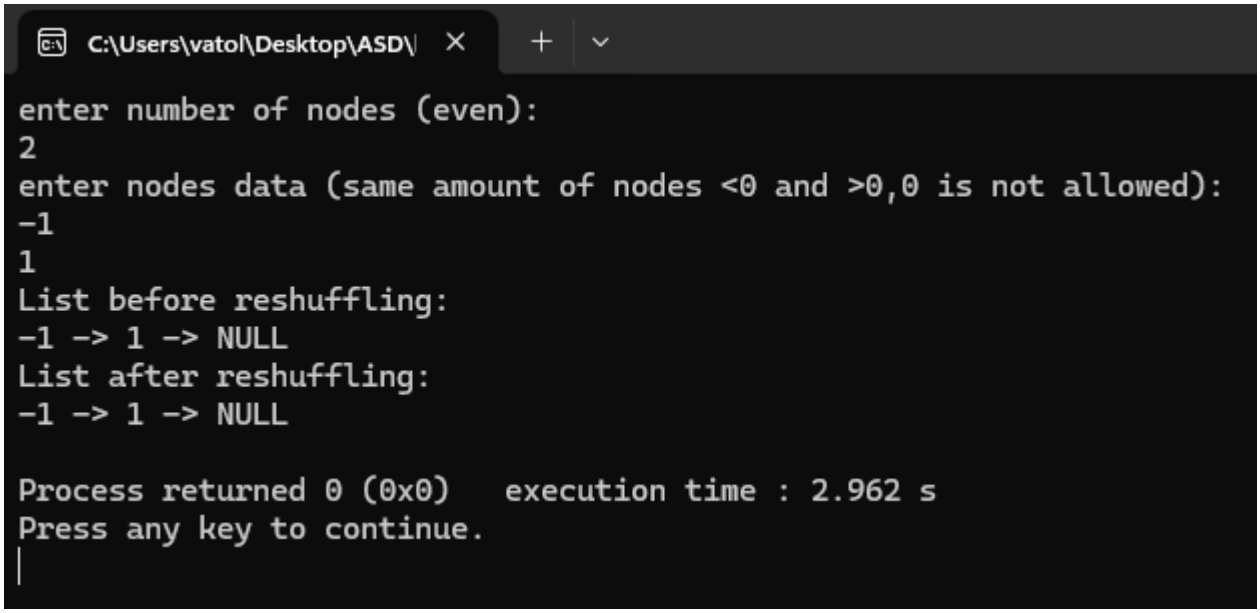
```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node *next;
};

void add(struct Node **head_ref, int new_data) {
   struct Node *new_node = (struct Node*)malloc(sizeof(struct Node));

   new_node->data = new_data;
   new_node->next = NULL;

   if (*head_ref == NULL) {
      *head_ref = new_node;
   }
   else{
   struct Node *last = *head_ref;

   while (last->next != NULL)
      last = last->next;

   last->next = new_node;}
}

void freeList(struct Node *head) {
   struct Node *temp;
```

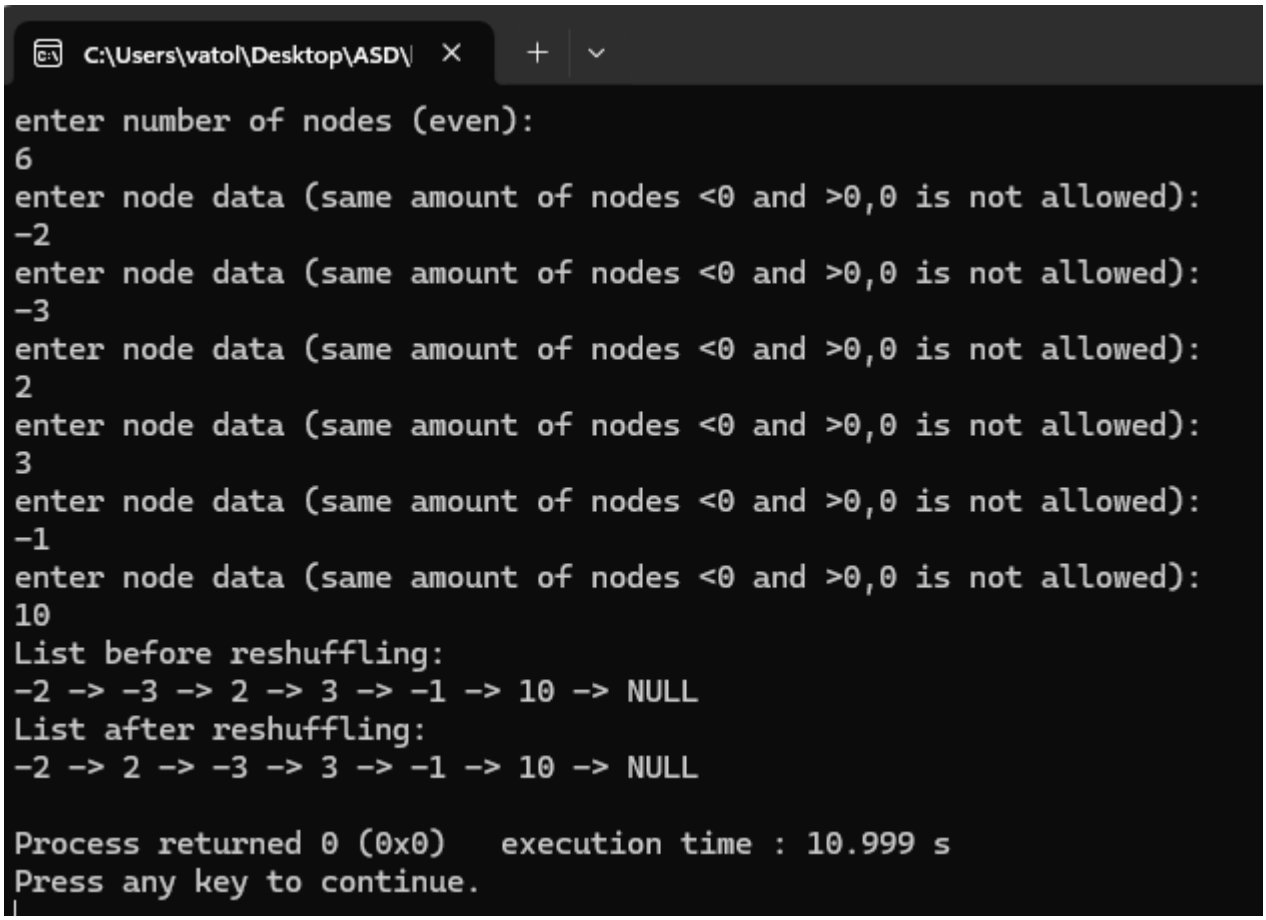
```
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int sameSign(int a, int b) {
    return (a >= 0 && b >= 0) || (a < 0 && b < 0);
}

void reshuffle(struct Node **head_ref) {

    struct Node *prev = *head_ref;
    struct Node *curr = (*head_ref)->next;

    while (curr != NULL) {
        if (sameSign(prev->data, curr->data)) {
            struct Node *temp = curr->next;
            struct Node *temp_prev = curr;

            while (temp != NULL && sameSign(prev->data, temp->data)) {
                temp_prev = temp;
                temp = temp->next;
            }

            if (temp == NULL) break;

            temp_prev->next = temp->next;
```

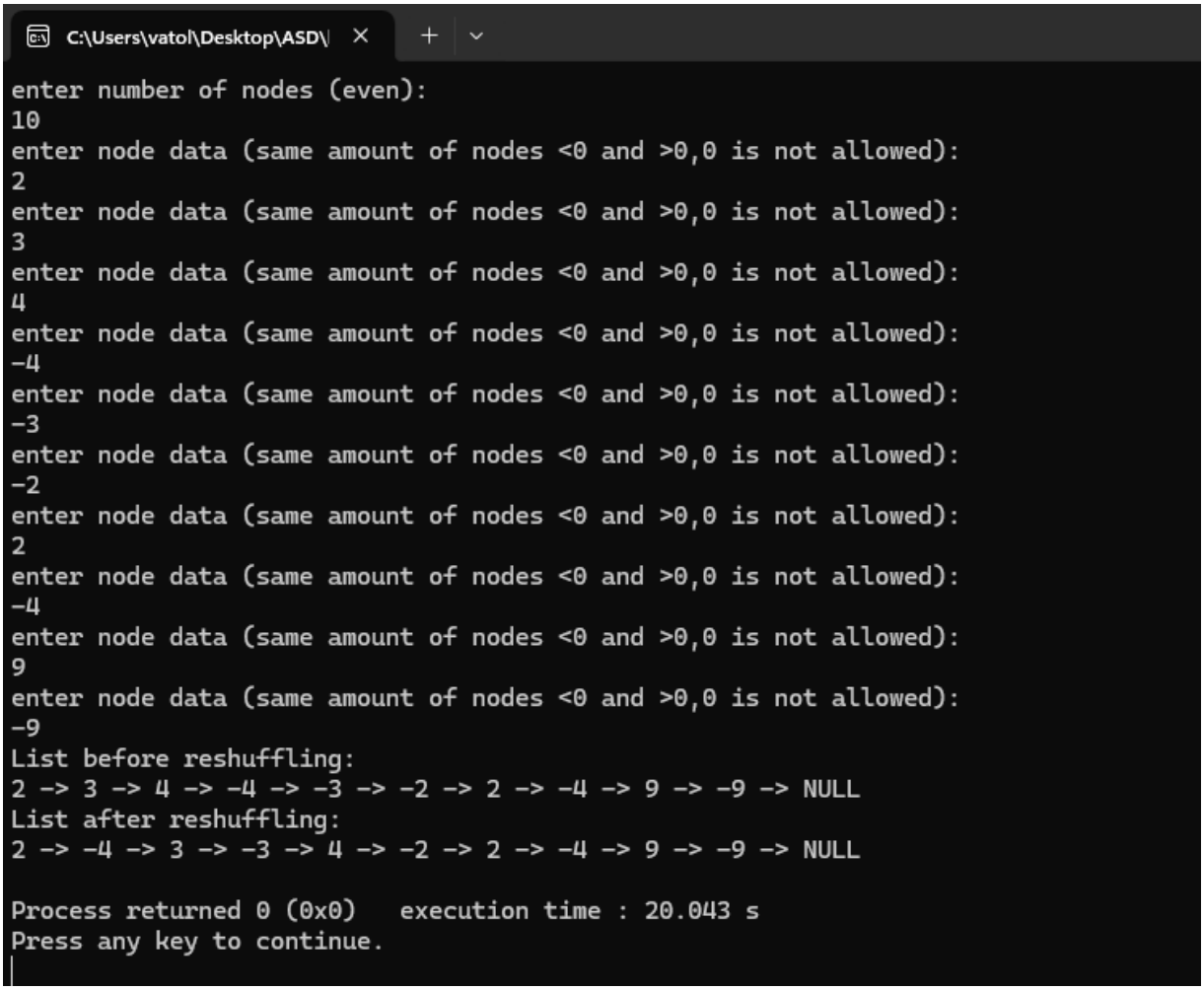
```
            temp->next = curr;
            prev->next = temp;

            prev = temp;
        } else {
            prev = curr;
        }

        curr = prev->next;
    }
}

void printList(struct Node *node) {
    while (node != NULL) {
        printf("%d -> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}

int main() {
    unsigned int n,i;
    int num;
    srand(time(NULL));

    printf("enter number of nodes (even):\n");
    scanf("%u", &n);

    if (n % 2 != 0) {
```

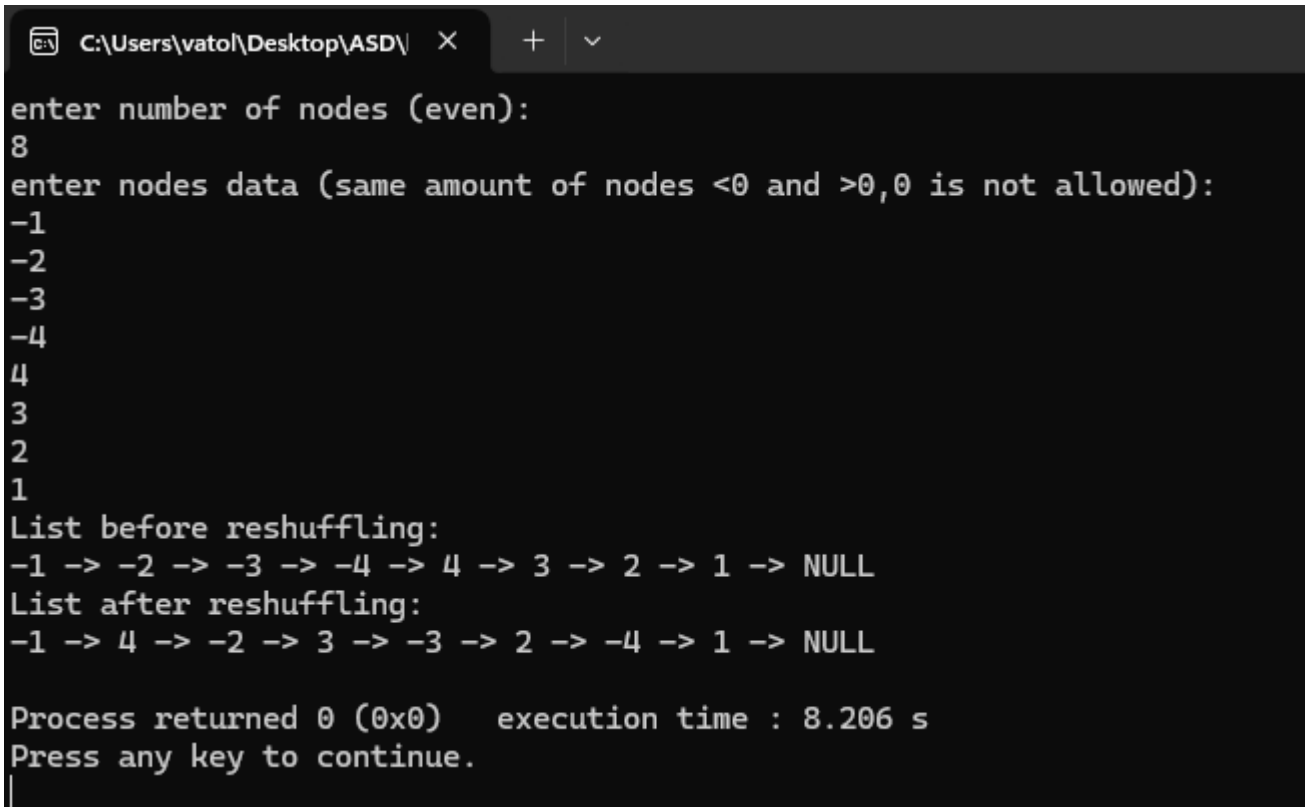
```
        printf("The number of nodes must be even!\n");
        main();
        return;
    }

    struct Node *head = NULL;
    printf("enter nodes data (same amount of nodes <0 and >0,0 is not
allowed):\n");
    for(i=1;i<=n;i++){
    scanf("%d", &num);
    add(&head,num);

    }

    printf("List before reshuffling:\n");
    printList(head);

    reshuffle(&head);

    printf("List after reshuffling:\n");
    printList(head);

    freeList(head);

    return 0;
}
```

Тестування програми :






C:\Users\vatol\Desktop\ASD\
enter number of nodes (even):
10
enter node data (same amount of nodes <0 and >0,0 is not allowed):
2
enter node data (same amount of nodes <0 and >0,0 is not allowed):
3
enter node data (same amount of nodes <0 and >0,0 is not allowed):
4
enter node data (same amount of nodes <0 and >0,0 is not allowed):
-4
enter node data (same amount of nodes <0 and >0,0 is not allowed):
-3
enter node data (same amount of nodes <0 and >0,0 is not allowed):
-2
enter node data (same amount of nodes <0 and >0,0 is not allowed):
2
enter node data (same amount of nodes <0 and >0,0 is not allowed):
-4
enter node data (same amount of nodes <0 and >0,0 is not allowed):
9
enter node data (same amount of nodes <0 and >0,0 is not allowed):
-9
List before reshuffling:
2 -> 3 -> 4 -> -4 -> -3 -> -2 -> 2 -> -4 -> 9 -> -9 -> NULL
List after reshuffling:
2 -> -4 -> 3 -> -3 -> 4 -> -2 -> 2 -> -4 -> 9 -> -9 -> NULL
Process returned 0 (0x0) execution time : 20.043 s
Press any key to continue.


C:\Users\vatol\Desktop\ASD\
enter number of nodes (even):
8
enter nodes data (same amount of nodes <0 and >0,0 is not allowed):
-1
-2
-3
-4
4
3
2
1
List before reshuffling:
-1 -> -2 -> -3 -> -4 -> 4 -> 3 -> 2 -> 1 -> NULL
List after reshuffling:
-1 -> 4 -> -2 -> 3 -> -3 -> 2 -> -4 -> 1 -> NULL
Process returned 0 (0x0) execution time : 8.206 s
Press any key to continue.

Висновок:

В ході виконання лабораторної роботи номер 2 було засвоєно теоретичний матеріал та набуті навички практичного досвіду використання зв’язаних динамічних структур даних у вигляді одно- та двозв’язнихсписків при складанні різних алгоритмів. Варто зазначити, що мова програмування С дозволяє точніше контролювати роботу з динамічними структурами даних, ніж більш сучасні мови програмування, завдяки своїй низькорівневості. Це зіграло свою роль у процесі створення відповідної програми.