**Міністерство освіти і науки України  
 Національний технічний університет України  
 «Київський політехнічний інститут імені Ігоря Сікорського»  
 Факультет інформатики та обчислювальної техніки  
 Кафедра обчислювальної техніки**

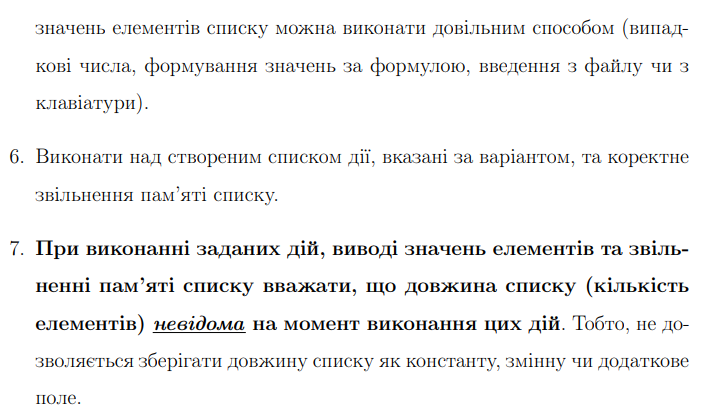
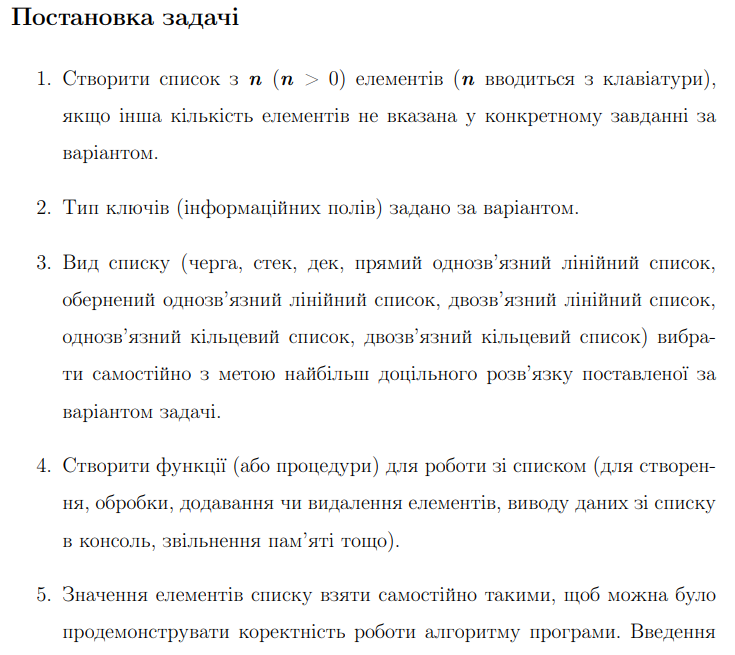
**Лабораторна робота №2**

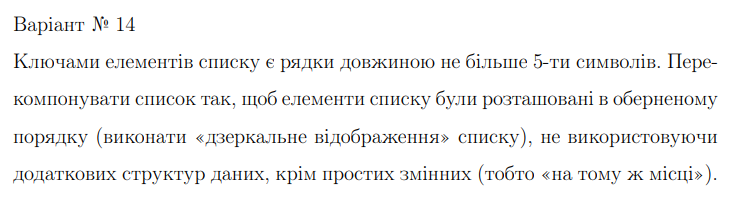
з дисципліни  
 «Алгоритми і структури даних»

Виконав: Перевірила:

студент групи ІМ-32 Молчанова А. А.  
Король Олександр Володимирович  
номер у списку групи: 14

Київ 2024





**Текст програми**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX\_LEN 6**

**typedef struct Node {**

**char data[MAX\_LEN];**

**struct Node\* next;**

**} Node; // прямий лінійний однозв'язний список**

**// Функція для створення нового вузла**

**Node\* createNode(char data[MAX\_LEN]) {**

**Node\* newNode = (Node\*)malloc(sizeof(Node));**

**strcpy(newNode->data, data);**

**newNode->next = NULL;**

**return newNode;**

**}**

**// Функція для вставки нового вузла у початок списку**

**Node\* insertNode(Node\* currentNode, char data[MAX\_LEN]) {**

**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("%s ", 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;**

**Node\* prev = NULL;**

**Node\* current = head;**

**Node\* next = NULL;**

**while (current) {**

**next = current->next; // зберігаємо наступний елемент**

**current->next = prev; // змінюємо посилання (reverse)**

**prev = current;**

**current = next; // рухаємось до наступного елементу**

**}**

**\*headRef = prev; // повертаємо посилання на перший елемент**

**}**

**int main() {**

**char data[MAX\_LEN];**

**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 (hint: it must be a string with max 5 symbols):");**

**scanf("%s", data);**

**while (strlen(data) > 5) {**

**printf("max length is exceeded, please, enter a valid value\n");**

**printf("Enter a value (hint: it must be a string with max 5 symbols):");**

**scanf("%s", 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);**

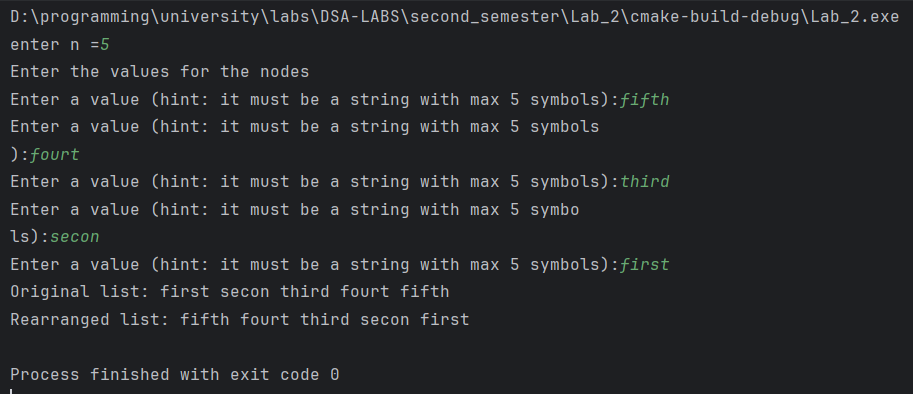
**// Звільняємо пам'ять, вивільняємо список**

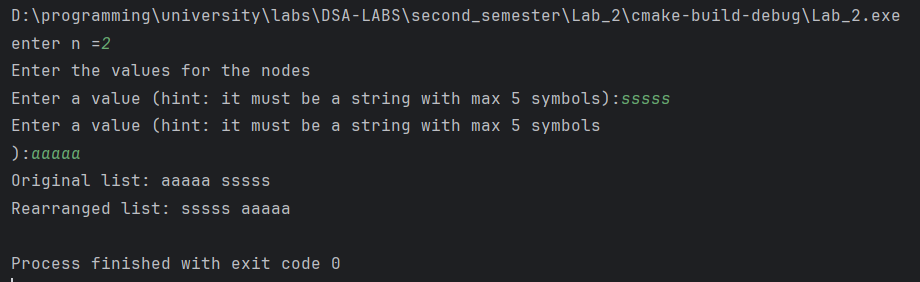
**freeList(head);**

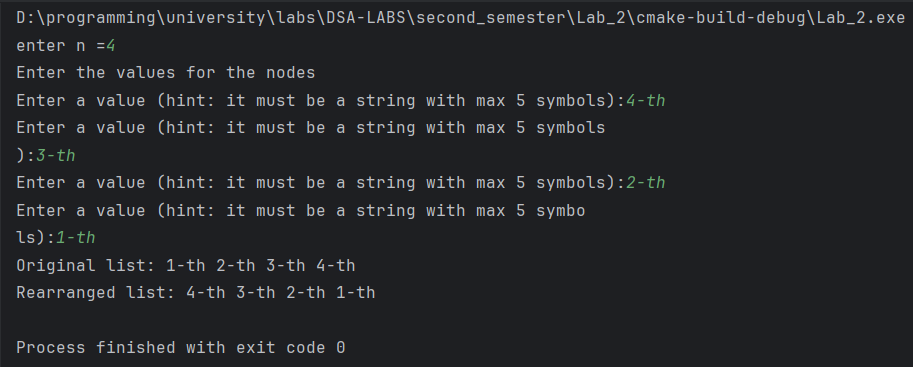
**return 0;**

**}**

**Результати тестування програми**







**Висновки**  
Під час виконання цієї лабораторної роботи я отримав цінний практичний досвід зі створення динамічних структур даних, зокрема однозв’язних лінійних списків. Змінюючи посилання наступного елемента на минулий, я зміг ефективно розвернути список, що виявилося досить зручним і простим підходом.

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.