Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота № 4**

**«Лініні однозв’язані і двозв’язані списки»**

Виконав

ст. гр. ІП-22-1

Микитій Ю.М.

Перевірив

доцент Процюк В.Р.

Івано-Франківськ

2023

**Мета:** отримати навички в реалізації і обробці однозв’язаних і двозв'язний списків, а також навчитися їх використовувати при вирішені завдань

**Умова: Варіант 15.**

Для кожного варіанта - реалізувати однозв'язний і двозв'язний список i

операції роботи з ними. Непарні варіанти зберігають в списку цiлi значення. Списки заповнити з клавіатури. Операції:

- додати на початок:

- додати в кінець:

- додати в середину (після зазначеного за значенням елемента);

- видалити (з будь-якого мiсця списку);

- знайти за значенням:

- роздрукувати список.

Програма має бути написана на С++

Використовуючи два створені списки, виконати два завдання за варіантом.

1)Створити новий(третій) список однозв'язний список. Помістити всі парні елементи з перших двох. Результуючий список вивести на екран.

7)Створити новий(третій) список однозв'язний список. Помістити всі елементи з перших двох, які менше заданого числа. Результуючий список вивести на екран.

**Розв’язок**

#include <iostream>

#include <locale>

using namespace std;

// Структура для представлення елементу списку

struct Node {

int data;

Node\* next;

};

// Функція для додавання елементу в початок списку

Node\* addToBeginning(Node\* head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = head;

return newNode;

}

// Функція для додавання елементу в кінець списку

Node\* addToEnd(Node\* head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

return head;

}

// Функція для додавання елементу в середину списку

Node\* insertAfterValue(Node\* head, int valueToInsertAfter, int value) {

Node\* newNode = new Node;

newNode->data = value;

Node\* current = head;

while (current != nullptr && current->data != valueToInsertAfter) {

current = current->next;

}

if (current == nullptr) {

cout << "Елемент зі значенням " << valueToInsertAfter << " не знайдено в списку." << endl;

return head;

}

newNode->next = current->next;

current->next = newNode;

return head;

}

// Функція для видалення елементу зі списку

Node\* deleteNode(Node\* head, int valueToDelete) {

if (head == nullptr) {

cout << "Список порожній." << endl;

return head;

}

if (head->data == valueToDelete) {

Node\* newHead = head->next;

delete head;

return newHead;

}

Node\* current = head;

while (current->next != nullptr && current->next->data != valueToDelete) {

current = current->next;

}

if (current->next == nullptr) {

cout << "Елемент зі значенням " << valueToDelete << " не знайдено в списку." << endl;

return head;

}

Node\* nodeToDelete = current->next;

current->next = nodeToDelete->next;

delete nodeToDelete;

return head;

}

// Функція для пошуку елементу за значенням

Node\* findNode(Node\* head, int value) {

Node\* current = head;

while (current != nullptr) {

if (current->data == value) {

return current;

}

current = current->next;

}

return nullptr;

}

// Функція для роздрукування списку

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функція для створення нового списку, що містить парні елементи з двох вхідних списків

Node\* createMergedEvenList(Node\* list1, Node\* list2) {

Node\* mergedList = nullptr;

Node\* current1 = list1;

Node\* current2 = list2;

while (current1 != nullptr || current2 != nullptr) {

if (current1 != nullptr && current1->data % 2 == 0) {

mergedList = addToEnd(mergedList, current1->data);

}

if (current2 != nullptr && current2->data % 2 == 0) {

mergedList = addToEnd(mergedList, current2->data);

}

if (current1 != nullptr) {

current1 = current1->next;

}

if (current2 != nullptr) {

current2 = current2->next;

}

}

return mergedList;

}

// Функція для створення нового списку, що містить елементи менші за задане число з двох вхідних списків

Node\* createMergedListLessThanValue(Node\* list1, Node\* list2, int value) {

Node\* mergedList = nullptr;

Node\* current1 = list1;

Node\* current2 = list2;

while (current1 != nullptr || current2 != nullptr) {

if (current1 != nullptr && current1->data < value) {

mergedList = addToEnd(mergedList, current1->data);

}

if (current2 != nullptr && current2->data < value) {

mergedList = addToEnd(mergedList, current2->data);

}

if (current1 != nullptr) {

current1 = current1->next;

}

if (current2 != nullptr) {

current2 = current2->next;

}

}

return mergedList;

}

int main() {

setlocale(LC\_CTYPE, "ukr");

Node\* list1 = nullptr;

Node\* list2 = nullptr;

// Заповнюємо перший список

int n1;

cout << "Введiть кiлькiсть елементiв першого списку: ";

cin >> n1;

cout << "Введiть елементи першого списку: ";

for (int i = 0; i < n1; i++) {

int value;

cin >> value;

list1 = addToEnd(list1, value);

}

// Заповнюємо другий список

int n2;

cout << "Введiть кiлькiсть елементiв другого списку: ";

cin >> n2;

cout << "Введiть елементи другого списку: ";

for (int i = 0; i < n2; i++) {

int value;

cin >> value;

list2 = addToEnd(list2, value);

}

cout << "Перший список: ";

printList(list1);

cout << "Другий список: ";

printList(list2);

// Завдання 1: Створити новий список з парними елементами

Node\* evenList = createMergedEvenList(list1, list2);

cout << "Список з парними елементами: ";

printList(evenList);

// Завдання 7: Створити новий список з елементами меншими за задане число

int valueToCompare;

cout << "Введiть число для порiвняння: ";

cin >> valueToCompare;

Node\* mergedListLessThanValue = createMergedListLessThanValue(list1, list2, valueToCompare);

cout << "Список з елементами меншими за " << valueToCompare << ": ";

printList(mergedListLessThanValue);

// Звільнення пам'яті

while (list1 != nullptr) {

Node\* temp = list1;

list1 = list1->next;

delete temp;

}

while (list2 != nullptr) {

Node\* temp = list2;

list2 = list2->next;

delete temp;

}

while (evenList != nullptr) {

Node\* temp = evenList;

evenList = evenList->next;

delete temp;

}

while (mergedListLessThanValue != nullptr) {

Node\* temp = mergedListLessThanValue;

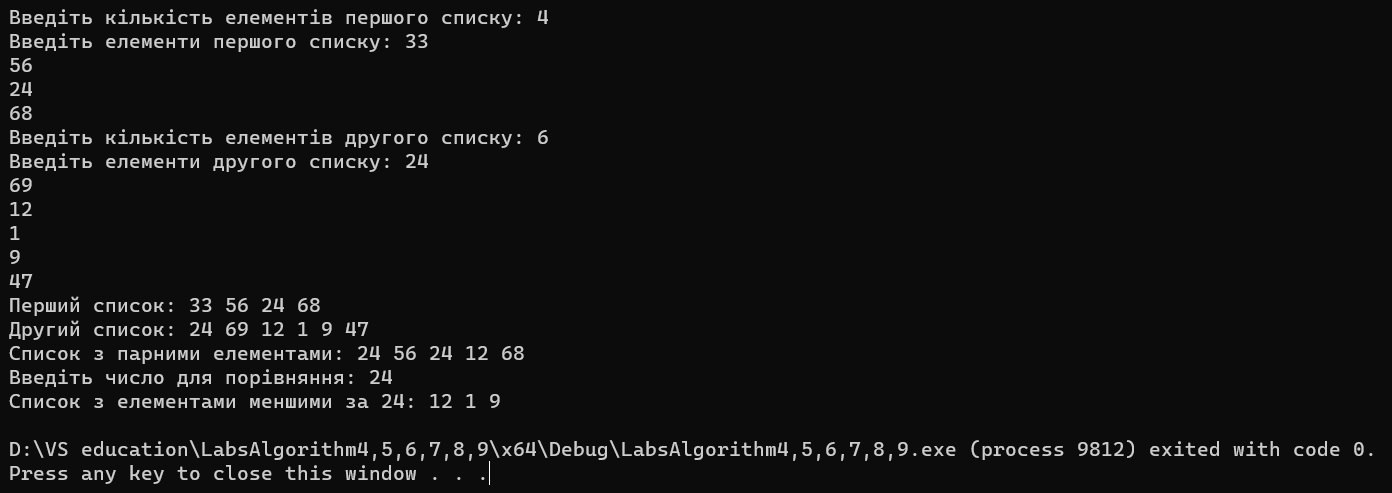
mergedListLessThanValue = mergedListLessThanValue->next;

delete temp;

}

return 0;

}

**Результат:  
**

**Висновок:** на цій лабораторній роботі я отримав навички в реалізації і обробці однозв’язаних і двозв'язний списків, а також навчився їх використовувати при вирішені завдань