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

Курчій С.В.

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 12.**

Лінійний список – це динамічна структура даних, кожний елемент якої за допомогою вказівника зв’язується з наступним елементом. Які використовуються для зберігання та організації даних

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.

2) Створити новий(третій) однозв’язний список. Помістити в нього перші три елемента з першого і другого списків. Результуючий список вивести на екран.

12) Створити новий (третій) однозв’язний список. Помістити в нього всі символи нижнього регістра з перших двох. Результуючий список вивести на екран

**Розв’язок**

#include <iostream>

// Визначення структури Node для однозв'язного списку, використовуємо зміну покажчик,яку будемо переміщати по списку

struct Node

{

    int data;

    Node \*next;

    Node(int val) : data(val), next(nullptr) {}

};

// Функція для додавання нового вузла з заданою значенням в кінець списку

void append(Node \*&head, int val)

{

    Node \*newNode = new Node(val);

    if (!head)

    {

        head = newNode;

    }

    else

    {

        Node \*current = head;

        while (current->next)

        {

            current = current->next;

        }

        current->next = newNode;

    }

}

// Функція для додавання нового вузла з заданою значенням на початок списку

void prepend(Node \*&head, int val)

{

    Node \*newNode = new Node(val);

    newNode->next = head;

    head = newNode;

}

// Функція для вставки нового вузла з заданою значенням після певного вузла

void insertAfter(Node \*node, int val)

{

    if (node == nullptr)

    {

        std::cout << "Error: Cannot insert after null pointer." << std::endl;

        return;

    }

    Node \*newNode = new Node(val);

    newNode->next = node->next;

    node->next = newNode;

}

// Функція для видалення першого вузла з заданим значенням із списку

void deleteNode(Node \*&head, int val)

{

    if (head == nullptr)

    {

        return;

    }

    if (head->data == val)

    {

        Node \*temp = head;

        head = head->next;

        delete temp;

        return;

    }

    Node \*current = head;

    while (current->next)

    {

        if (current->next->data == val)

        {

            Node \*temp = current->next;

            current->next = current->next->next;

            delete temp;

            return;

        }

        current = current->next;

    }

}

// Функція для пошуку вузла із заданим значенням у списку

Node \*search(Node \*head, int val)

{

    Node \*current = head; //Починаємо з голови списку

    while (current)

    {

        if (current->data == val)

        {

            return current;

        }

        current = current->next;

    }

    return nullptr;

}

// Функція для виведення елементів списку

void printList(Node \*head)

{

    Node \*current = head;

    while (current)

    {

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

        current = current->next;

    }

    std::cout << std::endl;

}

int main()

{

    // Ініціалізація трьох однозв'язних списків: list1, list2 та result

    Node \*list1 = nullptr;

    Node \*list2 = nullptr;

    std::cout << "Enter 4 items for the first list:" << std::endl;

    for (int i = 0; i < 4; i++)

    {

        int val;

        std::cin >> val;

        append(list1, val);

    }

    std::cout << "Enter 4 items for the second list:" << std::endl;

    for (int i = 0; i < 4; i++)

    {

        int val;

        std::cin >> val;

        append(list2, val);

    }

    Node \*result = nullptr;

    Node \*current1 = list1;

    Node \*current2 = list2;

    // Копіювати перші 3 елементи з першого списку в результат

    for (int i = 0; i < 3; i++)

    {

        append(result, current1->data);

        current1 = current1->next;

    }

    // Копіювати перші 3 елементи з другого списку в результат

    for (int i = 0; i < 3; i++)

    {

        append(result, current2->data);

        current2 = current2->next;

    }

    std::cout << "List results:" << std::endl;

    printList(result);

    // Звільнити виділену пам'ять, видаливши всі три списки

    while (list1)

    {

        Node \*temp = list1;

        list1 = list1->next;

        delete temp;

    }

    while (list2)

    {

        Node \*temp = list2;

        list2 = list2->next;

        delete temp;

    }

    while (result)

    {

        Node \*temp = result;

        result = result->next;

        delete temp;

    }

    return 0;

}

**2 завдання:**

#include <iostream>

struct Node

{

char data;

Node \*prev; // вказівник на попередній елемент

Node \*next; // вказівник на наступний елемент

Node(char val) : data(val), prev(nullptr), next(nullptr) {}

};

void append(Node \*&head, char val)

{

Node \*newNode = new Node(val);

if (!head)

{

head = newNode;

}

else

{

Node \*current = head;

while (current->next)

{

current = current->next;

}

current->next = newNode;

newNode->prev = current;

}

}

void printList(Node \*head)

{

Node \*current = head;

while (current)

{

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

current = current->next;

}

std::cout << std::endl;

}

int main()

{

Node \*list1 = nullptr;

Node \*list2 = nullptr;

std::cout << "Enter the characters for the first list (enter 0 to complete the entry):" << std::endl;

char input1;

while (std::cin >> input1 && input1 != '0')

{

if (islower(input1))

{

append(list1, input1);

}

}

std::cout << "Enter the characters for the second list (enter 0 to complete the entry):" << std::endl;

char input2;

while (std::cin >> input2 && input2 != '0')

{

if (islower(input2))

{

append(list2, input2);

}

}

Node \*result = nullptr;

Node \*current1 = list1;

Node \*current2 = list2;

// Додаємо символи нижнього регістру з перших двох списків до результату

while (current1)

{

append(result, current1->data);

current1 = current1->next;

}

while (current2)

{

append(result, current2->data);

current2 = current2->next;

}

std::cout << "The resulting list:" << std::endl;

printList(result);

// Очищення пам'яті

while (list1)

{

Node \*temp = list1;

list1 = list1->next;

delete temp;

}

while (list2)

{

Node \*temp = list2;

list2 = list2->next;

delete temp;

}

while (result)

{

Node \*temp = result;

result = result->next;

delete temp;

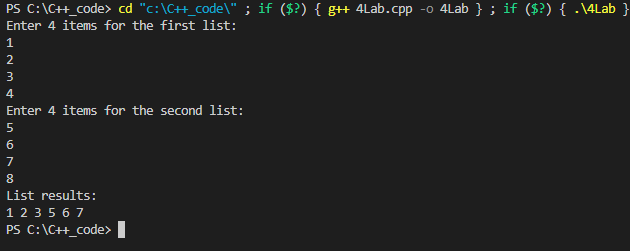
}

return 0;

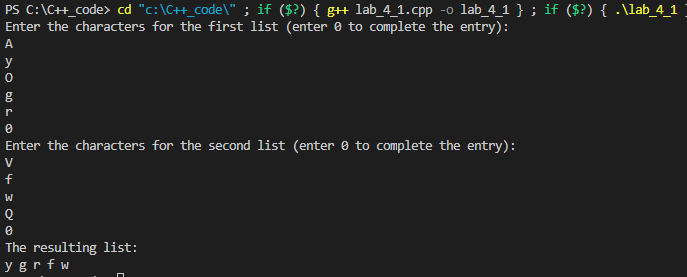
}

**Результат:**

1)



2)



**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