Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

**про виконання лабораторних та практичних робіт блоку № 6**

На тему: «*Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур*.»

***з дисципліни:*** «Мови та парадигми програмування»

до:

Практичних Робіт № 6

***Виконав:***

студент групи ШІ-14

Школяр Артур Тарасович

# **Тема роботи:**

Ознайомлення з динамічними структурами та їхніми алгоритмами обробки. Практичне застосування знань на основі виконання лабораторних та практичних робіт.

# **Мета роботи:**

* Theory Education Activities
* Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)
* Lab# programming: VNS Lab 10
* Lab# programming: Algotester Lab 5
* Lab# programming: Algotester Lab 7-8
* Practice# programming: Class Practice Task
* Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* Results Evaluation and Release

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Списки.
* Тема №2: Дерева
* Тема №3: Стек
* Тема №4: Черга

1. Індивідуальний план опрацювання теорії:

* Тема №1: Списки.
  + Джерела Інформації
    - <https://www.youtube.com/watch?v=HKfj0l7ndbc&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=3&pp=iAQB>
    - <https://www.youtube.com/watch?v=RNMIDj62o_o&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=4&pp=iAQB>
    - <https://www.youtube.com/watch?v=-StYr9wILqo&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=8&pp=iAQB>
    - Chat GPT
  + Що опрацьовано:
    - Ознайомлено з теорією
    - Використано у практичних та лабораторних роботах
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Звершення опрацювання теми: 13.12.23
* Тема №2: Дерева
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=UHxtjVsOTHc>
    - <https://youtu.be/_IhTp8q0Mm0?si=Ry5wReFC1KeM3icl>
    - Chat GPT
  + Що опрацьовано:
    - Ознайомлено з теорією
    - Використано у практичних та лабораторних роботах
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Завершення опрацювання теми: 13.12.23
* Тема №3: Стек
  + Джерела Інформації:
    - [https://www.youtube.com/watch?v=GBST5uQ\_yos&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz\_&index=5&pp=iAQB](https://www.youtube.com/watch?v=GBST5uQ_yos&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=5&pp=iAQB%20)
    - Chat GPT
    - Інтернет-ресурси
  + Що опрацьовано:
    - Ознайомлено з теорією
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Завершення опрацювання теми: ---
* Тема №4 Черга
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=jaK4pn1jXTo&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=6&pp=iAQB>
    - Chat GPT
    - Інтернет-ресурси
  + Що опрацьовано:
    - Ознайомлений з теорією
  + Статус: Ознайомлений
  + Початок опрацювання теми: 13.12.23
  + Завершення опрацювання теми: ---

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"

* Варіант завдання: 20
* Деталі завдання: Записи в лінійному списку містять ключове поле типу \*char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати по К елементів на початок й в кінець списку.
* Важливі деталі для врахування в імплементації програми: Написати програму, у якій створюються динамічні структури й виконати їхню обробку.

Потрібно розробити такі функції:

1. Створення списку.

2. Додавання елемента в список (у відповідності зі своїм варіантом).

3. Знищення елемента зі списку (у відповідності зі своїм варіантом).

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Завдання №2 Лабораторна робота №5 у Algotester

* Варіант завдання: 3
* Деталі завдання: Дано карту гори розміром N × M. Також відомі координати {x, y} , у яких знаходиться вершина гори. Завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число. Клiтини, якi мають суміжну сторону з вершиною мають висоту на один меншу, сумiжнi з ними i не розфарбовані мають ще на 1 меншу висоту i так далi.
* Важливі деталі для врахування в імплементації програми:

Вхiднi данi:

У першому рядку 2 числа N та M - розмiри карти

у другому рядку 2 числа x та y - координати пiку гори

Вихiднi данi:

N рядкiв по M елементiв в рядку через пробiл - висоти карти.

Завдання №3 Лабораторна робота №7\_8 у Algotester

* Варіант завдання: 1
* Деталі завдання: Ваше завдання - власноруч реалiзувати структуру даних "Двозв’язний список". Ви отримаєте Q запитiв, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого йдуть його аргументи.

Вам будуть поступати запити такого типу:

• Вставка:

Iдентифiкатор - insert

Ви отримуєте цiле число index елемента, на мiсце якого робити вставку.

Пiсля цього в наступному рядку рядку написане число N - розмiр списку, який треба вста-

Вити. У третьому рядку N цiлих чисел - список, який треба вставити на позицiю index.

• Видалення:

Iдентифiкатор - erase

Ви отримуєте 2 цiлих числа - index, iндекс елемента, з якого почати видалення та n -

кiлькiсть елементiв, яку треба видалити.

• Визначення розмiру:

Iдентифiкатор - size

Ви не отримуєте аргументів. Ви виводите кiлькiсть елементiв у списку.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте цiле число - index, iндекс елемента. Ви виводите значення елемента за iндексом.

• Модифiкацiя значення i-го елементу

Iдентифiкатор – set.

Ви отримуєте 2 цiлих числа - iндекс елемента, який треба змінити, та його нове значення.

• Вивiд списку на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв. Ви виводите усi елементи списку через пробiл. Реалiзувати використовуючи перегрузку оператора <<

* Важливі деталі для врахування в імплементації програми:

Вхiднi данi:

Цiле число Q - кiлькiсть запитiв. У наступних рядках Q запитiв у зазначеному в умовi форматi.

Вихiднi данi:

Вiдповiдi на запити у зазначеному в умовi форматi.

Важлива примітка:

Гарантується, що усi данi коректнi. Виходу за межi списку або розмiр, бiльший нiж розмiр списку недопустимi. Iндекси починаються з нуля.

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"

* Деталі завдання: ***Реалізувати метод реверсу списку:*** Node\* reverse(Node \*head);

*Умови задачі:*

1. використовувати цілочисельні значення в списку;
2. реалізувати метод реверсу;
3. реалізувати допоміжний метод виведення вхідного і обернутого списків;

* Важливі деталі для врахування в імплементації програми: ---

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

* Деталі завдання: bool compare(Node \*h1, Node \*h2);

*Умови задачі:*

1. використовувати цілочисельні значення в списку;
2. реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;
3. якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

* Важливі деталі для врахування в імплементації програми: ---

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

* Деталі завдання: Node\* add(Node \*n1, Node \*n2);

*Умови задачі:*

1. використовувати цифри від 0 до 9 для значень у списку;
2. реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379  ⟹  9→7→3);
3. функція повертає новий список, передані в функцію списки не модифікуються.

* Важливі деталі для врахування в імплементації програми: ---

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

* Деталі завдання: TreeNode \*create\_mirror\_flip(TreeNode \*root);

*Умови задачі:*

1. використовувати цілі числа для значень у вузлах дерева
2. реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева
3. функція повертає нове дерево, передане в функцію дерево не модифікується

* Важливі деталі для врахування в імплементації програми: ---

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”

* Деталі завдання: void tree\_sum(TreeNode \*root);

*Умови задачі:*

1. використовувати цілочисельні значення у вузлах дерева;
2. реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів
3. вузол-листок не змінює значення
4. значення змінюються від листків до кореня дерева

* Важливі деталі для врахування в імплементації програми: ---

## **2. Дизайн та планувальна оцінка часу виконання завдань:**

Програма №1 Лабораторна робота №10 "Інформаційні динамічні структури"

* Запланований час на реалізацію: 4 год
* Важливі деталі для врахування в імплементації:

1. Написати функцію для друку списку. Функція повинна передбачати вивід

повідомлення, якщо список порожній.

2. Написати функції для знищення й додавання елементів списку у

відповідності зі своїм варіантом

3. Виконати зміни в списку й друк списку після кожної зміни.

4. Написати функцію для запису списку у файл.

5. Написати функцію для знищення списку.

6. Записати список у файл, знищити його й виконати друк (при друці повинне

бути видане повідомлення "Список порожній").

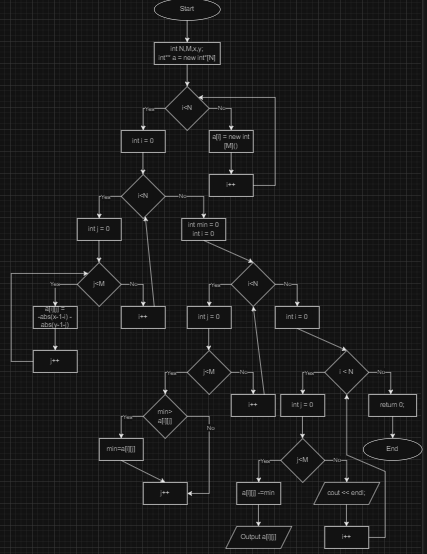
7. Написати функцію для відновлення списку з файлу.

8. Відновити список і роздрукувати його.

9.Знищити список.

Програма №2 Лабораторна робота №5 у Algotester

* Блок-схема:



* Планований час на реалізацію: 3год
* Важливі деталі для врахування в імплементації: обмеження у пам’яті, часі виконання та у значеннях змінних

Програма №3 Лабораторна робота №7\_8 у Algotester

* Запланований час на реалізацію: 5 год
* Важливі деталі для врахування в імплементації: Використовувати STL заборонено.

Програма №4 Практична робота №1 "Зв’язаний список: Реверс списку"

* Запланований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: Спочатку ми визначаємо просту структуру ***Node*** для нашого пов’язаного списку. Потім функція ***reverse*** ітеративно змінює список, маніпулюючи наступними покажчиками кожного вузла. ***printList*** — допоміжна функція для відображення списку. Основна функція створює зразок списку, демонструє реверсування та друкує вихідний і обернений списки.

Програма №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

* Запланований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації:
* Для пов’язаного списку визначено структуру ***Node***.
* Функція ***compare*** ітеративно проходить обидва списки одночасно, порівнюючи дані в кожному вузлі.
* Якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.
* Основна функція ***main*** створює два списки та демонструє порівняння.

Програма №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

* Запланований час на реалізацію: 2 год
* Важливі деталі для врахування в імплементації: ---

Програма №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

* Запланований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: ---

Програма №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму під вузлів”

* Запланований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: ---

## **3. Конфігурація середовища до виконання завдань:**

*Конфігурація з Епіку 1 не змінилась.*

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

struct Node

{

char data[100];

Node\* previous;

Node\* next;

};

Node\* createList()

{

return nullptr;

}

void addtolist(Node\*& head, const char\* data)

{

Node\* newNode = new Node;

strncpy(newNode->data, data, sizeof(newNode->data));

newNode->previous = nullptr;

newNode->next = nullptr;

if (head == nullptr)

{

head = newNode;

}

else

{

Node\* current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = newNode;

newNode->previous = current;

}

}

void printList(const Node\* head)

{

if (head == nullptr)

{

cout << "Список порожній(" << endl;

}

else

{

const Node\* current = head;

while (current != nullptr)

{

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

current = current->next;

}

cout << endl;

}

}

void earasenode(Node\*& head, const char\* key)

{

Node\* current = head;

while (current != nullptr)

{

if (strcmp(current->data, key) == 0)

{

if (current->previous != nullptr) {

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

}

else

{

head = current->next;

}

if (current->next != nullptr)

{

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

}

delete current;

return;

}

current = current->next;

}

}

void addition(Node\*& head, int K)

{

for (int i = 0; i < K; ++i)

{

char buffer[100];

sprintf(buffer, "Елемент %d", i);

Node\* newNode = new Node;

strncpy(newNode->data, buffer, sizeof(newNode->data));

newNode->previous = nullptr;

newNode->next = head;

if (head != nullptr)

{

head->previous = newNode;

}

head = newNode;

}

Node\* current = head;

while (current->next != nullptr)

{

current = current->next;

}

for (int i = K; i < 2 \* K; ++i)

{

char buffer[100];

sprintf(buffer, "Елемент %d", i);

Node\* newNode = new Node;

strncpy(newNode->data, buffer, sizeof(newNode->data));

newNode->previous = current;

newNode->next = nullptr;

current->next = newNode;

current = newNode;

}

}

void destroy(Node\*& head)

{

Node\* current = head;

while (current != nullptr)

{

Node\* nextNode = current->next;

delete current;

current = nextNode;

}

head = nullptr;

}

void writeinfile(const Node\* head, const char\* filename)

{

ofstream file(filename);

if (!file.is\_open())

{

cerr << "Помилка відкриття файлу" << endl;

exit(EXIT\_FAILURE);

}

const Node\* current = head;

while (current != nullptr)

{

file << current->data << endl;

current = current->next;

}

file.close();

}

Node\* readinfile(const char\* filename)

{

ifstream file(filename);

if (!file.is\_open())

{

cerr << "Помилка відкриття файлу" << endl;

exit(EXIT\_FAILURE);

}

Node\* head = nullptr;

char buffer[100];

while (file.getline(buffer, sizeof(buffer)))

{

addtolist(head, buffer);

}

file.close();

return head;

}

int main() {

Node\* myList = createList();

addition(myList, 3);

cout << "Список після додавання елементів:" << endl;

printList(myList);

earasenode(myList, "Елемент 1");

cout << "Список після знищення елемента:" << endl;

printList(myList);

writeinfile(myList, "myList.txt");

destroy(myList);

cout << "Список після знищення:" << endl;

printList(myList);

myList = readinfile("myList.txt");

cout << "Відновлений список:" << endl;

printList(myList);

destroy(myList);

return 0;

}

Завдання №2 Лабораторна робота №5 у Algotester

#include <iostream>

using namespace std;

int main()

{

int N, M, x, y;

cin >> N >> M >> x >> y;

int\*\* a = new int\*[N];

for (int i = 0; i < N; i++)

{

a[i] = new int[M]();

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < M; j++)

{

a[i][j] = -abs(x - 1 - i) - abs( y - 1 - j);

}

}

int min = 0;

for (int i = 0; i < N; i++)

{

for (int j = 0; j < M; j++)

{

if (min > a[i][j])

{

min = a[i][j];

}

}

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < M; j++)

{

a[i][j] -= min;

cout << a[i][j] << " ";

}

cout << endl;

}

return 0;

}

Завдання №3 Лабораторна робота №7\_8 у Algotester

#include <iostream>

using namespace std;

class Node

{

public:

int value;

Node\* prev, \* next;

};

class LinkedList

{

private:

struct Node

{

int value;

Node\* next;

Node\* prev;

Node(int val, Node\* node = nullptr)

{

value = val;

next = node;

}

};

Node\* head, \* tail;

int size;

public:

LinkedList()

{

head = nullptr;

tail = nullptr;

size = 0;

}

Node\* getNode(int index)

{

Node\* traverser = head;

int counter = 0;

while (counter != index)

{

if (traverser == nullptr) return traverser;

counter++;

traverser = traverser->next;

}

return traverser;

}

int getNodeValue(int index)

{

Node\* traverser = head;

int counter = 0;

while (counter != index)

{

if (traverser == nullptr) return traverser->value;

counter++;

traverser = traverser->next;

}

return traverser->value;

}

int SizeOfList()

{

int size = 0;

Node\* traverser = head;

while (traverser != nullptr)

{

size++;

traverser = traverser->next;

}

return size;

}

void insertNode(int index, int N, int arr[])

{

if (index > SizeOfList())

{

return;

}

for (int i = N - 1; i >= 0; i--)

{

Node\* traverser = new Node(arr[i]);

traverser->value = arr[i];

traverser->prev = nullptr;

traverser->next = nullptr;

if (index == 0) {

traverser->next = head;

if (head != nullptr)

{

head->prev = traverser;

}

head = traverser;

if (tail == nullptr)

{

tail = traverser;

}

}

else if (index == SizeOfList())

{

traverser->prev = tail;

if (tail != nullptr)

{

tail->next = traverser;

}

tail = traverser;

if (head == nullptr)

{

head = traverser;

}

}

else

{

Node\* ptr = head;

for (int j = 0; j < index - 1; j++)

{

ptr = ptr->next;

}

traverser->next = ptr->next;

traverser->prev = ptr;

if (ptr->next != nullptr)

{

ptr->next->prev = traverser;

}

ptr->next = traverser;

}

size = SizeOfList() + 1;

}

}

void eraseNode(int index, int N)

{

if (index < 0 || N < 0) return;

Node\* traverser = getNode(index);

for (int i = 0; i < N && traverser != nullptr; i++)

{

Node\* nextNode = traverser->next;

Node\* prevNode = traverser->prev;

if (prevNode)

prevNode->next = nextNode;

else

head = nextNode;

if (nextNode)

nextNode->prev = prevNode;

else

tail = prevNode;

delete traverser;

traverser = nextNode;

}

}

void setNodeValue(int index, int content)

{

Node\* traverser = getNode(index);

if (traverser != nullptr)

traverser->value = content;

}

friend ostream& operator<<(ostream& os, const LinkedList& list)

{

Node\* ptr = list.head;

while (ptr != nullptr)

{

os << ptr->value << " ";

ptr = ptr->next;

}

return os;

}

};

int main()

{

int Q;

cin >> Q;

LinkedList list;

while (Q--)

{

string request;

cin >> request;

if (request == "insert")

{

int index;

cin >> index;

int N;

cin >> N;

int nums[N];

for (int i = 0; i < N; i++)

{

cin >> nums[i];

}

list.insertNode(index, N, nums);

}

else if (request == "erase")

{

int index;

int n;

cin >> index >> n;

cout << endl;

list.eraseNode(index, n);

}

else if (request == "size")

{

int size = list.SizeOfList();

cout << size << endl;

}

else if (request == "set")

{

int index;

int content;

cin >> index >> content;

list.setNodeValue(index, content);

}

else if (request == "get")

{

int index;

cin >> index;

cout << list.getNodeValue(index) << endl;

}

else if (request == "print")

{

cout << list << endl;

cout << endl;

}

}

return 0;

}

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"

#include <iostream>

using namespace std;

class Node

{

public:

int value;

Node\* next;

Node(int val, Node\* node = nullptr)

{

value = val;

next = node;

}

};

class LinkedList

{

private:

Node\* head;

public:

LinkedList()

{

head = nullptr;

}

void append(int val)

{

Node\* newNode = new Node(val);

if (!head)

{

head = newNode;

}

else

{

Node\* current = head;

while (current->next)

{

current = current->next;

}

current->next = newNode;

}

}

void reverseAndOutput()

{

head = reverse(head);

outputList();

}

private:

Node\* reverse(Node\* head)

{

Node\* prev = nullptr;

Node\* current = head;

Node\* next = nullptr;

while (current != nullptr)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

}

return prev;

}

void outputList()

{

Node\* current = head;

while (current != nullptr)

{

cout << current->value << " ";

current = current->next;

}

cout << endl;

}

};

int main()

{

LinkedList list;

list.append(1);

list.append(2);

list.append(3);

list.append(4);

list.reverseAndOutput();

return 0;

}

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

#include <iostream>

using namespace std;

class Node

{

public:

int value;

Node\* next;

Node(int val, Node\* node = nullptr)

{

value = val;

next = node;

}

};

bool compare(Node\* h1, Node\* h2)

{

while (h1 != nullptr && h2 != nullptr)

{

if (h1->value != h2->value)

{

return false;

}

h1 = h1->next;

h2 = h2->next;

}

return h1 == nullptr && h2 == nullptr;

}

int main()

{

Node\* list1 = new Node(1, new Node(2, new Node(3)));

Node\* list2 = new Node(1, new Node(2, new Node(3)));

list1->next->next->next = new Node(4);

list2->next->next->next = new Node(5);

if (compare(list1, list2))

{

cout << "Списки рівні." << endl;

}

else

{

cout << "Списки відрізняються." << endl;

}

while (list1 != nullptr)

{

Node\* temp = list1;

list1 = list1->next;

delete temp;

}

while (list2 != nullptr)

{

Node\* temp = list2;

list2 = list2->next;

delete temp;

}

return 0;

}

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

#include <iostream>

using namespace std;

class Node

{

public:

int value;

Node\* next;

Node(int val, Node\* node = nullptr)

{

value = val;

next = node;

}

};

void outputList(Node\* head)

{

Node\* current = head;

while (current != nullptr)

{

cout << current->value;

if (current->next != nullptr)

{

cout << " -> ";

}

current = current->next;

}

cout << endl;

}

Node\* add(Node\* n1, Node\* n2)

{

Node\* resultHead = nullptr;

Node\* currentResult = nullptr;

int carry = 0;

while (n1 != nullptr || n2 != nullptr || carry != 0)

{

int sum = carry;

if (n1 != nullptr)

{

sum += n1->value;

n1 = n1->next;

}

if (n2 != nullptr)

{

sum += n2->value;

n2 = n2->next;

}

int digit = sum % 10;

carry = sum / 10;

Node\* newNode = new Node(digit);

if (resultHead == nullptr)

{

resultHead = newNode;

currentResult = newNode;

}

else

{

currentResult->next = newNode;

currentResult = newNode;

}

}

return resultHead;

}

int main()

{

Node\* number1 = new Node(3, new Node(7, new Node(9)));

Node\* number2 = new Node(8, new Node(6, new Node(3)));

cout << "Число 1: ";

outputList(number1);

cout << "Число 2: ";

outputList(number2);

Node\* result = add(number1, number2);

cout << "Сума: ";

outputList(result);

while (number1 != nullptr)

{

Node\* temp = number1;

number1 = number1->next;

delete temp;

}

while (number2 != nullptr)

{

Node\* temp = number2;

number2 = number2->next;

delete temp;

}

while (result != nullptr)

{

Node\* temp = result;

result = result->next;

delete temp;

}

return 0;

}

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

#include <iostream>

using namespace std;

class TreeNode

{

public:

int value;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val)

{

value = val;

left = nullptr;

right = nullptr;

}

};

TreeNode\* mirrorFlip(TreeNode\* gilka)

{

if (gilka == nullptr)

{

return nullptr;

}

TreeNode\* mirroredgilka = new TreeNode(gilka->value);

mirroredgilka->left = mirrorFlip(gilka->right);

mirroredgilka->right = mirrorFlip(gilka->left);

return mirroredgilka;

}

void outputTreaa(TreeNode\* gilka)

{

if (gilka == nullptr)

{

return;

}

cout << gilka->value << " ";

outputTreaa(gilka->left);

outputTreaa(gilka->right);

}

int main()

{

TreeNode\* gilka = new TreeNode(1);

gilka->left = new TreeNode(2);

gilka->right = new TreeNode(3);

gilka->left->left = new TreeNode(4);

gilka->left->right = new TreeNode(5);

cout << "Оригінальне дерево: ";

outputTreaa(gilka);

cout << endl;

TreeNode\* mirroredTree = mirrorFlip(gilka);

cout << "Віддзеркалене дерево: ";

outputTreaa(mirroredTree);

cout << endl;

delete gilka;

delete mirroredTree;

return 0;

}

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”

#include <iostream>

#include <queue>

using namespace std;

class TreeNode

{

public:

int value;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val)

{

value = val;

left = nullptr;

right = nullptr;

}

};

void outputTree(TreeNode\* gilka)

{

if (gilka == nullptr)

{

return;

}

cout << gilka->value << " ";

outputTree(gilka->left);

outputTree(gilka->right);

}

void sumOfTree(TreeNode\* gilka)

{

if (gilka == nullptr)

{

return;

}

queue<TreeNode\*> nodesQueue;

nodesQueue.push(gilka);

while (!nodesQueue.empty())

{

TreeNode\* currentNode = nodesQueue.front();

nodesQueue.pop();

int sum = 0;

if (currentNode->left != nullptr)

{

nodesQueue.push(currentNode->left);

sum += currentNode->left->value;

}

if (currentNode->right != nullptr)

{

nodesQueue.push(currentNode->right);

sum += currentNode->right->value;

}

currentNode->value += sum;

}

}

int main()

{

TreeNode\* gilka = new TreeNode(1);

gilka->left = new TreeNode(2);

gilka->right = new TreeNode(3);

gilka->left->left = new TreeNode(4);

gilka->left->right = new TreeNode(5);

cout << "Оригінальне дерево: ";

outputTree(gilka);

cout << endl;

sumOfTree(gilka);

cout << "Дерево після обчислення сум: ";

outputTree(gilka);

cout << endl;

delete gilka->left->left;

delete gilka->left->right;

delete gilka->left;

delete gilka->right;

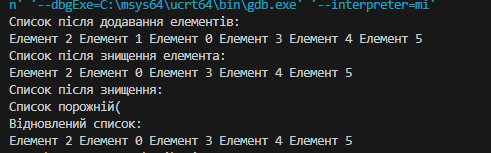
delete gilka;

return 0;

}

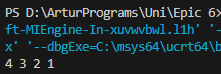
## **5. Результати виконання завдань, тестування та фактично витрачений час:**

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"



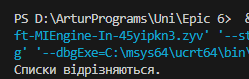
Затрачений час на виконання: 2.5год

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"



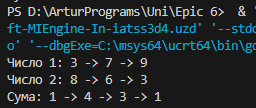
*Затрачений час на виконання: 1год*

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”



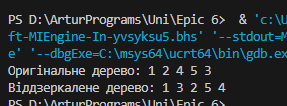
*Затрачений час на виконання: 1год*

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”



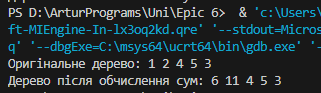
Затрачений час на виконання: 3год

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”



*Затрачений час на виконання: 2год*

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”



*Затрачений час на виконання: 1.5год*

# **Висновки:**

* Ознайомлений зі списками
* Ознайомлений з деревами
* Ознайомлений зі стеками
* Ознайомлений з чергами
* Успішно виконав лабораторні та практичні роботи