Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №6**

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

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

***Виконав:***

студент групи ШІ-13

Шабанов Микита Миколайович

# **Тема роботи:**

Інформаційні динамічні структури в програмуванні. Зв’язаний список, робота зі списками та бінарне дерево. Операції над списками та бінарним деревом.

# **Мета роботи:**

Навчитися користуватися зв’язними листами, подвійними зв’язними листами та бінарним деревом. Розвинути розуміння структур даних та алгоритмічного мислення, засвоїти механізми маніпуляції з покажчиками. Поглибити розуміння рівності в структурах даних, операцій зі структурами даних.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Шаблон класу.
* Тема №2: Однозв’язний список.
* Тема №3: Двозв’язний список.
* Тема №4: Бінарне дерево.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Шаблон класу.
  + Джерела Інформації:
    - Стаття: <https://www.bestprog.net/uk/2018/06/29/the-concept-of-a-class-template-the-template-keyword-advantages-of-using-templates-examples-of-declaring-and-using-of-class-templates-arguments-in-templates_ua/>  
      <http://nikolay.in.ua/navchaemos/visual-basic/ob-ektno-orientovane-programuvannya/289-struktura-klasu>  
      <https://acode.com.ua/urok-183-shablony-klasiv/>
    - Відео: <https://www.youtube.com/watch?app=desktop&v=CUGTegwAwz4>
    - <https://www.youtube.com/watch?v=mQqzP9EWu58>
  + Що опрацьовано: Шаблони класів, як їх використовувати і для чого вони потрібні. Опрацьовано деталі їх впровадження у код.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 14.12.23
  + Звершення опрацювання теми: 17.12.23
* Тема №2: Однозв’язний список.
  + Джерела Інформації:
    - Стаття: <https://www.bestprog.net/uk/2022/02/11/c-linear-singly-linked-list-general-information-ua/>   
      <https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html>  
      <https://www.programiz.com/dsa/linked-list>  
      <https://www.geeksforgeeks.org/what-is-linked-list/>
    - Відео: <https://www.youtube.com/watch?v=-25REjF_atI>
  + Що опрацьовано: Опрацьовано однозв’язний список. Логіка роботи та як впроваджувати у код.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 13.12.23
  + Звершення опрацювання теми: 17.12.23
* Тема №3: Двозв’язний список.
  + Джерела Інформації:
    - Стаття: <https://www.bestprog.net/uk/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ua/>
    - Відео: <https://www.youtube.com/watch?v=QLzu2-_QFoE>
  + Що опрацьовано: Опрацьовано двозв'язний список. Як його імплементувати у код. Особливості цього виду списку і різзниця між однозв'язним списком.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 14.12.23
  + Звершення опрацювання теми: 17.12.23
* Тема №4: Бінарне дерево.
  + Джерела Інформації:
    - Стаття: <https://uk.myservername.com/binary-search-tree-c>  
      <https://www.programiz.com/dsa/binary-tree>  
      <https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/> <https://www.bestprog.net/uk/2023/08/01/c-binary-search-tree-binarytree-class-integers-ua/>
    - Відео: <https://www.youtube.com/watch?v=qBFzNW0ALxQ>  
      <https://www.youtube.com/watch?v=COZK7NATh4k>   
      <https://youtu.be/zuuAPYiMYDA?si=80hDy--vJLyRbkd_>
  + Що опрацьовано: бінарне дерево.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 15.12.23
  + Звершення опрацювання теми: 17.12.23

# **Виконання роботи:**

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

Завдання №1 **VNS Lab 10**

* Варіант завдання: 6
* Деталі завдання: Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.
* Важливі деталі для імплементації: Записи в лінійному списку містять ключове поле типу int.

Завдання №2 **Algotester Lab 5**

* Варіант завдання: 2
* Деталі завдання: У пустелі розташована особлива печера, яка має двовимірну структуру. Її висота дорівнює N, ширина - M.
* Усередині печери розташовані різні об'єкти: порожнеча позначається буквою O, пісок - S, а каміння - X.
* Одного дня сталося землетрус, в результаті якого весь пісок опустився вниз. Пісок падає на найнижчу вільну клітинку, але він не може проходити через каміння.
* Ваша задача - описати, як виглядатиме печера після землетрусу.

Завдання №3 **Algotester Lab 7-8**

* Варіант завдання: 3
* Деталі завдання: Ваше завдання - власноруч реалізувати структуру даних "Двійкове дерево пошуку". Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його параметри.
* Вам будуть поступати запити такого типу:
* Вставка: Ідентифікатор - insert. Ви отримуєте ціле число value - число, яке треба вставити в дерево.
* Пошук: Ідентифікатор - contains. Ви отримуєте ціле число value - число, наявність якого у дереві необхідно перевірити. Якщо value наявне в дереві - ви виводите Yes, у іншому випадку No.
* Визначення розміру: Ідентифікатор - size. Ви не отримуєте аргументів. Ви виводите кількість елементів у дереві.
* Вивід дерева на екран: Ідентифікатор - print. Ви не отримуєте аргументів. Ви виводите усі елементи дерева через пробіл. Реалізувати використовуючи перегрузку оператора <<.

Завдання №4 **Class Practice Work**

Task 1:

* Деталі завдання: Реалізувати метод реверсу списку та допоміжний метод виведення вхідного і обернутого списків.  
  - Важливі деталі для імплементації: Використовувати цілочисельні значення в списку.

Task 2:

* Деталі завдання: Реалізувати функцію, яка ітеративно проходиться по обох списках і порівню дані в кожному вузлі.  
  Якщо виявлено невідповідність даних або якщо довжина списків різна, функція повертає false.  
  - Важливі деталі для імплементації: Використовувати цілочисельні значення в списку.

Task 3:

* Деталі завдання: Реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку. Молодший розряд числа записано в голові списка. Функція повертає новий список.
* Важливі деталі для імплементації: використовувати цифри від 0 до 9 для значень у списку.

Task 4:

* Деталі завдання: Реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева. Функція повертає нове дерево.
* Важливі деталі для імплементації: використовувати цілі числа для значень у вузлах дерева. Передане в функцію дерево не модифікується

Task 5:

* Деталі завдання: Реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів. Вузол-листок не змінює значення. Значення змінюються від листків до кореня дерева
* Важливі деталі для імплементації: Використовувати цілочисельні значення у вузлах дерева

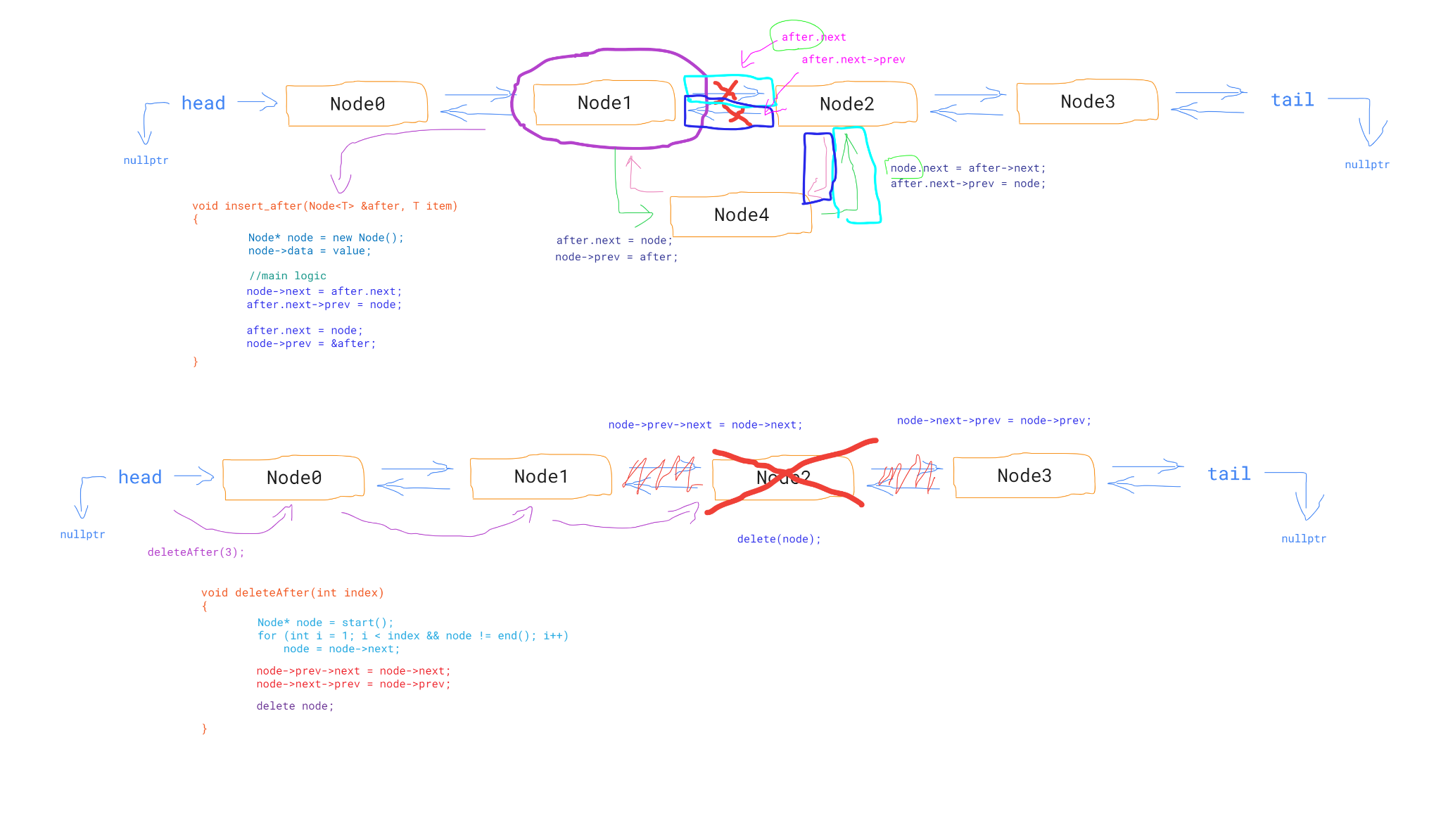
Завдання №5 **Self Practice Work**

* **Add Binary**

Given two binary strings a and b, return their sum as a binary string.

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

Програма №1 **VNS Lab 10**



*Рисунок 1 малюнок до VNS 10*

* Планований час на реалізацію: 8 годин
* Важливі деталі для врахування в імплементації: Записи в лінійному списку містять ключове поле типу int.

Програма №2 **Algotester Lab 5**

* Планований час на реалізацію: 10 годин
* Важливі деталі для врахування в імплементації: Клiтинкi якi мають сумiжну сторону з вершиною мають висоту на один меншу, сумiжнi з ними i не розфарбованi мають ще на 1 меншу висоту i так далi.

Програма №3 **Algotester Lab 7-8**

* Планований час на реалізацію: 1 день

Програма №4 **Class Practice Work**

* Планований час на реалізацію: 5 годин

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

Як і в першому епіку

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

Завдання №1 **VNS Lab 10**

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**using namespace std;**

**struct Node**

**{**

**int data;**

**Node\* next;**

**Node\* prev;**

**};**

**class dll**

**{**

**private:**

**//створюємо початок та кінець списку**

**//вони не є посиланнями**

**Node head;**

**Node tail;**

**public:**

**dll()**

**{**

**head.prev = nullptr;**

**head.next = &tail;**

**tail.prev = &head;**

**tail.next = nullptr;**

**}**

**int size = 0;**

**Node\* start() {**

**return head.next;**

**}**

**Node\* end() {**

**return &tail;**

**}**

**void pushBack(int value) {**

**insertAfter(\*tail.prev, value);**

**}**

**void push(int value) {**

**insertAfter(head, value);**

**}**

**void insertAfter(Node &after, int value)**

**{**

**Node\* node = new Node();**

**node->data = value;**

**//main logic**

**node->next = after.next;**

**after.next->prev = node;**

**after.next = node;**

**node->prev = &after;**

**size++;**

**}**

**void deleteAfter(int index)**

**{**

**//exeption**

**if (index <= 0 || index > size) {**

**cout << "Wrong index..." << endl;**

**return;**

**} if(head.next == nullptr) {**

**cout << "List is empty..." << endl;**

**return;**

**}**

**Node\* node = start();**

**for (int i = 1; i < index && node != end(); i++)**

**node = node->next;**

**node->prev->next = node->next;**

**node->next->prev = node->prev;**

**delete node;**

**size--;**

**}**

**void print()**

**{**

**if (size == 0)**

**{**

**cout << "List is empty..." << endl;**

**return;**

**}**

**for (Node\* list = head.next; list->next != nullptr; list = list->next)**

**cout << list->data << " ";**

**cout << endl;**

**}**

**bool writeToFile(string fileName)**

**{**

**ofstream fout(fileName);**

**//error while try to open while**

**if (!fout.is\_open())**

**return false;**

**for (Node\* list = start(); list->next != nullptr; list = list->next)**

**fout << list->data << endl;**

**return true;**

**}**

**bool readFromFile(string fileName)**

**{**

**ifstream fin(fileName);**

**Node\* result = start();**

**//error while try to open while**

**if (!fin.is\_open())**

**return false;**

**string line;**

**while (getline(fin, line))**

**pushBack(stoi(line));**

**return true;**

**}**

**void clear()**

**{**

**for (Node\* list = head.next->next; list != nullptr; list = list->next)**

**delete list->prev;**

**size = 0;**

**new dll();**

**}**

**};**

**int main()**

**{**

**//1. Створення списку**

**dll list = dll();**

**//2. Додавання елемента в список**

**for (int i = 0; i < 10; i++)**

**{**

**if(i < 5)**

**list.push(i);**

**else**

**list.pushBack(i);**

**}**

**list.print();**

**//3. Знищення елемента зі списку**

**list.deleteAfter(10);**

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

**list.push(66);**

**//4. Друк списку**

**list.print();**

**//5. Запис списку у файл**

**list.writeToFile("saveLinkedList.dat");**

**//6. Знищення списку**

**list.clear();**

**list.print();**

**//7. Відновлення списку з файлу.**

**dll list2 = dll();**

**list2.readFromFile("saveLinkedList.dat");**

**list2.print();**

**return 0;**

**}**

Код виконує створення двонаправленого списку та його виведення. Також реалізовані функції для видалення елемента після елемента з вказаним номером та додавання елементу в початок списку.

Список можна записати або прочитати з файлу. Можна очистити список

Додаткові джерела: <https://www.programiz.com/dsa/linked-list>  
<https://www.geeksforgeeks.org/data-structures/linked-list/>

Завдання №2 **Algotester Lab 5**

#include <iostream>

int main()

{

using namespace std;

//prompting

char cave[1000][1000];

int n, m;

cin >> n >> m;

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

{

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

cin >> cave[i][j];

}

//solution

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

{

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

{

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

{

if(cave[p][i] == 'S' && cave[p+1][i] == 'O')

{

char tmp = cave[p][i];

cave[p][i] = cave[p+1][i];

cave[p+1][i] = tmp;

}

}

}

}

cout << endl;

//output

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

{

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

cout << cave[i][j];

cout << endl;

}

return 0;

}

Програма робить розмальовку карти розміром N \* M, де найнижча точка має висоту 0, а пік гори має найбільше число. Пік гори знаходиться на координатах {x, y}. Клітинки, які мають суміжню сторону з вершиною піку гори, мають висоту на одиницю меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

Завдання №3 **Algotester Lab 7-8**

**#include <iostream>**

**using namespace std;**

**// Binary tree (search)**

**//custom type**

**template<class T>**

**class BTS**

**{**

**class Node**

**{**

**public:**

**T data;**

**Node \*left;**

**Node \*right;**

**Node \*parent;**

**Node(T value, Node \*left\_ptr, Node \*right\_ptr, Node \*parent\_ptr)**

**{**

**data = value;**

**left = left\_ptr;**

**right = right\_ptr;**

**parent = parent\_ptr;**

**}**

**explicit Node(T value) : Node(value, nullptr, nullptr, nullptr) {}**

**};**

**int size;**

**Node \*root;**

**void insert\_element(Node \*head, Node \*new\_node)**

**{**

**if (new\_node == nullptr) return;**

**if (head == nullptr)**

**{**

**root = new\_node;**

**size++;**

**return;**

**}**

**Node \*current = head;**

**if (current->data > new\_node->data)**

**{**

**if (current->left == nullptr)**

**{**

**current->left = new\_node;**

**new\_node->parent = current;**

**new\_node->left = nullptr;**

**new\_node->right = nullptr;**

**size++;**

**return;**

**}**

**insert\_element(current->left, new\_node);**

**} else if (current->data < new\_node->data)**

**{**

**if (current->right == nullptr)**

**{**

**current->right = (new\_node);**

**new\_node->parent = current;**

**new\_node->left = nullptr;**

**new\_node->right = nullptr;**

**size++;**

**return;**

**}**

**insert\_element(current->right, new\_node);**

**}**

**}**

**void delete\_BTS(Node\* head)**

**{**

**if (head == nullptr) return;**

**delete\_BTS(head->left);**

**delete\_BTS(head->right);**

**delete head;**

**}**

**void contain(Node \*node, T value)**

**{**

**Node \*current = node;**

**if (current == nullptr)**

**{**

**cout << "No";**

**return;**

**} else**

**{**

**if (current->data == value)**

**{**

**cout << "Yes";**

**return;**

**} else**

**{**

**if (current->data > value)**

**contain(current->left, value);**

**else**

**contain(current->right, value);**

**}**

**}**

**}**

**void remove(Node \*node)**

**{**

**if (node->parent == nullptr)**

**{**

**root = nullptr;**

**Node \*left = node->left;**

**Node \*right = node->right;**

**size--;**

**delete node;**

**if (left != nullptr)**

**{**

**left->parent = nullptr;**

**root = left;**

**insert\_element(root, right);**

**} else if (right != nullptr)**

**{**

**right->parent = nullptr;**

**root = right;**

**}**

**}**

**if (node->parent->left == node)**

**node->parent->left = nullptr;**

**else**

**node->parent->right = nullptr;**

**Node \*left = node->left;**

**Node \*right = node->right;**

**size--;**

**delete node;**

**insert\_element(root, left);**

**insert\_element(root, right);**

**}**

**void print(Node \*node, std::ostream& out) const**

**{**

**Node \*current = node;**

**if (current == nullptr) return;**

**print(current->left, out);**

**out << current->data << ' ';**

**print(current->right, out);**

**}**

**public:**

**BTS()**

**{**

**root = nullptr;**

**size = 0;**

**}**

**~BTS()**

**{**

**delete\_BTS(root);**

**}**

**void insertion(T value)**

**{**

**insert\_element(root, new Node(value));**

**}**

**void check(T value)**

**{**

**contain(root, value);**

**}**

**int count\_size()**

**{**

**return size;**

**}**

**void print\_bts(std::ostream& out) const**

**{**

**print(root, out);**

**}**

**};**

**template<class T>**

**std::ostream& operator<<(std::ostream &out, const BTS<T>& tree)**

**{**

**tree.print\_bts(out);**

**return out;**

**}**

**int main()**

**{**

**BTS<int> tree;**

**int queries;**

**string query;**

**cin >> queries;**

**for (int i = 0; i < queries; ++i)**

**{**

**cin >> query;**

**if (query == "insert")**

**{**

**int element;**

**cin >> element;**

**tree.insertion(element);**

**} else if (query == "size")**

**cout << tree.count\_size() << endl;**

**else if (query == "print")**

**cout << tree << endl;**

**else if (query == "contains")**

**{**

**int element;**

**cin >> element;**

**tree.check(element);**

**cout << endl;**

**}**

**}**

**return 0;**

**}**

В коді реалізована вставка, пошук, визначення розміру та вивід дерева на екран

Додаткові джерела: <https://www.geeksforgeeks.org/list-reverse-function-in-c-stl/>

Завдання №4 **Class Practice Work**

Task 1:

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node\* prev;

};

class dll

{

private:

Node head;

Node tail;

public:

dll()

{

head.prev = nullptr;

head.next = &tail;

tail.prev = &head;

tail.next = nullptr;

}

int size = 0;

Node\* start() {

return head.next;

}

void pushBack(int value) {

insertAfter(\*tail.prev, value);

}

void push(int value) {

insertAfter(head, value);

}

void insertAfter(Node &after, int value)

{

Node\* node = new Node();

node->data = value;

//main logic

node->next = after.next;

after.next->prev = node;

after.next = node;

node->prev = &after;

size++;

}

void print()

{

if (size == 0)

{

cout << "List is empty..." << endl;

return;

}

int i = 0;

for (Node\* list = head.next; list->next != nullptr && i < size; list = list->next, i++)

cout << list->data << " ";

cout << endl;

}

void reverse()

{

Node\* temp = nullptr;

Node\* current = head.next;

while (current)

{

temp = current->prev;

current->prev = current->next;

current->next = temp;

current = current->prev;

}

if (temp != nullptr)

head.next = temp->prev;

}

};

int main()

{

dll list = dll();

for (int i = 0; i <= 8; i++)

{

if(i < 5)

list.push(i);

else

list.pushBack(i);

}

list.print();

list.reverse();

//output reverse list

int i = 0;

for (Node\* print = list.start(); i < list.size; print = print->next, i++)

cout << print->next->data << " ";

return 0;

}

Код виконує реалізацію методу реверсу списку та допоміжного методу для виведення вхідного і обернутого списків. Програма дозволяє здійснити обертання порядку елементів усередині списку та вивести на екран вхідний та обернутий список для порівняння.

Task 2:

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node\* prev;

};

class dll

{

private:

Node head;

Node tail;

public:

dll()

{

head.prev = nullptr;

head.next = &tail;

tail.prev = &head;

tail.next = nullptr;

}

int size = 0;

Node\* start() {

return head.next;

}

void pushBack(int value) {

insertAfter(\*tail.prev, value);

}

void push(int value) {

insertAfter(head, value);

}

void insertAfter(Node &after, int value)

{

Node\* node = new Node();

node->data = value;

//main logic

node->next = after.next;

after.next->prev = node;

after.next = node;

node->prev = &after;

size++;

}

void print()

{

if (size == 0)

{

cout << "List is empty..." << endl;

return;

}

int i = 0;

for (Node\* list = head.next; list->next != nullptr && i < size; list = list->next, i++)

cout << list->data << " ";

cout << endl;

}

bool compare(dll &another\_dll)

{

if(size != another\_dll.size)

return false;

else if(size == 0)

return true;

Node\* n1 = start(); Node\* n2 = another\_dll.start();

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

{

if (n1->data != n2->data)

return false;

}

return true;

}

};

int main()

{

dll list1 = dll();

dll list2 = dll();

for (int i = 0; i <= 8; i++)

{

if(i < 5)

{

list1.push(i);

list2.push(i);

}

else

{

list1.pushBack(i);

list2.pushBack(i);

}

}

list1.print();

list1.compare(list2) ? cout << "the lists are the same" : cout << "lists are different";

cout << endl << "push back element>" << endl;

list2.pushBack(1);

list1.compare(list2) ? cout << "the lists are the same" : cout << "lists are different";

return 0;

}

Код виконує реалізацію функції, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі. Якщо виявлено невідповідність даних або якщо довжина списків різна, функція повертає false. Програма дозволяє визначити, чи є дані відповідними одне одному в обох списках та враховує різницю в їхніх довжинах. У випадку виявлення невідповідності або різниці в довжинах, функція повертає false, в іншому випадку - true

Task 3:

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node\* prev;

};

class dll

{

private:

Node head;

Node tail;

public:

dll()

{

head.prev = nullptr;

head.next = &tail;

tail.prev = &head;

tail.next = nullptr;

}

int size = 0;

Node\* start() {

return head.next;

}

void pushBack(int value) {

insertAfter(\*tail.prev, value);

}

void push(int value) {

insertAfter(head, value);

}

void insertAfter(Node &after, int value)

{

Node\* node = new Node();

node->data = value;

//main logic

node->next = after.next;

after.next->prev = node;

after.next = node;

node->prev = &after;

size++;

}

void print()

{

if (size == 0)

{

cout << "List is empty..." << endl;

return;

}

int i = 0;

for (Node\* list = head.next; list->next != nullptr && i < size; list = list->next, i++)

cout << list->data << " ";

cout << endl;

}

};

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

{

dll\* result = new dll();

int over = 0;

while (n1 || n2 || over) {

int sum = (n1 ? n1->data : 0) + (n2 ? n2->data : 0) + over;

over = sum / 10;

result->pushBack(sum % 10);

if (n1) n1 = n1->next;

if (n2) n2 = n2->next;

}

return result->start();

}

int main()

{

dll list1 = dll();

dll list2 = dll();

list1.push(1);

list1.push(5);

list1.push(0);

list2.push(5);

list2.push(5);

list1.print();

list2.print();

cout << "expected sum: 150 + 55 = 205" << endl;

cout << "result: ";

Node\* result = new Node();

result = nodeSum(list1.start(), list2.start());

while(result != nullptr)

{

result = result->next;

cout << result->data;

}

return 0;

}

Код виконує реалізацію функції, яка обчислює суму двох чисел, представлених у вигляді списків

Додаткові джерела: <https://www.geeksforgeeks.org/add-two-numbers-represented-by-linked-list/>

Task 4:

#include <iostream>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) : data(val), left(nullptr), right(nullptr) {}

};

TreeNode\* create\_mirror\_flip(TreeNode\* root) {

if (!root) {

return nullptr;

}

TreeNode\* temp = root->left;

root->left = create\_mirror\_flip(root->right);

root->right = create\_mirror\_flip(temp);

return root;

}

void print\_tree(TreeNode\* root) {

if (root) {

print\_tree(root->left);

cout << root->data << " ";

print\_tree(root->right);

}

}

void delete\_tree(TreeNode\* root) {

if (root) {

delete\_tree(root->left);

delete\_tree(root->right);

delete root;

}

}

int main() {

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

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

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

root->right->left = new TreeNode(6);

root->right->right = new TreeNode(7);

cout << "Original Tree: ";

print\_tree(root);

cout << endl;

TreeNode\* mirroredTree = create\_mirror\_flip(root);

cout << "Mirrored Tree: ";

print\_tree(mirroredTree);

cout << endl;

return 0;

}

Код виконує реалізацію функції, яка проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева. Функція повертає нове дерево, при цьому вхідне дерево не модифікується. Важливою деталлю є використання цілих чисел для значень у вузлах дерева, що враховує вимоги щодо типів даних у вузлах.  
Додаткові джерела: <https://www.tutorialspoint.com/print-binary-tree-in-cplusplus>  
<https://stackoverflow.com/questions/36802354/print-binary-tree-in-a-pretty-way-using-c>

Task 5:

#include <iostream>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) : data(val), left(nullptr), right(nullptr) {}

};

void print\_tree(TreeNode\* root) {

if (root) {

print\_tree(root->left);

cout << root->data << " ";

print\_tree(root->right);

}

}

void calculate\_tree\_sum(TreeNode\* root) {

if (!root || (!root->left && !root->right)) {

return;

}

calculate\_tree\_sum(root->left);

calculate\_tree\_sum(root->right);

root->data = (root->left ? root->left->data : 0) + (root->right ? root->right->data : 0);

}

int main() {

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

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

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

root->right->left = new TreeNode(6);

root->right->right = new TreeNode(7);

cout << "Original Tree: ";

print\_tree(root);

cout << endl;

calculate\_tree\_sum(root);

cout << "Tree after Sum Calculation: ";

print\_tree(root);

cout << endl;

return 0;

}

Код виконує реалізацію функції, яка ітеративно проходить по бінарному дереві і оновлює значення у батьківських вузлах, записуючи суму значень підвузлів. Значення вузлів-листків залишаються незмінними. Зміни в значеннях вузлів відбуваються в напрямку від листків до кореня дерева. Важливою деталлю є використання цілих чисел у вузлах дерева, що враховує вимоги до типів даних у вузлах дерева.

Завдання №5 **Self Practice**

**class Solution {**

**public:**

**string addBinary(string a, string b)**

**{**

**string result;**

**int i = a.size() - 1, j = b.size() - 1;**

**int carry = 0;**

**while(i >= 0 || j >= 0 || carry)**

**{**

**int numA = (i >= 0) ? a[i--] - '0' : 0,**

**numB = (j >= 0) ? b[j--] - '0' : 0;**

**int sum = numA + numB + carry;**

**carry = sum / 2;**

**result += (sum % 2) + '0';**

**}**

**reverse(begin(result), end(result));**

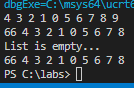
**return result;**

**}**

**};**

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

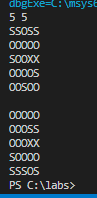
Завдання №1 **VNS 10**



*Рисунок 2 Виконання VNS 10*

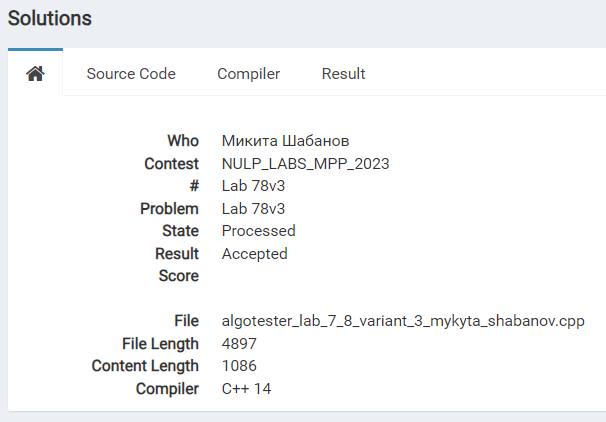
Час, затрачений на виконання завдання: 8 годин

Завдання №2 **Algotester lab 5v3**



*Рисунок 3 Виконання Algotester lab 5v3*

Час, затрачений на виконання завдання: 5 годин

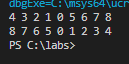
Завдання №3 **Algotester lab 7-8v3**

*Рисунок 4 Виконане завдання на Algotester lab 7-8v3*

Час, затрачений на виконання завдання: 7 годин

Завдання №4 **Class Practice Work**

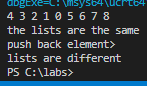
Task 1



*Рисунок 5 Виконання Class Practice Work Task 1*

Час, затрачений на виконання завдання: 2 години

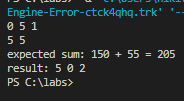
Task 2



*Рисунок 6 Виконання Class Practice Work Task 2*

Час, затрачений на виконання завдання: година

Task 3



*Рисунок 7 Виконання Class Practice Work Task 3*

Час, затрачений на виконання завдання: 1 година

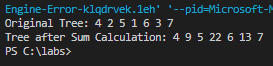
Task 4



*Рисунок 8 Виконання Class Practice Work Task 4*

Час, затрачений на виконання завдання: 2 години

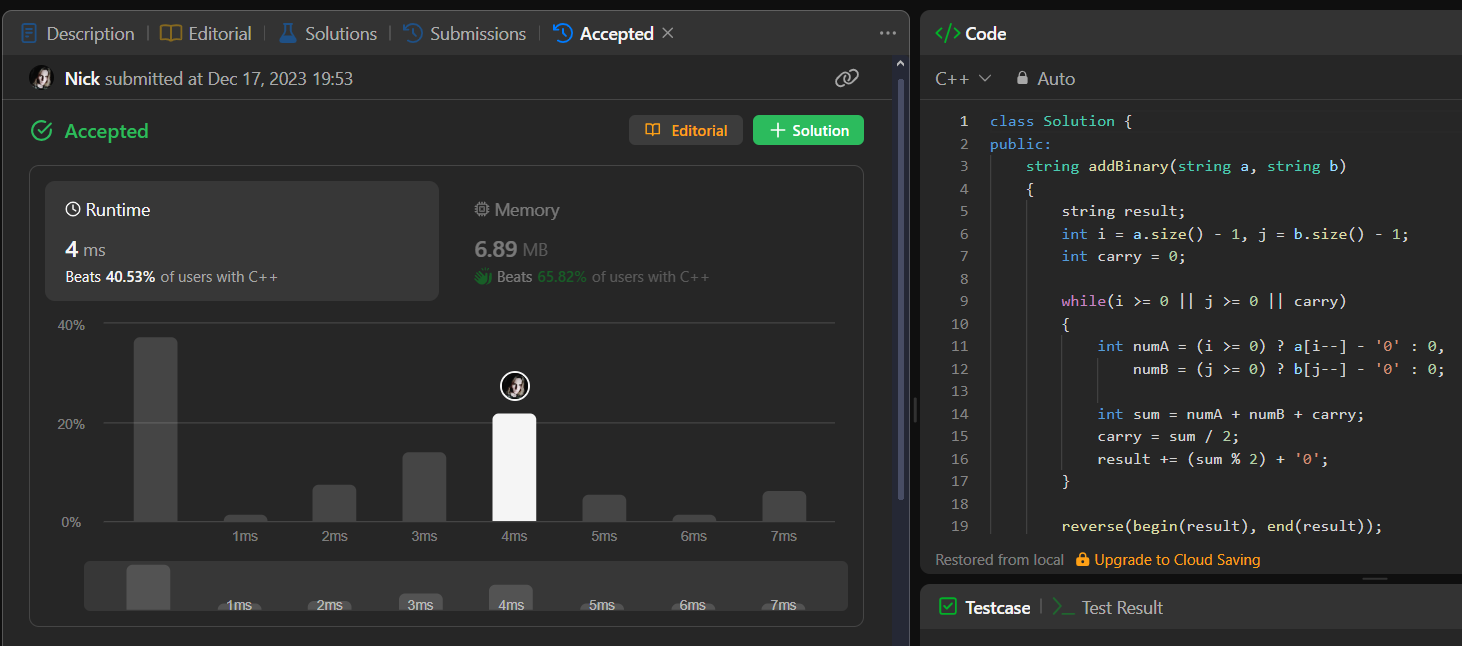
Task 5



*Рисунок 9 Виконання Class Practice Work Task 5*

Час, затрачений на виконання завдання: 3 години

Завдання №5 **Self Practice Work**

**

*Рисунок 10 Зарахування Self Practice*

Час, затрачений на виконання завдання: 30 хвилин

# **Висновки:**

Я попроцював з двонаправленим списком, його вузлами, вказівниками, структурами та класами, реалізовуючи різні методи.

Також дізнався про бінарне дерево та його структуру