Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**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:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

**Виконала:**

Студентка групи ШІ-13

Кузнєцова Анастасія Володимирівна

# **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: 08.12.23
  + Звершення опрацювання теми: 14.12.23
* 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: 08.12.23
  + Звершення опрацювання теми: 14.12.23
* 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: 08.12.23
  + Звершення опрацювання теми: 14.12.23
* 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: Що таке бінарне дерево. Як його імплементувати у код, які його особливості і спосіб застосування.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.23
  + Звершення опрацювання теми: 15.12.23

# **Виконання роботи:**

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

Завдання №1 **VNS Lab 10**

* Варіант завдання: 8
* Деталі завдання: Створити функцію для створення двонаправленого списку та його виведення. Створити функції для видалення елемента після елемента з вказаним номером та додавання К елементів на початок списку. Записати список у файл, виконати його очищення, та вивести повідомлення "Список порожній" при спробі виведення. Відновити список з файлу та вивести його, після - очистити список.
* Важливі деталі для імплементації: Записи в лінійному списку містять ключове поле типу int.

Завдання №2 **Algotester Lab 5**

* Варіант завдання: 3
* Деталі завдання: Ваше завдання - розмалювати карту розміром N\*M таким чином, щоб найнижча точка мала число 0, а пiк гори мав найбiльше число. Пік гори знаходиться на координатах {x, y}.
* Важливі деталі для 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.

Завдання №3 **Algotester Lab 7-8**

* Варіант завдання: 1  
  Деталі завдання: Реалізувати структуру даних "Двозв’язний список" з можливістю виконання операцій вставки, видалення, визначення розміру, отримання значення елемента за індексом, модифікації значення елемента та виводу списку на екран. Операції визначаються ідентифікаторами, такими як "insert", "erase", "size", "get", "set", "print".
* Важливі деталі для врахування в імплементації програми: Для виведення списку на екран використовується перегрузка 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

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

Програма №1 **VNS Lab 10**

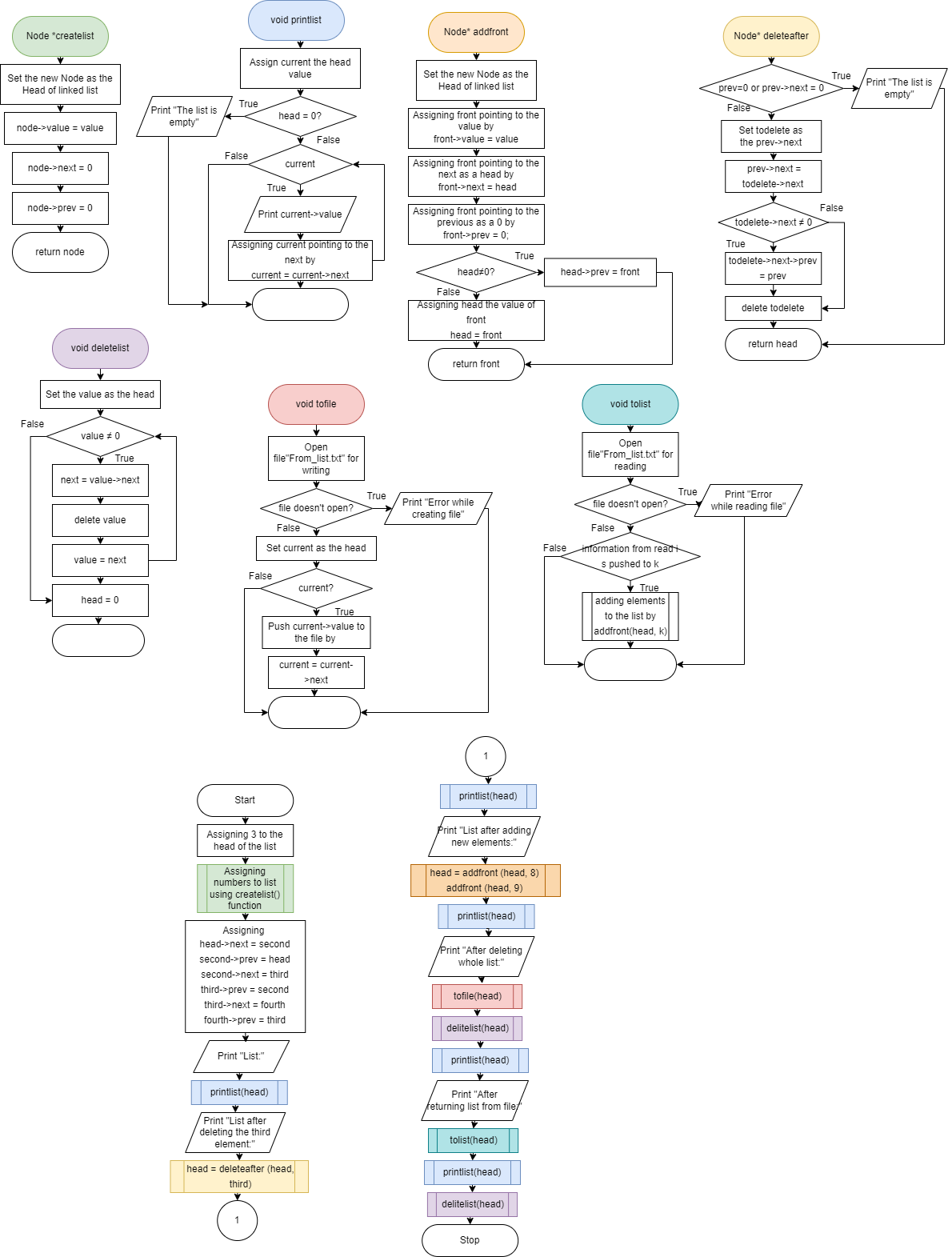


Рисунок Блок-схема до VNS 10

* 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
* Важливі деталі для врахування в імплементації: Для виведення списку на екран використовується перегрузка оператора "<<".

Програма №4 **Class Practice Work**

* Планований час на реалізацію: 5 годин

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

Файл settings.json в якому зберігаються налаштування VS Code

{

*"workbench.colorTheme"*: "Monokai",

*"code-runner.runInTerminal"*: true,

*"C\_Cpp.default.compilerPath"*: "d:\\C C++\\test.c",

*"cmake.configureOnOpen"*: true,

*"terminal.integrated.defaultProfile.windows"*: "Windows PowerShell",

*"terminal.integrated.profiles.windows"*: {

*"PowerShell"*: {

*"source"*: "PowerShell",

*"icon"*: "terminal-powershell"

        },

*"Command Prompt"*: {

*"path"*: [

                "${env:windir}\\Sysnative\\cmd.exe",

                "${env:windir}\\System32\\cmd.exe"

            ],

*"args"*: [],

*"icon"*: "terminal-cmd"

        },

*"Git Bash"*: {

*"source"*: "Git Bash"

        },

*"bash (MSYS2)"*: {

*"path"*: "C:\\msys64\\usr\\bin\\bash.exe",

*"args"*: [

                "--login",

                "-i"

            ]

        },

*"Windows PowerShell"*: {

*"path"*: "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"

        }

    },

*"workbench.colorCustomizations"*: {}

}

## **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* value;

    Node\* next;

    Node\* prev;

};

Node\* createlist(*int* *value*)

{

    Node\* node = new Node();

    node->value = *value*;

    node->next = nullptr;

    node->prev = nullptr;

    return node;

}

*void* printlist (Node\**head*)

{

    Node\*current=*head*;

    if(*head* == nullptr)

    {

        cout<<"The list is empty"<<endl;

        return;

    }

    while (current)

    {

        cout<<current->value<<"\n";

        current = current->next;

    }

}

Node\* addfront (Node\*&*head*, *int* *value*)

{

    Node\* front = new Node();

    front->value = *value*;

    front->next = *head*;

    front->prev = nullptr;

    if (*head* != nullptr)

    {

*head*->prev = front;

    }

*head* = front;

    return front;

}

Node\* deleteafter (Node\**head*, Node\**prev*)

{

    if(*prev*==nullptr || *prev*->next == nullptr)

    {

        cout<<"The list is empty"<<endl;

        return *head*;

    }

    Node\* todelete = *prev*->next;

*prev*->next = todelete->next;

    if (todelete->next != nullptr)

    {

        todelete->next->prev = *prev*;

    }

    delete todelete;

    return *head*;

}

*void* deletelist(Node\*&*head*)

{

    Node\*value = *head*;

    while (value != nullptr)

    {

        Node\* next = value->next;

        delete value;

        value = next;

    }

*head* = nullptr;

}

*void* tofile(Node\**head*)

{

    ofstream file("From\_list.txt");

    if(!file)

    {

        cerr<<"Error while creating file"<<endl;

        exit(1);

    }

    Node\* current = *head*;

    while (current)

    {

        file<<current->value<<' ';

        current = current->next;

    }

}

*void* tolist(Node\*&*head*)

{

    ifstream read("From\_list.txt");

    if(!read)

    {

        cerr<<"Error while reading file"<<endl;

        exit(2);

    }

*int* k;

    while (read>>k)

    {

        addfront(*head*, k);

    }

}

*int* main()

{

    Node\* head = createlist(3);

    Node\* second = createlist(4);

    Node\* third = createlist(5);

    Node\* fourth = createlist(1);

    head->next = second;

    second->prev = head;

    second->next = third;

    third->prev = second;

    third->next = fourth;

    fourth->prev = third;

    cout<<"List:\n";

    printlist(head);

    cout<<endl;

    cout<<"List after deleting the third element:\n";

    head = deleteafter (head, third);

    printlist(head);

    cout<<endl;

    cout<<"List after adding new elements:\n";

    head = addfront (head, 8);

    addfront (head, 9);

    printlist(head);

    cout<<endl;

    cout<<"After deleting whole list:\n";

    tofile(head);

    deletelist(head);

    printlist(head);

    cout<<endl;

    cout<<"After returning list from file:\n";

    tolist(head);

    printlist (head);

    deletelist(head);

return 0;

}

Код виконує створення двонаправленого списку та його виведення. Також реалізовані функції для видалення елемента після елемента з вказаним номером та додавання К елементів на початок списку. Список записується у файл, після чого виконується його очищення, і при спробі виведення виводиться повідомлення "Список порожній". Далі список відновлюється з файлу, виводиться, а після цього знову очищується.

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

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-f58be80ea44012ff217510a34c1401e5ebe8ab96ced54c9e4db8bcca63641dea

Завдання №2 **Algotester Lab 5**

#include <iostream>

#include <algorithm>

using *namespace* std;

*int* main()

{

*int* N, M, x, y;

    cin>>N>>M;

    cin>>x>>y;

*int* A[N][M];

    for(*int* i=0; i<N; i++)

    {

        for(*int* j=0; j<M; j++)

        {

            A[i][j]=0;

        }

    }

*int* a=x-1;

*int* b=y-1;

*int* maxA = max(a, N-x)+max(b, M-y);

    for(*int* i=x-1; i<N; i++)

    {

        for(*int* j=y - 1; j<M; j++)

        {

            A[i][j]=maxA-abs(i-a)-abs(j-b);

        }

        for (*int* j=y-1; j>=0; j--)

        {

            A[i][j]=maxA-abs(a-i)-abs(b-j);

        }

    }

    for (*int* i=x-1; i>=0; i--)

    {

        for (*int* j=y-1; j<M; j++)

        {

            A[i][j]=maxA-abs(i-a)-abs(j-b);

        }

        for (*int* j=y-1; j>=0; j--)

        {

            A[i][j]=maxA-abs(a-i)-abs(b-j);

        }

    }

    for (*int* i=0; i<N; i++)

    {

        for (*int* j=0; j<M; j++)

        {

            cout<<A[i][j]<<' ';

        }

        cout<<"\n";

    }

    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.

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-f8fa5585ab230a33a8ccf24b8ac47e51890ebb801e573a700c13d3dfa8a18b33

Завдання №3 **Algotester Lab 7-8**

#include <iostream>

#include <string>

using *namespace* std;

*template* <*typename* T>

*class* Node

{

*public:*

    T element;

    Node<T> \*next;

    Node<T> \*prev;

    Node<T> \*head;

    Node<T> \*tail;

*int* size;

    Node(const T &*newval*) : element(*newval*) {}

    Node()

    {

        next = nullptr;

        prev = nullptr;

        head = nullptr;

        tail = nullptr;

        size = 0;

    };

*void* insert(*int* *index*, *int* *N*, T \**list*)

    {

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

        {

            return;

        }

        Node<T> \*newNode;

        Node<T> \*current;

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

        {

            newNode = new Node<T>(*list*[i]);

            newNode->prev = nullptr;

            newNode->next = nullptr;

            if (*index* == 0)

            {

                newNode->next = head;

                if (head != nullptr)

                {

                    head->prev = newNode;

                }

                head = newNode;

                if (tail == nullptr) {

                        tail = newNode;

                    }

            }

            else if (*index* == size)

            {

                newNode->prev = tail;

                if (tail != nullptr)

                {

                    tail->next = newNode;

                }

                tail = newNode;

                if (head == nullptr) {

                        head = newNode;

                    }

            }

            else

            {

                current = head;

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

                {

                    current = current->next;

                }

                newNode->next = current->next;

                newNode->prev = current;

                if (current->next != nullptr)

                {

                    current->next->prev = newNode;

                }

                current->next = newNode;

            }

            ++size;

        }

    }

*void* erase(*int* *index*, *int* *n*)

    {

        Node \*todelete = head;

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

        {

            todelete = todelete->next;

        }

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

        {

            Node<T> \*nextnode = todelete->next;

            if (todelete->prev != nullptr)

            {

                todelete->prev->next = todelete->next;

            }

            else

            {

                head = todelete->next;

            }

            if (todelete->next != nullptr)

            {

                todelete->next->prev = todelete->prev;

            }

            else

            {

                tail = todelete->prev;

            }

            delete todelete;

            todelete = nextnode;

            size--;

        }

    }

*int* listsize()

    {

        return size;

    }

*int* getelem(*int* *index*)

    {

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

        {

            return 0;

        }

        Node \*current = head;

        while (*index*--)

        {

            current = current->next;

        }

        return current->element;

    }

*void* set(*int* *index*, T *newval*)

    {

        Node<T> \*nchange = head;

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

        {

            nchange = nchange->next;

        }

        nchange->element = *newval*;

    }

*void* print()

    {

        Node<T> \*current = head;

        while (current)

        {

            cout << current->element << ' ';

            current = current->next;

        }

        cout << endl;

    }

};

*int* main()

{

    Node<*int*> task;

*int* Q;

    cin >> Q;

    while (Q--)

    {

        string theword;

        cin >> theword;

        if (theword == "insert")

        {

*int* index, N;

            cin >> index;

            cin >> N;

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

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

            {

                cin >> list[i];

            }

            task.insert(index, N, list);

            delete[] list;

        }

        else if (theword == "erase")

        {

*int* index, n;

            cin >> index;

            cin >> n;

            task.erase(index, n);

        }

        else if (theword == "get")

        {

*int* index;

            cin >> index;

            cout << task.getelem(index) << "\n";

        }

        else if (theword == "print")

        {

            task.print();

        }

        else if (theword == "size")

        {

            cout << task.listsize() << "\n";

        }

        else if (theword == "set")

        {

*int* index, newval;

            cin >> index;

            cin >> newval;

            task.set(index, newval);

        }

    }

    return 0;

}

Код виконує реалізацію структури даних "Двозв’язний список" з можливістю виконання операцій вставки, видалення, визначення розміру, отримання значення елемента за індексом, модифікації значення елемента та виводу списку на екран. Операції визначаються ідентифікаторами, такими як "insert", "erase", "size", "get", "set", "print". Програма робить роботу зі списком ефективно та надає можливість взаємодії з його елементами через зазначені ідентифікатори операцій. Для виведення списку на екран використовується перегрузка оператора "<<".

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

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-6de9beb0f018eb3492015525d08aba55404ca038f7a60e6171c84a78e75f8f92

Завдання №4 **Class Practice Work**

Task 1:

#include <iostream>

using *namespace* std;

*struct* Node

{

*int* value;

    Node \*next;

    Node \*prev;

    Node(*int* *val*) : value(*val*), next(nullptr), prev(nullptr) {}

};

*void* printList(Node \**head*)

{

    while (*head* != nullptr)

    {

        cout << *head*->value << ' ';

*head* = *head*->next;

    }

    cout << endl;

}

Node \*reverse(Node \**head*)

{

    Node \*prev = nullptr;

    Node \*next = nullptr;

    Node \*current = *head*;

    while (current != nullptr)

    {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

    return prev;

}

*int* main()

{

    cout << "Enter the amount of elements: \n";

*int* amount;

    cin >> amount;

    Node \*head = nullptr;

    Node \*tail = nullptr;

    cout << "List the elements: \n";

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

    {

*int* element;

        cin >> element;

        Node \*newelement = new Node(element);

        if (head != nullptr)

        {

            tail->next = newelement;

            tail = newelement;

        }

        else

        {

            head = newelement;

            tail = newelement;

        }

    }

    cout << "\nBefore the reverse\n";

    printList(head);

    head = reverse(head);

    cout << "After the reverse\n";

    printList(head);

    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.

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-6575647a3d1f5d8b6297d0851c4c3365c70bb4822236c14478b40f0fbb4c57c2

Task 2:

#include <iostream>

using *namespace* std;

*struct* Node

{

*int* value;

    Node \*next;

    Node \*prev;

    Node(*int* *val*) : value(*val*), next(nullptr), prev(nullptr) {}

};

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

{

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

    {

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

        {

            return false;

        }

*h1* = *h1*->next;

*h2* = *h2*->next;

    }

    if (*h1* == nullptr && *h2* == nullptr)

    {

        return true;

    }

    return false;

}

*int* main()

{

    Node \*head1 = new Node(4);

    head1->next = new Node(2);

    head1->next->next = new Node(9);

    Node \*head2 = new Node(4);

    head2->next = new Node(2);

    head2->next->next = new Node(9);

    head2->next->next->next = new Node(5);

    if(compare(head1, head2)==true)

    {

        cout<<"True";

    }

    else

    {

        cout<<"False";

    }

    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.

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-be7e7e4a17ef6dc9c423c7968b93d03699c14080e6504208b4cba5f3615eef79

Task 3:

#include <iostream>

using *namespace* std;

*struct* Node

{

*int* value;

    Node \*next;

    Node \*prev;

    Node(*int* *value*) : value(*value*), next(nullptr), prev(nullptr) {}

};

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

{

    Node \*result = nullptr;

    Node \*current = nullptr;

    Node \*prev = nullptr;

*int* number = 0;

*int* sum = 0;

    while (*n1* != nullptr || *n2* != nullptr || number > 0)

    {

        sum = number + (*n1* ? *n1*->value : 0) + (*n2* ? *n2*->value : 0);

        number = (sum >= 10) ? 1 : 0;

        sum = sum % 10;

        current = new Node(sum);

        if (result == nullptr)

        {

            result = current;

        }

        else

        {

            prev->next = current;

        }

        prev = current;

        if (*n1*)

        {

*n1* = *n1*->next;

        }

        if (*n2*)

        {

*n2* = *n2*->next;

        }

    }

    return result;

}

*int* main()

{

    Node \*head1 = new Node(4);

    head1->next = new Node(2);

    head1->next->next = new Node(6);

    head1->next->next->next = new Node(2);

    Node \*head2 = new Node(0);

    head2->next = new Node(4);

    head2->next->next = new Node(1);

    head2->next->next->next = new Node(3);

    Node \*result = add(head1, head2);

    while (result != nullptr)

    {

        cout << result->value << ' ';

        Node \*temp = result;

        result = result->next;

    }

    return 0;

}

Код виконує реалізацію функції, яка обчислює суму двох чисел, представлених у вигляді списків. У кожному вузлі списку зберігається одна цифра числа, з молодшого розряду у голові списка. Функція повертає новий список, представляючи суму вхідних чисел. Важливою деталлю є використання цифр від 0 до 9 для значень у списку, що враховує діапазон цифрових значень.

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

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-555f589e93fddb783fced9b6ef7a41c6d6d2dfdabb60de4eb784a25914e8584c

Task 4:

#include <iostream>

using *namespace* std;

*struct* TreeNode

{

*int* number;

    TreeNode \*left;

    TreeNode \*right;

    TreeNode(*int* *x*) : number(*x*), left(nullptr), right(nullptr) {}

};

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

{

    if (*root* == nullptr)

    {

        return nullptr;

    }

    if ((*root*->left == nullptr) && (*root* == nullptr))

    {

        return 0;

    }

    TreeNode \*mirrored = new TreeNode{*root*->number};

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

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

    cout << *root*->number << ' ';

    return mirrored;

}

*int* main()

{

    TreeNode\* root = new TreeNode(1);

    root->left = new TreeNode(2);

    root->left->left = new TreeNode(8);

    root->left->right = new TreeNode(3);

    root->right = new TreeNode(7);

    root->right->right = new TreeNode(2);

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

    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

Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-555f589e93fddb783fced9b6ef7a41c6d6d2dfdabb60de4eb784a25914e8584c

Task 5:

#include <iostream>

#include <stack>

using *namespace* std;

*struct* TreeNode

{

*int* number;

    TreeNode \*left;

    TreeNode \*right;

    TreeNode(*int* *value*) : number(value), left(nullptr), right(nullptr) {}

};

*void* tree\_sum(TreeNode \**root*)

{

    if (!root)

    {

        return;

    }

    stack<TreeNode \*> tree;

    TreeNode \*current = nullptr;

    while (root || !tree.empty())

    {

        while (root)

        {

            tree.push(root);

            root = root->left;

        }

        root = tree.top();

        if (!root->right || root->right == current)

        {

            if (root->left)

            {

                root->number += root->left->number;

            }

            if (root->right)

            {

                root->number += root->right->number;

            }

            tree.pop();

            current = root;

            root = nullptr;

        }

        else

        {

            root = root->right;

        }

    }

}

*int* main()

{

    TreeNode \*root = new TreeNode(9);

    root->right = new TreeNode(4);

    root->left = new TreeNode(2);

    root->left->right = new TreeNode(2);

    root->left->left = new TreeNode(8);

    root->right->left = new TreeNode(1);

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

    tree\_sum(root);

    cout << "   " << root->number << "\n";

    cout << "  /  \\"

         << "\n";

    cout << " " << root->right->number << "    " << root->left->number << "\n";

    cout << "/ \\   / \\"

         << "\n";

    cout << root->left->left->number << " " << root->left->right->number << "   " << root->right->left->number << " " << root->right->right->number << "\n";

    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.  
  
Pull request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961/files#diff-cf7d0c99377f67845c4776577cdfe7a99c6972f7d4c37c4de206e757c47da088

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

Завдання №1 **VNS 10**



Рисунок Виконання VNS 10

Час, затрачений на виконання завдання: 8 годин

Завдання №2 **Algotester lab 5v3**

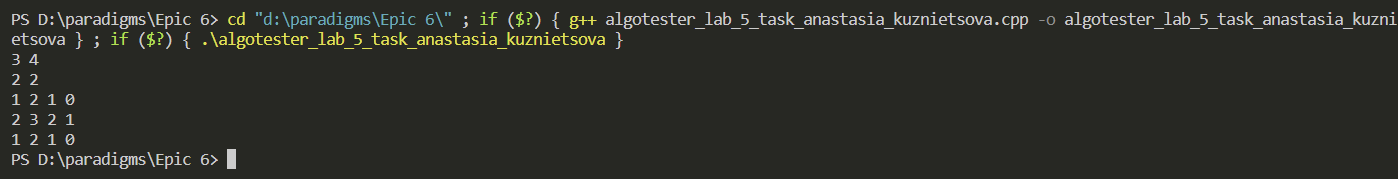


Рисунок Виконання Algotester lab 5v3

Так як обрані координати (2;2) то розмальовка мапи починається з клітинки 2 2, яка є найбільшим числом – 3. Далі всі менші за попереднє на 1.  
Час, затрачений на виконання завдання: 7 годин

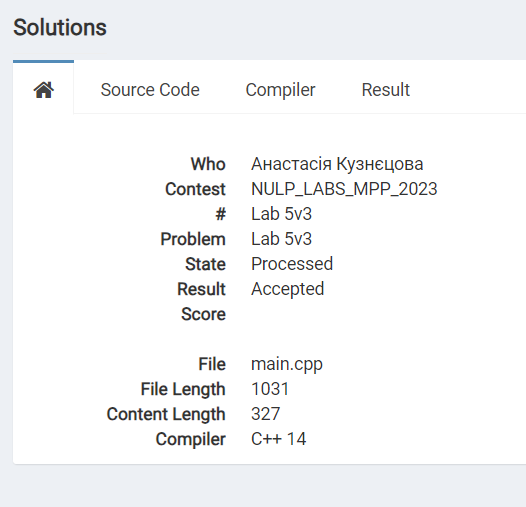


Рисунок Скріншот Algotester lab 5v3

Завдання №3 **Algotester lab 7-8v1**

****

Рисунок Виконання Algotester lab 7-8v1

Час, затрачений на виконання завдання: 10 годин

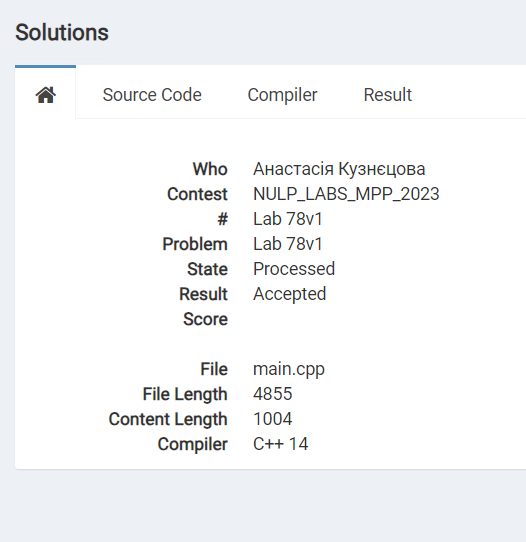


Рисунок Скріншот Algotester lab 7-8v1

Завдання №4 **Class Practice Work**

Task 1

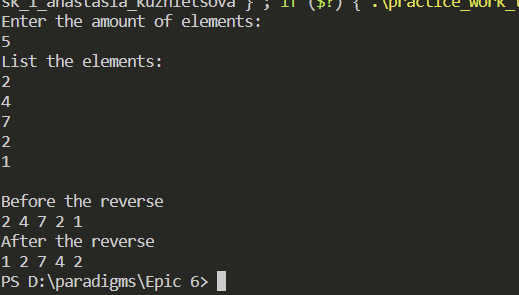


Рисунок Виконання Class Practice Work Task 1

Час, затрачений на виконання завдання: 2 години

Task 2

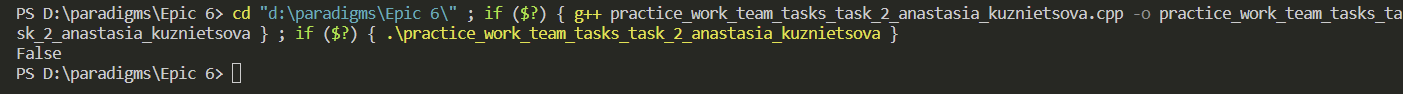


Рисунок Виконання Class Practice Work Task 2

Програма видає False бо два списки не є однаковими, так як у одного на 1 елемент більше.

Час, затрачений на виконання завдання: пів година

Task 3

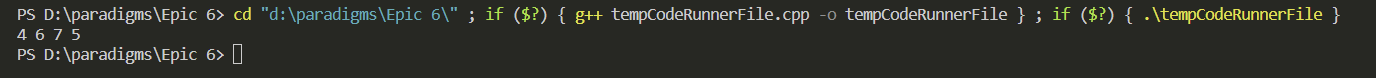


Рисунок Виконання Class Practice Work Task 3

Час, затрачений на виконання завдання: 1 година

Task 4

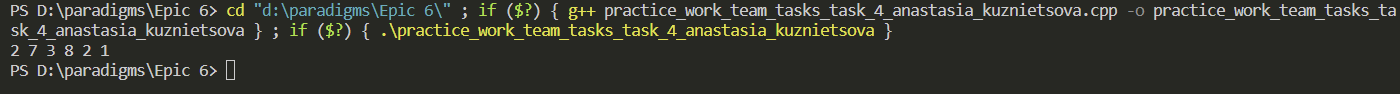


Рисунок Виконання Class Practice Work Task 4

Час, затрачений на виконання завдання: 2 години

Task 5

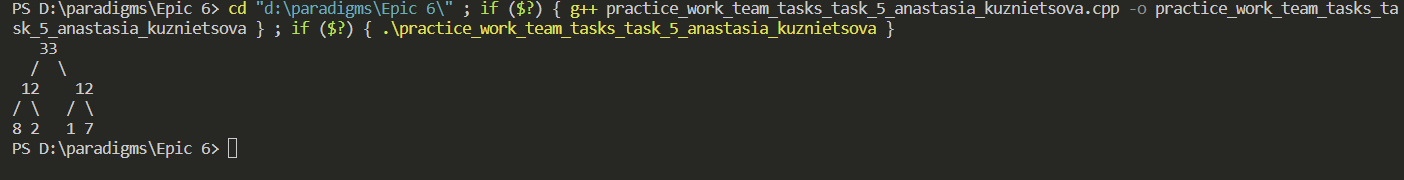


Рисунок Виконання Class Practice Work Task 5

Час, затрачений на виконання завдання: 3 години

# **Pull-request:** https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/961

# **Висновки:**

Було вивчено, як працювати з бінарним деревом та зв’язними списками. Було вивчено як рухатися по списку та дереву, змінювати їхню структуру.