Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**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

**Виконав:**

Студент групи ШІ-14

Шелеп Андрій Іванович

# **Тема роботи:**

Динамічні структури даних та алгоритми їх обробки. Однозв’язний та двозв’язний список. Стек, черга та дерево.

# **Мета роботи:**

Ознайомитись з динамічними структурами даних та попрацювати з базовими алгоритмами їх обробки. Реалізувати однозв’язний та двозв’язний списки та створити функції для їх обробки. Ознайомитися з деревами та реалізувати бінарне дерево. Ознайомитися з стеком та чергою

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Стек. Черга.
* Тема №2: Однозв’язний та двозв’язний списки.
* Тема №3: Дерева. Бінарне дерево.
* Тема №4: Класи. Шаблони класів.
* Тема №5: Алгоритми обробки динамічних структур даних.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Стек. Черга.
  + Джерела Інформації
    - Відео.

<https://www.youtube.com/watch?v=Yhw8NbjrSFA&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=ZYvYISxaNL0&t=462s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

* + - Стаття.

<https://acode.com.ua/urok-111-stek-i-kupa/>

<https://www.wikiwand.com/uk/%D0%94%D0%B2%D0%BE%D0%B1%D1%96%D1%87%D0%BD%D0%B0_%D1%87%D0%B5%D1%80%D0%B3%D0%B0>

<https://www.bestprog.net/uk/2019/09/26/c-queue-general-concepts-ways-to-implement-the-queue-implementing-a-queue-as-a-dynamic-array-ua/>

* + Що опрацьовано:
    - Стек і черга як динамічні структури даних, різниця між ними
    - Двобічна та пріоритетна черга
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 11.12.2023
* Тема №2: Однозв’язний та двозв’язний списки.
  + Джерела Інформації:
    - Відео.

<https://www.youtube.com/watch?v=-25REjF_atI&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=QLzu2-_QFoE&t=1328s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

* + - Стаття.

<https://www.wikiwand.com/uk/%D0%97%D0%B2'%D1%8F%D0%B7%D0%B0%D0%BD%D0%B8%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA>

* + Що опрацьовано:
    - Зв’язані списки. Різниця між однозв’язним та двозв’язними списками
  + Статус: Ознайомлений
  + Початок опрацювання теми: 02.12.2023
  + Звершення опрацювання теми: 11.12.2023
* Тема №3: Дерева. Бінарне дерево.
  + Джерела Інформації:
    - Відео.

<https://www.youtube.com/watch?v=o5XvtIPr1Ns&t=437s&ab_channel=%D0%86%D1%80%D0%B8%D0%BD%D0%B0%D0%9E%D0%BB%D1%96%D0%B9%D0%BD%D0%B8%D0%BA>

* + - Стаття.

<https://www.wikidata.uk-ua.nina.az/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85).html>

* + Що опрацьовано:
    - Дерева як структура даних
    - Бінарні дерева
    - Обходи дерева
  + Статус: Ознайомлений
  + Початок опрацювання теми: 03.12.2023
  + Звершення опрацювання теми: 12.12.2023
* Тема №4 Класи. Шаблони класів.
  + Джерела Інформації:
    - Відео.

<https://www.youtube.com/watch?v=ZbsukxxV5_Q&t=1282s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=a5jigk0Tw50&t=282s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=0YWOW8BezI4&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

* + - Стаття.

<https://acode.com.ua/uroki-po-cpp/>

* + Що опрацьовано:
    - Класи. Шаблони класів
    - Специфікатори доступу (private, public)
    - Конструктори та деструктори
    - Дружні функції
  + Статус: Ознайомлений
  + Початок опрацювання теми: 04.12.2023
  + Звершення опрацювання теми: 12.12.2023
* Тема №4 Алгоритми обробки динамічних структур даних.
  + Джерела Інформації:
    - Відео.
    - Стаття.

<https://www.bestprog.net/uk/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ua/>

<https://www.bestprog.net/uk/2022/02/13/c-an-example-of-the-implementation-of-a-linear-singly-linked-list-ua/>

* + Що опрацьовано:
    - Алгоритми обробки для динамічних структур даних (вставка елемента у чергу, списки, дерево, стек, видалення елемента…)
    - Бінарний пошук
  + Статус: Ознайомлений
  + Початок опрацювання теми: 03.12.2023
  + Звершення опрацювання теми: 12.12.2023

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант 14
* Записи в лінійному списку містять ключове поле типу \*char (рядок

символів). Сформувати двонаправлений список. Знищити з нього К

елементів із зазначеними номерами. Додати К елементів із зазначеними

номерами.

* Важливі деталі для врахування в імплементації програми

Використовую дві структури (для вузла (зберігає ключ з даними, посилання на наступний і попередній елементи) і для самого листа (зберігає розмір, посилання на перший і на останній елементи))

Завдання №2 Algotester Lab 5

* Варіант 2
* Деталі завдання

В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина - M Всередині печери є пустота, пісок та каміння. Пустота позначається буквою О, пісок S і каміння X;

Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння.

Ваше завдання сказати як буде виглядати печера після землетрусу.

* Важливі деталі для врахування в імплементації програми

Використовую двовимірний масив символів для відображення печери

Завдання №3 Algotester Lab 7-8

* Варіант 2
* Деталі завдання

Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".  
Ви отримаєте Q

запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.  
  
Вам будуть поступати запити такого типу:

**Вставка**:  
Ідентифікатор - insert  
Ви отримуєте ціле число index елемента, на місце якого робити вставку.  
Після цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити.  
У третьому рядку N цілих чисел - масив, який треба вставити на позицію index

**Видалення**:  
Ідентифікатор - erase  
Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та N - кількість елементів, яку треба видалити.

**Визначення розміру**:  
Ідентифікатор - size  
Ви не отримуєте аргументів.  
Ви виводите кількість елементів у динамічному масиві.

**Визначення кількості зарезервованої пам’яті**:  
Ідентифікатор - capacity  
Ви не отримуєте аргументів.  
Ви виводите кількість зарезервованої пам’яті у динамічному масиві.  
Ваша реалізація динамічного масиву має мати фактор росту ([Growth factor](https://en.wikipedia.org/wiki/Dynamic_array#Growth_factor)) рівний 2.

**Отримання значення** i-го елементу  
Ідентифікатор - get  
Ви отримуєте ціле число - index, індекс елемента.  
Ви виводите значення елемента за індексом.

**Модифікація значення** i-го елементу  
Ідентифікатор - set  
Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.

**Вивід динамічного масиву на екран**Ідентифікатор - print  
Ви не отримуєте аргументів.  
Ви виводите усі елементи динамічного масиву через пробіл.  
Реалізувати використовуючи перегрузку оператора <<

* Важливі деталі для врахування в імплементації програми

Реалізовую через шаблон класу

Завдання №4 Class Practice Work

* Без варіанту
* Деталі завдання

Задача №1 - Реверс списку (Reverse list). Задача №2 - Порівняння списків. Задача №3 – Додавання великих чисел. Задача №4 - Віддзеркалення дерева. Задача №5 - Записати кожному батьківському вузлу суму підвузлів

* Важливі деталі для врахування в імплементації програми

Здійснюю обхід дерева двома способами використовуючи рекурсію

Завдання №4 Self Practice Work

* Без варіанту
* Деталі завдання

Реалізація черги

* Важливі деталі для врахування в імплементації програми

Використовую структуру, яка зберігає масив і кількість елементів. Створюю функції для додавання до черги і видалення з черги.

Завдання №5 Self Practice Work

* Задача на алготестері - 0607 Свічкі на Halloween
* Деталі завдання

У селі, назву якого ми не говоримо, весела дітвора готує гарбузи та свічки для надзвичайно приємного ритуалу створення страшилок.

Складними математичними розрахунками діти визначили, що їм потрібно підготувати n страшилок. Кожна страшилка — це зовнішня частина гарбуза та свічка замість його внутрішньої частини. Дітям удалося знайти n однакових гарбузів, а от зі свічками не все так добре. Перед самим святом виявилось, що дітвора має m свічок. Кожна свічка має свою довжину (деякими з них уже користувалися за часів енергетичної кризи).

Малеча хоче, щоб свічки в кожному з n гарбузів були однакових розмірів. Для цього вони можуть розрізати свічки на дрібніші як їм заманеться, але ліпити їх докупи, на жаль, не можна. Окрім того, дітям хотілося б, щоб свічки в гарбузах були якомога довшими.

Вам необхідно визначити максимально можливу довжину свічки в гарбузі. Зауважте, що використовувати всі m свічок необов’язково.

* Важливі деталі для врахування в імплементації програми

Використовую принцип бінарного пошуку

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

Програма №1 VNS Lab 10

* Планований час на реалізацію: 2,5 год
* Важливі деталі для врахування в імплементації

Використовую дві структури (для вузла (зберігає ключ з даними, посилання на наступний і попередній елементи) і для самого листа (зберігає розмір, посилання на перший і на останній елементи))

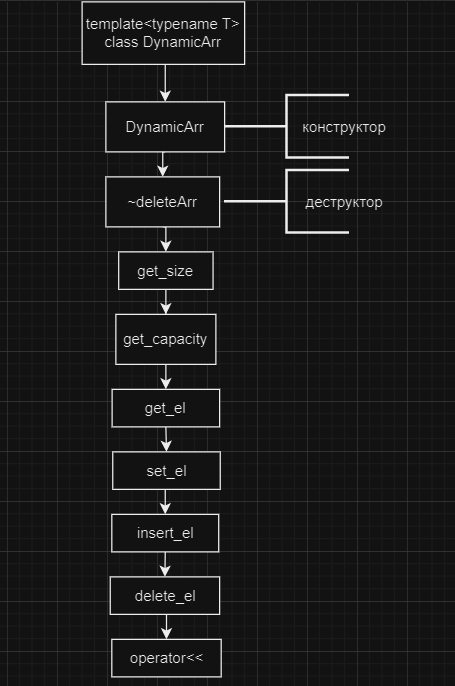
Програма №2 Algotester Lab 5

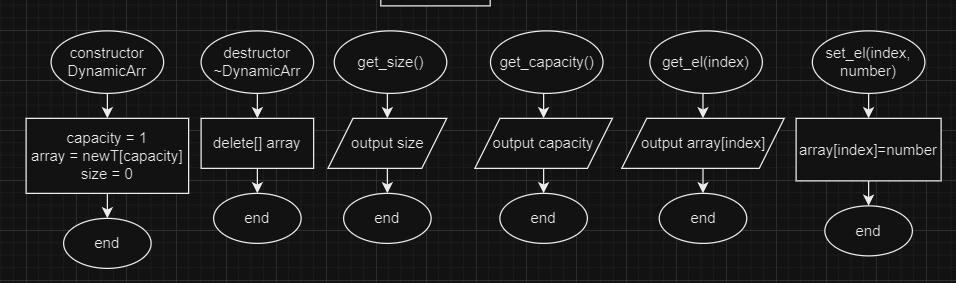
* Планований час на реалізацію: 1 год
* Важливі деталі для врахування в імплементації

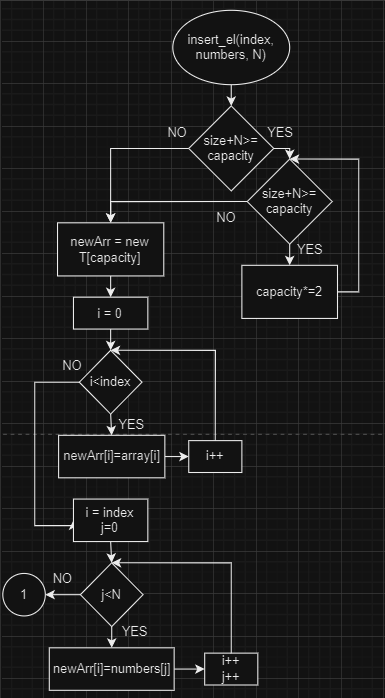
Використовую двовимірний масив символів для відображення печери

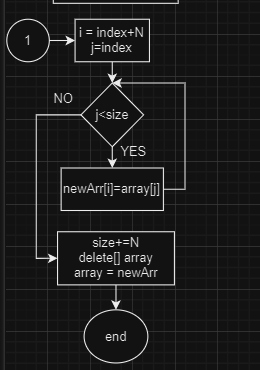
Програма №3 Algotester Lab 7-8

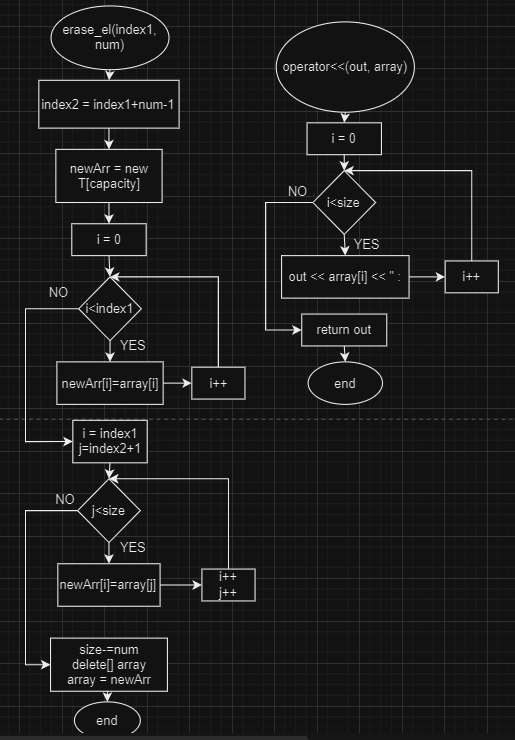
* Блок-схема

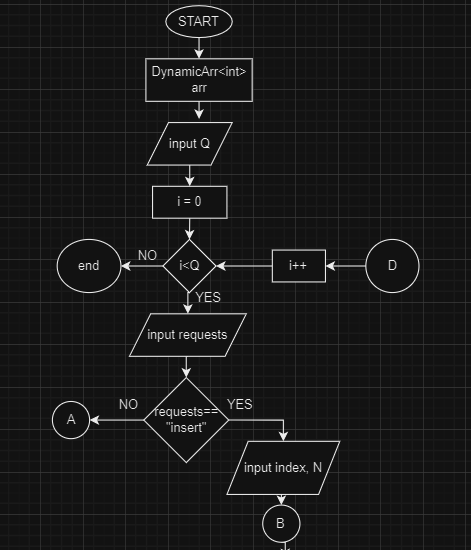


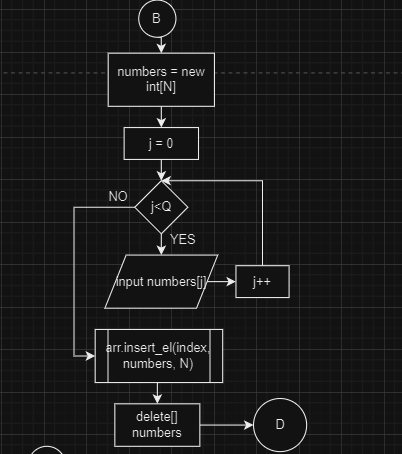


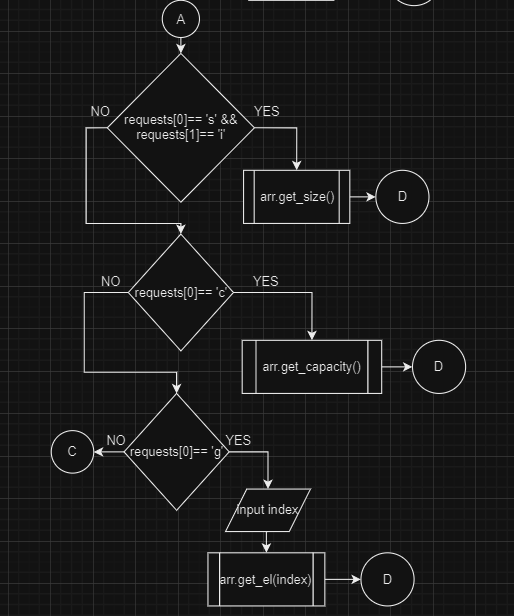


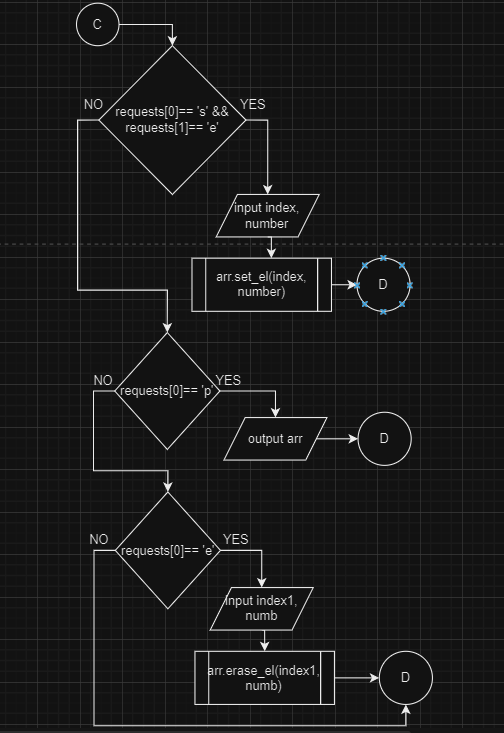












* Планований час на реалізацію: 1,5 год
* Важливі деталі для врахування в імплементації

Реалізовую через шаблон класу

Програма №4 Self Practice Work

* Планований час на реалізацію: 2,5 год
* Важливі деталі для врахування в імплементації

Здійснюю обхід дерева двома способами використовуючи рекурсію

Програма №4 Self Practice Work

* Планований час на реалізацію: 30 хв
* Важливі деталі для врахування в імплементації

Програма №5 Self Practice Work

* Планований час на реалізацію: 1год
* Важливі деталі для врахування в імплементації

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

Використано налаштування з попередніх лабораторних робіт

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

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/796>

Завдання №1 VNS Lab 10

#include <iostream>

#include <cstring>

#include <fstream>

using *namespace* std;

*struct* *Node*{

*char* key[10];

*Node*\* next;

*Node*\* prev;

//Node(char\* data): key(data), next(nullptr), prev(nullptr){}

};

*struct* *dblLinkedList*{

*Node*\* head;

*Node*\* tail;

*int* size;

};

*dblLinkedList*\* createList(){

*dblLinkedList*\* List = new *dblLinkedList*;

List->head=nullptr;

List->tail=nullptr;

List->size=0;

return List;

}

*void* destroyList(*dblLinkedList*\* *List*){

*Node*\* data = *List*->head;

while(data!=nullptr){

*Node*\* next\_data = data->next;

delete data;

data=next\_data;

*List*->size--;

}

*List*->head=nullptr;

*List*->tail=nullptr;

}

*void* printList(*dblLinkedList*\* *List*){

*Node*\* data = *List*->head;

if(data==nullptr){

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

return;

} else{

while(data!=nullptr){

cout << data->key << " ";

data=data->next;

}

}

}

*void* addEl(*dblLinkedList*\* *List*, *int* *index*, const *char*\* *str*){

*index*--;

if(*index* > *List*->size || *index*<0){

cout << "wrong index:(" << endl;

return;

}

*Node*\* newNode = new *Node*;

strcpy(newNode->key, *str*);

if(*List*->head==nullptr){

*List*->head=newNode;

newNode->next=nullptr;

newNode->prev=nullptr;

*List*->tail=newNode;

*List*->size++;

} else{

*Node*\* current = *List*->head;

*int* i = 0;

while(current!=nullptr && i<*index*){

current=current->next;

i++;

}

if(current==*List*->head){

newNode->next=*List*->head;

*List*->head->prev=newNode;

*List*->head=newNode;

newNode->prev=nullptr;

} else if(current==nullptr){

newNode->next=nullptr;

newNode->prev=*List*->tail;

*List*->tail->next=newNode;

*List*->tail=newNode;

} else{

newNode->next=current;

newNode->prev=current->prev;

current->prev->next=newNode;

current->prev=newNode;

}

*List*->size++;

}

}

*void* deleteEl(*dblLinkedList*\* *List*, *int* *index*){

*index*--;

if(*List*->size==0){

cout << "it is nothing to delete" << endl;

return;

}

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

cout << "wrong index" << endl;

return;

}

*Node*\* current = *List*->head;

*int* i = 0;

while(current!=nullptr && i<*index*){

current=current->next;

i++;

}

if(current->prev!=nullptr){

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

}

if(current->next!=nullptr){

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

}

if(current->prev==nullptr){

*List*->head=current->next;

}

if(current->next==nullptr){

*List*->tail=current->prev;

}

*List*->size--;

delete current;

}

*void* write\_to\_file(*dblLinkedList*\* *List*, const *char*\* *filename*){

*ofstream* oFile(*filename*);

if(oFile.is\_open()){

*Node*\* data = *List*->head;

while(data!=nullptr){

oFile << data->key << endl;

data=data->next;

}

oFile.close();

} else{

cout << "error opening file" << endl;

}

}

*dblLinkedList*\* restore(*dblLinkedList*\* *List*, const *char*\* *filename*){

*ifstream* iFile(*filename*);

if(iFile.is\_open()){

*char* str[20];

while(!iFile.eof()){

*Node*\* elem = new *Node*;

iFile.getline(str, sizeof(str));

//elem->key=new char [strlen(str)];

strcpy(elem->key, str);

if(*List*->head==nullptr){

*List*->head=elem;

*List*->tail=elem;

elem->next=nullptr;

elem->prev=nullptr;

*List*->size++;

} else{

elem->prev=*List*->tail;

elem->next=nullptr;

elem->prev->next=elem;

*List*->tail=elem;

*List*->size++;

}

}

iFile.close();

} else{

cout << "error opening files" << endl;

}

return *List*;

}

*int* main(){

*dblLinkedList*\* list = createList();

printList(list);

addEl(list, 1, "a");

addEl(list, 2, "b");

addEl(list, 3, "c");

addEl(list, 4, "d");

addEl(list, 5, "e");

cout << "formed list: ";

printList(list);

*int* K;

cout << endl << "enter the number of elements: " << endl;

cin >> K;

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

*char* word[10];

*int* index;

cout << "enter the short word: ";

cin >> word;

cout << "enter the index of new element: ";

cin >> index;

addEl(list, index, word);

}

cout << "updated list: ";

printList(list);

cout << endl << "enter the number of elements to delete: ";

cin >> K;

if(K>list->size){

cout << "there are not many elements in the list" << endl;

} else{

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

*int* index;

cout << endl << "enter the index of element to delete: ";

cin >> index;

deleteEl(list, index);

cout << "after deleting: ";

printList(list);

}

}

write\_to\_file(list, "dbllinkedlist.txt");

destroyList(list);

cout << endl << "after destroying: ";

printList(list);

list = restore(list, "dbllinkedlist.txt");

cout << "after restoring: ";

printList(list);

return 0;

}

*Блок №1. Код до програми №1 VNS Lab 10*

Завдання №2 Algotester Lab 5

#include <iostream>

using *namespace* std;

*int* main(){

*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];

}

}

for(*int* i = 0; i < M; i++){

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

if(cave[j][i]=='S'){

*int* k = j+1;

while(k<N && cave[k][i]!='X' && cave[k][i]!='S'){

k++;

}

k--;

if(k!=j){

*char* tmp = cave[j][i];

cave[j][i]=cave[k][i];

cave[k][i]=tmp;

}

}

}

}

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

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

cout << cave[i][j];

}

cout << endl;

}

return 0;

}

*Блок №2. Код до програми №2 Algotester Lab 5*

Завдання №3 Algotester Lab 7-8

#include <iostream>

#include <string>

using *namespace* std;

*template* <*typename* *T*>

*class* *DynamicArr*{

//private:

*T*\* array;

*int* size;

*int* capacity;

*public:*

//конструктор

DynamicArr(){

capacity=1;

array = new *T*[capacity];

size=0;

}

//деструктор

~DynamicArr(){

delete[] array;

}

*void* get\_size(){

cout << size << endl;

}

*void* get\_capacity(){

cout << capacity << endl;

}

*void* get\_el(*int* *index*){

cout << array[*index*] << endl;

}

*void* set\_el(*int* *index*, *T* *number*){

array[*index*]=*number*;

}

/\*void printArr(){

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

cout << array[i] << " ";

}

cout << endl;

}\*/

*void* insert\_el(*int* *index*, *T*\* *numbers*, *int* *N*){

if(size+*N*>=capacity){

while(size+*N*>=capacity){

capacity\*=2;

}

*T*\* newArray = new *T*[capacity];

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

newArray[i]=array[i];

}

delete[] array;

array = newArray;

}

*T*\* newArray = new *T*[capacity];

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

newArray[i]=array[i];

}

for(*int* i = *index*, j=0; j<*N* ; i++, j++){

newArray[i]=*numbers*[j];

}

for(*int* i = *index*+*N*, j = *index*; j < size; i++, j++){

newArray[i]=array[j];

}

size+=*N*;

delete[] array;

array=newArray;

}

*void* erase\_el(*int* *index1*, *int* *num*){

*int* index2 = *index1*+*num*-1;

*T*\* newArray = new *T*[capacity];

for(*int* i = 0; i<*index1*; i++){

newArray[i]=array[i];

}

for(*int* i = *index1*, j=index2+1; j < size; i++, j++){

newArray[i]=array[j];

}

size=size-*num*;

delete[] array;

array=newArray;

}

//operator T[]();

friend *ostream*& operator<<(*ostream*& *out*, *DynamicArr*<*T*>& *array*){

for(*int* i = 0; i < *array*.size; i++){

*out* << *array*.array[i] << " ";

}

return *out*;

}

};

*int* main(){

*DynamicArr*<*int*> arr;

*int* Q;

cin >> Q;

*string* requests;

for(*int* i = 0; i < Q; i++){

cin >> requests;

if(requests=="insert"){

*int* index, N;

cin >> index >> N;

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

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

cin >> numbers[i];

}

arr.insert\_el(index, numbers, N);

delete[] numbers;

} else if(requests[0]=='s' && requests[1]=='i'){

arr.get\_size();

} else if(requests[0]=='c'){

arr.get\_capacity();

} else if(requests[0]=='g'){

*int* index;

cin >> index;

arr.get\_el(index);

} else if(requests[0]=='s' && requests[1]=='e'){

*int* index, number;

cin >> index >> number;

arr.set\_el(index, number);

} else if(requests[0]=='p'){

cout << arr;

} else if(requests[0]=='e'){

*int* index1, num;

cin >> index1 >> num;

arr.erase\_el(index1, num);

}

}

return 0;

}

*Блок №3. Код до програми №3 Algotester Lab 7-8*

Завдання №4 Class Practice Work

#include <iostream>

using *namespace* std;

*struct* *Node*{

*int* data;

*Node*\* next;

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

};

*Node*\* createList(*int* *num*){

*Node*\* head = nullptr;

*Node*\* last = nullptr;

*int* val;

while(*num*!=0){

val = *num*%10;

*Node*\* newNode = new *Node*(val);

if(head==nullptr){

head=newNode;

last=newNode;

} else{

last->next=newNode;

last=newNode;

}

*num*/=10;

}

return head;

}

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

*Node*\* prev = nullptr;

*Node*\* current = *head*;

*Node*\* next\_el = nullptr;

while(current!=nullptr){

next\_el = current->next;

current->next = prev;

prev=current;

current=next\_el;

}

return prev;

}

*bool* compare(*Node*\* *head1*, *Node*\* *head2*){

*Node*\* node1 = *head1*;

*Node*\* node2 = *head2*;

while(node1 != nullptr && node2 != nullptr){

if(node1->data!=node2->data || (node1->next==nullptr && node2->next!=nullptr) || (node1->next!=nullptr && node2->next==nullptr)){

return false;

}

node1=node1->next;

node2=node2->next;

}

return true;

}

*void* print(*Node*\* *head*){

if(*head*==nullptr){

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

} else{

*Node*\* current = *head*;

while(current!=nullptr){

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

current = current->next;

}

}

}

*int* main(){

*Node*\* list = createList(256);

cout << "list: ";

print(list);

list = reverse(list);

cout << endl << "reversed list: ";

print(list);

*Node*\* list2 = createList(652);

cout << endl << "list2: ";

print(list2);

*bool* equal = compare(list, list2);

cout << endl;

if(equal == true){

cout << "the lists are equal" << endl;

} else{

cout << "the lists are not equal" << endl;

}

return 0;

}

*Блок №4. Код до програми №4.1, 4.2 Class Practice Work*

#include <iostream>

using *namespace* std;

*struct* *Node*{

*int* key;

*Node*\* next;

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

};

*Node*\* createList(*int* *num*){

*Node*\* head = nullptr;

*Node*\* last = nullptr;

*int* val;

while(*num*!=0){

val = *num*%10;

*Node*\* newNode = new *Node*(val);

if(head==nullptr){

head=newNode;

last=newNode;

} else{

last->next=newNode;

last=newNode;

}

*num*/=10;

}

return head;

}

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

*Node*\* node1 = *n1*;

*Node*\* node2 = *n2*;

*Node*\* head = nullptr;

*Node*\* last = nullptr;

*int* dec = 0;

while(node1!=nullptr || node2!=nullptr){

*int* number;

*int* num1, num2;

if(node1!=nullptr){

num1 = node1->key;

node1=node1->next;

} else {

num1 = 0;

}

if(node2!=nullptr){

num2 = node2->key;

node2=node2->next;

} else {

num2 = 0;

}

number = num1 + num2 + dec;

dec = number/10;

*Node*\* newNode = new *Node*(number%10);

if(head==nullptr){

head = newNode;

last = newNode;

} else {

last->next=newNode;

last = newNode;

}

}

if(dec==1){

*Node*\* newNode = new *Node*(dec);

last->next=newNode;

last=newNode;

}

return head;

}

*void* showReverseList(*Node*\* *head*){

*Node*\* prev = nullptr;

*Node*\* current = *head*;

*Node*\* next\_el = nullptr;

while(current!=nullptr){

next\_el = current->next;

current->next = prev;

prev=current;

current=next\_el;

}

*Node*\* node = prev;

while(node!=nullptr){

cout << node->key;

node=node->next;

}

}

*void* showList(*Node*\* *head*){

*Node*\* node = *head*;

while(node!=nullptr){

cout << node->key << " ";

node=node->next;

}

}

*int* main(){

*Node*\* list1 = createList(375);

*Node*\* list2 = createList(1858);

cout << "list1: ";

showList(list1);

cout << endl;

cout << "list2: ";

showList(list2);

*Node*\* result=add(list1, list2);

cout << endl;

cout << "new list: ";

showList(result);

cout << endl;

cout << "result: ";

showReverseList(result);

return 0;

}

*Блок №5. Код до програми №4.3 Class Practice Work*

#include <iostream>

#include <queue>

using *namespace* std;

*struct* *Node*{

*int* key;

*Node*\* left;

*Node*\* right;

Node(*int* *val*): key(*val*), left(nullptr), right(nullptr){}

};

*Node*\* insertEl(*Node*\* *root*, *int* *number*){

*Node*\* newNode = new *Node*(*number*);

if(*root*==nullptr){

*root* = newNode;

return *root*;

}

if(*number*>*root*->key){

*root*->right=insertEl(*root*->right, *number*);

}

if(*number*<*root*->key){

*root*->left=insertEl(*root*->left, *number*);

}

return *root*;

}

*Node*\* createTree(*int* *arr*[], *int* *size*){

*Node*\* root = nullptr;

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

root = insertEl(root, *arr*[i]);

}

return root;

}

*void* showTree(*Node*\* *root*){

*Node*\* current = *root*;

if(current!=nullptr){

showTree(current->left);

cout << current->key << " ";

showTree(current->right);

}

}

*Node*\* createMirrorFlip(*Node*\* *root*){

if(*root*!=nullptr){

*Node*\* tmp = *root*->left;

*root*->left=*root*->right;

*root*->right=tmp;

createMirrorFlip(*root*->left);

createMirrorFlip(*root*->right);

}

return *root*;

}

*void* showTree\_w(*Node*\* *root*){

if(*root*==nullptr){

cout << "the tree is empty" << endl;

return;

}

*queue*<*Node*\*> tree;

tree.push(*root*);

while(!tree.empty()){

*Node*\* current = tree.front();

tree.pop();

cout << current->key << " ";

if(current->left!=nullptr){

tree.push(current->left);

}

if(current->right!=nullptr){

tree.push(current->right);

}

}

}

*void* sumTree(*Node*\* *root*){

if(*root*==nullptr){

return;

}

sumTree(*root*->left);

sumTree(*root*->right);

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

*root*->key+=0;

} else{

*int* sum = 0;

if(*root*->left!=nullptr){

sum+=*root*->left->key;

}

if(*root*->right!=nullptr){

sum+=*root*->right->key;

}

*root*->key=sum;

}

}

*int* main(){

*int* arr[] = {15, 9, 25, 5, 12, 20, 30, 3, 7, 10, 13, 16, 22, 27, 31};

*int* size = sizeof(arr)/sizeof(*int*);

*Node*\* tree = createTree(arr, size);

cout << "treee: ";

showTree(tree);

cout << endl;

*Node*\* newtree = createMirrorFlip(tree);

cout << "mirror tree: ";

showTree(newtree);

cout << endl;

*int* arr2[] = {7, 4, 10, 2, 5, 8, 12, 1, 3};

*int* size2 = sizeof(arr2)/sizeof(*int*);

*Node*\* tree2 = createTree(arr2, size2);

cout << "tree: ";

showTree\_w(tree2);

sumTree(tree2);

cout << endl << "sum tree: ";

showTree\_w(tree2);

return 0;

}

*Блок №6. Код до програми №4.4, 4.5 Class Practice Work*

Завдання №5 Self Practice Work

#include <iostream>

using *namespace* std;

*struct* *Queue*{

*int*\* queue;

*int* count;

};

*void* createQueue(*Queue*& *q*){

*q*.count = 0;

*q*.queue = new *int*[*q*.count];

}

*void* Qpush(*Queue*& *q*, *int* *number*){

*int* new\_size = *q*.count + 1;

*int*\* arr = new *int*[new\_size];

for(*int* i = 0; i < *q*.count; i++){

arr[i]=*q*.queue[i];

}

arr[*q*.count]=*number*;

delete[] *q*.queue;

*q*.count = new\_size;

*q*.queue = arr;

}

*void* Qpop(*Queue*& *q*){

if(*q*.count==0){

cout << "queue is empty" << endl;

return;

}

*int* new\_size = *q*.count - 1;

*int*\* arr = new *int* [new\_size];

for(*int* i = 0; i<*q*.count-1; i++){

arr[i]=*q*.queue[i+1];

}

delete[] *q*.queue;

*q*.count=new\_size;

*q*.queue = arr;

}

*void* showQueue(*Queue*& *q*){

if(*q*.count == 0){

cout << "the queue is empty" << endl;

} else {

for(*int* i = 0; i < *q*.count; i++){

cout << *q*.queue[i] << " ";

}

cout << endl;

}

}

*void* deleteQueue(*Queue*& *q*){

delete[] *q*.queue;

*q*.count=0;

}

*int* main(){

*Queue* q;

createQueue(q);

*int* req;

cout << "enter the count of requests:" << endl;

cin >> req;

for(*int* i = 0; i < req; i++){

*string* str;

cout << "enter the name of operation (push, pop, show): ";

cin >> str;

if(str == "push"){

*int* number;

cout << "enter the number: " << endl;

cin >> number;

Qpush(q, number);

} else if(str == "pop"){

Qpop(q);

} else if(str == "show"){

showQueue(q);

}

}

return 0;

}

*Блок №7. Код до програми №5 Class Practice Work*

Завдання №6 Self Practice Work

#include <iostream>

#include <iomanip>

using *namespace* std;

*int* main(){

*int* n, m, l[100];

cin >> n >> m;

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

cin >> l[i];

}

*int* max = l[0];

for(*int* i = 1; i < m; i++){

if(l[i]>max)

max = l[i];

}

*double* left = 0, right = max;

*double* length, try\_length = max;

*int* number = 0;

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

*double* current = l[i];

while(current-try\_length>=0){

current-=try\_length;

number++;

}

}

if(number>=n){

length = try\_length;

} else{

number = 0;

try\_length = (left + right)/2;

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

*double* current = l[i];

while(current-try\_length>=0){

current-=try\_length;

number++;

}

}

while(right - left > 0.00000001){

if(number >= n){

left = try\_length;

length = try\_length;

} else {

right = try\_length;

}

number = 0;

try\_length = (left + right)/2;

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

*double* current = l[i];

while(current-try\_length>=0){

current-=try\_length;

number++;

}

}

}

}

cout << setprecision(8) << length << endl;

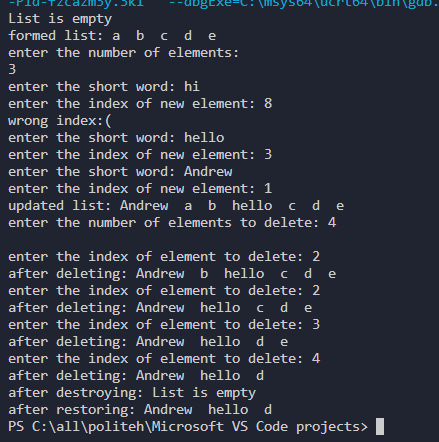
return 0;

}

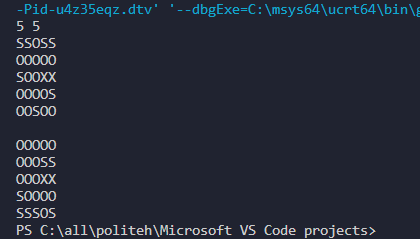
*Блок №8. Код до програми №6 Class Practice Work*

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

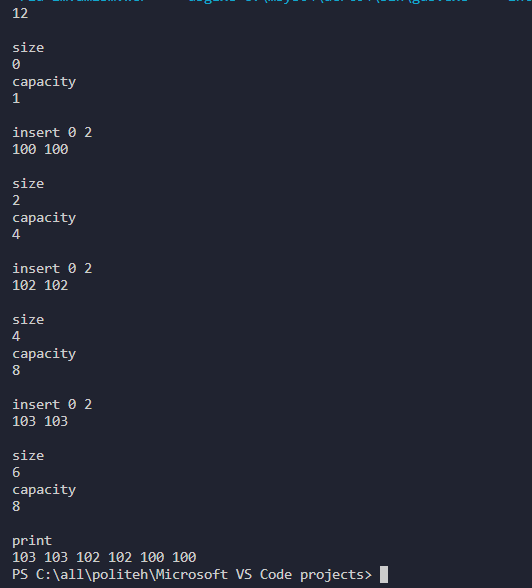
Завдання №1 VNS Lab 10

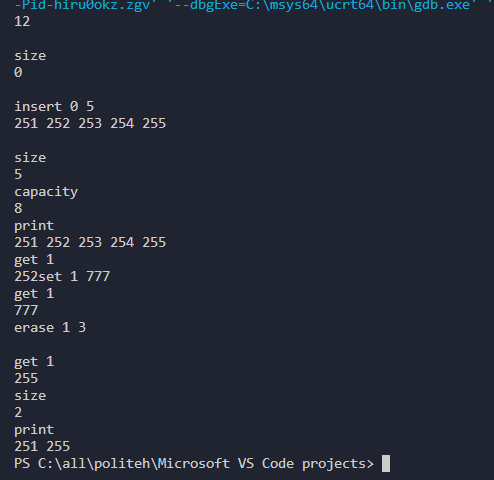


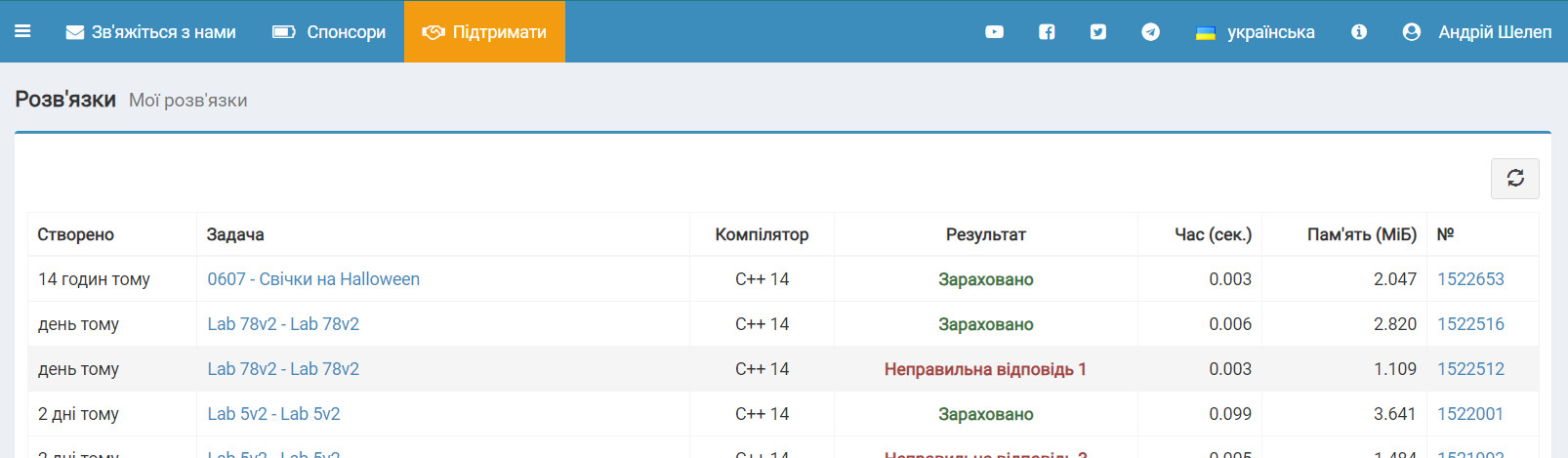
Завдання №2 Algotester Lab 5



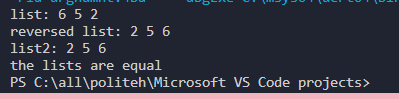
Завдання №3 Algotester Lab 7-8

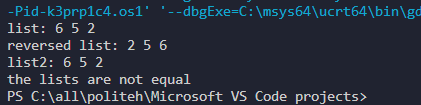


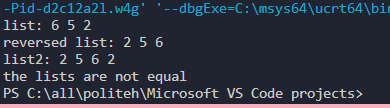




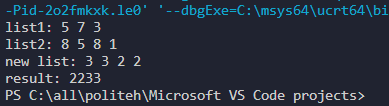
Завдання №4.1, 4.2 Class Practice Work





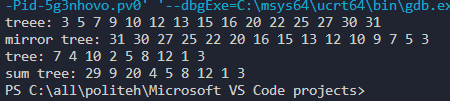


Завдання №4.3 Class Practice Work



375+1858=2233

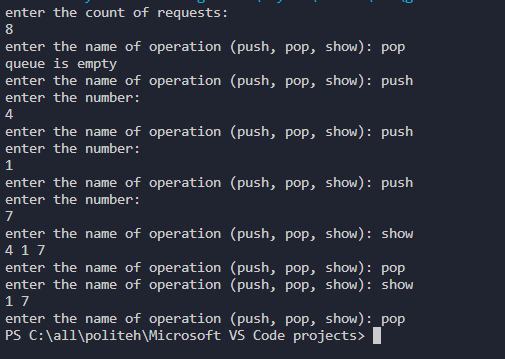
Завдання №4.4, 4.5 Class Practice Work



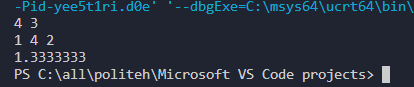
Обхід в глибину у внутрішньому порядку показує, що дерево віддзеркалилось

Обхід в ширину відображає значення елементів дерева по рівнях

Завдання №5 Self Practice Work



Завдання №6 Self Practice Work



Час затрачений на виконання завдання

## **6. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* Скрін з 3-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло (опційно)
* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи

# **Висновки:**

Опрацьовано динамічні структури даних та реалізовано деякі з них. Ознайомився з деревами, чергою, двозв’язним і однозв’язним списками та іншими структурами даних. Використано динамічні структури даних у виконанні завдань з лабораторної роботи.