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-12

Яремчук Павло Марекович

# **Тема роботи:**

Динамічні структури у програмуванні: динамічний масив, список, бінарне дерево.

# **Мета роботи:**

Опрацювати теми, названі вище, і виконати лабораторні завдання по ним.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні масиви
* Тема №2: Списки
* Тема №3: Бінарні дерева

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні масиви.
  + Джерела Інформації
    - Лекції
    - Практичні
    - <http://cpp.dp.ua/dynamichni-masyvy/>
  + Що опрацьовано:
    - Робота з динамічними масивами.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 17.12.2023
  + Завершення опрацювання теми: 18.12.2023
* Тема №2: Списки.
* Джерела Інформації:
  + - Лекції
    - Практичні
    - <https://www.geeksforgeeks.org/what-is-linked-list/>
  + Що опрацьовано:
    - Робота зі списками.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 18.12.2023
  + Завершення опрацювання теми: 18.12.2023
* Тема №3: Бінарні дерева.
  + Джерела Інформації:
    - Лекції
    - Практичні
    - <https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/?ref=shm>
  + Що опрацьовано:
    - Робота з бінарними деревами.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 19.12.2023
  + Завершення опрацювання теми: 20.12.2023

# **Виконання роботи:**

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

Завдання № 1 Algotester Lab 5

* Варіант завдання: 3
* Деталі завдання: У вас є карта гори розміром N × M . Також ви знаєте координати {x,y}, у яких знаходиться вершина гори. Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число. Клітинки які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

Завдання № 2 Algotester Lab 78

* Варіант завдання: 2
* Деталі завдання: Власноруч реалізувати структуру даних "Динамічний масив". Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

Завдання № 3 VNS Lab 10

* Варіант завдання: 5
* Деталі завдання: Написати програму, у якій створюються динамічні структури. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього К елементів, починаючи із заданого номера, додати К елементів, починаючи із заданого номера.

Завдання № 4 Class Practice

* Деталі завдання: Реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева.

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

Програма № 1 Algotester Lab 5

* Планований час на реалізацію: 60 хв

Програма № 2 Algotester Lab 78

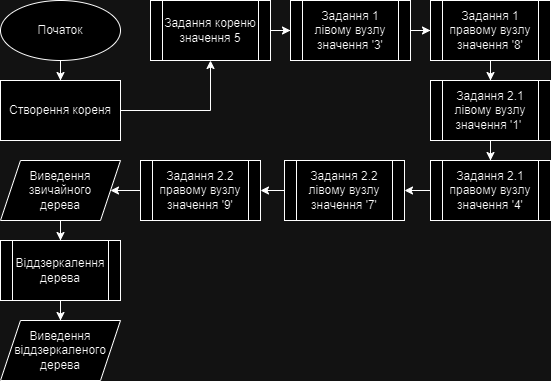
* Планований час на реалізацію: 180 хв

Програма №\_3 VNS Lab 10

* Планований час на реалізацію: 180 хв

Програма №\_4 Class Practice

* Блок-схема:



* Планований час на реалізацію: 180 хв

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

Додаткова конфігурація не потрібна.

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

Завдання №1

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int N, M, x, y, diff = 0, min = 0;

cin >> N >> M;

cin >> x >> y;

int mountain[N][M];

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

{

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

{

mountain[i][j] = 0 - (abs(i - (x - 1)) + abs(j - (y - 1)));

if (mountain[i][j] < min)

{

min = mountain[i][j];

diff = 0 - min;

}

}

}

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

{

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

{

cout << mountain[i][j] + diff << " ";

}

cout << endl;

}

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1217/files#diff-2a67ccbb79b650a75201bca6ae73423f4c882b3ac0c109575af5e21a34f021b8>

Завдання №2

#include <iostream>

#include <string>

using namespace std;

class DynamicArray {

private:

int\* array;

int capacity;

int size;

void reserve(int newCapacity) {

if (newCapacity >= capacity) {

while (newCapacity >= capacity) {

capacity \*= 2;

}

int\* newArray = new int[capacity];

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

newArray[i] = array[i];

}

delete[] array;

array = newArray;

}

}

public:

DynamicArray() : array(nullptr), capacity(1), size(0) {}

~DynamicArray() {

delete[] array;

}

void Insert(int index, int\* insArray, int insSize) {

int newSize = size + insSize;

reserve(newSize);

int\* resultArray = new int[capacity];

// Копіюємо елементи до вставки

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

resultArray[i] = array[i];

}

// Копіюємо елементи для вставки

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

resultArray[index + i] = insArray[i];

}

// Копіюємо елементи після вставки

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

resultArray[index + insSize + i] = array[index + i];

}

delete[] array;

array = resultArray;

size = newSize;

}

void Erase(int index, int n) {

if (size > 0) {

int newSize = size - n;

int\* resultArray = new int[capacity];

// Копіюємо елементи до видалення

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

resultArray[i] = array[i];

}

// Копіюємо залишок елементів після видалення

for (int i = 0; i < newSize - index; ++i) {

resultArray[index + i] = array[index + n + i];

}

delete[] array;

array = resultArray;

size = newSize;

}

}

void Size() const {

cout << size << endl;

}

void Capacity() const {

cout << capacity << endl;

}

bool IsEmpty() const {

return size == 0;

}

void Get(int index) const {

if (!IsEmpty()) {

cout << array[index] << endl;

}

}

void Set(int index, int newElement) {

if (!IsEmpty()) {

array[index] = newElement;

}

}

void Print() const {

if (!IsEmpty()) {

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

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

}

cout << endl;

}

}

};

int main() {

DynamicArray dynamicArray;

int Q;

string identifier;

cin >> Q;

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

cin >> identifier;

if (identifier == "insert") {

int index;

cin >> index;

int insSize;

cin >> insSize;

int insArray[insSize];

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

cin >> insArray[i];

}

dynamicArray.Insert(index, insArray, insSize);

}

else if (identifier == "erase") {

if (!dynamicArray.IsEmpty()) {

int index;

cin >> index;

int n;

cin >> n;

dynamicArray.Erase(index, n);

}

}

else if (identifier == "size") {

dynamicArray.Size();

}

else if (identifier == "capacity") {

dynamicArray.Capacity();

}

else if (identifier == "get") {

if (!dynamicArray.IsEmpty()) {

int index;

cin >> index;

dynamicArray.Get(index);

}

}

else if (identifier == "set") {

if (!dynamicArray.IsEmpty()) {

int index;

cin >> index;

int newElement;

cin >> newElement;

dynamicArray.Set(index, newElement);

}

}

else if (identifier == "print") {

dynamicArray.Print();

}

}

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1217/files#diff-4ac5ecd576da36292b18ea91f6c53093c351003516fd6d689dbdaded5faa93b3>

Завдання №3

#include <iostream>

#include <fstream>

using namespace std;

struct Node

{

int data;

Node\* next;

};

Node\* CreateLinkedList()

{

return nullptr;

}

void FillLinkedList(Node\*& head, int data)

{

Node\* newNode = new Node{data, nullptr};

if (head == nullptr)

{

head = newNode;

}

else

{

Node\* currnt = head;

while (currnt->next != nullptr)

{

currnt = currnt->next;

}

currnt->next = newNode;

}

}

void DeleteFromLinkedList(Node\*& head, int first, int K)

{

Node\* currnt = head;

Node\* prvs = nullptr;

int n = 1;

while (currnt != nullptr && n < first)

{

prvs = currnt;

currnt = currnt->next;

++n;

}

for (int i = 0; i < K && currnt != nullptr; ++i)

{

Node\* nxt = currnt->next;

delete currnt;

currnt = nxt;

}

if (prvs == nullptr)

{

head = currnt;

}

else

{

prvs->next = currnt;

}

}

void AddToLinkedList(Node\*& head, int first, int K)

{

Node\* currnt = head;

Node\* prvs = nullptr;

int n = 1;

while (currnt != nullptr && n < first)

{

prvs = currnt;

currnt = currnt->next;

++n;

}

int addedEl[K];

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

{

cin >> addedEl[i];

}

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

{

Node\* newNode = new Node;

newNode->data = addedEl[i];

if (prvs == nullptr)

{

newNode->next = head;

head = newNode;

}

else

{

newNode->next = currnt;

prvs->next = newNode;

}

prvs = newNode;

currnt = newNode->next;

}

}

void PrintLinkedList(Node\* head)

{

if (head == nullptr)

{

cout << "Список порожній";

}

else

{

Node\* currnt = head;

while (currnt != nullptr)

{

cout << currnt->data << " ";

currnt = currnt->next;

}

}

cout << endl;

}

void AddLlToFile(Node\* head, string filename)

{

ofstream file(filename);

if (!file.is\_open())

{

cout << "Помилка запису в файл" << endl;

return;

}

Node\* currnt = head;

while(currnt != nullptr)

{

file << currnt->data << " ";

currnt = currnt->next;

}

file.close();

}

void DeleteLinkedList(Node\*& head)

{

Node\* currnt = head;

while (currnt != nullptr)

{

Node\* nxt = currnt->next;

delete currnt;

currnt = nxt;

}

head = currnt;

}

void returnLlFromFile(string filename, Node\*& head)

{

ifstream file(filename);

if (!file.is\_open())

{

cout << "Помилка читання файлу" << endl;

return;

}

int dataFromFile;

while (file >> dataFromFile)

{

Node\* newNode = new Node{dataFromFile, nullptr};

if (head == nullptr)

{

head = newNode;

}

else

{

Node\* currnt = head;

while (currnt->next != nullptr)

{

currnt = currnt->next;

}

currnt->next = newNode;

}

}

file.close();

}

int main()

{

Node\* head = CreateLinkedList();

FillLinkedList(head, 11);

FillLinkedList(head, 12);

FillLinkedList(head, 13);

FillLinkedList(head, 14);

PrintLinkedList(head);

DeleteFromLinkedList(head, 2, 2);

PrintLinkedList(head);

AddToLinkedList(head, 2, 2);

PrintLinkedList(head);

AddLlToFile(head, "Lab10.txt");

DeleteLinkedList(head);

PrintLinkedList(head);

returnLlFromFile("Lab10.txt", head);

PrintLinkedList(head);

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1217/files#diff-6b8425a7645560b268571ebdf29fb3cf36992c9b3661f222ce31b59a3629448c>

Завдання №4

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* left;

Node\* right;

};

Node\* CreateNode(int data)

{

Node\* newNode = new Node;

newNode->data = data;

newNode->left = nullptr;

newNode->right = nullptr;

return newNode;

}

Node\* AddToTree(Node\* currnt, int data)

{

if (currnt == nullptr)

{

return CreateNode(data);

}

if (data < currnt->data)

{

currnt->left = AddToTree(currnt->left, data);

}

if (data > currnt->data)

{

currnt->right = AddToTree(currnt->right, data);

}

return currnt;

}

void PrintTree(Node\* root, int lvl = 0)

{

if (root == nullptr)

{

return;

}

else

{

PrintTree(root->right, lvl + 1);

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

{

cout << " ";

}

cout << root->data << endl;

PrintTree(root->left, lvl + 1);

}

}

Node\* ReverseTree(Node\* root)

{

if (root == nullptr)

{

return nullptr;

}

Node\* newRoot = CreateNode(root->data);

newRoot->left = ReverseTree(root->right);

newRoot->right = ReverseTree(root->left);

return newRoot;

}

int main()

{

Node\* root = nullptr;

root = AddToTree(root, 5);

AddToTree(root, 3);

AddToTree(root, 8);

AddToTree(root, 1);

AddToTree(root, 4);

AddToTree(root, 7);

AddToTree(root, 9);

cout << "Звичайне:" << endl;

PrintTree(root);

Node\* newRoot = ReverseTree(root);

cout << "Віддзеркалене:" << endl;

PrintTree(newRoot);

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1217/files#diff-dcc5f61d6bbcd3b02d4babbcfa9373da6889295893e842bb3d79988ea4e583f2>

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

Завдання №1 aaa bbb 10 abab 1 0

3 4

2 2

1 2 1 0

2 3 2 1

1 2 1 0

Час затрачений на виконання завдання: 60 хв

Завдання №2

12

size

0

insert 0 5

251 252 253 254 255

size

5

capacity

8

print

251 252 253 254 255

get 1

252

set 1 777

get 1

777

erase 1 3

get 1

255

size

2

print

251 255

Час затрачений на виконання завдання: 180 хв

Завдання №3

11 12 13 14

11 14

0 0

11 0 0 14

Список порожній

11 0 0 14

Час затрачений на виконання завдання: 180 хв

Завдання №4

Звичайне:

9

8

7

5

4

3

1

Віддзеркалене:

1

3

4

5

7

8

9

Час затрачений на виконання завдання: 180 хв

# **Висновки:**

Під час ознайомлення з цим розділом я дізнався про динамічні масиви, списки, бінарні дерева. Ці знання я застосував для виконання лабораторних і практичних робіт.