Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 6

На тему: «Програмування: алгоритм, програма, код. Системи числення. Двійкова система числення. Розробка та середовище розробки програми.»

з дисципліни: «Основи програмування»

до:

Практичних Робіт до блоку № 6

Виконав:
Студент групи ШІ-11
Климчук Юрій Олегович

Львів 2024

Тема: Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

Мета: Засвоїти основи роботи з динамічними структурами даних, такими як черга, стек, списки та дерева. Ознайомитися з алгоритмами їх обробки для розв'язання різноманітних задач.

Теоретичні відомості:

1)Перелік тем:

1. Основи Динамічних Структур Даних
2. Стек
3. Черга
4. Зв'язні Списки
5. Дерева
6. Алгоритми Обробки Динамічних Структур

2)Індивідуальний план опрацювання теорії:

1. **Основи Динамічних Структур Даних**

   Методичні вказівники до лабораторної роботи №10

   https://www.geeksforgeeks.org/static-data-structure-vs-dynamic-data-structure/

   https://www.javatpoint.com/dynamic-array-in-c

2. **Стек**

   https://www.geeksforgeeks.org/stack-in-cpp-stl/

   https://www.w3schools.com/cpp/cpp_stacks.asp

3. **Черга**

   https://www.geeksforgeeks.org/introduction-and-array-implementation-of-queue/

   https://www.tutorialspoint.com/cplusplus-program-to-implement-queue-using-array

4. **Зв'язні Списки**

   https://www.geeksforgeeks.org/cpp-linked-list/

https://www.geeksforgeeks.org/doubly-linked-list/

5. **Дерева**

   https://www.geeksforgeeks.org/binary-tree-in-cpp/

   https://www.geeksforgeeks.org/tree-c-cpp-programs/

   https://www.geeksforgeeks.org/red-black-tree-in-cpp/

6. **Алгоритми Обробки Динамічних Структур**

   Методичні вказівники до лабораторної роботи №10

   https://www.geeksforgeeks.org/c-cpp-dynamic-programming-programs/

Виконання роботи:

1)Перелік завдань:

- John Black - Epic 6 Task 1 - Theory Education Activities
- John Black - Epic 6 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)
- John Black - Epic 6 Task 3 - Lab# programming: VNS Lab 10
- John Black - Epic 6 Task 4 - Lab# programming: Algotester Lab 5
- John Black - Epic 6 Task 5 - Lab# programming: Algotester Lab 7-8
- John Black - Epic 6 Task 6 - Practice# programming: Class Practice Task
- John Black - Epic 6 Task 7 - Practice# programming: Self Practice Task
- John Black - Epic 6 Task 8 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
- John Black - Epic 6 Task 9 - Results Evaluation and Release

2)Умови завдань:

Task 3: Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом. Для кожного варіанту розробити такі функції:

1. Створення списку.

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

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

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Порядок виконання роботи

1. Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.

2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.

3. Написати функції для знищення й додавання елементів списку у відповідності зі своїм варіантом.

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

5. Написати функцію для запису списку у файл.

6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

9. Відновити список і роздрукувати його.

10.Знищити список.

Варіант 4

Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим номером, додати К елементів, починаючи із заданого номера;

Task 4: Lab 5v3

Обмеження: 1 сек., 256 МіБ

У вас є карта гори розміром N×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.

Клітинкі 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.

Вхідні дані

У першому рядку 2 числа N та M - розміри карти
у другому рядку 2 числа x та y - координати піку гори

Вихідні дані

N рядків по M елементів в рядку через пробіл - висоти карти.

Task 5: Lab 78v2
Обмеження: 1 сек., 256 МіБ
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 indexindex 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) рівний 2.

- **Отримання значення** і-го елементу
  Ідентифікатор - get
  Ви отримуєте ціле число - indexindex, індекс елемента.
  Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора []
- **Модифікація значення** і-го елементу
  Ідентифікатор - setset
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора []
- **Вивід динамічного масиву на екран**
  Ідентифікатор - print
  Ви не отримуєте аргументів.
  Ви виводите усі елементи динамічного масиву через пробіл.
  Реалізувати використовуючи перегрузку оператора <<

Вхідні дані
Ціле число Q - кількість запитів.
У наступних рядках Q запитів у зазначеному в умові форматі.
Вихідні дані
Відповіді на запити у зазначеному в умові форматі.
Для отримання 100%балів ця структура має бути написана як шаблон класу, у якості параметру використати intint.
Використовувати STL заборонено.

Task 6:

Зв’язаний список

Задача №1 - Реверс списку (Reverse list)

Реалізувати метод реверсу списку: Node* reverse(Node *head);
Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати метод реверсу;
- реалізувати допоміжний метод виведення вхідного і обернутого списків;

Задача №2 - Порівняння списків

bool compare(Node *h1, Node *h2);
Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;

- якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

Задача №3 – Додавання великих чисел

Node* add(Node *n1, Node *n2);
Умови задачі:

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

Бінарні дерева

Задача №4 - Віддзеркалення дерева

TreeNode *create_mirror_flip(TreeNode *root);
Умови задачі:

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

Задача №5 - Записати кожному батьківському вузлу суму підвузлів

void tree_sum(TreeNode *root);
Умови задачі:

- використовувати цілочисельні значення у вузлах дерева;
- 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

Task 7: Lab 5v1

Обмеження: 2 сек., 256 МіБ

У світі Атод сестри Ліна і Рілай люблять грати у гру. У них є дошка із 8-ми рядків і 8-ми стовпців. На перетині i-го рядка і j-го стовпця лежить магічна куля, яка може світитись магічним світлом (тобто у них є 64 кулі). На початку гри деякі кулі світяться, а деякі ні... Далі вони обирають NN куль і для кожної читають магічне заклинання, після чого всі кулі, які лежать на перетині стовпця і рядка обраної кулі міняють свій стан (ті що світяться - гаснуть, ті, що не світяться - загораються). Також вони вирішили трохи Вам допомогти і придумали спосіб як записати стан дошки одним числом a із 8-ми байт, а саме (див. Примітки):

- Молодший байт задає перший рядок матриці;
- Молодший біт задає перший стовпець рядку;
- Значення біту каже світиться куля чи ні (0 - ні, 1 - так);

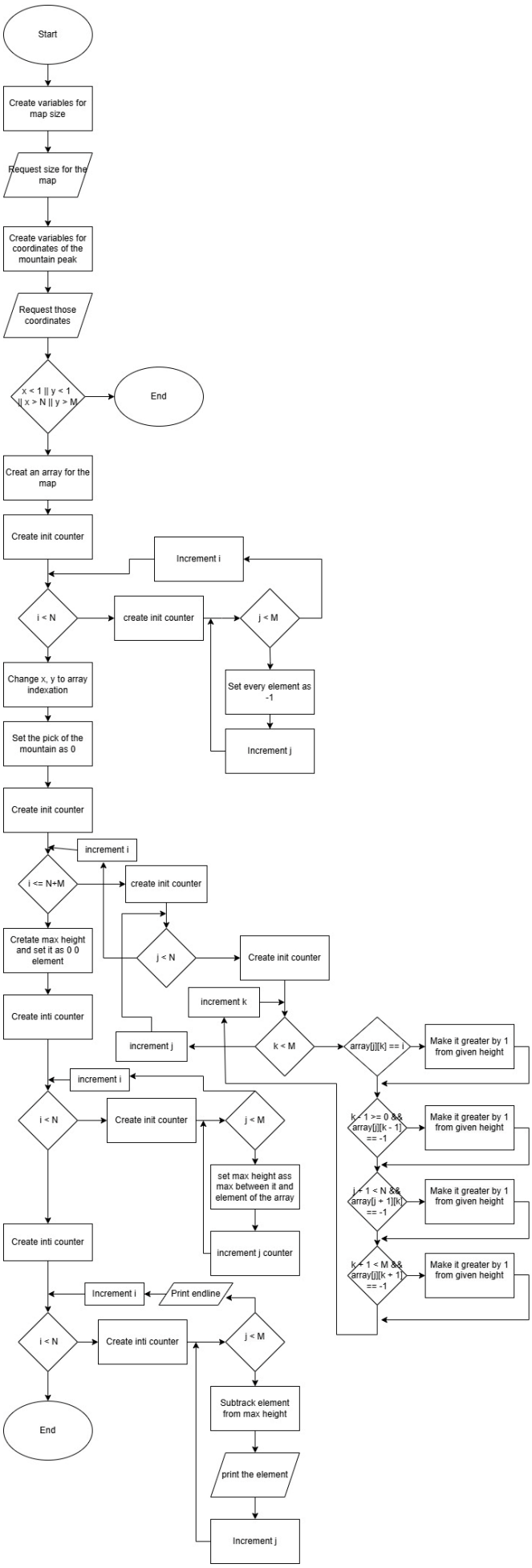
Тепер їх цікавить яким буде стан дошки після виконання N заклинань і вони дуже просять Вас їм допомогти.

Вхідні дані

У першому рядку одне число a - поточний стан дошки.
У другому рядку N - кількість заклинань.
У наступних N рядках по 2 числа R_i, C_i - рядок і стовпець кулі над якою виконується заклинання.

3)Дизайн та планова оцінка часу виконання завдань:

John Black - Epic 6 Task 4 - Lab# programming: Algotester Lab 5(варіант 3)


Start
Create variables for map size
Request size for the map
Create variables for coordinates of the mountain peak
Request those coordinates
x < 1 || y < 1 || x > N || y > M
End
Creat an array for the map
Create init counter
Increment i
i < N
create init counter
j < M
Set every element as -1
Increment j
Change x, y to array indexation
Set the pick of the mountain as 0
Create init counter
increment i
i <= N+M
create init counter
Cretate max height and set it as 0 0 element
j < N
Create init counter
increment k
Create inti counter
increment j
k < M
array[j][k] == i
Make it greater by 1 from given height
increment i
i < N
Create init counter
j < M
k - 1 >= 0 && array[j][k - 1] == -1
Make it greater by 1 from given height
set max height ass max between it and element of the array
j + 1 < N && array[j + 1][k] == -1
Make it greater by 1 from given height
Create inti counter
increment j counter
k + 1 < M && array[j][k + 1] == -1
Make it greater by 1 from given height
Increment i
Print endline
i < N
Create inti counter
j < M
End
Subtrack element from max height
print the element
Increment j

Орієнтовний час виконання: 1год 30хв

4)Код програм з посиланням на зовнішні ресурси:

John Black - Epic 6 Task 3 - Lab# programming: VNS Lab 10(варіаніт 4)

```
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;

//структура для створення елемента з'язного списку
struct Node {
    int key;
    Node* next;
};

//структура для додавання нового елемента у список
void AddNewNode(Node*& head, int key, int position) {
    //створюємо нову ноду
    Node* newNode = new Node;
    newNode->key = key;
    newNode->next = nullptr;

    //якщо позиція 0 ставимо як початок списку
    if (head == nullptr || position == 0) {
        newNode->next = head;
        head = newNode;
        return;
    }

    //записуємо новий елемент у список
    Node* temp = head;
    int currentPosition = 0;
    while (temp->next != nullptr && currentPosition < position - 1) {
        temp = temp->next;
        currentPosition++;
    }
    newNode->next = temp->next;
```

```
    //додавання елемента у випадку коли його позиція поза межами
списку
    temp->next = newNode;
}

//видалення елемента списку за позицією
void DeleteNode(Node*& head, int position) {
    //перевірка на наявність елементів
    if (head == nullptr) {
        cout << "The list is empty\n";
        return;
    }
    //якщо позиція 0, зміщуємо початок списку перед видаленням
    if (position == 0) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    //шукаємо вказівник на необхідну позицію
    Node* temp = head;
    for (int i = 0; temp != nullptr && i < position - 1; i++) {
        temp = temp->next;
    }
    if (temp == nullptr || temp->next == nullptr) {
        cout << "The position you entered is out of range of the
list, it`s impossible to delete the element\n";
        return;
    }
    Node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    delete nodeToDelete;
}

//функція для додавання кількох елементів
void AddKNodes(Node*& head, int startPosition, int K) {
    for (int i = 0; i < K; i++) {
        int key;
        cout << "Enter the value for the new element" << (i + 1) <<
": ";
        cin >> key;
```

```
        //викликаємо функцію для додавання для кожного нового
елемента
        AddNewNode(head, key, startPosition + i);
    }
}

//вивід списку
void PrintList(Node* head) {
    //перевірка на наявність елементів
    if (head == nullptr) {
        cout << "The list is empty\n";
        return;
    }

    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->key << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

//запис списку у файл
void WriteToFile(Node* head, const string& filename) {
    ofstream List_file(filename);
    if (!List_file) {
        cerr << "Error while opening the file\n";
        return;
    }

    Node* temp = head;
    while (temp != nullptr) {
        List_file << temp->key << " ";
        temp = temp->next;
    }
    List_file.close();

    cout << "The list is successfully written to the file\n";
}

//видалення списку
void DeleteList(Node*& head) {
```

```
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    cout << "The list is successfully deleted\n";
}

//відновлення списку з файлу
void RestoreFromFile(Node*& head, const string& filename) {
    ifstream file(filename);
    if (!file) {
        cerr << "Error while opening the file\n";
        return;
    }

    int key;
    DeleteList(head);//видаляємо старий список
    while (file >> key) {
        AddNewNode(head, key, 0); //додаємо кожеш елемент з файлу
    }
    file.close();
    cout << "The list is successfully restored from the file\n";
}

int main() {
    Node* list = nullptr; //створюємо пустий список
    int user_choice, key, position, k; //вводимо змінні для
користувача

    //реалізація меню
    do {
        cout << "\nWelcome to linked list programme:\n"
        << "1.Add an element to the list\n"
        << "2.Delete an element\n"
        << "3.Print the list\n"
        << "4.Add K element to the list\n"
        << "5.Add the list to the file\n"
        << "6.Delete the list\n"
        << "7.Restore the list from the file\n"
        << "8.Exit\n"
        << "Your choice: ";
```

```
        cin >> user_choice;

        switch (user_choice) {
            case 1:
                cout << "Enter the new element: ";
                cin >> key;
                cout << "Enter the position of this element: ";
                cin >> position;
                AddNewNode(list, key, position);
                break;
            case 2:
                cout << "Enter the position of the element to delete:
";
                cin >> position;
                DeleteNode(list, position);
                break;
            case 3:
                PrintList(list);
                break;
            case 4:
                cout << "Enter the position from where we will be
adding elements: ";
                cin >> position;
                cout << "Enter the number of elements to add: ";
                cin >> k;
                AddKNodes(list, position, k);
                break;
            case 5:
                WriteToFile(list, "list.txt");
                break;
            case 6:
                DeleteList(list);
                break;
            case 7:
                RestoreFromFile(list, "list.txt");
                break;
            case 8:
                cout << "Exiting the programme\n";
                break;
            default:
                cout << "Incorrect choice, try again\n";
        }
```

```
    }while(user_choice != 8);

    DeleteList(list);
    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-c11b354851ced93298343927ded0da8ac7dee12d279c274f3d757c0a74218ced

John Black - Epic 6 Task 4 - Lab# programming: Algotester Lab 5(варіант 3)

```
#include <iostream>

using namespace std;

int main() {
    //The size of a map(NxM)
    int N, M;
    cin >> N >> M;

    //The peak of the mountain
    int x, y;
    cin >> x >> y;
    if (x < 1 || y < 1 || x > N || y > M) return 0;

    //Map
    int array[N][M];

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            array[i][j] = -1;
        }
    }

    //Set peak as 0
    --x, --y;
    array[x][y] = 0;

    //Main algorithm
    for (int i = 0; i <= N + M; i++) {
```

```
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < M; k++) {
                if (array[j][k] == i) {
                    //Check if the coordinate we want to change
exists
                    if (j - 1 >= 0 && array[j - 1][k] == -1) {
                        array[j - 1][k] = i + 1;
                    }
                    if (k - 1 >= 0 && array[j][k - 1] == -1) {
                        array[j][k - 1] = i + 1;
                    }
                    if (j + 1 < N && array[j + 1][k] == -1) {
                        array[j + 1][k] = i + 1;
                    }
                    if (k + 1 < M && array[j][k + 1] == -1) {
                        array[j][k + 1] = i + 1;
                    }
                }
            }
        }
    }

    //Find the max height of the map
    int max_height = array[0][0];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            max_height = max(max_height, array[i][j]);
        }
    }

    //Print our height chart
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            array[i][j] = max_height - array[i][j];
            cout << array[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-fd31dcea8699450bd6f566366b98f7bc84e6447043e44caf7cc0baaed588e83d

Посилання на алготестер: https://algotester.com/uk/ProblemSolution/Display/1909285

John Black - Epic 6 Task 5 - Lab# programming: Algotester Lab 7-8(варіант 2)(100%)

```
#include <iostream>
using namespace std;

//Usage of class template
template <class T>
class DynamicArray {
private:
    T *data;
    //Resize with growth factor 2
    void ResizeIfNeeded(int N) {
        while (size + N >= capacity) {
            capacity *= 2;
        }
    }

public:
    int size;
    int capacity;
    //Constructor
    DynamicArray() {
        size = 0;
        capacity = 1;
        data = new T[1];
    }
    //Inserting multiple elements into array
    void insert(int index, int N, T *elements) {
        ResizeIfNeeded(N);

        T *temporary = new T[capacity];

        for (int i = 0; i < index; i++) {
            temporary[i] = data[i];
        }
```

```
        for (int i = 0; i < N; i++) {
            temporary[index + i] = elements[i];
        }

        for (int i = index; i < size; i++) {
            temporary[i + N] = data[i];
        }
        //Update the size, delete old data, add new
        size += N;
        delete[] data;
        data = temporary;
    }

    //Erasing multiple elements
    void erase(int index, int N) {
        T *temporary = new T[capacity];
        int newSize = 0;

        //Deleting elements by index
        for (int i = 0; i < size; i++) {
            if (i < index || i >= index + N) {
                temporary[newSize] = data[i];
                newSize++;
            }
        }
        //Update the size and data
        size -= N;
        delete[] data;
        data = temporary;
    }

    int Size() const {
        return size;
    }

    int Capacity() const {
        return capacity;
    }
    //Return element by its index
    T get(int index) const {
        return data[index];
```

```
    }

    //Rewrite the value by index
    void set(int index, T value) {
        data[index] = value;
    }

    //Function to display array
    void print(const string &separator = " ") const {
        for (int i = 0; i < size; i++) {
            cout << data[i];
            if (i < size - 1) {
                cout << separator;
            }
        }
        cout << endl;
    }
};

int main() {
    DynamicArray<int> array;

    int Q;
    cin >> Q;

    while (Q--) {
        string command;
        cin >> command;
        //Processing users input
        if (command == "insert") {
            int index, N;
            cin >> index >> N;
            int *elements = new int[N];

            for (int i = 0; i < N; i++) {
                cin >> elements[i];
            }

            array.insert(index, N, elements);
            delete[] elements;
        }
        else if (command == "erase") {
```

```
            int index, N;
            cin >> index >> N;
            array.erase(index, N);
        }
        else if (command == "size") {
            cout << array.Size() << endl;
        }
        else if (command == "capacity") {
            cout << array.Capacity() << endl;

        }
        else if (command == "get") {
            int index;
            cin >> index;
            cout << array.get(index) << endl;
        }
        else if (command == "set") {
            int index, N;
            cin >> index >> N;
            array.set(index, N);
        }
        else if (command == "print") {
            array.print();
        }
    }
    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-6fb3c425231ff942879b5cb010e3e47882cf4f998403eeefc2fa223d6c63e87c

Посилання на алготестер: https://algotester.com/uk/ProblemSolution/Display/1909036

John Black - Epic 6 Task 6 - Practice# programming: Class Practice Task(1-3 завдання)

```
#include <iostream>
using namespace std;
```

```
//Structure for an element of the list
struct Node {
    int data;
    Node* next;
    //Constructor
    Node(int val) : data(val), next(nullptr) {}
};

//Function to display a list
void PrintList(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

//Function to insert elements at a specified index
void insert(Node*& head, int index, int listSize, int values[]) {
    //Invalid index given
    if (index < 0 || index > listSize) {
        return;
    }

    Node* newNode = nullptr;
    Node* current = head;

    //If inserting at the beginning, create a new head node
    if (index == 0) {
        for (int i = 0; i < listSize; i++) {
            newNode = new Node(values[i]);
            newNode->next = head;
            head = newNode;
        }
        return;
    }

    //Find node at the given index
    for (int i = 0; i < index - 1; i++) {
        if (current != nullptr) {
            current = current->next;
```

```
        }
    }

    //Insert new nodes
    for (int i = 0; i < listSize; i++) {
        newNode = new Node(values[i]);
        newNode->next = current->next;
        current->next = newNode;
        current = newNode;
    }
}

//Function to reverse the list
Node* reverse(Node* head) {
    Node* previous = nullptr;
    Node* current = head;
    Node* next = nullptr;

    //Swap each two nearby elements
    while (current != nullptr) {
        next = current->next;
        current->next = previous;
        previous = current;
        current = next;
    }

    return previous;
}

//Practical task #2: compare lists
bool compare(Node* h1, Node* h2) {
    /*Take each individual element and compare to
    the element on the same position on the other list*/
    while (h1 != nullptr && h2 != nullptr){
        if (h1->data != h2->data) {
            return false;
        }
        h1 = h1->next;
        h2 = h2->next;
    }
    //Check if each list run out of elements
   return h1 == nullptr && h2 == nullptr;
```

```
}

//Practical task #3: sum numbers
Node* add(Node* n1, Node* n2) {
    // Reverse both input lists first
    n1 = reverse(n1);
    n2 = reverse(n2);

    // Create a new list for the sum
    Node* temp_head = new Node(0);
    Node* current_head = temp_head;
    int carry = 0;

    while (n1 != nullptr || n2 != nullptr || carry != 0) {
        int sum = carry;

        // Add the digits from both lists, if present
        if (n1 != nullptr) {
            sum += n1->data;
            n1 = n1->next;
        }
        if (n2 != nullptr) {
            sum += n2->data;
            n2 = n2->next;
        }

        // Carry for the next place
        carry = sum / 10;

        // Add the current digit to the result list
        current_head->next = new Node(sum % 10);
        current_head = current_head->next;
    }

    // Reverse the result list to maintain the reversed order
    Node* result = reverse(temp_head->next);
    return result;
}

int main() {
    // First programme: reverse the list
    Node* head = nullptr;
```

```
    int new_elements[] = {8,5,3,2,1,1};
    int new_elements_size = sizeof(new_elements) /
sizeof(new_elements[0]);
    insert(head, 0, new_elements_size, new_elements);

    cout << "Original list: ";
    PrintList(head);

    head = reverse(head);
    cout << "Reversed list: ";
    PrintList(head);

    // Second programme: compare lists
    Node* head1 = nullptr;
    int new_elements_head1[] = {-47,34,22};
    int new_elements_size_head1 = sizeof(new_elements_head1) /
sizeof(new_elements_head1[0]);
    insert(head1, 0, new_elements_size_head1, new_elements_head1);

    Node* head2 = nullptr;
    int new_elements_head2[] = {-47,34,22};
    int new_elements_size_head2 = sizeof(new_elements_head2) /
sizeof(new_elements_head2[0]);
    insert(head2, 0, new_elements_size_head2, new_elements_head2);

    Node* head3 = nullptr;
    int new_elements_head3[] = {-36,4,-47};
    int new_elements_size_head3 = sizeof(new_elements_head3) /
sizeof(new_elements_head3[0]);
    insert(head3, 0, new_elements_size_head3, new_elements_head3);

    //With identical lists
    cout << "List 1: ";
    PrintList(head1);

    cout << "List 2: ";
    PrintList(head2);

    if (compare(head1, head2)){
        cout << "The lists are equal" << endl;
    }
    else {
```

```
        cout << "The lists are not equal" << endl;
    }

    //With different lists
    cout << "List 1: ";
    PrintList(head1);

    cout << "List 3: ";
    PrintList(head3);

    if (compare(head1, head3)){
        cout << "The lists are equal" << endl;
    }
    else {
        cout << "The lists are not equal" << endl;
    }

    // Third programme: sum numbers
    // 207
    Node* number_1 = nullptr;
    int new_elements_number_1[] = {2, 0, 7};
    int new_elements_size_number_1 = sizeof(new_elements_number_1) /
sizeof(new_elements_number_1[0]);
    insert(number_1, 0, new_elements_size_number_1,
new_elements_number_1);

    // 664
    Node* number_2 = nullptr;
    int new_elements_number_2[] = {6,6,4};
    int new_elements_size_number_2 = sizeof(new_elements_number_2) /
sizeof(new_elements_number_2[0]);
    insert(number_2, 0, new_elements_size_number_2,
new_elements_number_2);

    cout << "Number 1: ";
    PrintList(number_1);

    cout << "Number 2: ";
    PrintList(number_2);

    Node* result = add(number_1, number_2);
```

```
    cout << "Sum: ";
    PrintList(result);
    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-b3270773a9b440e61e76d04847236332c76a487c6cad736b8d99fd612919c7e5

John Black - Epic 6 Task 6 - Practice# programming: Class Practice Task(4-5 завдання)

```
#include <iostream>

using namespace std;

//Structure to create a tree vertex
struct TreeNode {
    int data;
    TreeNode* left;
    TreeNode* right;
    //Constructor
    TreeNode(int value) : data(value), left(nullptr), right(nullptr)
{}
};

struct BinarySearchTree {
    //Construct a tree
    BinarySearchTree() : root(nullptr) {}

    void insert(int value) {
        root = insert(root, value);
    }

    void mirrorFlip() {
        root = createMirrorFlip(root);
    }

    void treeSum() {
```

```
        iterativeTreeSum(root);
    }

    void printTree() {
        printTree(root);
    }

private:
    TreeNode* root;

    //Fuction to insert a tree element
    TreeNode* insert(TreeNode* node, int value) {
        if (node == nullptr) {
            return new TreeNode(value);
        }

        if (value < node->data) {
            node->left = insert(node->left, value);
        }
        else if (value > node->data){
            node->right = insert(node->right, value);
        }

        return node;
    }

    //Function to create new mirrored tree
    TreeNode* createMirrorFlip(TreeNode* node) {
        if (node == nullptr) {
            return nullptr;
        }
        //Swap each right and left leaves
        TreeNode* newNode = new TreeNode(node->data);
        newNode->left = createMirrorFlip(node->right);
        newNode->right = createMirrorFlip(node->left);

        return newNode;
    }
    //Calculate sum of subnodes
    void iterativeTreeSum(TreeNode* node) {
        if (node == nullptr) return;
```

```
        TreeNode* current = node;
        TreeNode* stack[100];
        int top = -1;
        while (current != nullptr || top >= 0) {
            while (current != nullptr) {
                stack[++top] = current;
                current = current->left;
            }
            current = stack[top--];
            if (current->left == nullptr && current->right ==
nullptr) {
                current = nullptr;
            } else {
                int sum = 0;
                if (current->left) sum += current->left->data;
                if (current->right) sum += current->right->data;
                current->data += sum;
                current = nullptr;
            }
        }
    }

    void printTree(TreeNode* node) {
        if (node != nullptr) {
            printTree(node->left);
            cout << node->data << " ";
            printTree(node->right);
        }
    }
};

int main() {
    BinarySearchTree myTree;

    myTree.insert(4);
    myTree.insert(2);
    myTree.insert(5);
    myTree.insert(1);
    myTree.insert(6);
    myTree.insert(3);

    //Fourth task: create mirrored tree
```

```
    cout << "Original tree: ";
    myTree.printTree();
    cout << endl;

    myTree.mirrorFlip();
    cout << "Mirrored flip tree: ";
    myTree.printTree();
    cout << endl;

    //Fifth task: calculate sum of subnodes
    myTree.treeSum();
    cout << "Tree with sums: ";
    myTree.printTree();
    cout << endl;

    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-24ce1fa7c3165f736ad0560817efcf1e60b631d266f18bd94dd456ef2fbe82b7

John Black - Epic 6 Task 7 - Practice# programming: Self Practice Task

```
#include <iostream>
#include <cstdint>
#include <vector>

using namespace std;

void flip(uint64_t &a, int R, int C) {
    //Calculate position, create a mask, reverse the bit
    a ^= (1ULL << (R * 8 + C));
}

int main() {
    //Variables for board and number of spells
    uint64_t a;
    int N;
    cin >> a;
    cin >> N;
```

```
    //Create vector for the position of the spell
    vector<pair<int, int>> spells(N);
    for (int i = 0; i < N; ++i) {
        cin >> spells[i].first >> spells[i].second;
        spells[i].first--;
        spells[i].second--;
    }

    //
    for (int i = 0; i < N; ++i) {
        int R = spells[i].first;
        int C = spells[i].second;
        //Changes all Rows
        for (int j = 0; j < 8; ++j) {
            flip(a, R, j);
        }
        //Changes all Cols
        for (int k = 0; k < 8; ++k) {
            flip(a, k, C);
        }
        //Gets rid of intersections
        flip(a, R, C);
    }

    //Prints final situation on the board
    cout << a << endl;
    return 0;
}
```

Посилання на файл програми: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532/files#diff-38112a28dde60297db89832200a2ebf5c73077463f75381ac7422bcc3533f85b

Посилання на алготестер: https://algotester.com/uk/ProblemSolution/Display/1909265

5)Результати виконання завдань та фактично затрачений час

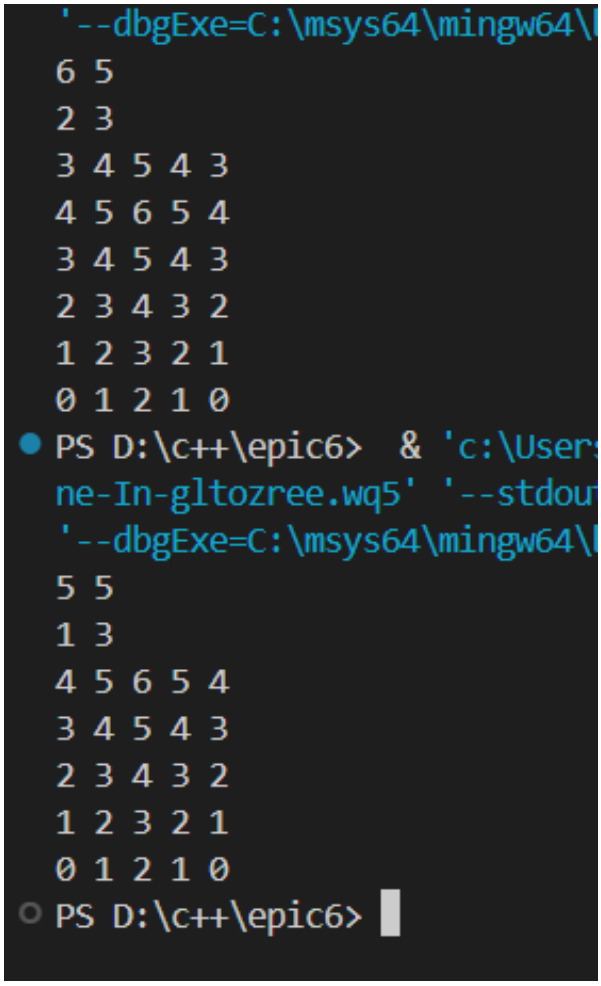
John Black - Epic 6 Task 3 - Lab# programming: VNS Lab 10

```
Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
7.Restore the list from the file
8.Exit
Your choice: 1
Enter the new element: 4
Enter the position of this element: 0

Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
7.Restore the list from the file
8.Exit
Your choice: 3
4 -> NULL

Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
7.Restore the list from the file
8.Exit
Your choice: 4
Enter the position from where we will be adding elements: 1
Enter the number of elements to add: 5
Enter the value for the new element1: 8
Enter the value for the new element2: 3
Enter the value for the new element3: 2
```

```
8.Exit
Your choice: 4
Enter the position from where we will be adding elements: 1
Enter the number of elements to add: 5
Enter the value for the new element1: 8
Enter the value for the new element2: 3
Enter the value for the new element3: 2
Enter the value for the new element4: 6
Enter the value for the new element5: 4

Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
7.Restore the list from the file
8.Exit
Your choice: 3
4 -> 8 -> 3 -> 2 -> 6 -> 4 -> NULL

Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
7.Restore the list from the file
8.Exit
Your choice: 5
The list is successfully written to the file

Welcome to linked list programme:
1.Add an element to the list
2.Delete an element
3.Print the list
4.Add K element to the list
5.Add the list to the file
6.Delete the list
```

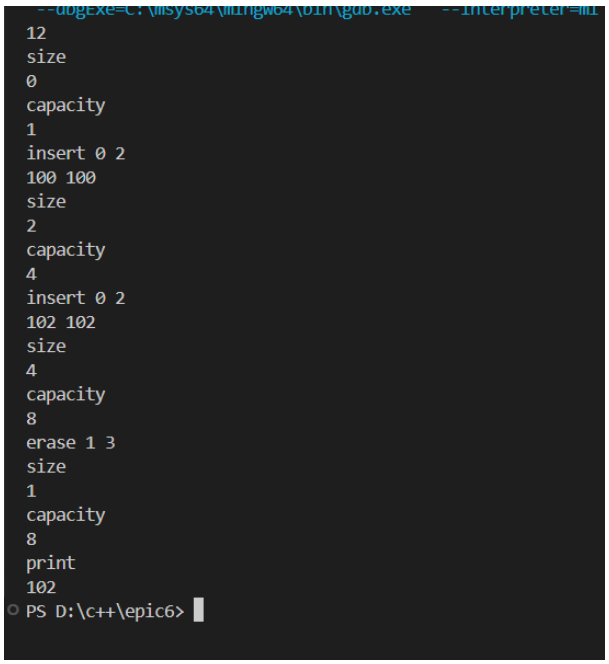
Фактичний час виконання: 1год 48хв

John Black - Epic 6 Task 4 - Lab# programming: Algotester Lab 5



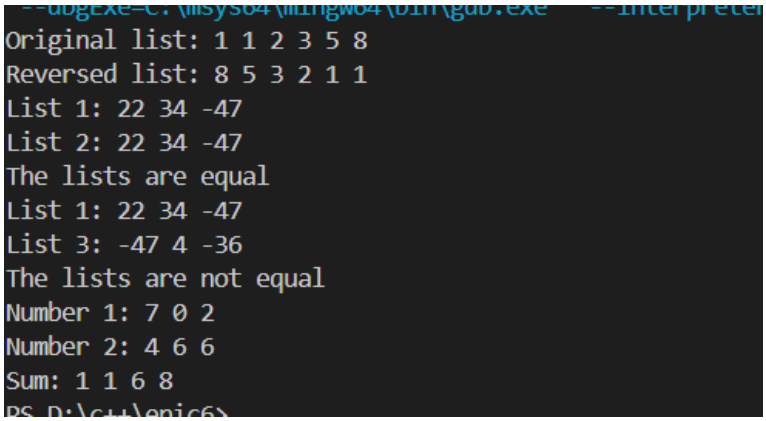
Фактичний час виконання: 2год 21хв

John Black - Epic 6 Task 5 - Lab# programming: Algotester Lab 7-8(варіант 2)(100%)



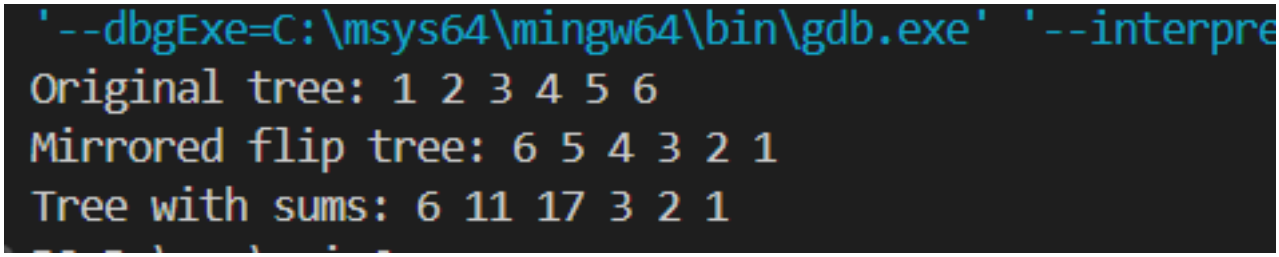
Фактичний час виконання: 1год 38хв

John Black - Epic 6 Task 6 - Lab# programming: Class Practice Task(1-3 завдання)



Фактичний час виконання: 58хв

John Black - Epic 6 Task 6 - Lab# programming: Class Pratice Task(4-5 завдання)



Фактичний час виконання: 1год 6хв

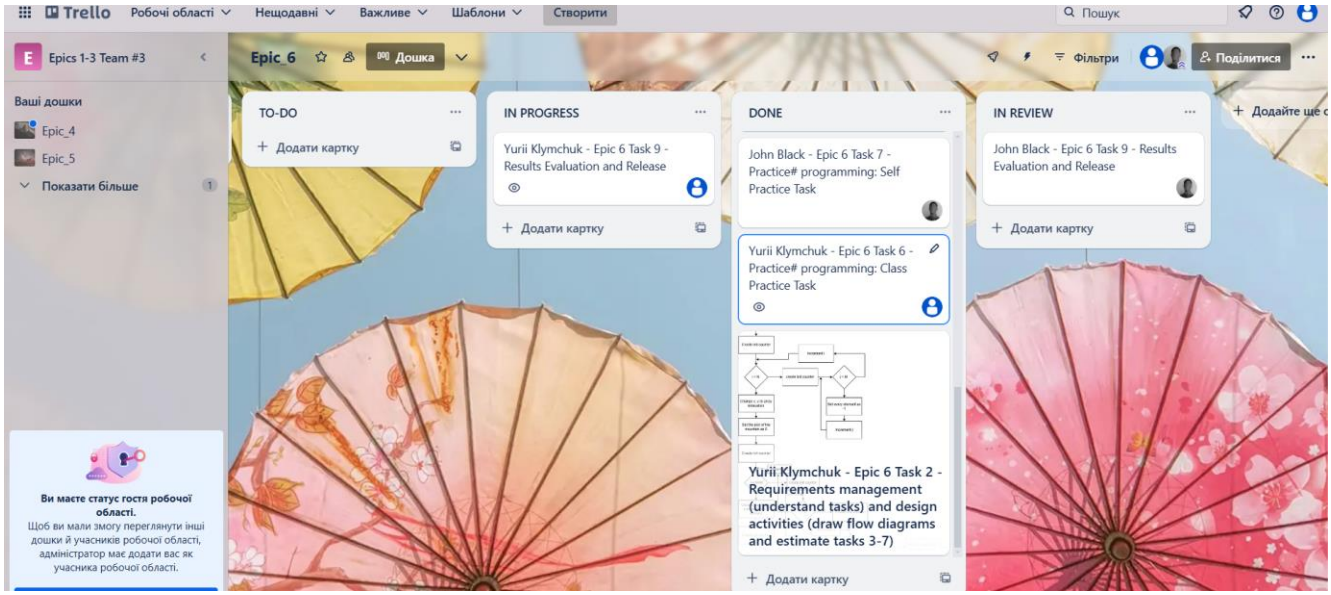
John Black - Epic 6 Task 7 - Practice# programming: Self Practice Task

```
ne-In-ce1152bc.yie'   --stdout=Microsoft-MIE
'--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '-
0
4
1 1
1 2
2 1
2 2
771
PS D:\c++\epic6>  & 'c:\Users\User\.vscode\
ne-In-twlrdca1.aft' '--stdout=Microsoft-MIE
'--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '-
771
4
1 1
1 2
2 1
2 2
0
PS D:\c++\epic6>
```

Фактичний час виконання: 1год 29хв

6)Робота з комадою

Trello:



Висновок: Робота з динамічними структурами даних (черга, стек, списки, дерева) є фундаментальною навичкою в алгоритмічному програмуванні. Вона

дозволяє ефективно зберігати, обробляти та організовувати дані для розв'язання широкого спектра задач

Посилання на пул реквест: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/532