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

Звіт

про виконання лабораторних та практичних робіт блоку № 6

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 10
Алготестер Лабораторної Роботи № 5
Алготестер Лабораторної Роботи № 7-8
Практичних Робіт до блоку № 6

Виконала:
Студент(ка) групи ШІ-11
Цибух Андріана

Львів 2024

Тема роботи:

Основи динамічних структур даних: стек, черга, зв’язний список, дерево.

Мета роботи:

Освоєння основних принципів робити з динамічною пам’яттю, отримання навичок реалізації та використання стеку, черги, зв’язних списків та дерев.

Теоретичні відомості:

1) Теоретичні відомості з переліком важливих тем:

- Тема №*.1: Динамічні структури даних.
- Тема №*.2: Алгоритми обробки.

2) Індивідуальний план опрацювання теорії:

- Тема №*.1: Динамічні структури даних.
    - Джерела Інформації
        - Лекції О. Пшеничного.
        - Практичні заняття М. Фаріон.
        - Ютуб
        - Сайт **GeeksforGeeks : Linked List in C++.**
    - Статус: Ознайомлена

- Тема №*.2: Алгоритми обробки.
    - Джерела Інформації:
        - Лекції О. Пшеничного.
        - Практичні заняття М. Фаріон.
        - Ютуб

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання №1 Зв’язний список та Бінарні дерева

- Деталі завдання : Реалізувати :
    - Метод реверсу списку
    - Порівняння списків
    - Додавання великих чисел
    - Віддзеркалювання дерева
    - Запис кожному батьківському вузлу суму підвузлів
- Час на реалізацію : 2 год

Завдання №2 VNS Labs 10

- Варіант завдання : 13
- Деталі завдання : Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом. Розробити такі функції:
  1. Створення списку.
  2. Додавання елемента в список .
  3. Знищення елемента зі списку.
  4. Друк списку.
  5. Запис списку у файл.

6. Знищення списку.
7. Відновлення списку з файлу.
Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити з нього K перших елементів. Додати елемент після елемента, що починається із зазначеного символу.

- Час на реалізацію : 4 год

Завдання №3 Algotester Lab 5

- Варіант завдання : 1
- Деталі завдання : 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 ii-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 N куль і для кожної читають магічне заклиння, 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

- Час на реалізацію : 40 хв

Завдання №4 Algotester Lab 7-8

- Варіант завдання : 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) рівний 2.

- **Отримання значення** і-го елементу
  Ідентифікатор - get
  Ви отримуєте ціле число - index, індекс елемента.\ Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора []
- **Модифікація значення** і-го елементу
  Ідентифікатор - set
  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 <<
- Час на реалізацію : 1 год

Завдання №5 Self Practice Algotester Lab 5

- Варіант завдання : 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 O , 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.
- Час на реалізацію : 30 хв

2. Код програм з посиланням на зовнішні ресурси:

Завдання №1 Зв’язний список та Бінарні дерева 1

Посилання на файл програми у пул-запиті GitHub

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

struct Node {
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

// task 1
Node* reverse(Node* head) {
    Node* prev = nullptr;
    Node* current = head;
    Node* next = nullptr;

    while (current != nullptr) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    return prev;
}

// task 2
bool compare(Node* h1, Node* h2) {
    while (h1 != nullptr && h2 != nullptr) {
        if (h1->data != h2->data) return false;
        h1 = h1->next;
        h2 = h2->next;
    }
    return (h1 == nullptr && h2 == nullptr);
}

// task 3
Node* add(Node* n1, Node* n2) {
    Node* dummy = new Node(0);
    Node* current = dummy;
    int carry = 0;

    while (n1 != nullptr || n2 != nullptr || carry > 0) {
        int sum = carry;
        if (n1 != nullptr) {
            sum += n1->data;
            n1 = n1->next;
```

```
    while (n1 != nullptr || n2 != nullptr || carry > 0) {
        int sum = carry;
        if (n1 != nullptr) {
            sum += n1->data;
            n1 = n1->next;
        }
        if (n2 != nullptr) {
            sum += n2->data;
            n2 = n2->next;
        }

        carry = sum / 10;
        current->next = new Node(sum % 10);
        current = current->next;
    }

    return dummy->next;
}

void printList(Node* head) {
    while (head != nullptr) {
        cout << head->data << " -> ";
        head = head->next;
    }
    cout << "nullptr" << endl;
}
```

```cpp
int main() {
    // task 1
    Node* list1 = new Node(1);
    list1->next = new Node(2);
    list1->next->next = new Node(3);
    list1->next->next->next = new Node(4);

    cout << "Original list: ";
    printList(list1);

    list1 = reverse(list1);
    cout << "Reversed list: ";
    printList(list1);

    // task 2
    Node* list2 = new Node(1);
    list2->next = new Node(2);
    list2->next->next = new Node(3);

    Node* list3 = new Node(1);
    list3->next = new Node(2);
    list3->next->next = new Node(3);

    cout << "Lists are " << (compare(list2, list3) ? "equal" : "not equal") << endl;

    // task 3
    Node* num1 = new Node(9);
    num1->next = new Node(7);
    num1->next->next = new Node(3);

    Node* num2 = new Node(5);
    num2->next = new Node(4);
    num2->next->next = new Node(8);

    cout << "Sum list: ";
    Node* sum = add(num1, num2);
    printList(sum);

    return 0;
}
```

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

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

// task 4
TreeNode* create_mirror_flip(TreeNode* root) {
    if (!root) return nullptr;

    TreeNode* newRoot = new TreeNode(root->data);
    newRoot->left = create_mirror_flip(root->right);
    newRoot->right = create_mirror_flip(root->left);

    return newRoot;
}

// task 5
int tree_sum(TreeNode* root) {
    if (!root) return 0;

    if (!root->left && !root->right) return root->data;

    int leftSum = tree_sum(root->left);
    int rightSum = tree_sum(root->right);

    root->data = leftSum + rightSum;
    return root->data;
}

void printTree(TreeNode* root) {
    if (!root) return;
    printTree(root->left);
    cout << root->data << " ";
    printTree(root->right);
}
```

```cpp
int main() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->right->left = new TreeNode(6);
    root->right->right = new TreeNode(7);

    cout << "Original tree (in-order): ";
    printTree(root);
    cout << endl;

    // task 4
    TreeNode* mirrored = create_mirror_flip(root);
    cout << "Mirrored tree (in-order): ";
    printTree(mirrored);
    cout << endl;

    // task 5
    tree_sum(root);
    cout << "Tree with subtree sums (in-order): ";
    printTree(root);
    cout << endl;

    return 0;
}
```

```
Original list: 1 -> 2 -> 3 -> 4 -> nullptr
Reversed list: 4 -> 3 -> 2 -> 1 -> nullptr
Lists are equal
Sum list: 4 -> 2 -> 2 -> 1 -> nullptr
```

```
Original tree (in-order): 4 2 5 1 6 3 7
Mirrored tree (in-order): 7 3 6 1 5 2 4
Tree with subtree sums (in-order): 4 9 5 22 6 13 7
```

Завдання №2 VNS Labs 10

Посилання на файл програми у пул-запиті GitHub

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

struct Node {
    string data;
    Node* next;
    Node* prev;

    Node(string d) : data(d), next(nullptr), prev(nullptr) {}
};

class DoubleLinkedList {
private:
    Node* head;

public:
    DoubleLinkedList() : head(nullptr) {}

    ~DoubleLinkedList() {
        destroy();
    }

    bool isEmpty() const {
        return head == nullptr;
    }

    void add(const string& data) {
        Node* newNode = new Node(data);
        if (isEmpty()) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next) {
                temp = temp->next;
            }
            temp->next = newNode;
            newNode->prev = temp;
        }
    }

```

```
    void deleteFirstKElement(int k) {
        for (int i = 0; i < k && head; ++i) {
            Node* temp = head;
            head = head->next;
            if (head) {
                head->prev = nullptr;
            }
            delete temp;
        }
    }

    void addAfter(char startChar, const string& data) {
        Node* current = head;
        while (current) {
            if (!current->data.empty() && current->data[0] == startChar) {
                Node* newNode = new Node(data);
                newNode->next = current->next;
                if (current->next) {
                    current->next->prev = newNode;
                }
                current->next = newNode;
                newNode->prev = current;
                return;
            }
            current = current->next;
        }
        cout << "Елемент, що починається на '" << startChar << "', не знайдено." << endl;
    }

    void print() const {
        if (isEmpty()) {
            cout << "Список порожній." << endl;
            return;
        }
        Node* current = head;
        while (current) {
            cout << current->data << " <-> ";
            current = current->next;
        }
        cout << "None" << endl;
    }
```

```
86      void toFile(const string& filename) const {
87          ofstream outFile(filename);
88          if (!outFile) {
89              cout << "Помилка запису у файл." << endl;
90              return;
91          }
92          Node* current = head;
93          while (current) {
94              outFile << current->data << endl;
95              current = current->next;
96          }
97          outFile.close();
98      }
99
100     void fromFile(const string& filename) {
101         destroy();
102         ifstream inFile(filename);
103         if (!inFile) {
104             cout << "Помилка відкриття файлу." << endl;
105             return;
106         }
107         string line;
108         while (getline(inFile, line)) {
109             add(line);
110         }
111         inFile.close();
112     }
113
114     void destroy() {
115         while (head) {
116             Node* temp = head;
117             head = head->next;
118             delete temp;
119         }
120     }
121 };
122
```

```cpp
int main() {
    DoubleLinkedList dll;

    // створення списку
    dll.add("apple");
    dll.add("banana");
    dll.add("cherry");
    dll.add("durian");

    cout << "Список після створення:" << endl;
    dll.print();

    // видалення перших K елементів
    int k = 2;
    dll.deleteFirstKElement(k);
    cout << "\nСписок після видалення перших " << k << " елементів:" << endl;
    dll.print();

    // додавання елемента після елемента на 'c'
    dll.addAfter('c', "cranberry");
    cout << "\nСписок після додавання елемента після елемента, що починається на 'c':" << endl;
    dll.print();

    // запис в файл
    string filename = "doubly_linked_list.txt";
    dll.toFile(filename);
    cout << "\nСписок записано у файл: " << filename << endl;

    // знищення списку
    dll.destroy();
    cout << "\nСписок після знищення:" << endl;
    dll.print();

    // відновлення списку з файлу
    dll.fromFile(filename);
    cout << "\nСписок після відновлення з файлу:" << endl;
    dll.print();

    // знищення списку остаточно
    dll.destroy();
    cout << "\nСписок після остаточного знищення:" << endl;
    dll.print();

    return 0;
}
```

```
Список після створення:
apple <-> banana <-> cherry <-> durian <-> None

Список після видалення перших 2 елементів:
cherry <-> durian <-> None

Список після додавання елемента після елемента, що починається на 'c':
cherry <-> cranberry <-> durian <-> None

Список записано у файл: doubly_linked_list.txt

Список після знищення:
Список порожній.

Список після відновлення з файлу:
cherry <-> cranberry <-> durian <-> None

Список після остаточного знищення:
Список порожній.
```

i_11 > andriana_tsybukh > epic_6 > doubly_linked_list.txt

```
1  cherry
2  cranberry
3  durian
4
```

Завдання №3 Algotester Lab 5

Посилання на файл програми у пул-запиті GitHub

```cpp
#include <iostream>
#include <cstdint>

using namespace std;

void toggleRow(uint64_t &board, int row) {
    for (int col = 0; col < 8; col++) {
        board ^= (1LL << (row * 8 + col));
    }
}

void toggleCol(uint64_t &board, int col) {
    for (int row = 0; row < 8; row++) {
        board ^= (1LL << (row * 8 + col));
    }
}

int main() {
    uint64_t board;
    int n;

    cin >> board >> n;

    for (int i = 0; i < n; i++) {
        int row, col;
        cin >> row >> col;

        row--;
        col--;

        toggleRow(board, row);

        toggleCol(board, col);

        board ^= (1LL << (row * 8 + col));
    }

    cout << board << endl;

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
3 години тому	C++ 23	Зараховано	0.003	1.402	Перегляд
3 години тому	C++ 23	Неправильна відповідь 1	0.002	0.910	Перегляд

Завдання №4 Algotester Lab 7-8

Посилання на файл програми у пул-запиті GitHub

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

template <class T>
class dynamicArray {
private:
    T *data;

public:
    int size;
    int capacity;



    dynamicArray() {
        this->size = 0;
        this->capacity = 1;
        this->data = new T[1];
    }

    void insert(int index, int N, T *elements) {
        while (size + N >= capacity)
            capacity *= 2;
        T *temp = new T[capacity];

        for (int i = 0; i < index; i++)
            temp[i] = data[i];

        for (int i = 0; i < N; i++)
            temp[index + i] = elements[i];

        for (int i = index; i < size; i++)
            temp[i + N] = data[i];

        this->size += N;
        delete[] data;
        data = temp;
    }
```

```cpp
    void erase(int index, int N) {
        T *temp = new T[capacity];
        int newSize = 0;
        for (int i = 0; i < this->size; i++)
        {
            if (i < index || i >= index + N)
            {
                temp[newSize] = data[i];
                newSize++;
            }
        }
        this->size -= N;
        delete[] data;
        data = temp;
    }

    T get(int index) {
        return this->data[index];
    }

    void set(int index, T value) {
        this->data[index] = value;
    }

    void print(const string &separator) {
        for (int i = 0; i < this->size; i++) {
            cout << data[i];
            if (i < size - 1) {
                cout << separator;
            }
        }
        cout << endl;
    }
};
```

```cpp
int main() {
    dynamicArray<int> array;

    int Q;
    cin >> Q;

    while (Q--) {


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

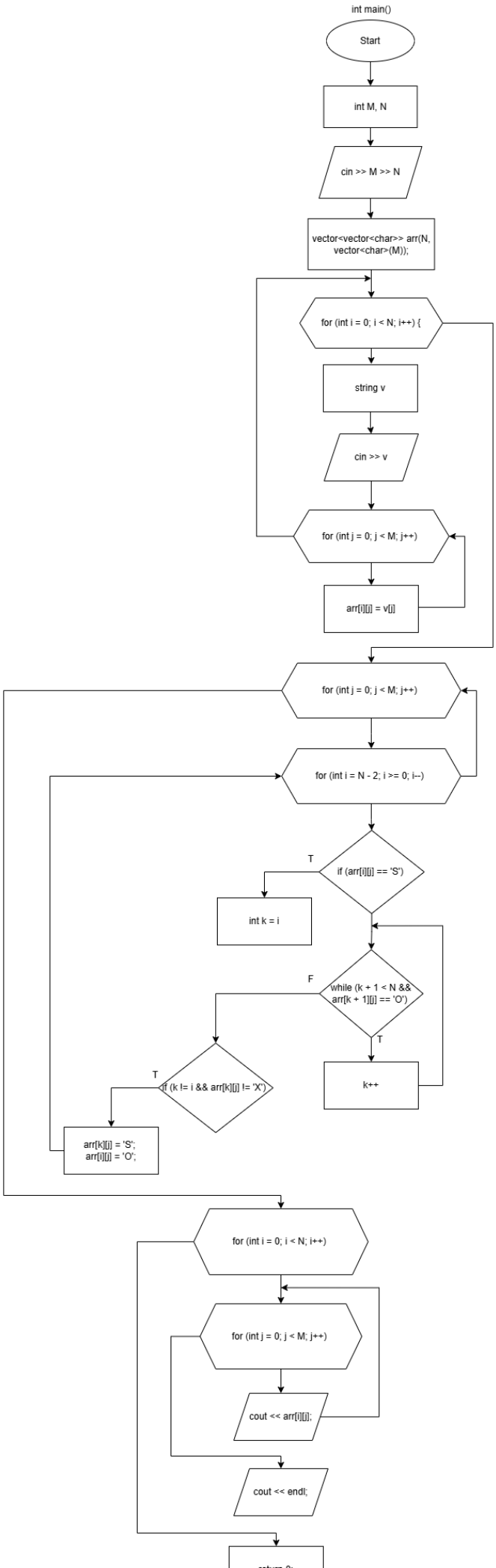
Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
2 години тому	C++ 23	Зараховано	0.006	1.270	Перегляд

Showing 1 to 1 of 1 rows

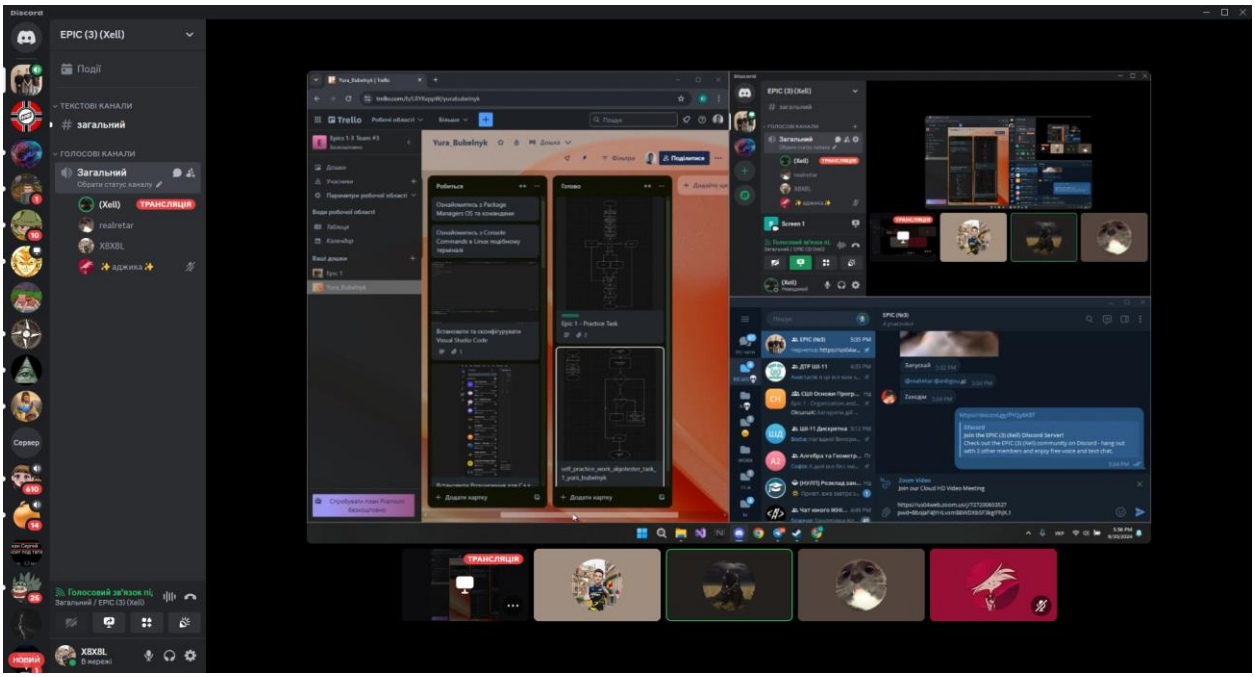
Завдання №5 Self Practice Algotester Lab 5 v2

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main() {
    int N, M;
    cin >> N >> M;

    vector<vector<char>> arr(N, vector<char>(M));

    for (int i = 0; i < N; i++) {
        string v;
        cin >> v;
        for (int j = 0; j < M; j++) {
            arr[i][j] = v[j];
        }
    }

    for (int j = 0; j < M; j++) {
        for (int i = N - 2; i >= 0; i--) {
            if (arr[i][j] == 'S') {
                int k = i;
                while (k + 1 < N && arr[k + 1][j] == 'O') {
                    k++;
                }
                if (k != i && arr[k][j] != 'X') {
                    arr[k][j] = 'S';
                    arr[i][j] = 'O';
                }
            }
        }
    }


    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            cout << arr[i][j];
        }
        cout << endl;
    }

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
2 години тому	C++ 23	Зараховано	0.038	1.902	Перегляд


int main()
Start
int M, N
cin >> M >> N
vector<vector<char>> arr(N, vector<char>(M));
for (int i = 0; i < N; i++) {
string v
cin >> v
for (int j = 0; j < M; j++)
arr[i][j] = v[j]
for (int j = 0; j < M; j++)
for (int i = N - 2; i >= 0; i--)
T
if (arr[i][j] == 'S')
int k = i
F
while (k + 1 < N && arr[k + 1][j] == 'O')
T
k++
T
if (k != i && arr[k][j] != 'X')
arr[k][j] = 'S';
arr[i][j] = 'O';
for (int i = 0; i < N; i++)
for (int j = 0; j < M; j++)
cout << arr[i][j];
cout << endl;

3. Кооперація з командою:



Зустріч в діскорді з командою. Обговорювали завдання та допомагали один одному із ними.

Висновки:

Після даної лабораторної роботи я навчилась використовувати динамічні структури даних, як створювати їх та ключові операції.