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

Виконав:
Студент групи ШІ-11
Кравченко Артем Миколайович

Тема:

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

Мета:

Ознайомитися з динамічними структурами даних, такими як черга, стек, списки та дерево, зрозуміти їхні особливості та області застосування. Також необхідно навчитися основним алгоритмам обробки цих структур, зокрема додавання, видалення та пошуку елементів, з метою ефективного управління даними в програмах.

Теоретичні відомості:

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

Індивідуальний план опрацювання теорії:

1. https://www.youtube.com/watch?v=eSxLVD5vfqM
2. https://www.youtube.com/watch?v=jUJngLO_c_0
3. https://www.youtube.com/watch?v=Yhw8NbjrSFA
4. https://www.youtube.com/watch?v=-25REjF_atI
5. https://m.youtube.com/watch?v=qBFzNW0ALxQ
6. https://www.youtube.com/watch?v=999IE-6b7_s

Виконання роботи:

Завдання 1: VNS Lab 10 - Task 1. Варіант - 24
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.
24.Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим номером. Додати по К елементів перед і після елемента із заданим ключем.

Завдання 2: Algotester lab 5 Варіант 3
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 3: Algotester lab 7 8 Варіант 2
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 <<

Завдання 4: Class Practice Work

Умови задачі:

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

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***.

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 $\implies$ 9→7→3);
- функція повертає новий список, передані в функцію списки не модифікуються.

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

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

Завдання 5: Self Practice Work

Цілий день члени виборчої комісії нудяться, спостерігають за виборами, а потім ще й підраховують голоси. Зате вже після підрахунку голосів члени виборчої комісії починають веселитися та возити протоколи з виборчих дільниць в регіональні представництва.

Звісно, кожна з виборчих дільниць з’єднана дорогою зі своїм регіональним представництвом. Та все не так просто. Так склалося, що дороги прямі, а їхні довжини є кратними тисячі метрів.

Пан Городний, який є членом виборчої комісії, задумався, скільки кілометрів цього дня проїдуть автомобілі, що возитимуть протоколи. Для того, щоб це дізнатися, він знайшов карту з позначеними на ній виборчими комісіями та відповідними їм регіональними представництвами.

Ваше завдання — написати програму, яка порахує, скільки кілометрів проїдуть автомобілі, що возитимуть протоколи в день виборів.

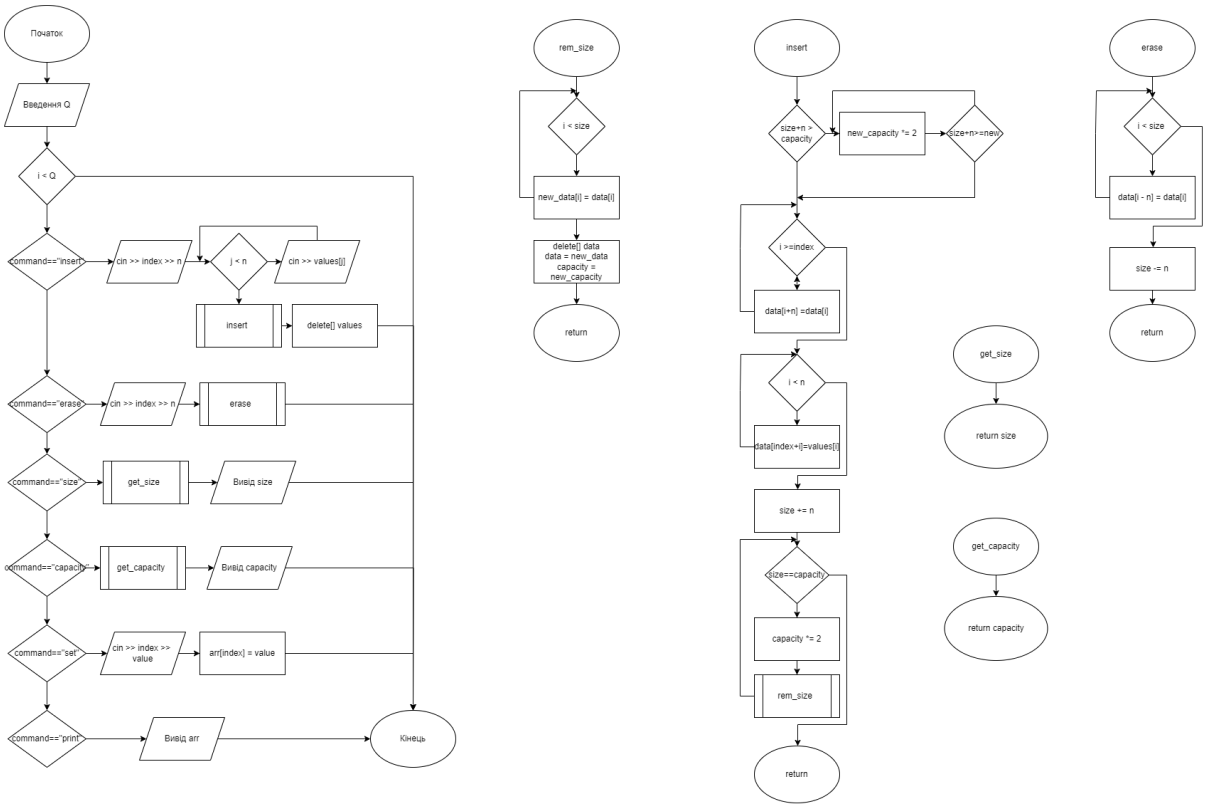
До кожної виборчої дільниці належить рівно один автомобіль, який може їхати лише до відповідного регіонального представництва.

Дизайн та планувальна оцінка часу виконання завдань:

Завдання 1: Запланований час виконання 1-1.5 години.

Завдання 2: Запланований час виконання 1 година.

Завдання 3:



Запланований час виконання 1-2 години.

Завдання 4: Запланований час виконання 2 години.

Завдання 5: Запланований час виконання 40-50 хвилин.

Код програм з посиланням на зовнішні ресурси:
Завдання 1:

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

using namespace std;

struct vuzol{
    char* key;
    vuzol* next;
    vuzol* prev;

    vuzol(const char* data){
        key = new char[strlen(data) + 1];
        strcpy(key, data);
        next = nullptr;
        prev = nullptr;
    }
    ~vuzol(){
        delete[] key;
    }
};
class d_list{
    private:
    vuzol* head;
    vuzol* tail;
    public:
    d_list() : head(nullptr), tail(nullptr){}

    void create(){
        head = nullptr;
        tail = nullptr;
    }

    void add(const char* data){
        vuzol* new_vuzol = new vuzol(data);
        if (!head)
        {
            head = tail = new_vuzol;
        }else{
```

```
        }else{
            tail->next = new_vuzol;
            new_vuzol->prev = tail;
            tail = new_vuzol;
        }

    }

    void delete_el(const char* delete_key){
        vuzol* current = head;
        while (current)
        {
            if (strcmp(current->key, delete_key) == 0)
            {
                if (current->prev)
                {
                    current->prev->next = current->next;
                }else{
                    head = current->next;
                }
                if (current->next)
                {
                    current->next->prev = current->prev;
                }else{
                    tail = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }

    }

    void print() const{
        if (!head)
        {
            cout << "Список прожній"<< endl;
```

```
            return;
        }
        vuzol* current = head;
        while(current){
            cout << current->key << " ";
            current = current->next;
        }
        cout << endl;
    }

    void save_file(const char* name_f) const{
        ofstream file (name_f);
        if (!file)
        {
            cerr << "Помилка відкриття файлу" << endl;
            return;
        }
        vuzol* current = head;
        while (current)
        {
            file << current->key << endl;
            current = current->next;
        }
        file.close();
    }

    void load_file(const char* name_f) {
        ifstream file(name_f);
        if (!file)
        {
            cerr << "Помилка відкриття файлу" << endl;
            return;
        }
        delete_list();
        char list[256];
        while (file.getline(list, 256))
        {
            add(list);
        }
```

```
        }
        file.close();
    }
    void delete_list(){
        vuzol* current = head;
        while (current)
        {
            vuzol* next = current->next;
            delete current;
            current = next;
        }
        head = tail = nullptr;
    }
    ~d_list(){
        delete_list();
    }
};

void menu(){
    cout << "\nМеню:\n";
    cout << "1. Додати елемент\n";
    cout << "2. Видалити елемент\n";
    cout << "3. Вивести список\n";
    cout << "4. Зберегти список у файл\n";
    cout << "5. Завантажити список з файлу\n";
    cout << "6. Вийти\n";
    cout << "Ваш вибір: ";
}

int main(){
    d_list list;
    char input[256];
    int choice;

    do
    {
        menu();
        cin >> choice;
```

```
152         cin >> choice;
153         cin.ignore();
154 
155         switch (choice)
156         {
157         case 1:
158             cout << "Введіть елемент: ";
159             cin.getline(input, 256);
160             list.add(input);
161             break;
162         case 2:
163             cout << "Введіть елемент для видалення: ";
164             cin.getline(input, 256);
165             list.delete_el(input);
166             break;
167         case 3:
168             list.print();
169             break;
170         case 4:
171             cout << "Введіть ім'я файлу: ";
172             cin.getline(input, 256);
173             list.save_file(input);
174             break;
175         case 5:
176             cout << "Введіть ім'я файлу: ";
177             cin.getline(input, 256);
178             list.load_file(input);
179             break;
180         case 6:
181             break;
182         default:
183             cout << "Невірний вибір. Спробуйте ще раз." << endl;
184         }
185     } while (choice != 6);
186 
187     return 0;
188 }
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/333/files#diff-de14581451ae12a4078eb95f8e2631fb321c9aeccd3736194fa236b5139c9c82

Завдання 2:

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

using namespace std;

int main(){
    int N, M, x, y;
    cin >> N >> M >> x >> y;
    x--;
    y--;

    vector<vector<int>> mount(N, vector<int>(M, -1));
    queue<pair<int, int>> q;
    q.push({x,y});
    mount[x][y] = 0;

    int max_height = max(x, N - 1 - x) + max(y, M - 1 - y);
    mount[x][y] = max_height;

    int dx[] = {-1, 1, 0, 0};
    int dy[] = {0, 0, -1, 1};

    while (!q.empty())
    {
        int cx = q.front().first;
        int cy = q.front().second;
        q.pop();
        for (int i = 0; i < 4; i++)
        {
            int nx = cx + dx[i];
            int ny = cy + dy[i];
            if (nx >= 0 && nx < N && ny >= 0 && ny < M && mount[nx][ny] == -1)
            {
                mount[nx][ny] = mount[cx][cy] - 1;
                q.push({nx, ny});
            }
        }
    }
```

```cpp
    }

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

    return 0;
}
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/333/files#diff-7da61b5114600ee5bcffd18e6cb3c87793130cdbe48776a8944433bdf7887b60

Завдання 3:

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

using namespace std;
class dymanic_aray{
    private:
    int* data;
    int capacity;
    int size;

    void grow(){
        capacity *= 2;
        int* new_data = new int[capacity];
        for (int i = 0; i < size; i++)
        {
            new_data[i] = data[i];
        }
        delete[] data;
        data = new_data;
    }
    void rem_size(int new_capacity) {
        int* newData = new int[new_capacity];
        for (int i = 0; i < size; ++i) {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
    }

    public:
    dymanic_aray() : capacity(1), size(0){
        data = new int[capacity];
    }
    ~dymanic_aray(){
        delete[] data;
    }
    void insert(int index, int n, int* values){
        if (size == capacity) {
            capacity *= 2;
```

```
            capacity *= 2;
            rem_size(capacity);
        }
        while (size + n > capacity)
        {
            grow();
        }
        for (int i = size - 1; i >= index; --i)
        {
            data[i + n] = data[i];
        }
        for (int i = 0; i < n; i++)
        {
            data[index + i] = values[i];
        }
        size += n;

    }
    void erase(int index, int n){
        for (int i = index; i < size - n; i++)
        {
            data[i] = data[i + n];
        }
        size -= n;
    }
    int get_size() const{
        return size;
    }
    int get_capacity() const{
        return capacity;
    }
    int& operator[](int index){
        return data[index];
    }
    friend ostream& operator<<(ostream& os, const dymanic_aray& arr){
```

```
74          for (int i = 0; i < arr.size; i++)
75          {
76              os << arr.data[i] << " ";
77          }
78          return os;
79      }
80  };
81
82  int main(){
83      int Q;
84      cin >> Q;
85      dymanic_aray arr;
86
87     for (int i = 0; i < Q; i++)
88      {
89          string command;
90          cin >> command;
91
92          if (command == "insert")
93          {
94              int index, n;
95              cin >> index >> n;
96              int* values = new int[n];
97              for (int j = 0; j < n; j++)
98              {
99                  cin >> values[j];
100             }
101             arr.insert(index, n, values);
102             delete[] values;
103         }else if (command == "erase")
104         {
105             int index, n;
106             cin >> index >> n;
107             arr.erase(index, n);
108         }else if (command == "size")
109         {
110             cout << arr.get_size() << endl;
```

```
110             cout << arr.get_size() << endl;
111         }else if (command == "capacity")
112         {
113             cout << arr.get_capacity() << endl;
114         }else if (command == "get")
115         {
116             int index;
117             cin >> index;
118             cout << arr[index] << endl;
119         }else if (command == "set")
120         {
121             int index, value;
122             cin >> index >> value;
123             arr[index] = value;
124         }else if (command == "print")
125         {
126             cout << arr << endl;
127         }
128     }
129     return 0;
130 }
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/333/files#diff-60257614de71d6493735d2bc5196605d5ea48d45939e1363f9666811d856389a

Завдання 4:

```cpp
#include <iostream>

using namespace std;

struct node
{
    int data;
    node* next;
    node(int value) : data(value), next(nullptr){}
};

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;
}

void print(node* head){
    node* temp = head;
    while (temp != nullptr)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

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);
}

node* add(node *n1, node *n2){
    node* result = nullptr;
    node* tail = nullptr;
    int carry = 0;

    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;
        node* new_node = new node(sum % 10);
        if (result == nullptr)
        {
            result = new_node;
            tail = new_node;
        }else{
            tail->next = new_node;
            tail = new_node;
```

```
            tail = new_node;
        }

    }
    return result;
}

struct tree{
    int val;
    tree* left;
    tree* right;

    tree(int value) : val(value), left(nullptr), right(nullptr){}
};

tree* mirror(tree* root){
    if (root == nullptr)
    {
        return nullptr;
    }
    tree* new_root = new tree(root->val);
    new_root->left = mirror(root->right);
    new_root->right = mirror(root->left);
    return new_root;
}

void print_tree(tree* root){
    if (root == nullptr)
    {
        return;
    }
    cout << root->val << " ";
    print_tree(root->left);
    print_tree(root->right);
}
```

```
110  }
111
112  int tree_sum(tree* root){
113      if (root == nullptr)
114      {
115          return 0;
116      }
117      if (root->left == nullptr && root->right == nullptr)
118      {
119          return root->val;
120      }
121      int left_sum = tree_sum(root->left);
122      int right_sum = tree_sum(root->right);
123      root->val = left_sum + right_sum;
124      return root->val;
125  }
126
127  int main(){
128      cout << "Завдання 1:" << endl;
129      node* head = new node(1);
130      head->next = new node(2);
131      head->next->next = new node(3);
132      head->next->next->next = new node(4);
133      head->next->next->next->next = new node(5);
134
135      cout << "Заданий список:" << endl;
136      print(head);
137      head = reverse(head);
138
139      cout << "Обернений список:" << endl;
140      print(head);
141
142      cout << "Завдання 2:" << endl;
143      node* list1 = new node(1);
144      list1->next = new node(2);
145      list1->next->next = new node(3);
146
147      node* list2 = new node(1);
```

```
147     node* list2 = new node(1);
148     list2->next = new node(2);
149     list2->next->next = new node(3);
150     cout << "Порівняння списків: " << (compare(list1, list2) ? "списки рівні" : "списки не рівні") << endl;
151
152     cout << "Завдання 3:" << endl;
153     node* num1 = new node(9);
154     num1->next = new node(9);
155     num1->next->next = new node(9);
156
157     node* num2 = new node(1);
158
159     node* sum = add(num1, num2);
160     cout << "Результат: " << endl;
161     print(sum);
162
163     cout << "Завдання 4:" << endl;
164     tree* root = new tree(1);
165     root->left = new tree(2);
166     root->right = new tree(3);
167     root->left->left = new tree(4);
168     root->left->right = new tree(5);
169
170     cout << "Задане дерево:" << endl;
171     print_tree(root);
172     cout << endl;
173
174     cout << "Завдання 5:" << endl;
175     tree* root_sum = new tree(1);
176     root_sum->left = new tree(2);
177     root_sum->right = new tree(3);
178     root_sum->left->left = new tree(4);
179     root_sum->left->right = new tree(5);
180
181     tree_sum(root_sum);
182     cout << "Результат: " << endl;
183     print_tree(root_sum);
184     return 0;
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/333/files#diff-956d3a7a60e60788ef8824e2a4326e48845cb9e144c7c4aae8d050e6796bcb26

Завдання 5:

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main(){
    int n;
    int x1, x2, y1, y2;

    cin >> n;
    int sum = 0;
    for (int i = 0; i < n; i++)
    {
        cin >> x1 >> y1 >> x2 >> y2;
        int oparstion = sqrt(pow(x2 -x1, 2) + pow(y2 - y1, 2));
        sum += oparstion;
    }

    cout << sum << endl;

    return 0;
}
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/333/files#diff-8743f7088bee665e25c0545d273848d5b78b57566cb689af66ed003ca274b4dc

Результат виконання завдань, тестування та фактично витрачений час:

Завдання 1:

```
Меню:
1. Додати елемент
2. Видалити елемент
3. Вивести список
4. Зберегти список у файл
5. Завантажити список з файлу
6. Вийти
Ваш вибір: 1
Введіть елемент: Hello world

Меню:
1. Додати елемент
2. Видалити елемент
3. Вивести список
4. Зберегти список у файл
5. Завантажити список з файлу
6. Вийти
Ваш вибір: 3
Hello world

Меню:
1. Додати елемент
2. Видалити елемент
3. Вивести список
4. Зберегти список у файл
5. Завантажити список з файлу
6. Вийти
Ваш вибір: 4
```

Фактично витрачений час: 2 години.

Завдання 2:

Фактично витрачений час: 1.5 години.

Завдання 3:

```
12

size
0
capacity
1

insert 0 2
100 100

size
2
capacity
4

insert 0 2
102 102

size
4
capacity
8

insert 0 2
103 103

size
6
capacity
8

print
103 103 102 102 100 100
PS D:\VS Code project\C ++>
```

Фактично витрачений час: 4 години.

Завдання 4:

```
Завдання 1:
Заданий список:
1 2 3 4 5
Обернений список:
5 4 3 2 1
Завдання 2:
Порівняння списків: списки рівні
Завдання 3:
Результат:
0 0 0 1
Завдання 4:
Задане дерево:
1 2 4 5 3
Завдання 5:
Результат:
12 9 4 5 3
PS D:\VS Code project\C ++>
```

Фактично витрачений час: 1.5 години.

Завдання 5:

```
3
-3 2 1 5
0 0 6 8
4 7 6 7
17
PS D:\VS Code project\C ++>
```

Фактично витрачений час: 20 хвилин.

Зустріч з комадою:

Висновок: У цій лабораторній роботі я навчився працювати з динамічними структурами даних — чергою, стеком, списками та деревами, а також застосовувати алгоритми для їхньої обробки. Це дало мені розуміння ефективного управління даними в програмах.