Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №6**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

***Виконав:***

студент групи ШІ-14

Мруць Остап Мар’янович

**Тема роботи:**

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

**Мета роботи:**

**Теоретичні відомості:**

* Теоретичні відомості з переліком важливих тем:
* Тема №1: Динамічні структури
* Тема №2: Алгоритми обробки динамічних структур
* Індивідуальний план опрацювання теорії:
* Тема №1: Динамічні структури
* Джерела Інформації
* Лекції
* Лекційний матеріал на ВНС
* Що опрацьовано:
* Черга
* Стек
* Списки
* Дерево
* Статус: Ознайомлений
* Тема №2: Алгоритми обробки динамічних структур
* Джерела Інформації:
* Лекції
* Лекційний матеріал на ВНС
* Що опрацьовано:
* Алгоритми для вставки, вилучення, пошуку та інших операцій
* Linked list
* Статус: Ознайомлений

**Виконання роботи:**

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

Завдання №1 - Theory Education Activities

* Вивчення теорії

Завдання №2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)

* Створення діаграм для програм в draw io

Завдання №3 - Lab# programming: VNS Lab 10

* Варіант № 4
* Деталі завдання: Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом. Для кожного варіанту розробити такі функції: створення списку, додавання елемента в список (у відповідності зі своїм варіантом), знищення елемента зі списку (у відповідності зі своїм варіантом), друк списку, запис списку у файл, знищення списку, відновлення списку з файлу. Варіант: 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
* Важливі деталі для врахування в імплементації програми: Функція може створювати порожній список, а потім додавати в нього елементи, після виведення функцій потрібно роздрукувати список, якщо при виведення список порожній, то потрібно вивести відповідне повідомлення

Завдання №4 - Lab# programming: Algotester Lab 5

* Варіант № 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. 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.
* Важливі деталі для врахування в імплементації програми: 1≤N,M≤103, 1≤x≤N, 1≤y≤M, спочатку варто знайти число найвищої точки, після цього згенерувати гору буде простіше

Завдання №5 - Lab# programming: Algotester Lab 7-8

* Варіант № 2
* Деталі завдання: Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".  
  Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи. Вам будуть поступати запити такого типу: **Вставка**: Ідентифікатор – insert. Ви отримуєте ціле число index елемента, на місце якого робити вставку. Після цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити. У третьому рядку N цілих чисел - масив, який треба вставити на позицію index. **Видалення**: Ідентифікатор – erase. Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити. **Визначення розміру**: Ідентифікатор – size. Ви не отримуєте аргументів. Ви виводите кількість елементів у динамічному масиві. **Визначення кількості зарезервованої пам’яті**: Ідентифікатор – capacity. Ви не отримуєте аргументів. Ви виводите кількість зарезервованої пам’яті у динамічному масиві. Ваша реалізація динамічного масиву має мати фактор росту ([Growth factor](https://en.wikipedia.org/wiki/Dynamic_array#Growth_factor)) рівний 2. **Отримання значення** i-го елементу: Ідентифікатор – get. Ви отримуєте ціле число - index, індекс елемента. Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора [][]. **Модифікація значення** i-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 <<
* Важливі деталі для врахування в імплементації програми: 0≤Q≤105, 0≤li≤105, ∥l∥≤105. Для отримання 100% балів ця структура має бути написана як [шаблон класу](https://cplusplus.com/doc/oldtutorial/templates/), у якості параметру використати int

Завдання №6 - Practice# programming: Class Practice Task

* Деталі завдання №1: 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
* Деталі завдання №2: bool compare(Node \*h1, Node \*h2); 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
* Важливі деталі для врахування в імплементації програми: Потрібно зробити структуру Node для даних файлу, для кожної дії потрібно створити відповідну функцію
* Захист звіту.

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

Програма №1 VNS Lab 10

* Важливі деталі для врахування в імплементації програми: Завдання можна виконати в одному файлі для того щоб використовувати спільні функції. Записи в списку містять поля типу int, що треба вказати в структурі

Програма №2 Algotester Lab 5

* Важливі деталі для врахування в імплементації програми: Завдяки функції abs можна знайти максимальне значення гори, а потім завдяки цьому числу і масиву можна сформувати всі решта числа

Програма №3 Algotester Lab 7-8

* Важливі деталі для врахування в імплементації програми: capacity завжди має бути більше ніж size, якщо ж ні, то потрібно його домножити на growthFactor

Програма №4 Class Practice Task

* Важливі деталі для врахування в імплементації програми: Для зручності list був створений самою програмою, провірку можна робити до nullptr, що означає кінець списку.

**3. Код програм з посиланням на зовнішні ресурси:**

Завдання №1: VNS Lab 10

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* prev;

Node\* next;

};

class DoublyLinkedList {

private:

Node\* head;

Node\* tail;

public:

DoublyLinkedList() : head(nullptr), tail(nullptr) {}

void addToStart(int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->prev = nullptr;

newNode->next = head;

if (head != nullptr) {

head->prev = newNode;

} else {

tail = newNode;

}

head = newNode;

}

void deleteElements(int k, int\* positions, int size) {

Node\* current = head;

while (current != nullptr && k > 0) {

Node\* temp = current;

current = current->next;

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

if (temp != nullptr && temp->data == positions[i]) {

if (temp->prev != nullptr) {

temp->prev->next = temp->next;

} else {

head = temp->next;

}

if (temp->next != nullptr) {

temp->next->prev = temp->prev;

} else {

tail = temp->prev;

}

delete temp;

k--;

}

}

}

}

void printList() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

cout << endl;

}

void deleteList() {

Node\* current = head;

while (current != nullptr) {

Node\* temp = current;

current = current->next;

delete temp;

}

head = nullptr;

tail = nullptr;

}

};

int main() {

DoublyLinkedList myList;

for (int i = 5; i > 0; --i) {

myList.addToStart(i);

}

cout << "Список перед видаленням: ";

myList.printList();

int positionsToDelete[] = {2, 4};

int k = sizeof(positionsToDelete) / sizeof(positionsToDelete[0]);

myList.deleteElements(k, positionsToDelete, k);

int newElements[] = {9, 8};

int numNewElements = sizeof(newElements) / sizeof(newElements[0]);

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

myList.addToStart(newElements[i]);

}

cout << "Список після видалення та додавання: ";

myList.printList();

myList.deleteList();

return 0;

}

*Рисунок 2: Код завдання з Algotester 5 2*

#include <cstdio>

#include <vector>

using namespace std;

int main() {

int N, M;

scanf("%d %d", &N, &M);

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

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

char row[M + 1];

scanf("%s", row);

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

cave[i][j] = row[j];

}

}

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

int emptyCellIdx = N - 1;

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

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

cave[i][j] = 'O';

cave[emptyCellIdx--][j] = 'S';

} else if (cave[i][j] == 'X') {

emptyCellIdx = i - 1;

}

}

}

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

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

printf("%c", cave[i][j]);

}

printf("\n");

}

return 0;

}

*Рисунок 3: Код завдання з Algotester №5*

Завдання №3: Algotester Lab 7-8

##include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* left;

Node\* right;

Node(int value) {

data = value;

left = nullptr;

right = nullptr;

}

};

class BinarySearchTree {

private:

Node\* root;

Node\* insertRecursive(Node\* current, int value) {

if (current == nullptr) {

return new Node(value);

}

if (value < current->data) {

current->left = insertRecursive(current->left, value);

} else if (value > current->data) {

current->right = insertRecursive(current->right, value);

}

return current;

}

bool containsRecursive(Node\* current, int value) {

if (current == nullptr) {

return false;

}

if (value == current->data) {

return true;

}

if (value < current->data) {

return containsRecursive(current->left, value);

} else {

return containsRecursive(current->right, value);

}

}

int calculateSize(Node\* current) {

if (current == nullptr) {

return 0;

}

return 1 + calculateSize(current->left) + calculateSize(current->right);

}

void printTree(Node\* current) {

if (current == nullptr) {

return;

}

printTree(current->left);

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

printTree(current->right);

}

public:

BinarySearchTree() {

root = nullptr;

}

void insert(int value) {

root = insertRecursive(root, value);

}

bool contains(int value) {

return containsRecursive(root, value);

}

int size() {

return calculateSize(root);

}

void print() {

printTree(root);

cout << endl;

}

};

int main() {

int Q;

cin >> Q;

BinarySearchTree bst;

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

string operation;

cin >> operation;

if (operation == "insert") {

int value;

cin >> value;

bst.insert(value);

} else if (operation == "contains") {

int value;

cin >> value;

if (bst.contains(value)) {

cout << "Yes" << endl;

} else {

cout << "No" << endl;

}

} else if (operation == "size") {

cout << bst.size() << endl;

} else if (operation == "print") {

bst.print();

}

}

return 0;

}

Рисунок 4: Код завдання з Algotester №7-8

Завдання №4: Class Practice Task

##include <iostream>

struct Node {

int value;

Node\* next;

Node(int val) : value(val), next(nullptr) {}

};

Node\* reverse(Node\* head) {

Node \*prev = nullptr, \*curr = head, \*next = nullptr;

while (curr != nullptr) {

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

return prev;

}

void printList(Node\* head) {

while (head != nullptr) {

std::cout << head->value << " ";

head = head->next;

}

std::cout << std::endl;

}

int main() {

Node\* head = new Node(1);

head->next = new Node(2);

head->next->next = new Node(3);

std::cout << "Оригінальний список: ";

printList(head);

head = reverse(head);

std::cout << "Реверсований список: ";

printList(head);

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

*Рисунок 5: Код програми з практичної роботи*

Завдання №5: Self Practice Task

##include <iostream>

struct Node {

int data;

Node\* next;

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

};

class LinkedList {

private:

Node\* head;

public:

LinkedList() : head(nullptr) {}

void append(int value) {

Node\* newNode = new Node(value);

if (head == nullptr) {

head = newNode;

return;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

void display() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

std::cout << std::endl;

}

};

int main() {

LinkedList list;

list.append(5);

list.append(10);

list.append(15);

list.display();

return 0;

}

*Рисунок 6: Код програми для самостійного опрацювання*

**4. Результати виконання завдань:**

**VNS LAB 10v19:**



**Class practice:**  


**Self practice:**  


**Висновки:**

Протягом лабораторних та практичних робіт №6було вивчено динамічні структури, такі як черга, стек, списки і дерево, а також алгоритми обробки динамічних структур