Звіт

про виконання лабораторної роботи №7.2

«Контейнери-списки »

з дисципліни

«Об’єктно-орієнтоване програмування»

студента групи РІ-12

Синчука Івана Романовича

**Мета роботи:** **Навчитися використовувати стандартні контейнери-списки.**

**Умова завдання:**

**Зображення, що містить текст, Шрифт, знімок екрана, ряд

Вміст, створений ШІ, може бути неправильним.**

**UML-діаграма класів:**



**Структурна схема програми:**



**Текст програми:**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <Windows.h>

using namespace std;

// Структура вузла однонаправленого списку

struct Node {

int data; // Дані вузла (ціле число)

Node\* next; // Вказівник на наступний вузол

Node(int val) : data(val), next(nullptr) {} // Конструктор для ініціалізації

};

// Функція для виведення списку на екран

void printList(Node\* head) {

cout << "Список: ";

// Проходимо по всіх вузлах списку

while (head != nullptr) {

cout << head->data << " "; // Виводимо дані поточного вузла

head = head->next; // Переходимо до наступного вузла

}

cout << endl;

}

// Функція для створення списку з випадковими числами

Node\* createList(int size) {

if (size <= 0) return nullptr;

Node\* head = new Node(rand() % 100 - 50);

Node\* current = head; // Вказівник для проходження по списку

// Додаємо решту вузлів

for (int i = 1; i < size; ++i) {

current->next = new Node(rand() % 100 - 50); // Створюємо новий вузол

current = current->next; // Пересуваємо вказівник на новий вузол

}

return head;

}

// Функція для створення тестового відсортованого списку

Node\* createSortedList() {

Node\* head = new Node(1);

head->next = new Node(2);

head->next->next = new Node(2);

head->next->next->next = new Node(5);

return head;

}

// Функція для перевірки, чи список впорядкований за неспаданням

bool isNonDecreasing(Node\* head) {

// Список з 0 або 1 елемента завжди впорядкований

if (head == nullptr || head->next == nullptr) {

return true;

}

Node\* current = head; // Вказівник для проходження по списку

// Перевіряємо всі сусідні пари

while (current->next != nullptr) {

// Якщо поточний елемент більший за наступний - список не впорядкований

if (current->data > current->next->data) {

return false;

}

current = current->next; // Переходимо до наступного вузла

}

return true;

}

// Функція для видалення списку та звільнення пам'яті

void deleteList(Node\* head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next; // Переходимо до наступного

delete temp;

}

}

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

srand(static\_cast<unsigned>(time(nullptr))); // Ініціалізація генератора випадкових чисел

// Тест 1: Список з випадковими числами

int n;

cout << "Введіть розмір списку n = ";

cin >> n;

Node\* randomList = createList(n); // Створюємо випадковий список

cout << "Створений випадковий ";

printList(randomList); // Виводимо його

// Перевіряємо чи впорядкований

if (isNonDecreasing(randomList)) {

cout << "Результат: впорядкований за неспаданням.\n\n";

}

else {

cout << "Результат: НЕ впорядкований за неспаданням.\n\n";

}

// Тест 2: (тестовий приклад)

Node\* sortedList = createSortedList();

cout << "Тестовий відсортований ";

printList(sortedList);

if (isNonDecreasing(sortedList)) {

cout << "Результат: впорядкований за неспаданням.\n\n";

}

else {

cout << "Результат: НЕ впорядкований за неспаданням.\n\n";

}

// Очищення пам'яті (видалення обох списків)

deleteList(randomList);

deleteList(sortedList);

return 0;

}

**Результат unit-тесту:**

**Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Вміст, створений ШІ, може бути неправильним.**

**Посилання на git-репозиторій з проектом:**

**Висновок: На цій лабораторній я навчився використовувати стандартні контейнери-списки.**