Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Інформаційних Систем та Технологій

Лабораторна робота №1

З дисципліни «Основи розробки програмного забезпечення на платформі

Microsoft.NET»

На тему: «Проектування та створення веб-застосунку в межах

багаторівневої архітектури»

Виконав студент гр. ІТ-02

Терешкович М.О

Перевірив

Вовк Є. А.

Київ

2023

**Мета:**

Навчитися проектувати та реалізовувати узагальнені типи, а також типи з підтримкою подій.

**Завдання:**

1. Розробити клас власної узагальненої колекції, використовуючи стандартні інтерфейси колекцій із бібліотек System.Collections та System.Collections.Generic. Стандартні колекції при розробці власної не застосовувати. Для колекції передбачити методи внесення даних будь-якого типу, видалення, пошуку та ін. (відповідно до типу колекції).
2. Додати до класу власної узагальненої колекції підтримку подій та обробку виключних ситуацій.
3. Опис класу колекції та всіх необхідних для роботи з колекцією типів зберегти у динамічній бібліотеці.
4. Створити консольний додаток, в якому продемонструвати використання розробленої власної колекції, підписку на події колекції.

**Варіант 8:**

****

**Реалізація:**

**Лістинг програми:**

using Lab1\_\_\_Ring\_collection;

using System;

using System.Collections;

using System.Collections.Generic;

class Program

{

static void Main()

{

CircularLinkedList<int> circularList = new CircularLinkedList<int>();

circularList.Add(1);

circularList.Add(2);

circularList.Add(3);

circularList.Add(5);

circularList.Add(-2);

circularList.Add(100);

circularList.Add(0);

circularList.Print();

bool isCircularList = circularList.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList);

circularList.Delete(0);

circularList.Print();

isCircularList = circularList.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList);

circularList.Add(666);

circularList.Add(843);

circularList.Print();

isCircularList = circularList.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList);

Console.WriteLine("-------------------------------------------");

CircularLinkedList<string> circularList2 = new CircularLinkedList<string>();

circularList2.Add("2");

circularList2.Add("3");

circularList2.Add("32");

circularList2.Add("adw");

circularList2.Add("apple");

circularList2.Add("hundred");

circularList2.Add("trousers");

circularList2.Print();

bool isCircularList2 = circularList2.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList2);

circularList2.Delete("adw");

circularList2.Print();

isCircularList = circularList2.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList2);

circularList2.Add("###");

circularList2.Add("0asd2");

circularList2.Print();

isCircularList = circularList2.IsCircularList();

Console.WriteLine("Is it circular Linked List? " + isCircularList2);

}

}

using System;

using System.Collections.Generic;

namespace Lab1\_\_\_Ring\_collection

{

public class CircularLinkedList<T>

{

private class Node

{

public T Value;

public Node Next;

}

private Node head;

private Node tail;

private int count;

public int Count => count;

public bool IsCircularList()

{

if (count == 0 || count == 1)

return false;

Node current = head;

for (int i = 0; i < count; i++)

{

if (current == null || current.Next == null)

return false;

if (current.Next == head)

return true;

current = current.Next;

}

return false;

}

public void Add(T item)

{

Node newNode = new Node

{

Value = item,

Next = null

};

if (head == null)

{

head = newNode;

tail = newNode;

newNode.Next = head;

}

else

{

tail.Next = newNode;

tail = newNode;

tail.Next = head;

}

count++;

}

public bool Delete(T item)

{

if (count == 0)

throw new InvalidOperationException("Empty!");

if (count == 1 && head.Value.Equals(item))

{

head = null;

tail = null;

count--;

return true;

}

Node current = head;

Node previous = null;

bool found = false;

do

{

if (current.Value.Equals(item))

{

found = true;

break;

}

previous = current;

current = current.Next;

} while (current != head);

if (found)

{

if (current == head)

{

head = current.Next;

tail.Next = head;

}

else if (current == tail)

{

tail = previous;

tail.Next = head;

}

else

{

previous.Next = current.Next;

}

count--;

}

return found;

}

public void Print()

{

if (count == 0)

{

Console.WriteLine("Empty!");

return;

}

Node curr = head;

for (int i=0; i < count; i++)

{

if (i == 0 || i == count-1)

{

Console.Write(curr.Value+"("+ head.Value +"/"+ tail.Value + ")" + " -> ");

curr = curr.Next;

}

else

{

Console.Write(curr.Value + " -> ");

curr = curr.Next;

}

}

Console.WriteLine(".....");

}

}

}

**Вивід програми:**

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

Автоматично згенерований опис**

**Висновок:**

На даній лабараторній роботі, я навчився розробляти алгоритм для взаємодії з кільцевим списком та навчився проектувати та реалізовувати узагальнені типи, а також типи з підтримкою подій.