МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 3**

з дисципліни « Алгоритми і структури даних »

*назва дисципліни*

на тему:« ДВОРІВНЕВІ СТРУКТУРИ. СПИСКИ СПИСКІВ. ВБУДОВАНІ КОЛЕКЦІЇ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: канд. техн. наук, доцент на кафедрі 603 Волобуєва Ліна Олексіївна

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**Мета роботи** − вивчення вбудованих типів вбудованих колекцій в узагальненій та в неузагальненій реалізаціях на прикладі створення списку підсписків.

Постановка задачі

1. Описати вказаний абстрактний тип даних список підсписків і основні операції роботи з ним на абстрактному і логічному рівнях:

* початкова ініціалізація структури (список порожній).
* додавання елементу в список/підсписок
* видалення елементу зі списку/підсписку.
* перегляд першого елементу в списку/підсписку.
* перевірка списку/підсписку- порожній він або не порожній
* виведення структури на екран.

1. Відповідно до варіанту створити структуру даних для організації дворівневої структури даних (списки підсписків).
2. Для реалізації структури використовувати вбудовані типи в узагальненій і неузагальнених реалізаціях (Stack, Stack , Queue, Queue , ArrayList, List , LinkedList). Як елементи структур використати наступні типи: (class або struct), що мають поле ключа та поле інформації для підсписків (нижній рівень) і поле ключа для списку (верхній рівень).
3. Створити додаток, який демонструє роботу операцій, перелічених вище (див. п.1.). Для створення додатку дозволяється використовувати лише ті операції, що притаманні заданій структурі. Наприклад, заборонено отримувати доступ до елементу із довільною позицією у черзі.
4. Оформити звіт.

Зміст звіту

Зміст звіту з лабораторної роботи №3 має включати наступні підрозділи:

1. Абстрактне представлення структури.
2. Представлення даних в графічному виді.
3. Проектування необхідних класів і структур.
4. Алгоритм для кожного методу, використовуючи один з наступних видів представлення алгоритмів: псевдокод або блок-схему.
5. Лістинг програми та екранні форми результату роботи програми.
6. Висновки

Варіант 9 індивідуального завдання до лабораторної роботи 3:

Список типу ArrayList та підсписки Stack <T>

хід роботи

Опис програми:

Мова програмування: С#, операційна система Windows 11 Prо, Версія 23H2, Збірка ОС 22621.1325, процесор: Apple Silicon M1 Pro 3.20 GHz (ядер: 4), компілятор: Microsoft Visual Studio Community 2022 (64-розрядна версія ARM).

Абстрактне представлення структури

Стек (Stack) – АТД з доступом до елементів за правилом «останній прийшов – першим вийшов» (Last In – First Out, LIFO)

Опис АТД стек:

Top - тип стек, el – елемент стеку.

Операції:

* 1. Створити (New):  → Up\_Sp
  2. Додавання елементу в підсписок (Push): Up\_Sp x i x D\_St → Up\_Sp;
  3. Додавання елементу в список (Add): Up\_Sp x D\_St → Up\_Sp
  4. Видалення елементу з підсписку (Pop): Up\_Sp x i x D\_St → Up\_Sp;
  5. Видалення елементу зі списку (Remove(At)): Up\_Sp x i → Up\_Sp;
  6. Перший елемент в підсписку (Peek): Up\_Sp x i x D\_St → El\_Down (int)

Графічне представлення структури

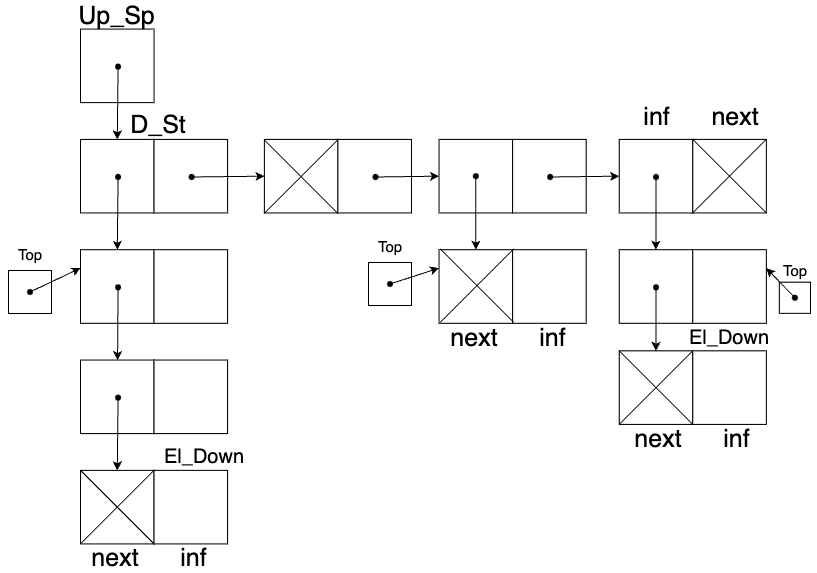


Рисунок 1 – Структура зі списком ArrayList та підсписком Stack <T>

Проектування необхідних класів і структур.

Клас El\_Down потрібен для ініціалізації елементів в підсписку; містить інформацію та номер списку - ключ.

public class El\_Down // елемент підсписку

{

public int key;

public string inf;

public El\_Down(int key, string inf)

{

this.key = key;

this.inf = inf;

}

public override string ToString()

{

string res = "Значення=" + key.ToString() + "; inf=" + inf + " ";

return res;

}

}

Клас Down містить в собі інформацію про номер стеку, та універсальний стек типу El\_Down. Також містить методи для реалізації дій із даними, які зазначені у постановці завдання (фрагмент коду наведено нижче).

public class Down // елемент типу стек <T> - цілий підсписок

{

public int L\_inf;

public Stack<El\_Down> Down\_St;

public Down(int inf)

{

L\_inf = inf;

Down\_St = new Stack<El\_Down>();

}

public void Add\_El\_to\_Down(El\_Down El)

{

Down\_St.Push(El);

}

public void Remove\_El\_Down()

{

if (Down\_St.Count > 0)

Down\_St.Pop();

else throw new Exception("Підсписок пустий");

}

// Возвращает верхний элемент подсписка без удаления

public El\_Down Peek()

{

if (Down\_St.Count > 0)

return Down\_St.Peek();

else throw new Exception("Підсписок пустий");

}

// Возвращает количество элементов в подсписке

public int Count()

{

return Down\_St.Count;

}

// Очищает весь подсписок

public void Clear()

{

Down\_St.Clear();

}

public override string ToString()

{

string res = "\nСписок " + L\_inf.ToString() + "\t--> V V V V V V V V V V V V V V\n";

foreach (El\_Down El in Down\_St)

res += El.ToString();

return res + '\n';

}

}

Клас Up\_ArLst містить список ArrayList. Також включає методи для реалізації дій із даними, які зазначені у постановці завдання (фрагмент коду наведено нижче).

public class Up\_ArLst // основний список Ерей ліст, який містить в собі стеки <T>

{

public ArrayList UP\_ArLst;

public Up\_ArLst()

{

UP\_ArLst = new ArrayList();

}

public void Add\_Down\_to\_ArLst(Down El\_St)

{

UP\_ArLst.Add(El\_St);

}

public void Remove\_Down\_St(Down El\_St)

{

if (El\_St.Down\_St.Count != 0)

El\_St.Down\_St.Clear();

UP\_ArLst.Remove(El\_St);

}

public void Remove\_Down\_El(int Arr\_id)

{

if (Arr\_id >= 0 && Arr\_id < UP\_ArLst.Count)

{

Down down = UP\_ArLst[Arr\_id] as Down;

if (down != null)

{

if (down.Down\_St.Count > 0)

{

down.Remove\_El\_Down(); // Удаляем элемент из стека

}

else

{

throw new Exception("Стек пустий.");

}

}

else throw new Exception("Елемент з вказаним індексом в основному списку не підписок");

}

else throw new Exception("Індекс елемента в основному списку недопустимий");

}

public void Remove\_UP(int id)

{

if (id >= 0 && id < UP\_ArLst.Count)

{

UP\_ArLst.RemoveAt(id);

}

else

{

throw new Exception("Індекс елемента в основному списку недопустимий.");

}

}

public void Reverse\_UP()

{

UP\_ArLst.Reverse();

}

public override string ToString()

{

string res = "";

if (UP\_ArLst.Count != 0)

foreach (object El in UP\_ArLst)

res += El.ToString();

return res;

}

}

Методи подані у вигляді псевдокоду

Процедура ВвідЕлементу підсписку (змінна кількості списків, режим роботи)

Вивести "Введіть кількість елементів для додавання в підсписок:"

прочитати кількість елементів для додавання в стек з консолі як ціле число

Створити новий об'єкт MyDown типу Down з параметром (кількість списків+1)

Для кожного i від 1 до кількості елементів для додавання до стеку

Якщо режим = 1 то

Створити об'єкт r типу Random

k = Згенерувати\_Випадкове\_Число\_В\_Межах(0, 20)

Інакше, Якщо режим = 2 то

k = Ввід\_Числа("Введіть число для додавання в підсписок:", Int32.MinValue, Int32.MaxValue)

Вивести "Додано елемент " + число для додавання до стеку

Створити новий об'єкт MyElDown типу El\_Down з параметрами (k, "SP" + count\_SP + "PSp" + k)

Додати Елемент\_До\_Підсписку(MyElDown, MyDown)

Завершити процедуру і повернути MyDown

Метод перевірки введеного значення з клавіатури за діапазоном:

Функція Ввід\_Числа(текст, нижня\_межа, верхня\_межа) повертає ціле число

Поки істина

Якщо Спроба\_Перетворення\_Рядка\_У\_Число(Console.ReadLine(), input), а також input <= верхня\_межа і input >= нижня\_межа То

Повернути input

Інакше

Вивести "Помилка введення числа. Спробуйте ще раз"

Метод перетворення елементу підсписку на рядок:

В\_Рядок()

змінна результату = «значення»+ значення елементу стеку+ «інфо»+інфо + пробіл

повернути рядок результату

Метод додавання елементу до підсписку:

Додавання елементу до підсписку (Елемент типу елемент підсписку)

Підсписок\_стек додати (елемент типу елемент підсписку);

Метод видалення елементу з підсписку:

Додавання елементу до підсписку (Елемент типу елемент підсписку)

Якщо кількість елементів підсписку більше 0

Підсписок\_стек додати (елемент типу елемент підсписку);

інакше виключення «Стек пустий»

Метод переглянути верхній елемент підсписку:

Видалити елемент з підсписку ()

Якщо кількість елементів підсписку більше 0

повернути переглянтий елемент підсписку

інакше виключення «Підсписок пусти»

Метод кількості елементів в підсписку:

Порахувати кількість елементів в підсписку()

повернути кількість елементів підсписку

Метод очищення елементів в підсписку:

очищення елементів в підсписку()

Підсписок очистити();

Метод перетворення підсписку в рядок:

перетворити в рядок()

результат = «Список»+номер списку

для кожного елементу підсписку в підсписку

результат = елемент перетворити на рядок();

повернути результат + перехід на новий рядок;

Метод додавання підсписку в список:

додати підсписок в список(підсписок)

список додати(підсписок)

Метод видалення підсписку:

видалення підсписку(підсписок)

якщо підсписок не пустий

очистити підсписок();

Список прибрати (підсписок)

Метод видалення елементу в підсписку:

видалення елементу в підсписку(індекс списку)

якщо індекс більше або дорівнює 0 і меньше кількості списків

підсписок = список [індекс] є типом підсписку

якщо (підсписок не null)

якщо (кількість елментів в підсписку більше 0)

видаляємо елемент з підсписку

інакше виключення «підсписок пустий»

накше виключення «елемент не підсписок»

інакше «помилка індекса списку»

Метод видалення списку:

видалення списку(індекс)

якщо індекс більше або дорівнює 0 і меньше кількості списків

видалити список в (індексі)

інакше виключення «недопустимий індекс»

Метод перетворення списку в рядок:

список в рядок()

якщо (кількість елементів в списку не 0)

для кожного елементу в списку

результат додати і присвоїти елемент перетворений в рядок

повернути результат

Тестування програми

Створемо 5 списків з різними значеннями – введеними і випадковими. Процес додавання зображено на рисунках 2 – 4. Також будемо вводити і неправильні значення, які не мають бути допущені до програми.

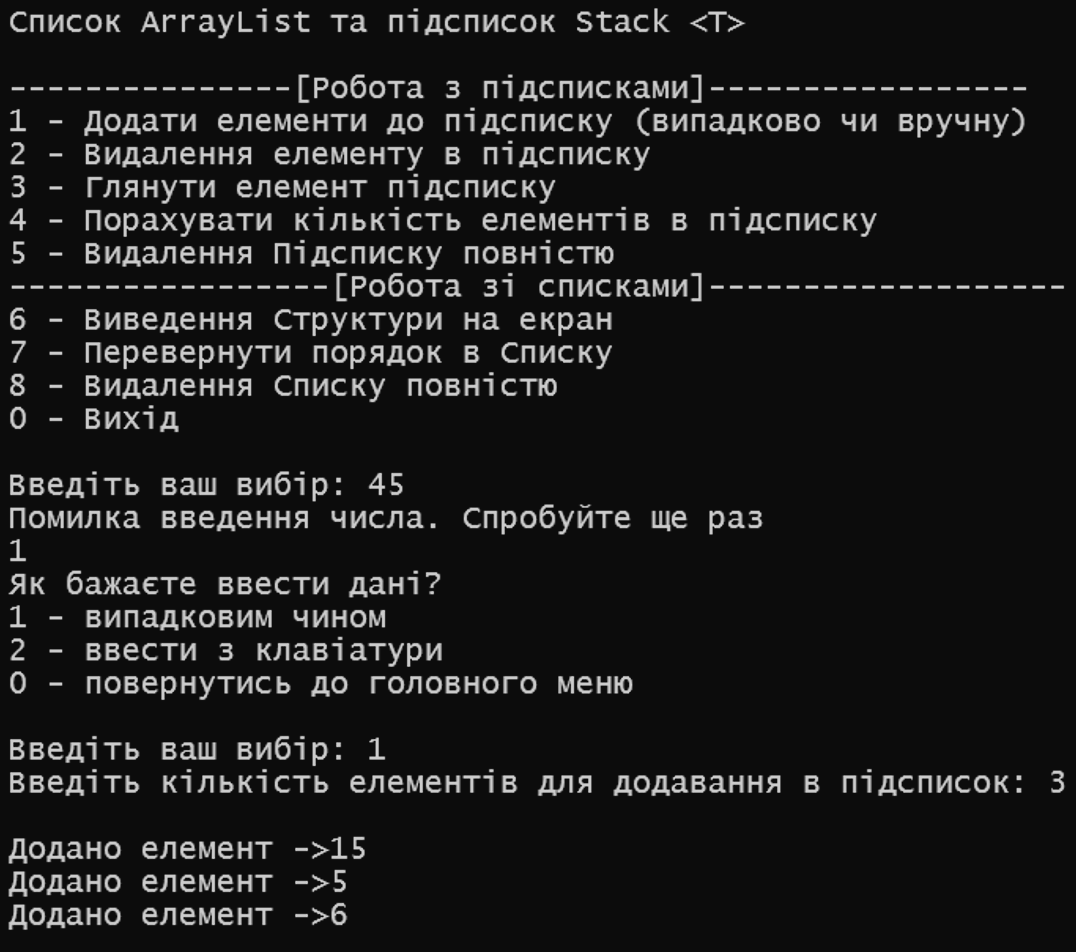


Рисунок 2 – додавання 3 випадкових елементів до підсписку в першому списку

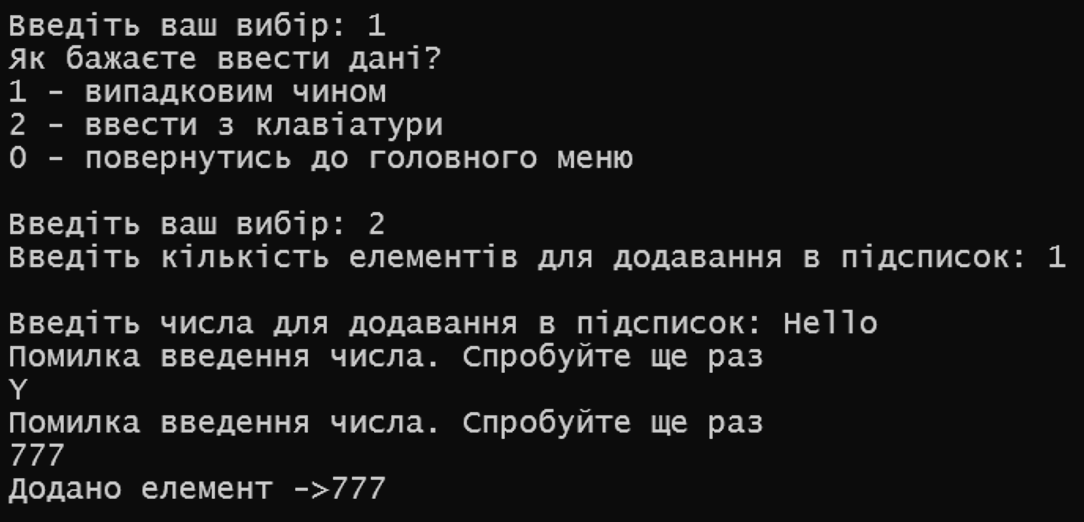


Рисунок 3 – додавання 1 елемента з клавіатури до підсписку в другому списку

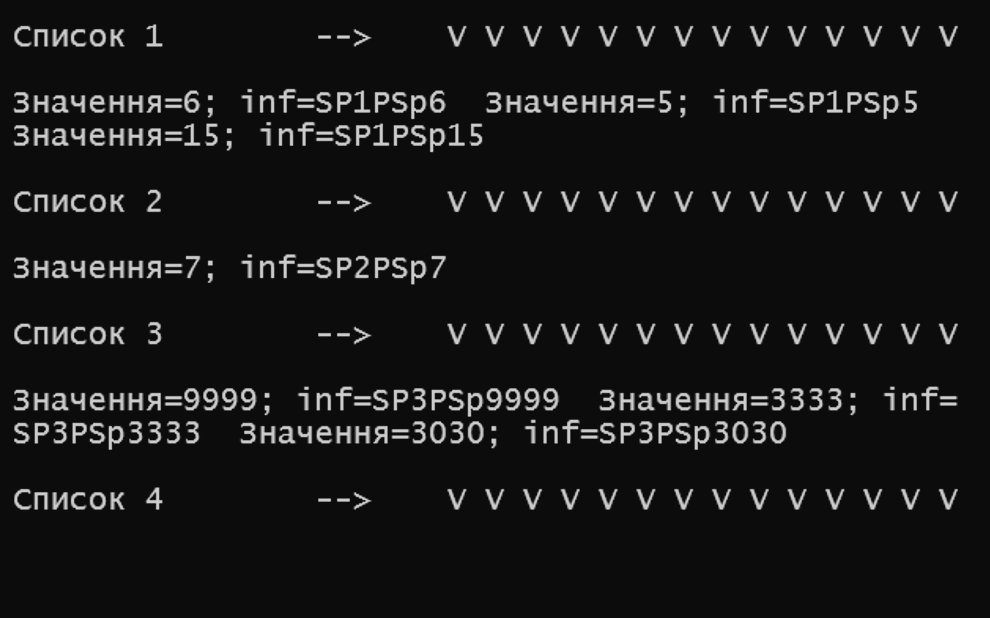


Рисунок 4 –створена структура

Після створення структури можна виконати дії над ними

Видалення елементу з підсписку можна переглянути на рисунку 5

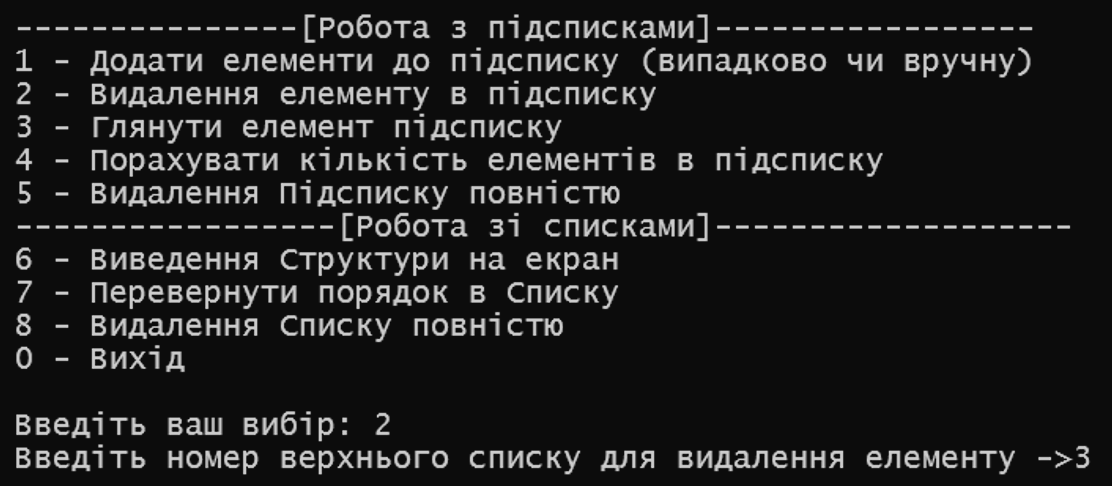




Рисунок 5 – видалення зі списку типу стек

Перегляд верхнього елементу в підсписку показано на рисунку 6

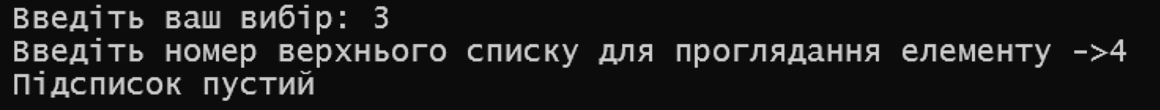
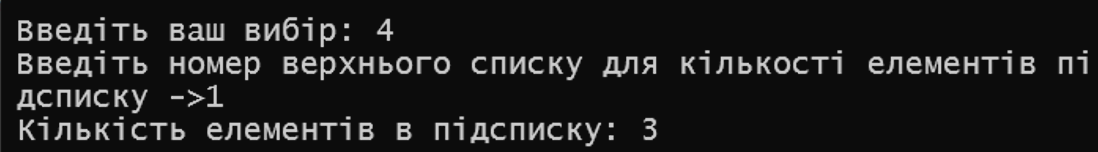
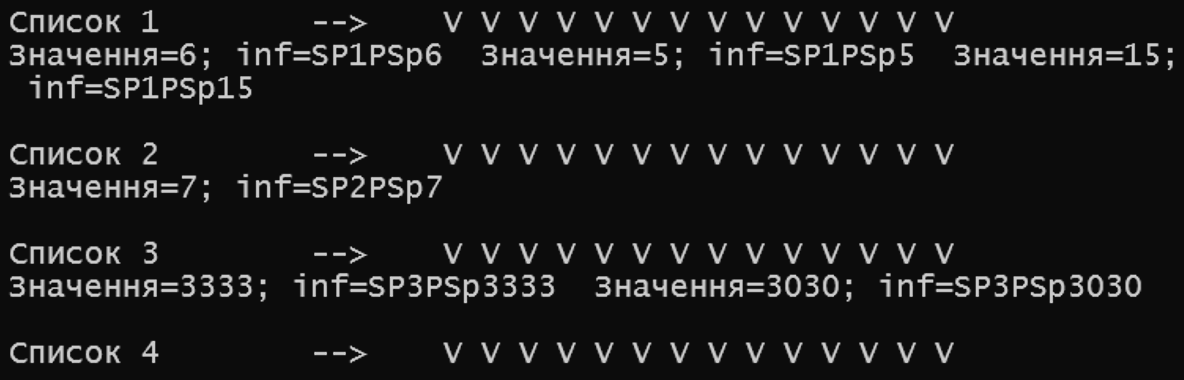


Рисунок 6 – підсписок був пустий – нічого переглядати

Можемо порахувати кількість елементів в підсписку. Результат на рисунку 7



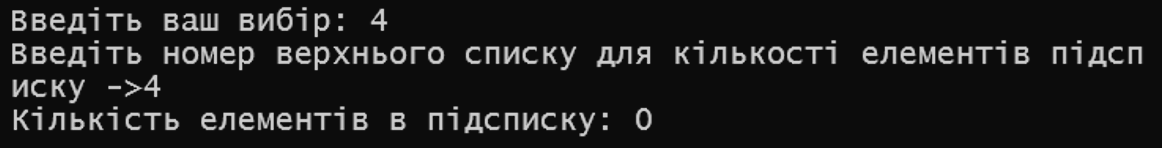
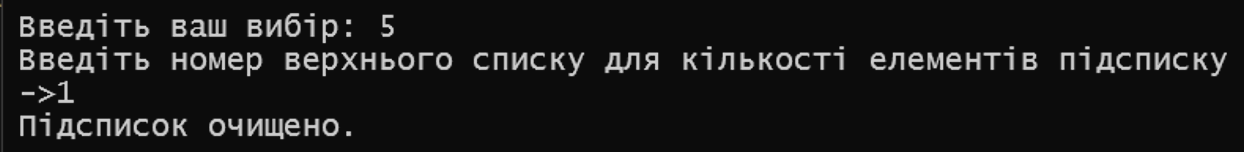


Рисунок 7 – кількість елементів в підсписку

Результат повного видалення підсписку 1 на рисунку 8



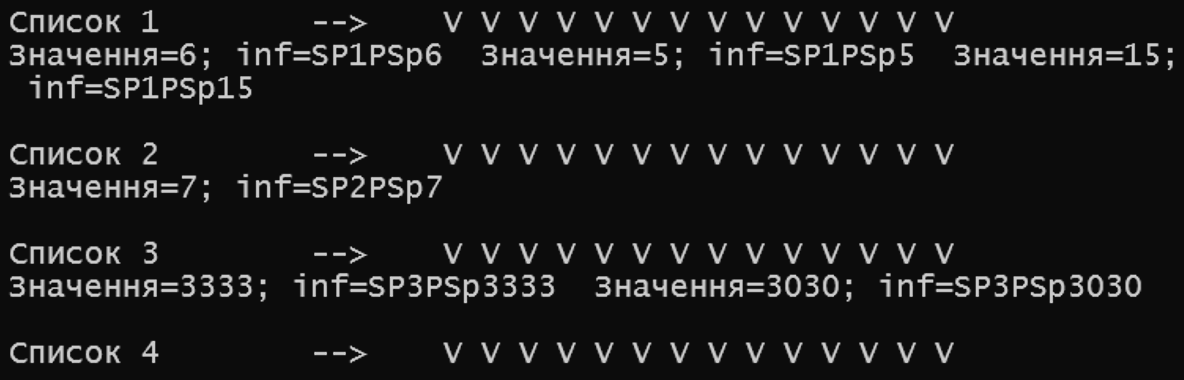


Рисунок 8 –видалення підсписку 1

Результат перевертання списку подано на рисунку 9

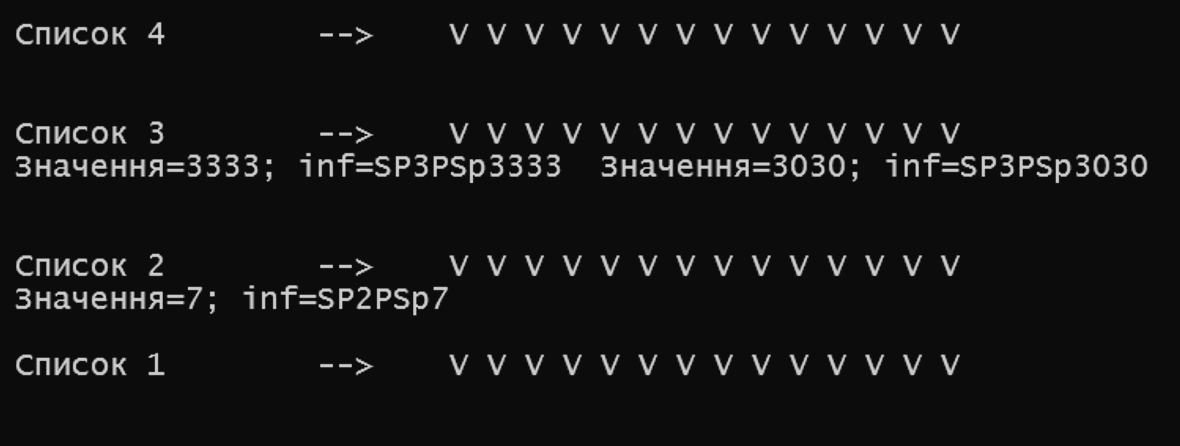
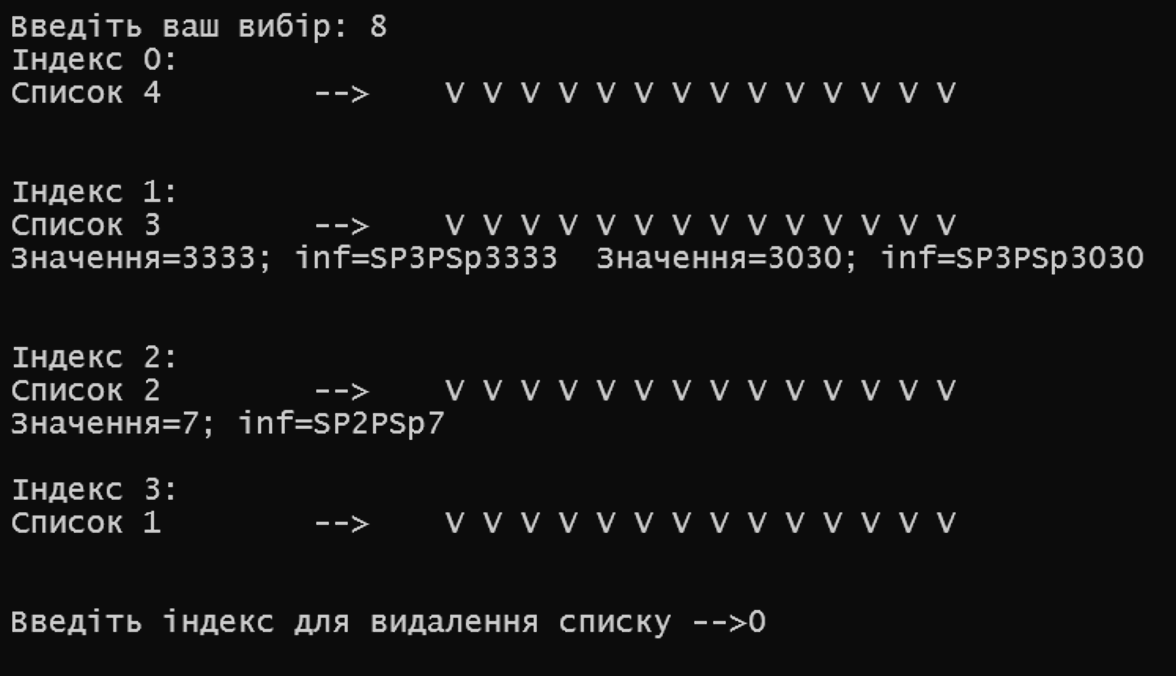


Рисунок 9 –перевернутий список

Видалення повністю списку показано на рисунку 10



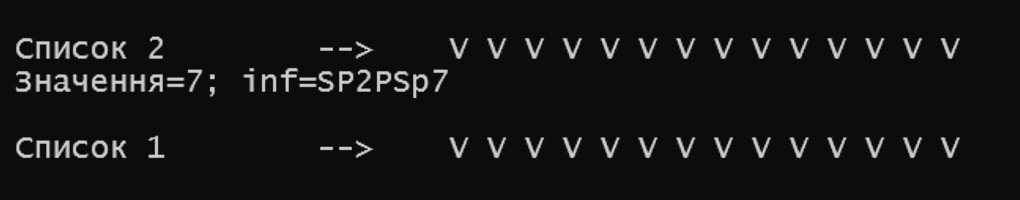


Рисунок 10 – повне видалення списку зі структури

ВИСНОВОК

У ході виконання лабораторної роботи я:

* Описав вказаний абстрактний тип даних список підсписків і основні операції роботи з ним на абстрактному і логічному рівнях
* Відповідно до варіанту створив структуру даних для організації дворівневої структури даних (списки підсписків).
* Для реалізації структури використовував вбудований тип ArrayList та Stack <T>. Як елементи структур використав тип Class, що мають поле ключа та поле інформації для підсписків (нижній рівень) і поле ключа для списку (верхній рівень).
* Створив додаток, який демонструє роботу базових операцій з вбудованими типами структур даних.
* Написав звіт.

ДОДАТОК А

Машинний лістинг програми

Program.cs:

using System.Text;

namespace ASD\_Lab\_3

{

internal class Program

{

public static void Main(string[] args)

{

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

Console.InputEncoding = System.Text.Encoding.GetEncoding(1251);

Console.OutputEncoding = System.Text.Encoding.GetEncoding(1251);

Up\_ArLst MyUP = new Up\_ArLst();

int count\_SP\_UP = 0;

int rm = 0;

Console.WriteLine($"Список ArrayList та підсписок Stack <T>");

while (true)

{

int choise = Menu();

try

{

switch (choise)

{

case 1:

{

if (!(MyUP.UP\_ArLst.Count < 5)) break;

rm = Rand\_or\_manual\_inp();

MyUP.Add\_Down\_to\_ArLst(Input\_El(ref count\_SP\_UP, rm)); // введення елементів в нижній стек & добавляем на верхний уровень элемент

break;

}

case 2:

{

int del\_el\_up = Input\_num("Введіть номер верхнього списку для видалення елементу ->", 1, MyUP.UP\_ArLst.Count);

MyUP.Remove\_Down\_El(del\_el\_up - 1); // удаление элемента стека из подсписка по номеру списка

break;

}

case 3:

{

int peek\_N\_SP = Input\_num("Введіть номер верхнього списку для проглядання елементу ->", 1, MyUP.UP\_ArLst.Count);

Down down = MyUP.UP\_ArLst[peek\_N\_SP - 1] as Down;

if (down != null)

{

El\_Down topElement = down.Peek();

Console.WriteLine("Верхній елемент: " + topElement.ToString());

}

else Console.WriteLine("Об'єкт вказаного верхнього списку не є екземпляром класу Down.");

break;

}

case 4:

{

int quantity\_N\_SP = Input\_num("Введіть номер верхнього списку для кількості елементів підсписку ->", 1, MyUP.UP\_ArLst.Count);

Down down = MyUP.UP\_ArLst[quantity\_N\_SP - 1] as Down; // Оскільки список MyUP.UP\_ArLst індексується з 0, а користувач вводить з 1

if (down != null)

{

int count = down.Count();

Console.WriteLine("Кількість елементів в підсписку: " + count);

}

else Console.WriteLine("Об'єкт вказаного верхнього списку не є екземпляром класу Down.");

break;

}

case 5:

{

int delete\_N\_SP = Input\_num("Введіть номер верхнього списку для кількості елементів підсписку ->", 1, MyUP.UP\_ArLst.Count);

Down down = MyUP.UP\_ArLst[delete\_N\_SP - 1] as Down; // Оскільки список MyUP.UP\_ArLst індексується з 0, а користувач вводить з 1

if (down != null)

{

down.Clear();

Console.WriteLine("Підсписок очищено.");

}

else Console.WriteLine("Об'єкт вказаного верхнього списку не є екземпляром класу Down.");

break;

}

case 6:

{

Console.WriteLine();

// виводимо список на екарн

Console.WriteLine(MyUP.ToString());

break;

}

case 7:

{

MyUP.Reverse\_UP();

break;

}

case 8:

{

foreach (var El in MyUP.UP\_ArLst)

{

int index = MyUP.UP\_ArLst.IndexOf(El);

Console.WriteLine($"Індекс {index}: {El}");

}

int del\_sp = Input\_num("Введіть індекс для видалення списку -->", 0, MyUP.UP\_ArLst.Count);

MyUP.Remove\_UP(del\_sp);

break;

}

case 0:

{ Environment.Exit(0); break; }

}

}

catch (Exception ex) { Console.WriteLine(ex.Message); continue; }

}

}

public static Down Input\_El(ref int count\_SP, int mode)

{

Console.Write("Введіть кількість елементів для додавання в підсписок: "); //додавання кількості елементів до підсписку

int push\_el\_qu = int.Parse(Console.ReadLine());

Down MyDown = new Down(++count\_SP);

Console.WriteLine();

for (int i = 0; i < push\_el\_qu; i++)

{

int k = 0;

if (mode == 1)

{

Random r = new Random();

k = r.Next(20);

}

else if (mode == 2)

k = Input\_num("Введіть числа для додавання в підсписок: ", Int32.MinValue, Int32.MaxValue);

Console.WriteLine("Додано елемент ->" + k.ToString());

El\_Down MyElDown = new El\_Down(k, "SP" + count\_SP.ToString() + "PSp" + k.ToString());

MyDown.Add\_El\_to\_Down(MyElDown);

}

return MyDown;

}

public static int Input\_num(string text, int down, int up)

{

int input;

Console.Write(text);

while (true)

{

if (int.TryParse(Console.ReadLine(), out input) && input <= up && input >= down)

{

return input;

}

else Console.WriteLine("Помилка введення числа. Спробуйте ще раз");

}

}

public static int Menu()

{

Console.WriteLine($"\n---------------[Робота з підсписками]-----------------\n" +

$"1 - Додати елементи до підсписку (випадково чи вручну)\n" +

$"2 - Видалення елементу в підсписку\n" +

$"3 - Глянути елемент підсписку\n" +

$"4 - Порахувати кількість елементів в підсписку\n" +

$"5 - Видалення Підсписку повністю\n" +

$"-----------------[Робота зі списками]-------------------\n" +

$"6 - Виведення Структури на екран\n" +

$"7 - Перевернути порядок в Списку\n" +

$"8 - Видалення Списку повністю\n" +

$"0 - Вихід\n");

return Input\_num("Введіть ваш вибір: ", 0, 9);

}

public static int Rand\_or\_manual\_inp()

{

int inp = 0;

Console.WriteLine($"Як бажаєте ввести дані?\n" +

$"1 - випадковим чином\n" +

$"2 - ввести з клавіатури\n" +

$"0 - повернутись до головного меню\n");

switch (Input\_num("Введіть ваш вибір: ", 0, 2))

{

case 0: { break; }

case 1: { inp = 1; break; }

case 2: { inp = 2; break; }

}

return inp;

}

}

}

ДОДАТОК Б

Машинний лістинг створеного класу

StructureClass.cs

using System;

using System.Collections;

namespace ASD\_Lab\_3

{

public class El\_Down // елемент підсписку

{

public int key;

public string inf;

public El\_Down(int key, string inf)

{

this.key = key;

this.inf = inf;

}

public override string ToString()

{

string res = "Значення=" + key.ToString() + "; inf=" + inf + " ";

return res;

}

}

public class Down // елемент типу стек <T> - цілий підсписок

{

public int L\_inf;

public Stack<El\_Down> Down\_St;

public Down(int inf)

{

L\_inf = inf;

Down\_St = new Stack<El\_Down>();

}

public void Add\_El\_to\_Down(El\_Down El)

{

Down\_St.Push(El);

}

public void Remove\_El\_Down()

{

if (Down\_St.Count > 0)

Down\_St.Pop();

else throw new Exception("Підсписок пустий");

}

// Возвращает верхний элемент подсписка без удаления

public El\_Down Peek()

{

if (Down\_St.Count > 0)

return Down\_St.Peek();

else throw new Exception("Підсписок пустий");

}

// Возвращает количество элементов в подсписке

public int Count()

{

return Down\_St.Count;

}

// Очищает весь подсписок

public void Clear()

{

Down\_St.Clear();

}

public override string ToString()

{

string res = "\nСписок " + L\_inf.ToString() + "\t--> V V V V V V V V V V V V V V\n";

foreach (El\_Down El in Down\_St)

res += El.ToString();

return res + '\n';

}

}

public class Up\_ArLst // основний список Ерей ліст, який містить в собі стеки <T>

{

public ArrayList UP\_ArLst;

public Up\_ArLst()

{

UP\_ArLst = new ArrayList();

}

public void Add\_Down\_to\_ArLst(Down El\_St)

{

UP\_ArLst.Add(El\_St);

}

public void Remove\_Down\_St(Down El\_St)

{

if (El\_St.Down\_St.Count != 0)

El\_St.Down\_St.Clear();

UP\_ArLst.Remove(El\_St);

}

public void Remove\_Down\_El(int Arr\_id)

{

if (Arr\_id >= 0 && Arr\_id < UP\_ArLst.Count)

{

Down down = UP\_ArLst[Arr\_id] as Down;

if (down != null)

{

if (down.Down\_St.Count > 0)

{

down.Remove\_El\_Down(); // Удаляем элемент из стека

}

else

{

throw new Exception("Стек пустий.");

}

}

else throw new Exception("Елемент з вказаним індексом в основному списку не підписок");

}

else throw new Exception("Індекс елемента в основному списку недопустимий");

}

public void Remove\_UP(int id)

{

if (id >= 0 && id < UP\_ArLst.Count)

{

UP\_ArLst.RemoveAt(id);

}

else

{

throw new Exception("Індекс елемента в основному списку недопустимий.");

}

}

public void Reverse\_UP()

{

UP\_ArLst.Reverse();

}

public override string ToString()

{

string res = "";

if (UP\_ArLst.Count != 0)

foreach (object El in UP\_ArLst)

res += El.ToString();

return res;

}

}

}