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 № 4**

з дисципліни « Алгоритми і структури даних »

*назва дисципліни*

на тему:«ДЕРЕВА. ФОРМИ ПРЕДСТАВЛЕННЯ ДЕРЕВ. ОБХОДИ ДЕРЕВ»

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 різних форм представлення структури даних «дерево» та основних операцій роботи з цим типом даних.

Постановка задачі

1. Описати вказаний абстрактний тип даних дерево і основні функції роботи з ним на абстрактному рівні і логічному рівні:

* початкова ініціалізація дерева;
* перевірка дерева на пустоту (дерево порожнє);
* додавання вузла дерева (нового нащадка);
* видалення вузла дерева;
* пошук вузла в дереві.
* виведення дерева на екран ( згідно з варіантом).

1. Створити додаток, який демонструє роботу основних функцій, перелічених вище (див. п.1.).

Зміст звіту

Зміст звіту з лабораторної роботи №4 має включати наступні підрозділи:

1. Опис m-арного дерева згідно з завданням варіанту.
2. Перетворення m-арного дерева з п.1 у бінарне.
3. Абстрактне представлення структури.
4. Представлення даних в графічному вигляді.
5. Проектування необхідних класів і структур.
6. Алгоритм для кожного методу (структури), використовуючи один з наступних видів представлення алгоритмів: псевдокод або блок- схему.
7. Лістинг програми та екранні форми результату роботи програми.
8. Висновки

**Варіант 9** індивідуального завдання до лабораторної роботи 4:

Описати структуру інституту або іншого підприємства (начальник - підлеглий) у вигляді m-арного дерева.

Фізичне зберігання дерева - динамічне

Тип обходу дерева - зворотний

Алгоритм реалізації обходу дерева - ітеративний

Алгоритм реалізації пошуку вузла - рекурсивний

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).

Опис m-арного дерева згідно з завданням варіанту.

До завдання варіанта 9 було створену наступну ієрархію в організації

* Керівник
* Підрозділ 1
* Відділ виробництва
* Відділ якості
* Відділ безпеки
* Підрозділ 2
* Відділ маркетингу
* Відділ продажів
* Відділ реклами
* Підрозділ 3
* Відділ фінансів
* Відділ бухгалтерії
* Відділ інвестицій

Дерево зображене на рисунку 1.

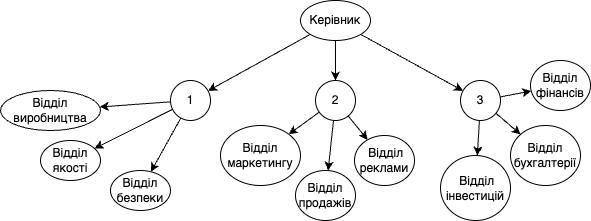


Рисунок 1 – Графічне подання m-арного дерева

Перетворення m-арного дерева з у бінарне.

За наступним алгоритмом виконаємо перетворення дерев:

1. В кожному вузлу залишити тільки гілку до старшого сина;
2. З’єднати горизонтальними ребрами всіх братів одного батька;
3. Таким чином перебудувати дерево за правилом: лівий син – вершина, розташована під даною; правий син – вершина, розташована праворуч від даної (тобто на одному ярусі з нею).
4. Розвернути дерево на 45 градусів так, щоб усі вертикальні гілки відображали лівих синів, а горизонтальні – правих

Результат перетворення зображено на рисунку 2 та 3.

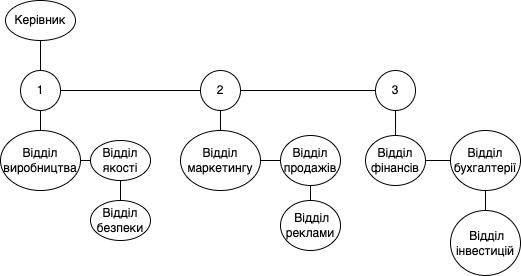


Рисунок 2 – Бінарне подання m-арного дерева за алгоритмом перетворення

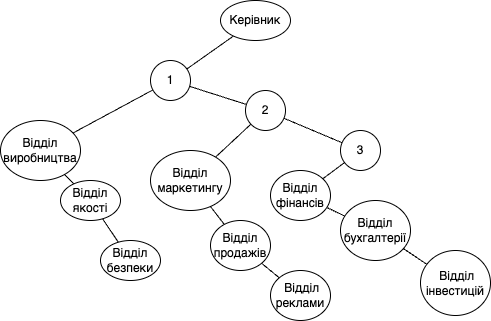


Рисунок 3 – Бінарне подання m-арного дерева за алгоритмом перетворення

Абстрактне представлення структури

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

**Опис АТД Бінарне дерево:**

BT – binary tree, El\_inf – значення елементу дерева, Side – сторона піддеревабб Father – батьківський елемент

**Операції:**

* 1. Створити (New):  → BT;
  2. Перевірка на порожність (IsEmpty): BT → bool;
  3. Додавання елементу вліво/вправо (Add): El\_inf x Side x Father → ElTree;
  4. Пошук вузла в дереві (Find): BT x El\_inf x Father → BT;
  5. Видалення елементу (Remove): BT x El\_inf → BT;
  6. Обхід дерева (ToString): BT → String;

Проектування необхідних класів і структур.

Клас ElTree представляє собою вузол/елемент дерева. Основні поля класу та їхні призначення:

* ElTree L, R – Покажчики на ліве (L) та праве (R) піддерева;
* string inf – містить інформацію, яка визначає значення вузла;

public class ElTree

{

public ElTree L, R; //покажчик на піддерева

public string inf;

public ElTree(string inf)

}

Клас Tree використовується для створення та управління деревом. Основні поля та методи:

* public ElTree T – Вказівник на кореневий вузол дерева.
* Конструктор public Tree(string inf) – Ініціалізує об'єкт Tree, створюючи кореневий вузол з вказаною інформацією.
* Метод public ElTree Find(ElTree V, string inf) – Пошук вузла з інформацією у піддереві, починаючи з вказаного вузла V.
* Метод public ElTree Find\_Father(ElTree V, string inf) – Пошук предка вузла з інформацією у піддереві, починаючи з вказаного вузла V.
* Метод public bool Add(ElTree V, string inf, char S) – Додавання вузла з інформацією вліво (S = 'L') або вправо (S = 'R') вказаного вузла V.
* Метод public bool Remove(ref ElTree V, string inf) – Видалення вузла з інформацією з дерева, починаючи з кореня V.
* Метод public void TreeInString(ElTree V, ref string s) –Перетворення дерева у рядок (стрічку) у зворотньому порядку.
* Метод public bool TreeExists() – Перевірка, чи існує дерево (чи кореневий вузол не є порожнім).

public class Tree

{

public ElTree T;

public Tree(string inf)

public ElTree Find(ElTree V, string inf) //пошук вузла з інформацією

public ElTree Find\_Father(ElTree V, string inf)//пошук предка вузла з інформацією

public bool Add(ElTree V, string inf, char S)//додавання вузла з інформацією вліво/вправо s=L/R.

public bool Remove(ref ElTree V, string inf)//видалення вузла з інформацією

public void TreeInString(ElTree V, ref string s)

public bool TreeExists()

}

Методи подані у вигляді псевдокоду

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"

Find: Пошук вузла з інформацією у дереві.

ElTree Find(ElTree V, string inf):

якщо V не є null:

якщо V.inf дорівнює inf, то повернути V

temp = Find(V.L, inf)

якщо temp не є null, то повернути temp

повернути Find(V.R, inf)

повернути null

**Find\_Father**: Пошук предка вузла з інформацією у дереві.

ElTree Find\_Father(ElTree V, string inf):

якщо V не є null:

якщо V.L не є null і V.L.inf дорівнює inf, то повернути V

якщо V.R не є null і V.R.inf дорівнює inf, то повернути V

temp = Find\_Father(V.L, inf)

якщо temp не є null, то повернути temp

повернути Find\_Father(V.R, inf)

повернути null

Add: Додавання вузла з інформацією вліво або вправо.

bool Add(ElTree V, string inf, char S):

temp = new ElTree(inf)

якщо S дорівнює 'L':

якщо V.L є null, то V.L = temp і повернути true

інакше повернути false

інакше (S дорівнює 'R'):

якщо V.R є null, то V.R = temp і повернути true

інакше повернути false

Remove: Видалення вузла з інформацією з дерева.

bool Remove(ref ElTree V, string inf):

father = Find\_Father(V, inf)

temp = Find(V, inf)

якщо temp є null, то повернути false

якщо father є null:

V = null

інакше, якщо father.L не є null і father.L.inf дорівнює inf:

father.L = null

інакше:

father.R = null

повернути true

**TreeInString**: Перетворення дерева у рядок у зворотньому порядку.

void TreeInString(ElTree V, ref string s):

якщо V є null, то повернути

stack = новий Stack<ElTree>()

stack.Push(V)

outputStack = новий Stack<string>()

while stack не є порожнім:

current = stack.Pop()

outputStack.Push(current.inf)

якщо current.L не є null, то stack.Push(current.L)

якщо current.R не є null, то stack.Push(current.R)

while outputStack не є порожнім:

s += outputStack.Pop() + " "

TreeExists: Перевірка наявності дерева.

bool TreeExists():

повернути T не є null

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.

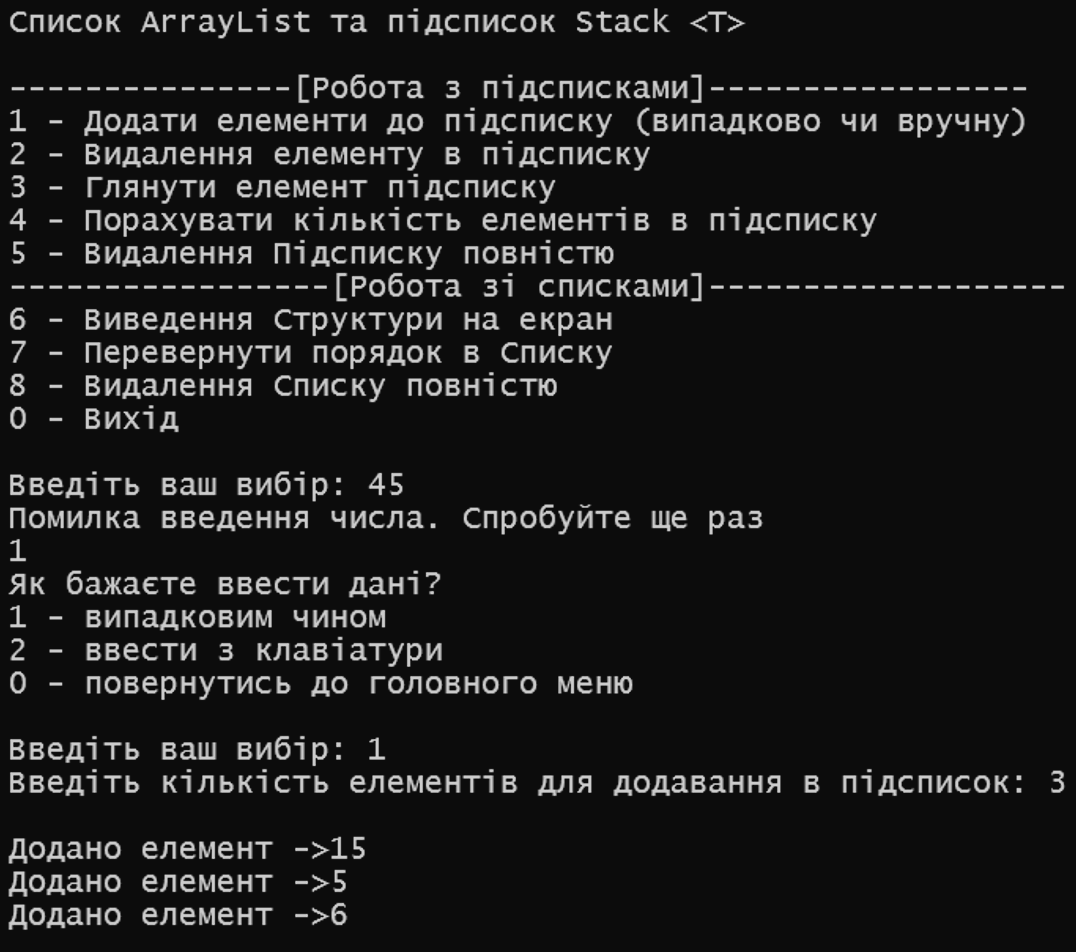


Рисунок 2 – додавання 3 випадкових елементів до підсписку в першому списку

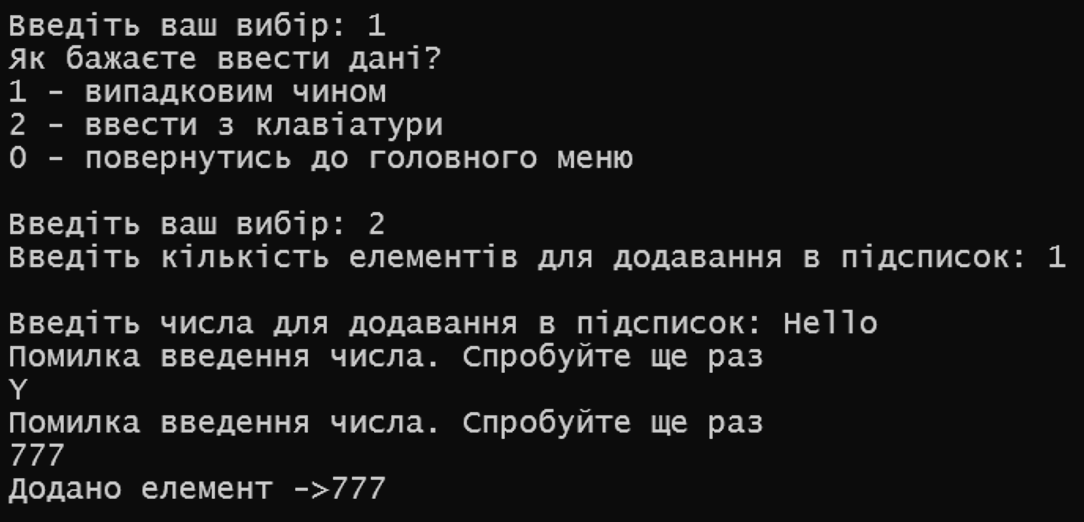


Рисунок 3 – додавання 1 елемента з клавіатури до підсписку в другому списку

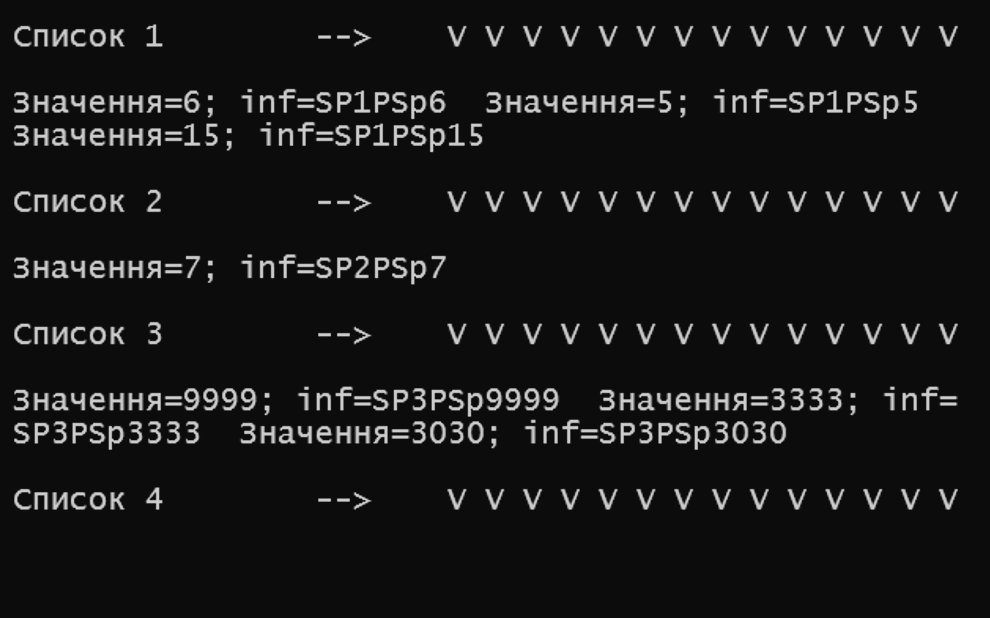


Рисунок 4 –створена структура

Після створення структури можна виконати дії над ними

Видалення елементу з підсписку можна переглянути на рисунку 5

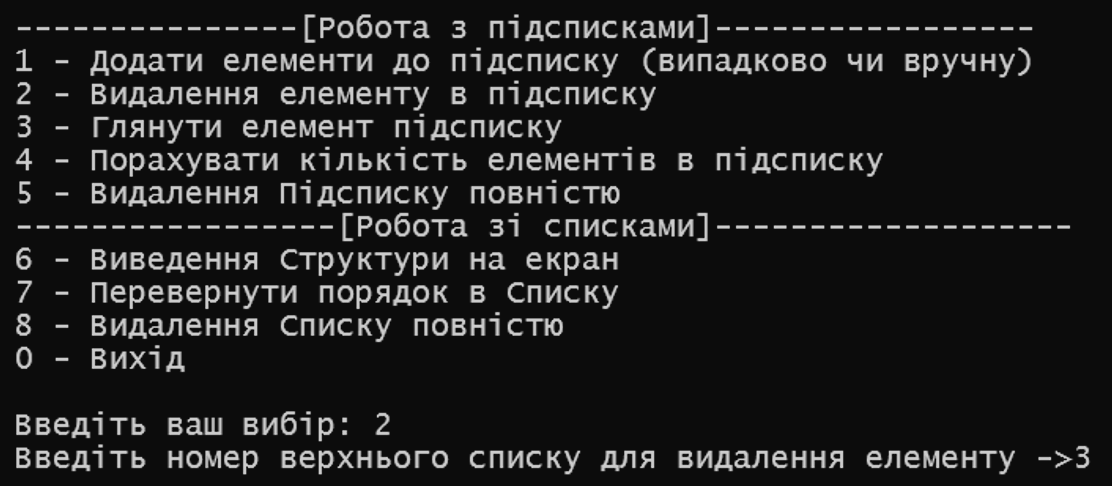




Рисунок 5 – видалення зі списку типу стек

Перегляд верхнього елементу в підсписку показано на рисунку 6

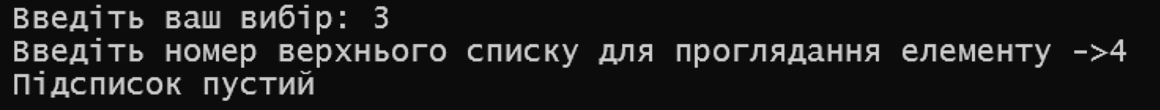
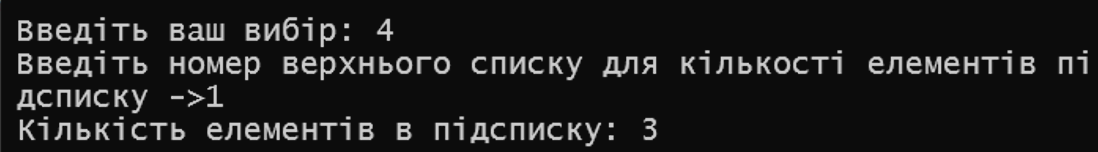
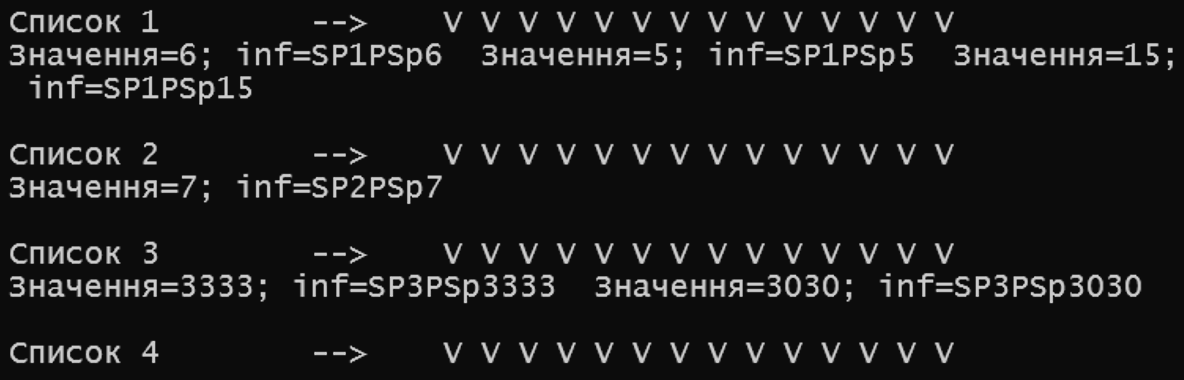


Рисунок 6 – підсписок був пустий – нічого переглядати

Можемо порахувати кількість елементів в підсписку. Результат на рисунку 7



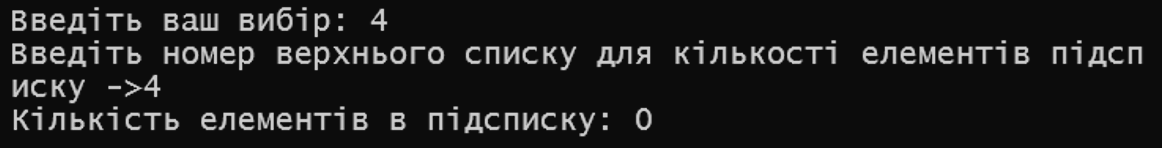
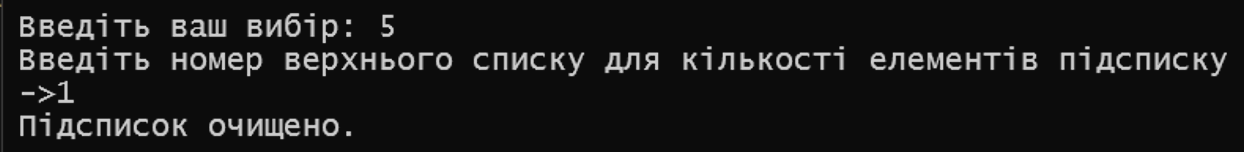


Рисунок 7 – кількість елементів в підсписку

Результат повного видалення підсписку 1 на рисунку 8



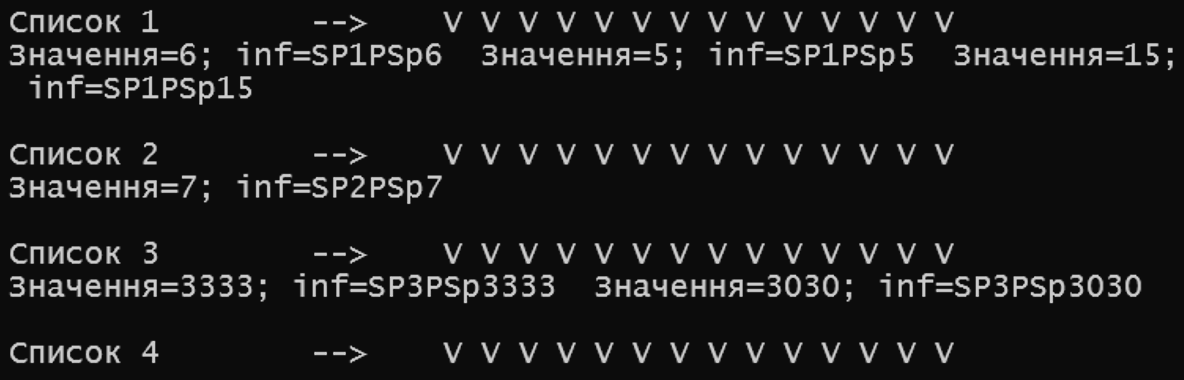


Рисунок 8 –видалення підсписку 1

Результат перевертання списку подано на рисунку 9

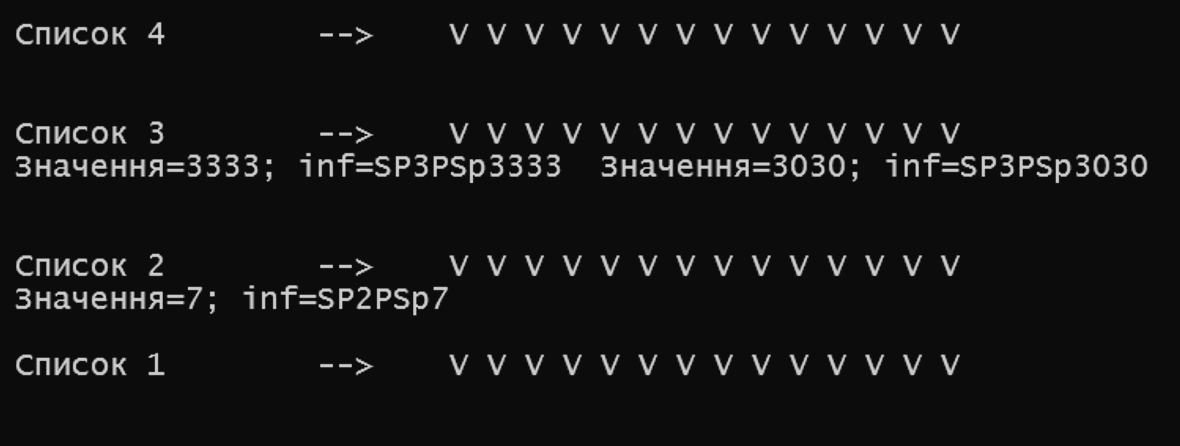
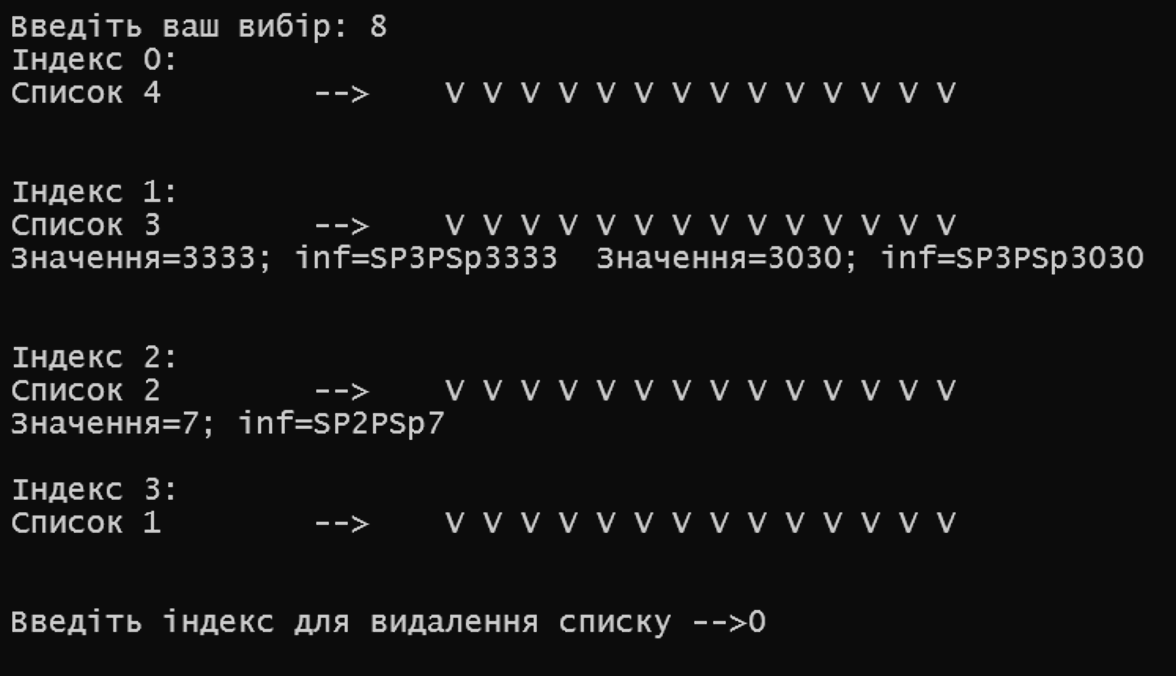


Рисунок 9 –перевернутий список

Видалення повністю списку показано на рисунку 10



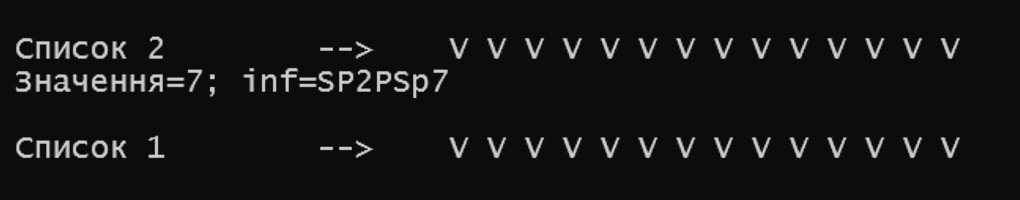


Рисунок 10 – повне видалення списку зі структури

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;

}

}

}