МІНІСТЕРСТВО КУЛЬТУРИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КУЛЬТУРИ І МИСТЕЦТВ

ФАКУЛЬТЕТ ЕКОНОМІКИ І ІНФОРММЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Звіт

з дисципліни «Системний аналіз»

на тему «Обробка стеків»

Виконав:

студент 3 курсу

групи КН-23

Михаленков Микола

Київ –2015

ЗМІСТ

[РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc431929648)

[РОЗДІЛ 2 ОПИС АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ 4](#_Toc431929649)

[2.1 Інструменти розв’язання задачі 4](#_Toc431929650)

[2.2 Початковий стан 4](#_Toc431929651)

[2.3 Опис алгоритму додавання *k*-елементів до стеку висоти *n* після *l*-того елементу 4](#_Toc431929652)

[2.4 Опис алгоритму видалення *k*-елементів з стеку висоти *n* після *l*-того елементу 4](#_Toc431929653)

[РОЗДІЛ 3 БЛОК-СХЕМИ ПРОГРАМНОГО ПРОДУКТУ 5](#_Toc431929654)

[РОЗДІЛ 4 ЛІСТИНГ КОДУ 6](#_Toc431929655)

[4.1 Файл index.html 6](#_Toc431929656)

[4.2 Файл style.css 8](#_Toc431929657)

[4.3 Файл script.js 8](#_Toc431929658)

[РОЗДІЛ 5 SCREENSHOTS 10](#_Toc431929659)

[5.1 Загальні 10](#_Toc431929660)

[5.2 Варіації 11](#_Toc431929661)

# РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

Написати програму обробки стека, яка передбачає:

1. Додавання *k*-елементів до стеку висоти *n* після *l*-того елементу.
2. Видалення *k*-елементів після *l*-того зі стеку висоти *n*.

При написанні програми врахувати всі можливі варіації запису стеків.

# РОЗДІЛ 2 ОПИС АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

## 2.1 Інструменти розв’язання задачі

Задача була розв’язана за допомогою мови програмування JavaScript. Для графічного зображення даних та їх введення використовувалася мова розмітки XHTML, стандарт HTML5 та каскадні таблиці стилів. Переглянути роботу програми можна в будь-якому сучасному веб-браузері (Google Chrome, Mozilla Firefox, Safari, Opera) відкривши файл index.html.

Для полегшення роботи з графікою та приємного оку оформлення роботи використовувалася бібліотека каскадних таблиць стилів Bootstrap 3.

Для полегшення роботи з DOM-елементами та пришвидшення користувацького вводу використовувалася бібліотека Angular.js.

## 2.2 Початковий стан

При завантаженні сторінки у браузері створюються 3 порожніх масиви X, Y,Z.

Для задання розміру стеку X та Y потрібно ввести значення *n* та *k* у відповідні поля для вводу і потім натиснути відповідні кнопки справа від форм.

*n* – розмір стека X.

*k* – розмір стека Y.

За напрям запису слова у стек відповідають перемикачі справа зверху сторінки. Для кожного стеку позначена своя пара перемикачів.

У поле *l* вводиться номер елемента після якого буду додаватися або прибиратися *k*-елементів.

## 2.3 Опис алгоритму додавання *k*-елементів до стеку висоти *n* після *l*-того елементу

Алгоритм додавання *k*-елементів до стеку висоти *n* після *l*-того елементу:

1. Присвоюємо стеку Z порожню множину.
2. Додаємо до стеку Z елементи від першого до *l*-того зі стекуX.
3. Додаємо до стеку Z елементи від першого до останнього зі стеку Y.
4. Додаємо до стеку Z елементи віл (*l*+1)-того до останнього зі стеку X.

## 2.4 Опис алгоритму видалення *k*-елементів з стеку висоти *n* після *l*-того елементу

Алгоритм видалення *k*-елементів з стеку висоти *n* після *l*-того елементу:

1. Присвоюємо стеку Z порожню множину.
2. Додаємо до стеку Z елементи від першого до *l*-того зі стекуX.
3. Додаємо до стеку Z елементи від *(l+k*)-того до останнього зі стеку X.

# РОЗДІЛ 3 БЛОК-СХЕМИ ПРОГРАМНОГО ПРОДУКТУ

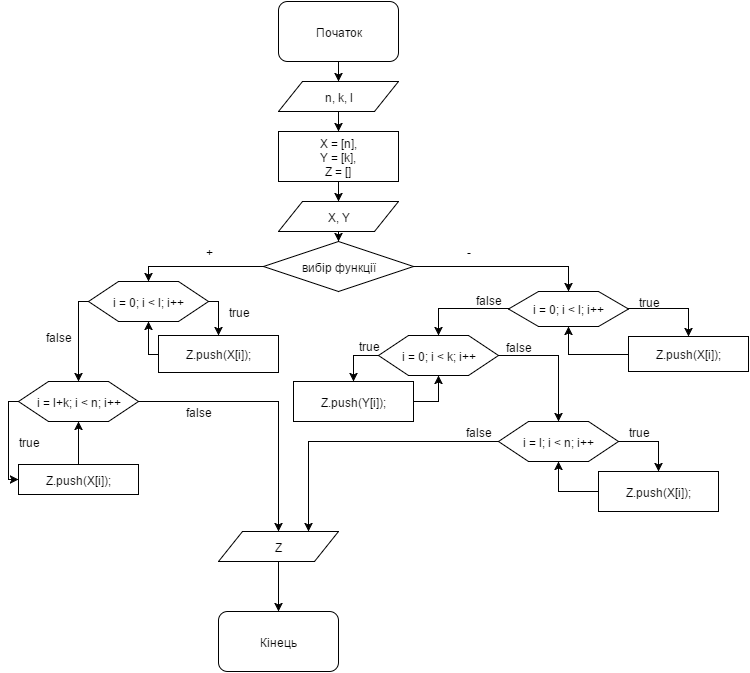


Рис. 1. Головна блок-схема

# РОЗДІЛ 4 ЛІСТИНГ КОДУ

## 4.1 Файл index.html

<!DOCTYPE html>

<html lang="en" ng-app="DataStructureStack">

<head>

<title>Stack | SA</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" type="text/css" href="../../../Libraries/bootstrap-3.3.5-dist/css/bootstrap.css">

<link rel="stylesheet" type="text/css" href="style.css">

<script src="../../../Libraries/angular.js"></script>

<script src="../../../Libraries/jquery-2.1.4.min.js"></script>

<script src="script.js"></script>

</head>

<body ng-controller="DefaultCtrl">

<h1 class="text-center">Stack</h1>

<div class="container">

<div class="col-lg-1">

<form>

<div class="form-group">

<input type="text" ng-model="nValue" placeholder="n" class="form-control">

</div>

<div class="form-group">

<input type="text" ng-model="kValue" placeholder="k" class="form-control">

</div>

<div class="form-group">

<input type="text" ng-model="lValue" placeholder="l" class="form-control">

</div>

</form>

</div>

<div class="col-lg-2">

<button type="button" class="btn btn-primary" ng-click="setX()">Set X to {{nValue}} items</button><br><br>

<button type="button" class="btn btn-primary" ng-click="setY()">Set Y to {{kValue}} items</button><br><br>

<p>Caution: First element has index 0.</p>

</div>

<div class="col-lg-3">

<button type="button" class="btn btn-success" ng-click="Add()">Add {{Y.length}} elements to X after {{lValue}}</button><br><br>

<button type="button" class="btn btn-danger" ng-click="Delete()">Delete {{Y.length}} elements from X after {{lValue}}</button>

</div>

<div class="col-lg-2">

<form>

<h3>Read X from</h3>

<input type="radio" checked value="Xleftside" ng-model="Xside" ng-change="changeXside()">Left</input><br>

<input type="radio" value="Xrightside" ng-model="Xside" ng-change="changeXside()">Right</input>

</form>

</div>

<div class="col-lg-2">

<form>

<h3>Read Y from</h3>

<input type="radio" value="Yleftside" ng-model="Yside">Left</input><br>

<input type="radio" value="Yrightside" ng-model="Yside">Right</input>

</form>

</div>

<div class="col-lg-2">

<form>

<h3>Write Z from</h3>

<input type="radio" value="Zleftside" ng-model="Zside">Left</input><br>

<input type="radio" value="Zrightside" ng-model="Zside" ng-change="showSides()">Right</input>

</form>

</div>

</div>

<!-- Result -->

<div class="col-lg-1"></div>

<div class="col-lg-2">

<h2 class="text-center">X</h2>

<form class="form-inline" >

<div class="input-group" ng-repeat="item in X">

<div class="input-group-addon">{{X.length - 1 - $index}}</div>

<input type="text" ng-model="X[$index]" class="form-control" size="3"></input>

</div>

</form>

</div>

<div class="col-lg-2"></div>

<div class="col-lg-2">

<h2 class="text-center">Y</h2>

<form class="form-inline" >

<div class="input-group" ng-repeat="item in Y">

<div class="input-group-addon">{{Y.length - 1 - $index}}</div>

<input type="text" ng-model="Y[$index]" class="form-control" size="3"></input>

</div>

</form>

</div>

<div class="col-lg-2"></div>

<div class="col-lg-2">

<h2 class="text-center">Z</h2>

<form class="form-inline">

<div class="input-group" ng-repeat="item in Z track by $index">

<div class="input-group-addon">{{Z.length - 1 - $index}}</div>

<input type="text" ng-model="Z[$index]" class="form-control" size="3" ng-readonly></input>

</div>

</form>

</div>

<div class="col-lg-1"></div>

</body>

</html>

## 4.2 Файл style.css

.form {

border-color: #ddd;

border-width: 1px;

border-radius: 4px;

margin: 0 auto;

}

div.input-group {

margin: 3px;

}

## 4.3 Файл script.js

angular.module('DataStructureStack', [])

.controller('DefaultCtrl', function ($scope) {

$scope.X = [];

$scope.Y = [];

$scope.Z = [];

$scope.Xside = "Xleftside";

$scope.Yside = "Yleftside";

$scope.Zside = "Zleftside";

$scope.showSides = function () {

console.log($scope.Xside + "\n" + $scope.Yside + "\n" + $scope.Zside);

};

$scope.showArrays = function () {

console.log($scope.X + "\n" + $scope.Y + "\n" + $scope.Z);

};

$scope.setX = function () {

$scope.X = [];

if ($scope.Xside === "Xleftside") {

for (var i = 0; i < $scope.nValue; i++) {

$scope.X[i] = $scope.nValue - i - 1;

}

} else {

for (var i = 0; i < $scope.nValue; i++) {

$scope.X[i] = i;

}

}

};

$scope.setY = function () {

$scope.Y = [];

if ($scope.Yside === "Yleftside") {

for (var i = 0; i < $scope.kValue; i++) {

$scope.Y[i] = $scope.kValue - i - 1;

}

} else {

for (var i = 0; i < $scope.kValue; i++) {

$scope.Y[i] = i;

}

}

};

$scope.Add = function () {

$scope.Z = [];

for (var i = 0; i < $scope.lValue; i++) {

$scope.Z.push($scope.X[i]);

}

for (var i = 0; i < $scope.Y.length; i++) {

$scope.Z.push($scope.Y[i]);

}

for (var i = $scope.lValue; i < $scope.X.length; i++) {

$scope.Z.push($scope.X[i]);

}

if ($scope.Zside === "Zleftside") {

$scope.TempArray = $scope.Z;

$scope.Z = [];

$scope.Z = $scope.TempArray.reverse();

}

};

$scope.Delete = function () {

$scope.Z = [];

for (var i = 0; i < $scope.lValue; i++) {

$scope.Z.push($scope.X[i]);

}

console.log(parseInt($scope.lValue, 10) + $scope.Y.length);

for (var i = parseInt($scope.lValue, 10) + $scope.Y.length; i < $scope.X.length; i++) {

$scope.Z.push($scope.X[i]);

}

if ($scope.Zside === "Zleftside") {

$scope.TempArray = $scope.Z;

$scope.Z = [];

$scope.Z = $scope.TempArray.reverse();

}

};

});

# РОЗДІЛ 5 SCREENSHOTS

## 5.1 Загальні

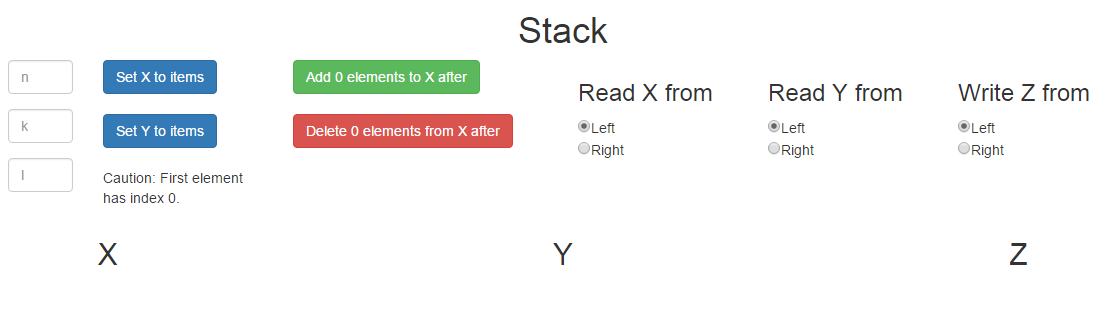


Рис. 1. Початкове вікно програми.

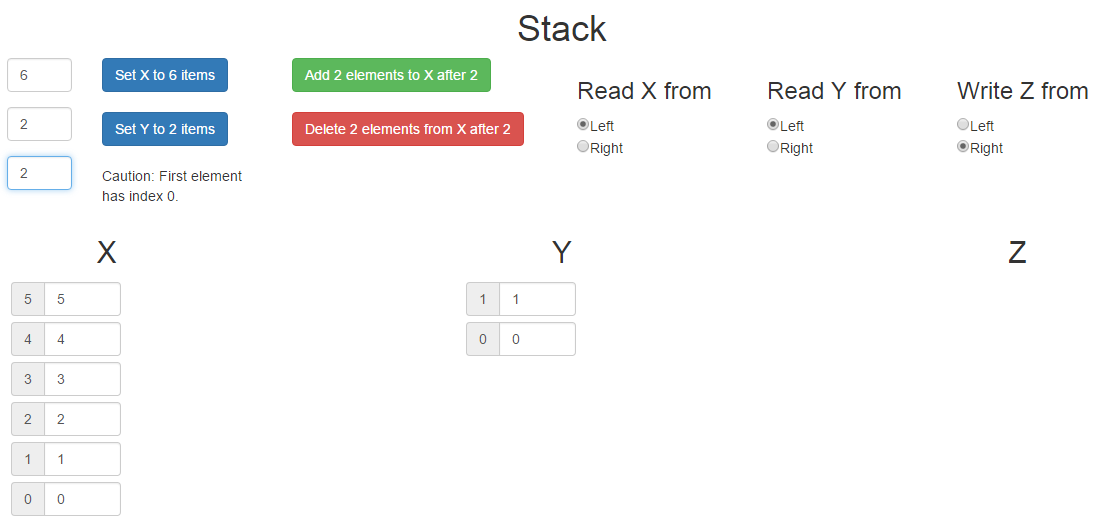


Рис. 2. Вікно програми після вводу умови.

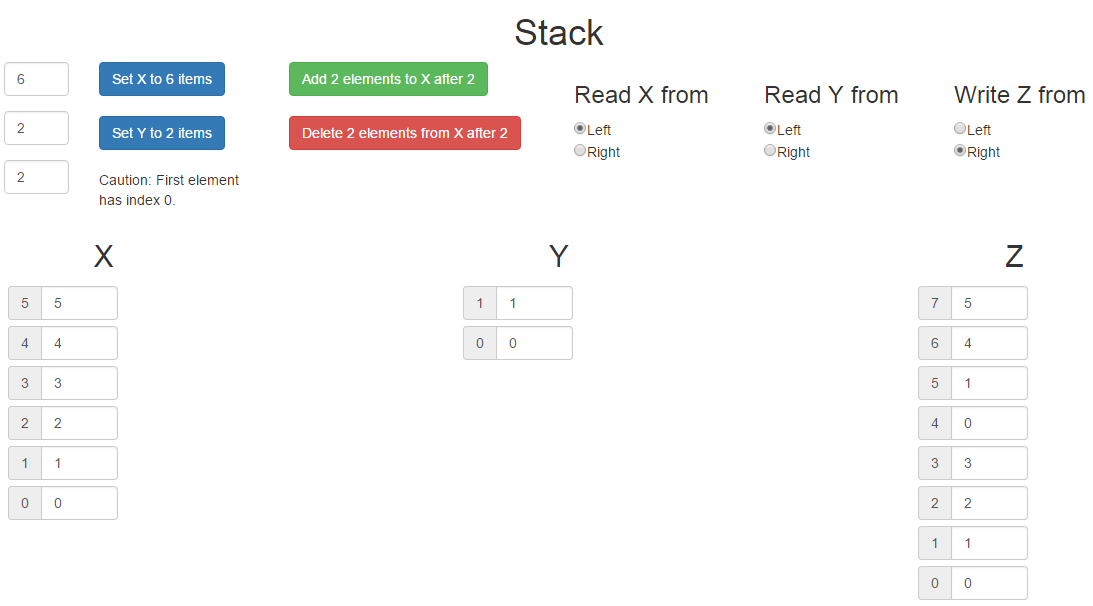


Рис. 3. Результат виконання функції додавання.

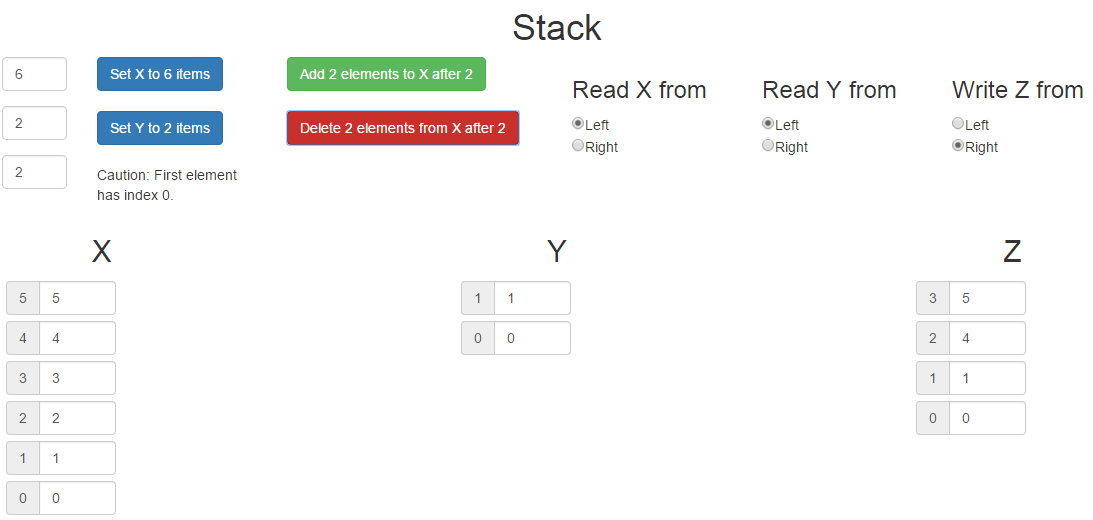


Рис. 4. Результат виконання функції видалення.

## 5.2 Варіації

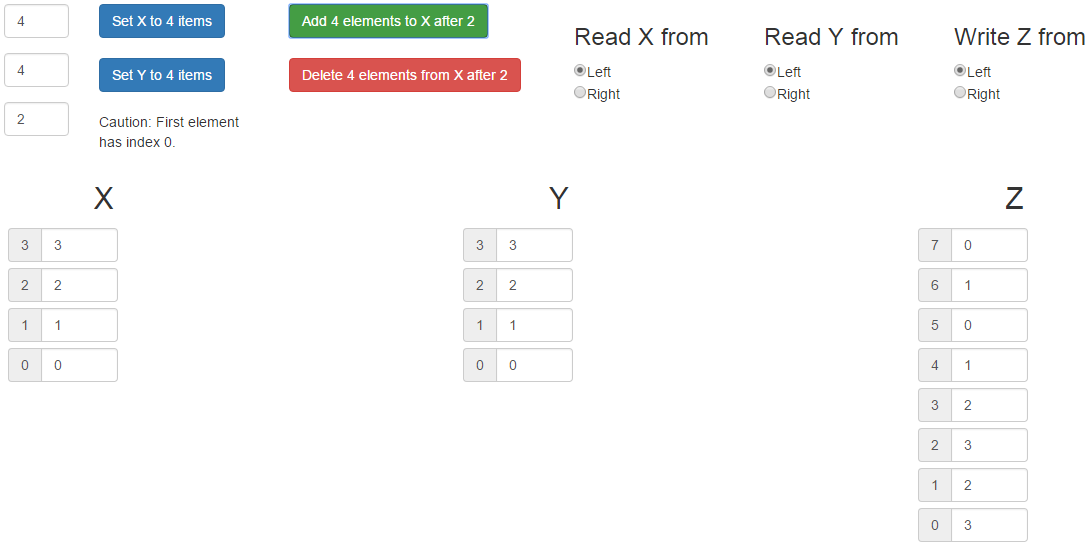


Рис. 5. Варіант Left-Left-Left-Add

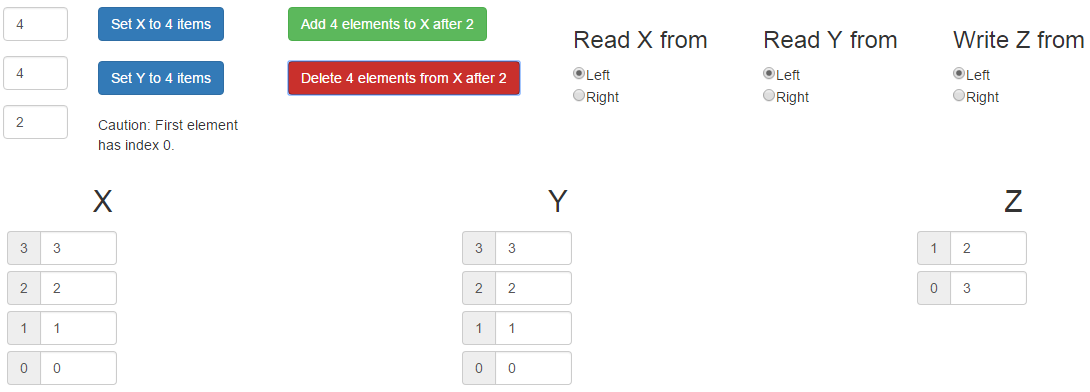


Рис. 6. Варіант Left-Left-Left-Delete

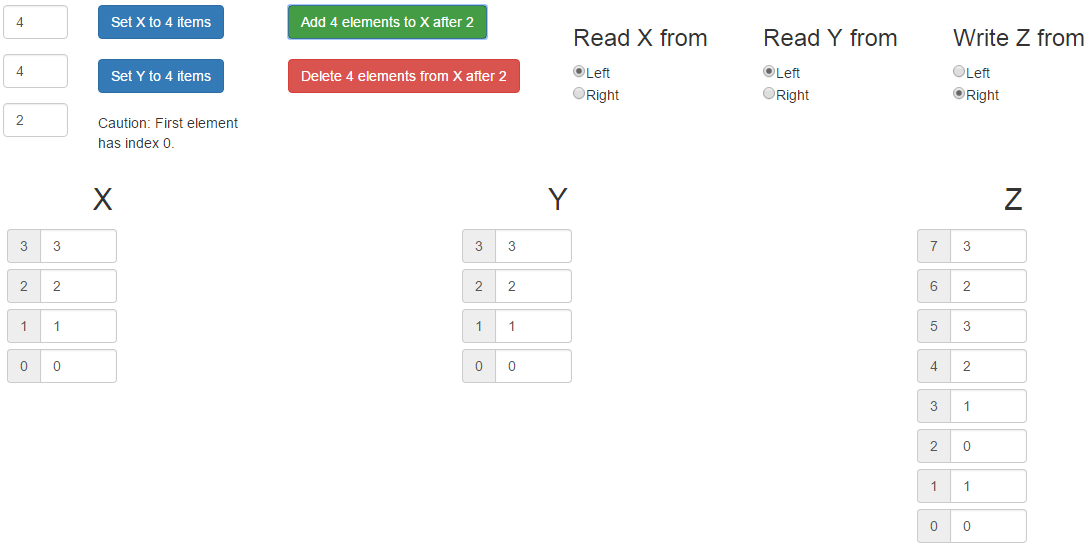


Рис. 7. Варіант Left-Left-Right-Add

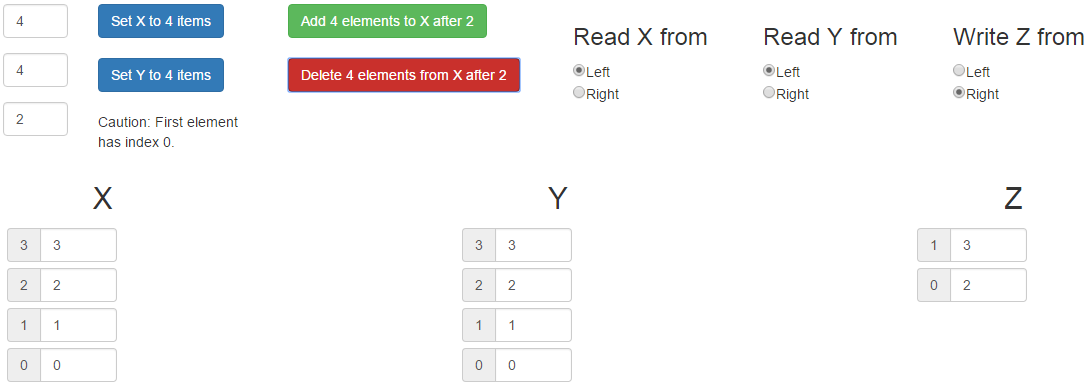


Рис. 8. Варіант Left-Left-Right-Delete

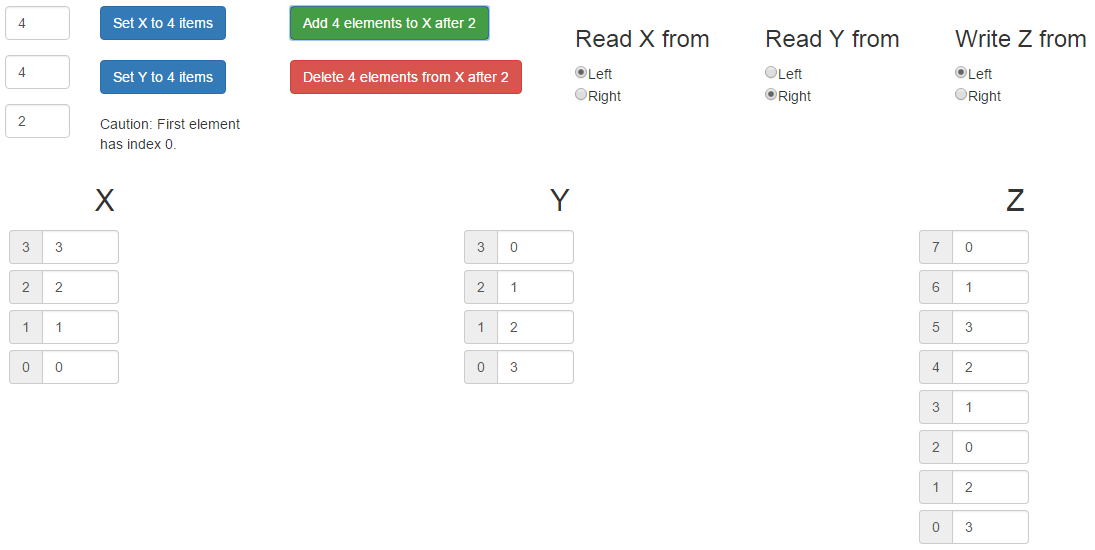


Рис. 9 Варіант Left-Right-Left-Add

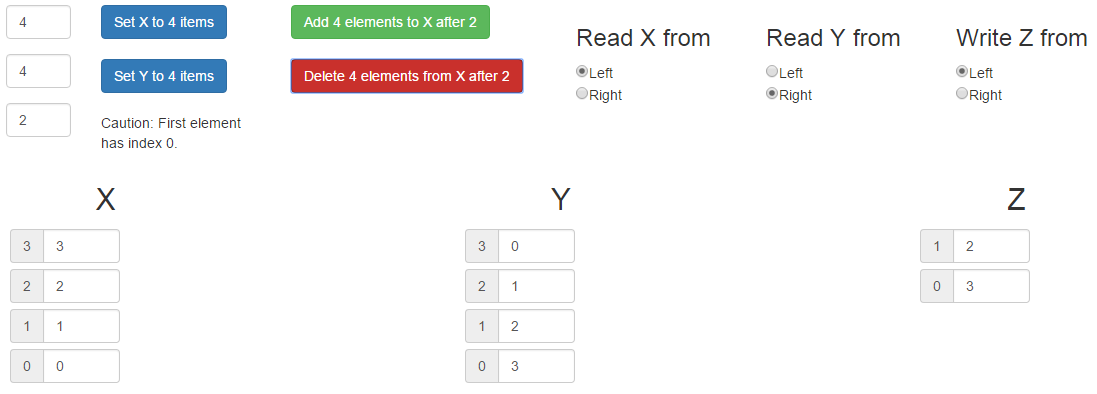


Рис. 10. Варіант Left-Right-Left-Delete

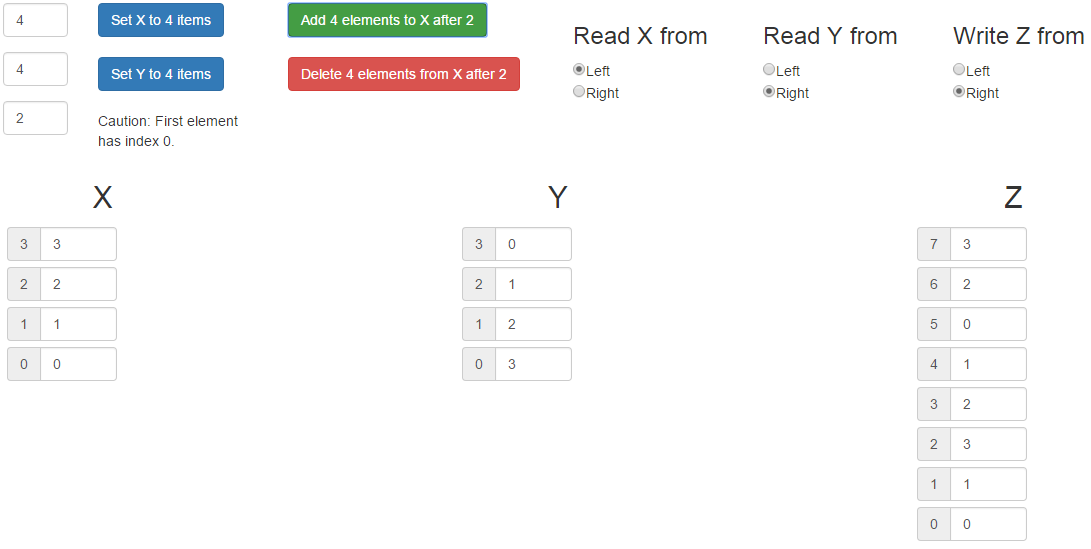


Рис. 11. Варіант Left-Right-Right-Add

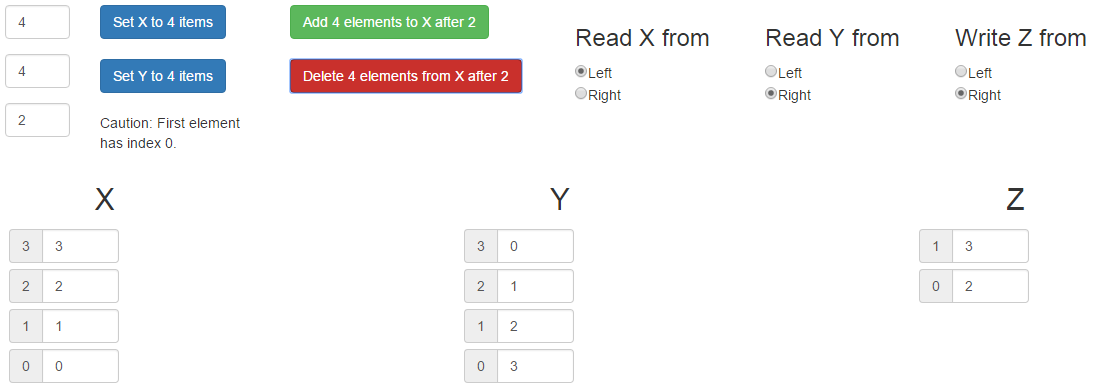


Рис. 12. Варіант Left-Right-Right-Delete

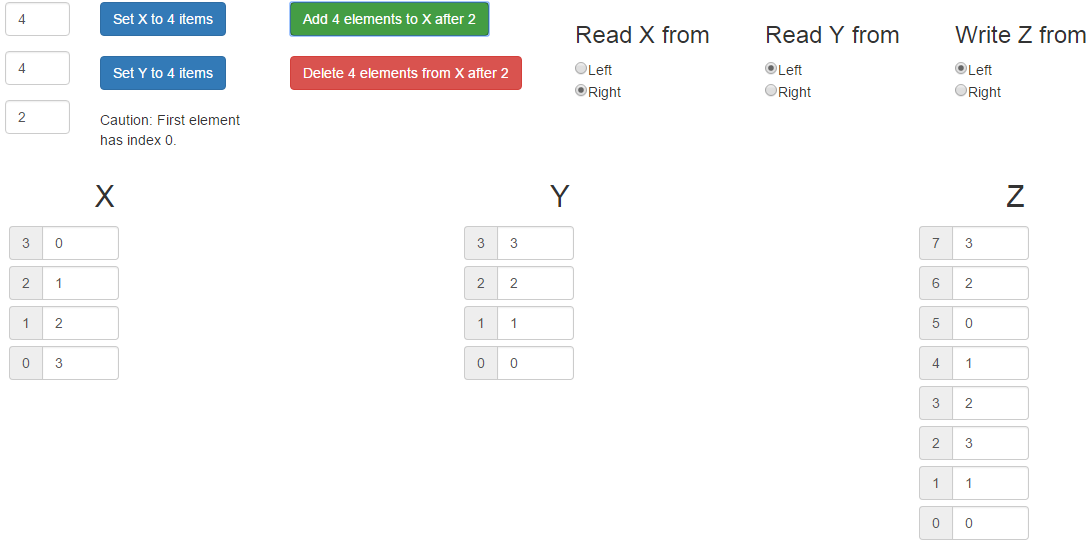


Рис. 13. Варіант Right-Left-Left-Add

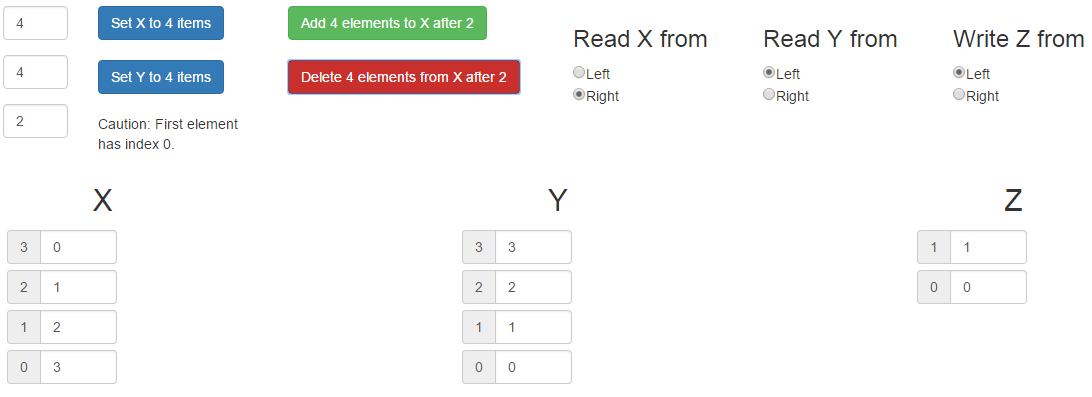


Рис. 14. Варіант Right-Left-Left-Delete

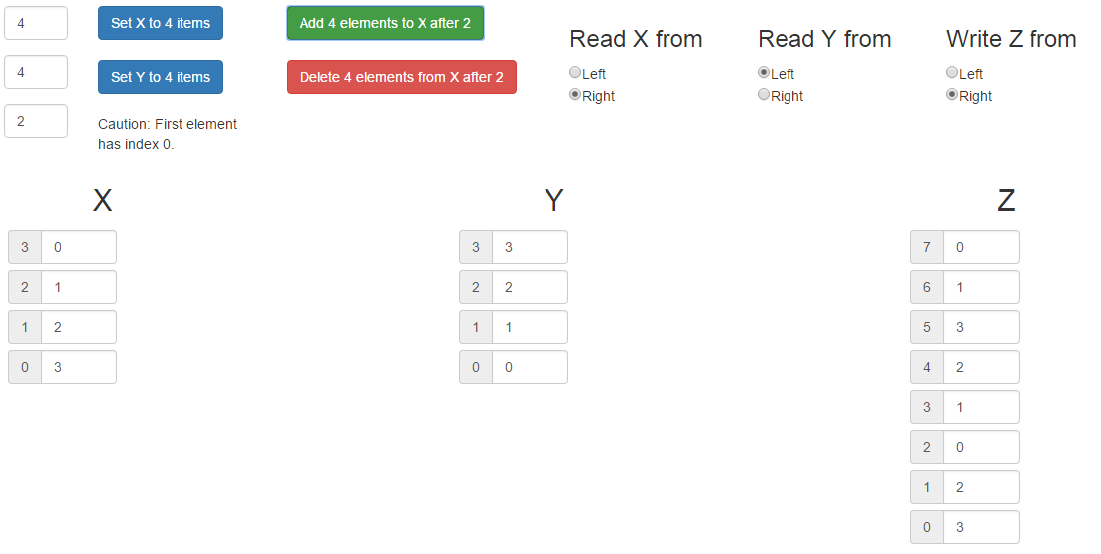


Рис. 15. Варіант Right-Left- Right -Add

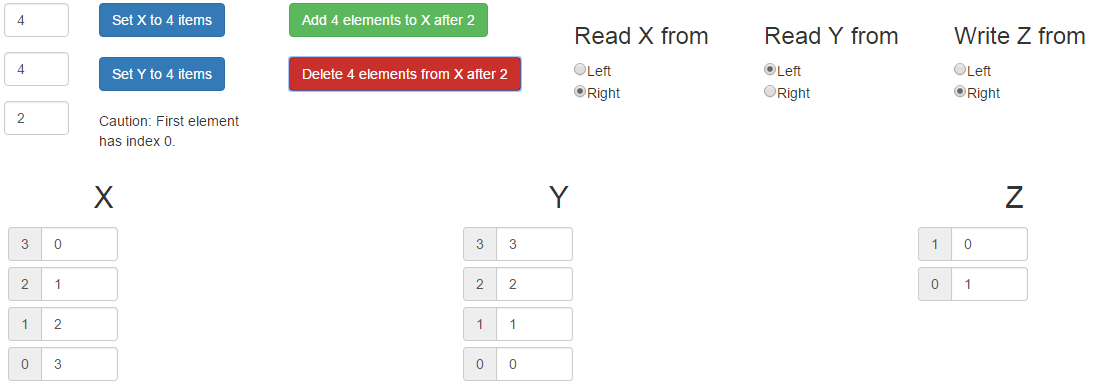


Рис. 15. Варіант Right-Left-Right-Delete

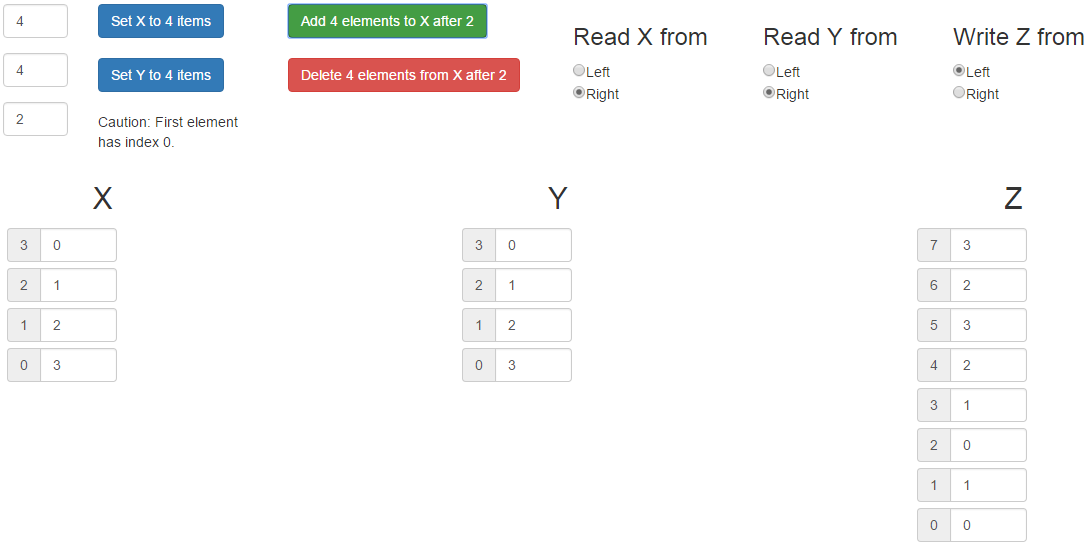


Рис. 16. Варіант Right-Right-Left-Add

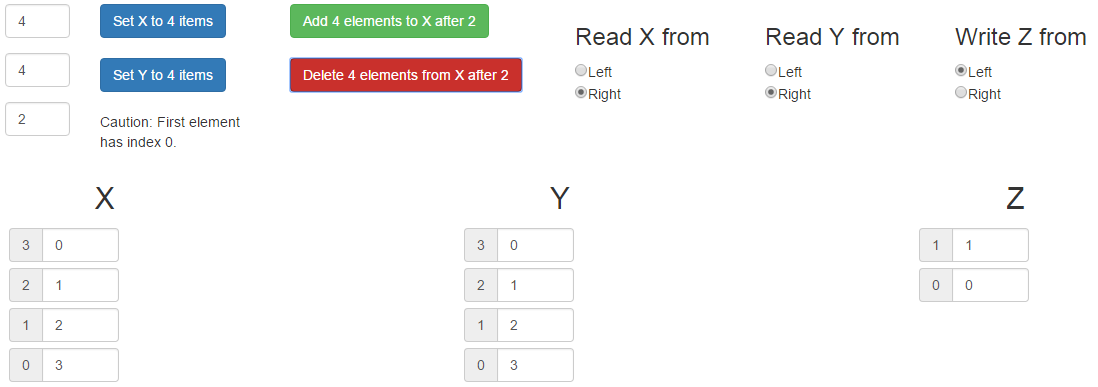


Рис. 17. Варіант Right-Right-Left-Delete

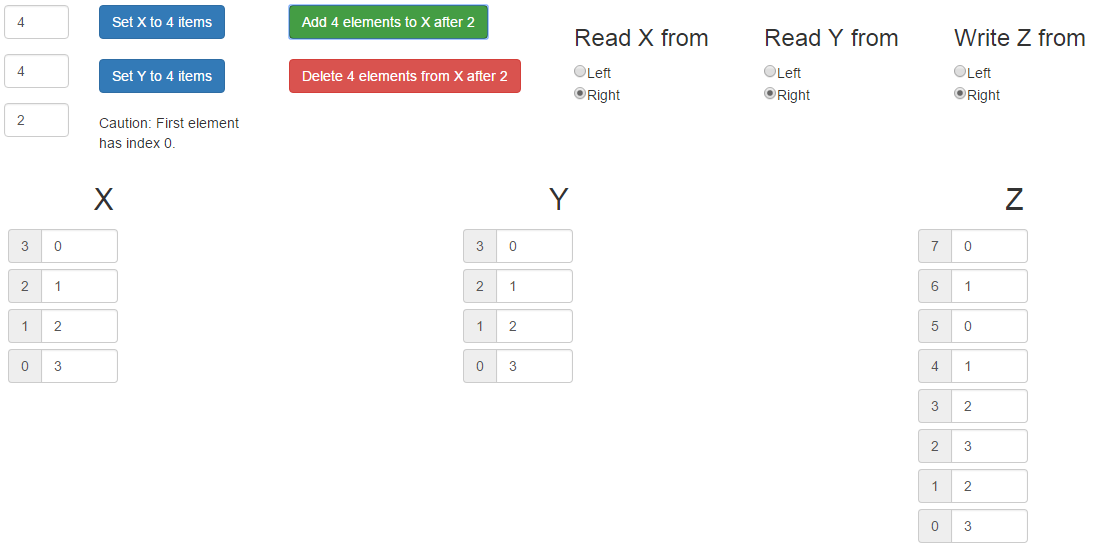


Рис. 18. Варіант Right-Right-Right-Add

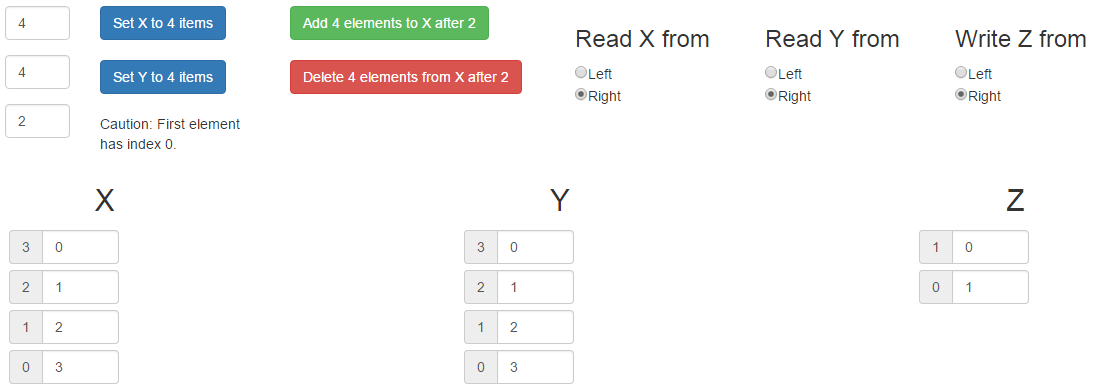


Рис. 19. Варіант Right-Right-Right-Delete