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 черг»

Виконав:

студент 3 курсу

групи КН-23

Михаленков Микола

Київ –2015

ЗМІСТ

[РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc431966937)

[РОЗДІЛ 2 ОПИСИ АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ 4](#_Toc431966938)

[2.1 Інструменти розв’язання задачі 4](#_Toc431966939)

[2.2 Початковий стан 4](#_Toc431966940)

[2.3 Опис алгоритму додавання *k*-елементів до черги довжини *n* після *l-*того елементу 4](#_Toc431966941)

[2.4 Опис алгоритму видалення *k*-елементів з стеку висоти *n* після *l*-того елементу 4](#_Toc431966942)

[РОЗДІЛ 3 БЛОК-СХЕМА ПРОГРАМНОГО ПРОДУКТУ 5](#_Toc431966943)

[РОЗДІЛ 4 ЛІСТИНГ КОДУ 6](#_Toc431966944)

[4.1 Файл index.html 6](#_Toc431966945)

[4.2 Файл style.css 7](#_Toc431966946)

[4.3 Файл script.js 7](#_Toc431966947)

[РОЗДІЛ 5 SCREENSHOTS 9](#_Toc431966948)

# РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

Написати програму обробки черги, яка передбачає:

1. Додавання *k*-елементів до черги довжини *n* після *l*-того елементу.
2. Видалення *k-*елементів з черги довжини *n* після *l*-того елементу.

# РОЗДІЛ 2 ОПИСИ 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.

Для коректної роботи програми потрібно підключення до інтернету.

## 2.2 Початковий стан

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

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

*n* – розмір черги X.

*k* – розмір черги Y.

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

## 2.3 Опис алгоритму додавання *k*-елементів до черги довжини *n* після *l-*того елементу

Алгоритм додавання *k*-елементів до черги довжини *n* після *l-*того елементу:

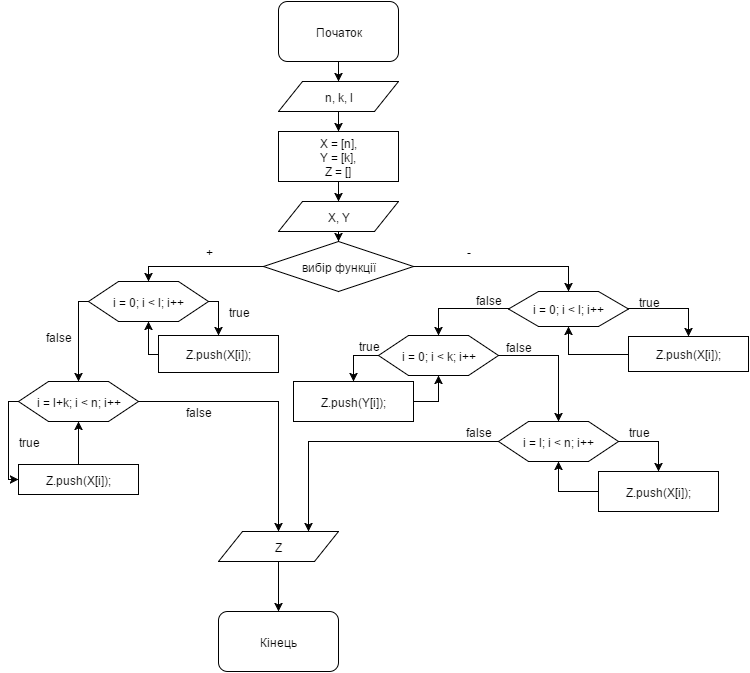
1. Присвоюємо черзі Z порожню множину.
2. Додаємо до черги Z послідовно всі елементи від першого до l-того з черги X.
3. Додаємо до черги Z послідовно всі елементи від першого до *k*-того з черги Y.
4. Додаємо до черги Z послідовно всі елементи від (*l*+1)-того до останнього з черги X.

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

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

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

# РОЗДІЛ 3 БЛОК-СХЕМА ПРОГРАМНОГО ПРОДУКТУ



# РОЗДІЛ 4 ЛІСТИНГ КОДУ

## 4.1 Файл index.html

<!DOCTYPE html>

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

<head>

<title>Queue | SA</title>

<meta charset="utf-8">

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

<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

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

</head>

<body ng-controller="DefaultCtrl">

<h1 class="text-center">Queue</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-3">

<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 1.</p>

</div>

<div class="col-lg-8">

<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><br><br>

</div>

</div>

<h2>X</h2>

<div class="queueParent">

<form class="form-inline">

<input type="text" ng-repeat="item in X" ng-model="X[$index]" class="form-control" size="5"></input>

</form>

</div>

<h2>Y</h2>

<div class="queueParent">

<form class="form-inline">

<input type="text" ng-repeat="item in Y" ng-model="Y[$index]" class="form-control" size="5"></input>

</form>

</div>

<h2>Z</h2>

<div class="queueParent">

<form class="form-inline">

<input type="text" ng-repeat="item in Z track by $index" class="form-control" ng-model="Z[$index]" ng-readonly></input>

</form>

</div>

</div>

</body>

</html>

## 4.2 Файл style.css

.form-inline {

border-color: #ddd;

border-width: 1px;

border-radius: 4px;

margin: 0 auto;

}

.form-inline > input {

margin: 5px;

}

## 4.3 Файл script.js

angular.module('DataStructureQueue', [])

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

$scope.X = [];

$scope.Y = [];

$scope.Z = [];

$scope.setX = function () {

$scope.X = [];

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

$scope.X[i] = i;

}

};

$scope.setY = function () {

$scope.Y = [];

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]);

console.log(i);

}

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

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

console.log(i);

}

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

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

console.log(i);

}

};

$scope.Delete = function () {

$scope.Z = [];

if ($scope.lValue < $scope.X.length) {

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

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

console.log(i);

}

if (($scope.lValue + $scope.Y.length) < $scope.X.length) {

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

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

console.log(i);

}

}

} else {

console.log("l > X");

$scope.Z = ["ERROR: nothing to delete."];

}

}

});

# РОЗДІЛ 5 SCREENSHOTS

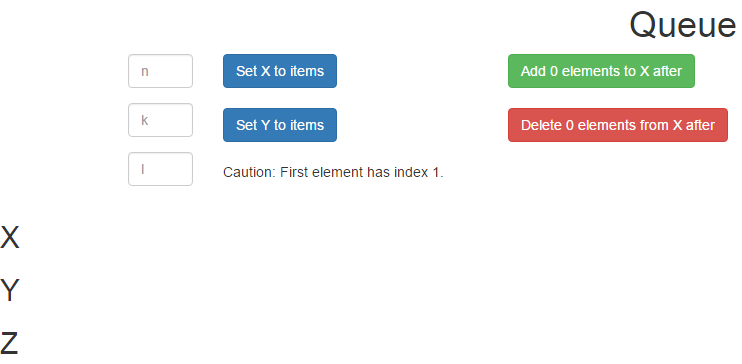


Рис. 1. Початкове вікно програми.

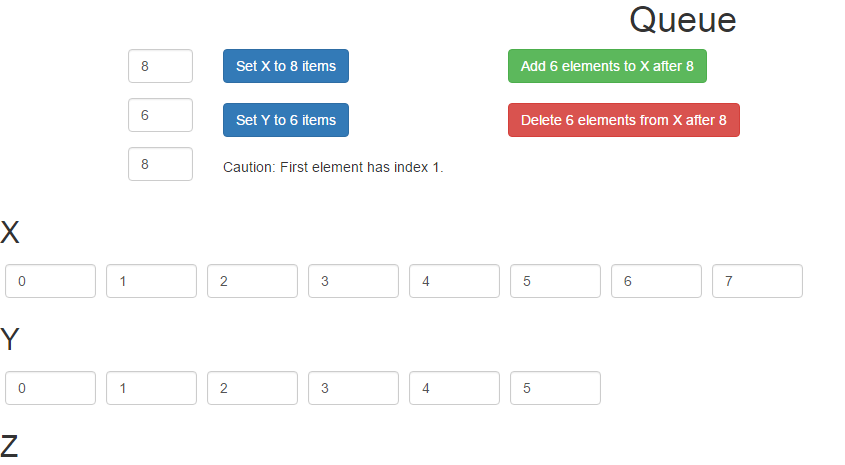


Рис. 2. Вікно програми після вводу умови.

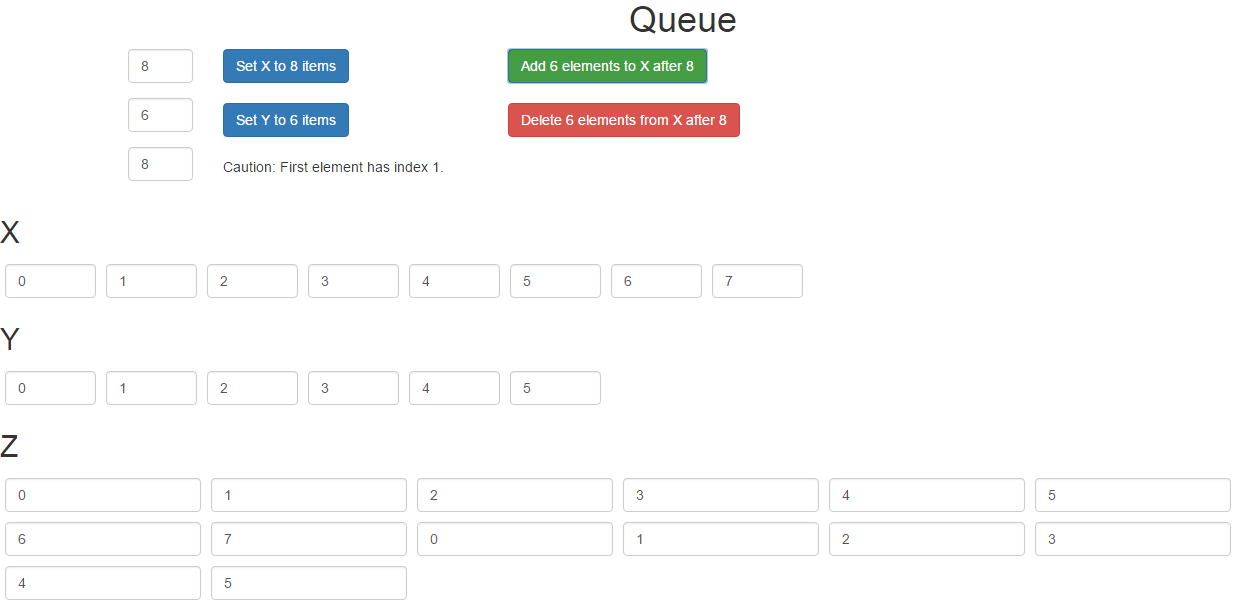


Рис. 3. Результат виконання функції додавання.

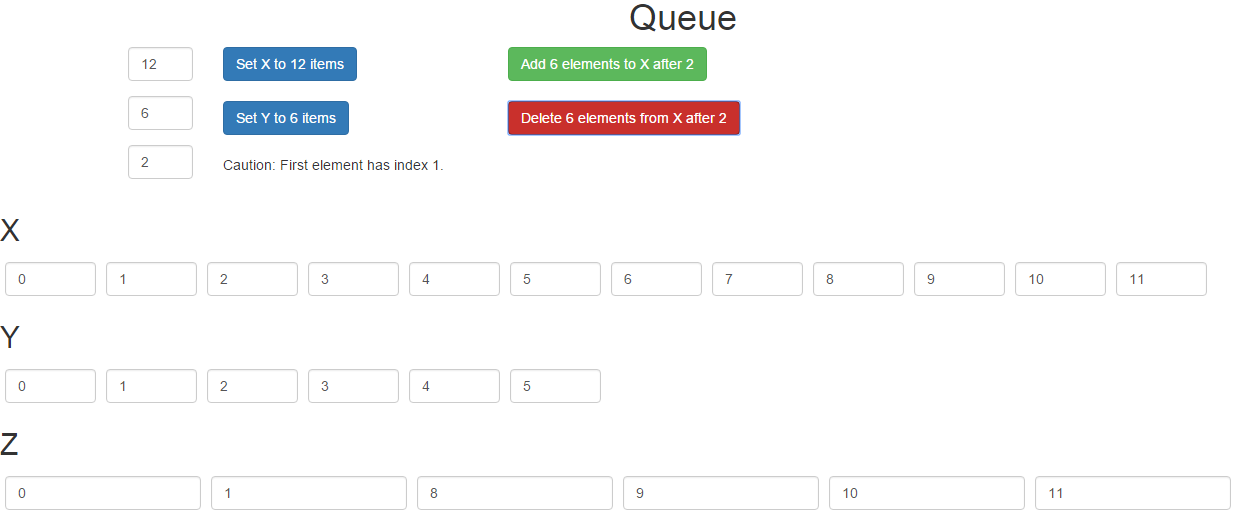


Рис. 5. Результат виконання функції видалення.