**Паркинг**

**Общ преглед**

Tрябва да реализирате собствена структура от данни, която да позволява добавяне и премахване на елементи, проверка за налични и др. – всичко това ще работи чрез **команди**, които вие ще получавате. Поредицата от команди приключва с „**END**”. За ваше удобство ще получите готов Program.cs файл, и ще трябва да реализирате само необходимия класове **DataList.cs**

**Основната идея се базира на това, че DataList е структура, която трябва да се държи като динамичен списък. Тоест елементите следва да бъдат подредени в стрингов масив. При нужда масивът да бъде разширяван, при триене от него не бива да остават празни места и тн.**

**Подзадача 1: 30 точки**

**DataList**

DataList класът държи информация за всички елементи, брой и капацитет

data – **стринг масив**

count – **брой**

capacity – **капацитет**

|  |
| --- |
| DataList.cs |
| public class DataList {    private const int DEFAULT\_CAPACITY = 3;    public DataList(int capacity = DEFAULT\_CAPACITY)  {      //TODO: Добавете вашия код тук …  }    public int Count  {      //TODO: Добавете вашия код тук …  }    public int Capacity  {      //TODO: Добавете вашия код тук …  }    public void Add(string element)  {      //TODO: Добавете вашия код тук …  }    public bool Contains(string element)  {      //TODO: Добавете вашия код тук …  }    public bool RemoveByIndex(int index)  {      //TODO: Добавете вашия код тук …  }    public void ReplaceLastWithFirst()  {      //TODO: Добавете вашия код тук …  }    public StringBuilder Info()  {      //TODO: Добавете вашия код тук …  }    public void Resize()  {      //TODO: Добавете вашия код тук …  } |

**Класът DataList трябва да имплементира следните методи**

**При създаване на обект от типа DataList.cs се очаква, че първоначалният капацитет е 3 елемента.**

**При достигане на капацитета и опит за добавяне на нов елемент – увеличете капацитета по формулата**

***capacity = capacity \* 2***

* **Метод за добавяне на елемент**

**Add** <**element**> - този метод има за цел да добави стойността към масива от данни на първата свободна позиция. Тоест първият добавен елемент отива на позиция 0, следващият на позиция 1 и тн.

* **Метод за генериране на информация относно структурата**

**Info**– Трябва да съберете информация за всички налични елементи в структурата в StringBuilder като спазвате следния формат:

**Count**: {брой елементи }  
**Capacity: {**текущ капацитет**}**  
**{element}  
{element}  
…**

**В случай на празна колекция добавете само първите два служебни реда. Всяка добавена стойност трябва да е на нов ред**

**Подзадача 2: 30 точки**

* **Метод за проверка на наличност на елемент**

**Contains <element**> - Трябва да бъде намерен елемент, който отговаря на подадения. При успешно намиране на такъв трябва да върнете **true**в обратен случай **false**

**Метод за замяна -  [ foo, bar]**-> **[bar, foo]**

**ReplaceLastWithFirst** – Следва последният елемент от колекцията да стане начален, а старият начален да премине на последна позиция. Ще ви бъде гарантирано, че имате поне два елемента.

**Подзадача 3: 20 точки**

* **Метод за премахване на елемент по индекс**

**Remove** **<index**> - Трябва да бъде премахнат елемента на подадения индекс. Всички следващи елементи следва да бъдат преместени наляво, с което целим да няма дупки в масива

**[foo, foo1, bar] -> Remove 1 -> [foo, bar]**

При успешно премахване на елемент трябва да върнете стойност**true**

При получаване на индекс извън обхвата на колекцията – напр. oтрицателно число – върнете**false.**

**Подзадача 4: 20 точки**

* **Метод за преоразмеряване**

**Resize**– Трябва да увеличите капацитета на масива по формулата **capacity = capacity \* 2.**Командата може да бъде подадена и върху празна колекция, което все пак трябва да промени капацитета.

**Kоманди подавани на конзолата**

Вашето приложение реализира следните команди:

* **Add < element > -**Добавя <**element**> към структурата
* **Info** – Изпечатва се информация за всички налични елементи
* **Check <element> -**При наличие на елемента - **<Element is present>** , в обратен случай - **<Element is not present>**
* **Replace** **–**Заменя местата на първият и последният елемент **[foo, bar] -> [bar, foo]**
* **Resize** – Преоразмерява колекцията като увеличава капацитета двойно
* **Remove <int index>**- Премахва елемент на посочения индекс. При успех - **<Removed element>**, в обратен случай - **<Could not remove element>**
* Програмата ще получава множество редове с информация. Всеки ред представлява команда. Самият вход се обработва изцяло от примерния Program.cs.
* Всички команди приключват с въвеждането на END

**Вход / Изход**

**Вход**

**Изход**

За някои от командите не е нужно да извеждате нищо. За други е необходимо форматиране на изход – напр. **Info**

**Ограничения**

* Имената няма да съдържат интервал

**Примери**

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Add element1  Add element2 |  |
| Info | Count: 2  Capacity: 3  element1  element2 |
| Add element3  Add element4 |  |
| Info | Count: 4  Capacity: 6  element1  element2  element3  element4 |
| Replace |  |
| Info | Count: 4  Capacity: 6  element4  element2  element3  element1 |
| Check element1 | Element is present |
| Check adsadadsa | Element is not present |
| Resize |  |
| Info | Count: 4  Capacity: 12  element4  element2  element3  element1 |
| Remove 1 | Removed element |
| Info | Count: 3  Capacity: 12  element4  element3  element1 |
| END |  |

**Requested files**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Exam

{

class Program

{

static DataList list = new DataList();

static void Main(string[] args)

{

string line;

while ("END" != (line = Console.ReadLine()))

{

string[] cmdArgs = line.Split(' ');

switch (cmdArgs[0])

{

case "Add":

AddElement(cmdArgs[1]);

break;

case "Resize":

ResizeStructure();

break;

case "Remove":

RemoveByIndex(cmdArgs[1]);

break;

case "Check":

Contains(cmdArgs[1]);

break;

case "Replace":

ReplaceLastWithFirst();

break;

case "Info":

StructureInfo();

break;

}

}

}

private static void ReplaceLastWithFirst()

{

list.ReplaceLastWithFirst();

}

private static void StructureInfo()

{

StringBuilder info = list.Info();

Console.WriteLine(info.ToString().Trim());

}

private static void RemoveByIndex(string index)

{

bool released = list.RemoveByIndex(int.Parse(index));

if (released)

{

Console.WriteLine("Removed element");

}

else

{

Console.WriteLine("Could not remove element");

}

}

private static void Contains(string element)

{

bool contained = list.Contains(element);

Console.WriteLine(contained ? "Element is present"

: "Element is not present");

}

private static void ResizeStructure()

{

list.Resize();

}

private static void AddElement(string element)

{

list.Add(element);

}

}

}

**DataList.cs**