# Aлгоритми върху линейни структури от данни -2

## Намиране на най-малко число

**Напишете** програма, която чете от конзолата **последователност от цели числа** на един ред, разделени с интервал**.** Намерете най-малкото от тях и го изведете.

Подсказка

1. В променлива minimum запишете първото число
2. Сравнявайте това число с всички останали (от второто до последното) и ако някое от тях е по-малко, то в minimum запазете неговата стойност
3. Изведете стйността на minimum.

## Наредени двойки

**Напишете** програма, която чете от конзолата последователност от **две цели числа m и n** . Да се изведат всички възможни наредени двойки цели числа **( p, q )** които се менят съответно за **p [1..m], q в [1..n]**

## Сливане на списъци

На **два реда** от конзолата се въвеждат **два подредени** **списъка от цели числа List<int>** с разделител интервал. Да се изведе нов списък **List<int>, в който да са слети двата списъка, отново подредени.**

## Подреждане на думи

**Определете сложността на програма**, която чете от конзолата **последователност от думи (**символни низове на един ред, разделени с интервал**).** Подредете ги по азбучен ред. Запазете последователността в List<string>

Подсказка

Използвайте алгоритъма от задачата „Намиране на най-малко число“, променете и допълнете

1. Намерете „най-малката“ по азбучен ред дума и я запазете в променлива minimum и запомнете поицията ѝ в променлива minimumPos
2. На позиция minimumPos в списъка запишете първата дума от списъка
3. На първа позиция в списъка запишете стойността minimum
4. Повторете стъпки от 1 до 3 за елементите на списъка от втора до последа позиция

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| wow softuni alpha | alpha softuni wow |
| hi | hi |

## Най-дълга последователност

**Съставете програма**, която намира най-дългата последователност от равни числа в даден списък от цели числа List<int> и връща резултата като нов List<int>. Ако няколко поредици имат същата най-дълга дължина, върнете най-лявата от тях.

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 12 2 7 4 **3 3** 8 | 3 3 |
| **2 2 2** 3 3 3 | 2 2 2 |
| 4 4 **5 5 5** | 5 5 5 |
| **1** 2 3 | 1 |
| **0** | 0 |
| 4 2 3 **4 4** | 4 4 |

## Remove/Add Method

**Определете сложността (максималния брой стъпки) на програма**, която чете от конзолата **възходящ** **списък** **от цели числа** на един ред, разделени с интервал и на втори ред число, за което се проверява дали е в списъка или не. Ако е, то то се **премахва** от него , а ако го няма – се **добавя** на такова място, че списъка отново да е подреден. Изведете:

1. Новополучения списък
2. Двата списъка – входния и новополучения

## Средно аритметично и сума на спъсък

Напишете програма, която прочита от конзолата поредица от цели положителни числа. Поредицата спира когато се въведе празен ред. Програмата трябва да изчислява сумата и средното аритметично на поредицата. Използвайте **List<int>**.

## Обръщане на последователността

Напишете програма, която прочита N цели числа от конзолата и ги отпечатва в обратен ред. Използвайте класа**Stack<int>**.

## Филтриране

Напишете програма, която премахва всички отрицателни числа от дадена редица.

Пример: array = {19, -10, 12, -6, -3, 34, -2, 5} 🡪 {19, 12, 34, 5}

##  Филтриране на нечетен броя срещания

Напишете програма, която при дадена редица изтрива всички числа, които се срещат нечетен брой пъти.

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4, 2, 2, 5, 2, 3, 2, 3, 1, 5, 2 | 5, 3, 3, 5 |

## Честота на срещания

Напишете програма, която по даден масив от цели числа в интервала [0..1000], намира по колко пъти се среща всяко число.

Пример: array = {3, 4, 4, 2, 3, 3, 4, 3, 2}

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3, 4, 4, 2, 3, 3, 4, 3, 2 | 2 - 2 пъти3 - 4 пъти4 - 3 пъти |

## BFS обхождане в ширина

Използвайки опашка реализирайте пълно обхождане на всички дирек­тории на твърдия ви диск и ги отпечатвайте на конзолата. Реализи­райте алгоритъма "обхождане в ширина" – Breadth-First-Search (BFS) – може да намерите стотици статии за него в Интернет.

## Честота на срещания

Използвайки опашка реализирайте пълно обхождане на всички дирек­тории на твърдия ви диск и ги отпечатвайте на конзолата. Реализи­райте алгоритъма "обхождане в дълбочина" – Depth-First-Search (DFS) – може да намерите стотици статии за него в Интернет.

## Мажорант на масив

Мажорант на масив от N елемента е стойност, която се среща поне N/2+1 пъти. Напишете програма, която по даден масив от числа намира мажоранта на масива и го отпечатва. Ако мажоранта не съществува – отпечатва "The majorant does not exists!”.

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2, 2, 3, 3, 2, 3, 4, 3, 3 | 3 |
| 2, 3, 4, 5, 6, 7, 8, 3 | The majorant does not exists! |

## Мода на масив

Мода на масив от N елемента е стойност, която се среща най-често. Напишете програма, която по даден масив от числа намира модата на масива и го отпечатва. Ако има няколко моди се извежда средно аритметичната им стойност

|  |  |  |
| --- | --- | --- |
| **Вход** | **Пояснения** | **Изход** |
| 2, 2, 3, 3, 2, 3, 4, 3, 3 |  | 3  |
| 3, 3, 4, 5, 6, 7, 4, 2, 2 | 3, 4 и 5 се срещат по два пъти => (3+4+5)/2=6 | 6  |

## Министерство на образованието и науката (МОН)

* Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

* Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен** **лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).

