# Упражнения: Качествен код и преработване

## Завъртаща разходка в матрица

### Разбиране на задачата

Дадена ви е квадратна матрица с **n\*n** клетки. Започвате да обхождате матрицата като **започвате** от **горния ляв ъгъл** и вървите в посока **надолу дясно**. Когато няма **продължение** в текущата посока (достигате до стена на матрицата или попълнена клетка), **посоката** се променя по посока на часовниковата стрелка. Осемте възможни посоки са:

8-directions

Когато **няма празна** клетка във **всички посоки**, "разходката" започва **наново** от свободна клетка в най-малкия **възможен ред** и възможно най-близо **до началото** на този ред. Когато **няма свободна клетка** вляво в матрицата, движението е **завършено**.

Вашата задача е да напишете програма, която чете от конзолата числото n (1 ≤ **n** ≤ 100) и отпечатва попълнената **матрица на конзолата**.

#### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| **3** | **1 7 8**  **6 2 9**  **5 4 3** | **Започваме** от **горния** **ляв ъгъл** и започваме да се движем по **диаголнала** **(1-2-3)**. След това достигаме до дъното на **долния десен ъгъл**. Посоката се променя по **часовникавата посока** и попълването на матрицата **продължава** **(4-5)**. Посоката отново се **променя** докато не се попълни матрицата **(6-7-8-9)**. |
| **6** | **1 16 17 18 19 20**  **15 2 27 28 29 21**  **14 31 3 26 30 22**  **13 36 32 4 25 23**  **12 35 34 33 5 24**  **11 10 9 8 7 6** | **Започваме** от **горния ляв ъгъл** и започваме да се движем по **диагонала** **(1-2…6)**. След като достигнем до **долния десен ъгъл**, **променяме посоката** по **часовниковата стрелка** и продължавме да попълваме матрицата **(7-8…11)**. Посоката се променя докато няма **свободна клетка в никаква посока** **(12-13…15)**, **(16-17…20)** … **(30)**. Когато няма свободна клетка в **никаква посока**, **разходката започва наново** от свободна клетка в **най-малкия възможен ред** и възможно **най-близо до началото на този ред** **(31)**. След намирането на свободна клетка матрицата **продължава да се попълва** **(31-32…36)**. |

### Вход

Входът ще бъде **позитивно число** – n – **броя на клетките** [0…100].

### Изход

**Отпечатайте** на конзолата **попълнената матрицат**а. Всеки **елемент** (число) в матрицата трябва бъде в поле от **4 символа**:



## Преработване и изчистване на кода

Използвайте **решението** "Matrica.sln".Подобрете качество на кода в него. Може да следвате следните стъпки**:**

1. **Отворете проекта** в VS и **разгледайте класовете**, **структурата на проекта**, и т.н.
2. **Стартирайте** кода, за да видите дали да се е компилира правилно.
3. **Вижте** дали има разлики между **изхода от конзолата** и **примерния изход**.
4. **Започнете** да четете кода
   * **Разберете** какво прави всеки един метод.
   * **Разберете** какво прави всяка една променлива.
   * Ако трябва, можете да дебъгвате.
5. Помислете какви **промени можете да направите**.
   * Вземете бележки, ако трябва.
6. **Преработете** кода:
7. **Преформатирайте** кода.
8. **Променете имената на лошо** написаните променливи.
9. **Потърсете** за **магическите числа**.
10. **Променете имената на лошо** написаните методи.
11. **Променете методите** с **еднакъв код**.
12. **Извлечете части от код**,които **работят заедно** в **нови методи**.
13. **Подредете** **методите правилно.**
14. **Разделете сложните булеви израза** на няколко **израза** или **метода**
15. **Добавете коментарите** на **частите от кода**, които са **трудни** за **разбиране.**
16. Направете **кода тестов**
17. **Напишете unit тестове. Проверите за** бъгове **докато тествате**.
    * **Тествайте** всички **случай** на **методите.**
    * Помислете как да тествате входа/изхода, базиран на конзолата.

**Преработете** **кода** по **следните стъпки** от упражнението. Направете го **стъпка** **по** **стъпка**. Стартирайте **unit** **тестовете** след всяка **голяма промяна**.

### Променете името на файловете

Първо трябва да погледнем **структурата на проекта** "Matrica.sln":



Както виждате, имаме **лошо** **написано име** на **проект** и **клас**. Помислете как може да променим "Matrica" и "Program.cs", така че да спазват добрите практики за наименуване. Не забравяйте да промените **namespaces-а**.

### Насоки

#### Лошо инициализирани и написани променливи

Вижте фрагмента от кода:



Трябва да променим кода. Не е четим. **Не е** добра практика да се **наименуват много променливи наведнъж**.

В допълнение, нашите променливи са кръстени лошо (не можете да определите каква стойност запазва **k**, защото името ѝ не го подсказва).

Промененият код трябва да изглежда така:

Graphical user interface, text

Description automatically generated

#### Магически числа

Mоже да видим, че в кода са използвани **магически числа** (директно се използва на число в кода):



Това е лошо, защото не е ясно **за какво са тези числа**. За да подобрим кода, може да преместим числата в **променлива** или **константа**, както е показано:

 🡪



#### Лошо написани методи

По-долу можете да видите метод, който има лошо наименувано име според стандарта на **наименуване на методи в C#**. Освен това той няма изрично зададен модификатор за достъп, което означава, че е **вътрешен** метод:

A picture containing icon

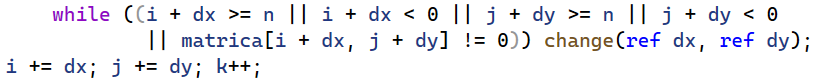
Description automatically generated

За да го оправим, трябва да променим **името на методи** чрез PascalCase. Трябва да го направим на public, за да може да се вижда във всички други класове (например: класовете за тестване):



#### Повтаряне на код и сложни булеви изрази

По-долу имаме **дублиране на код** и **сложни булеви изрази**:



Трябва да изчистим кода и да го променим, без да променяме неговата логика. Това става с разделяне на кода чрез методи и преизпълваното им. Накрая трябва да изглежда така:

Graphical user interface, text, application

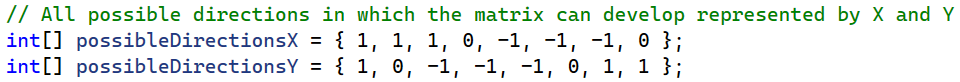
Description automatically generated with medium confidence

#### Коментари

Имаме някой места в кода, където **не е лесно** да се разбиране **логиката** или **променливите**:



В такъв случай може да използваме коментари, за да обясним кода с едно изречение:



Трябва и да изтрием **лошо написаните и безполезни коментари**:



### Примери за компонентни тестове

Не забравяйте да напишете **компонентни тестове**, за да тествате **логиката на приложението**.

За да постигнете това, трябва да направите **всички методи тестови** – методите трябва да имат ясен вход и изход.

След като напишете **позитивни** и **негативни теста** за всеки метод, трябва да проверите дали работят правилно. Тествайте всички частни случай и помислете как ще тествате тестовете от тип **вход/изход базирани на конзола** (можете да намерите лесен начин на

<https://www.pmichaels.net/2022/05/26/unit-testing-a-console-application/>).

Например трябва да имаме 4 тестови метода, които проверяват дали излизаме от матрицата в четирите посоки, и поне един **отрицателен тестов случай**:

