|  |
| --- |
| **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**  **Факултет – „Компютърни системи и управление“** |

**Инспекция на кода на програмата**

**Дисциплина – „Валидация и верификация на**

**програмни системи“**

**Тема – „Изчисляване на множествена регресия”**

**Студент:** Иван Николаев Колев, фак. № 121213114, гр. 43

**Специалност:** „Компютърно и софтуерно инженерство“

**Дата:** 4.22.2016 г**. Заверил:**

**/ доц. А. Алексиева/**

**София  
2017**

# Критерии за инспекция кода на програмата:

## 1. Структура на кода

* Следвани са наложените конвенции за писане на C# код.
* Кодът е добре структуриран (форматиране, консистентен стил на писане, отстояния).
* Кодът не съдържа методи, полета, пропъртита, които не се използват.
* Кодът не съдържа методи, които могат да бъдат заменени от вградени такива в езика компоненти или функции от външни библиотеки.
* Кодът не съдържа повтаряеми фрагменти код, които могат да бъдат изнесени в отделен метод/клас.
* Кодът използва ясно дефинирани константи и променливи, вместо „магически“ числа и низове.
* Всички модули са достатъчно прости и разделени на отделни процедури.

## 2. Документация

* Кодът е ясно написан и правилно документиран спазвайки наложените конвенции за писане на коментари в C#.
* Кодът не съдържа излишни коментари.
* Кодът не съдържа стари, неактуални коментари.
* Кодът не съдържа повтарящи се коментари.
* Кодът не съдържа закоментирани редове код.
* Всички коментари са написани консистентно и ползват един и същи стил на форматиране.

## 3. Променливи

* Всички променливи са дефинирани с ясни, точни и значещи имена.
* Всички променливи са дефинирани за работа с подходящ тип данни.
* Кодът не съдържа неизползвани или излишни променливи (които могат да бъдат заменени от вече съществуващи такива).

## 4. Аритметични операции

* В кода се избягва пряко сравнение на числа с плаваща запетая. Използват се допустими отклонения от желаната стойност, при сравнение на числа с плаваща запетая (пример: +- delta = 0.001).
* В кода се прихващат случаи, в които могат да възникнат грешки при закръгляне на числа с много голяма точност след десетичната запетая.
* В кода се прихващат случаи, в които могат да възникне превъртане на типа данни (e.g. integer overflow)
* В кода са прихванати слуачи, в които може да се извърши делене на нула.

## 5. Цикли и условни конструкции

* Всички условни конструкции, цикли и логически разклонения са пълни, коректни и правилно вложени.
* Най-честите случаи, които се очаква да покрият условните конструкции, са прихванати първи.
* Всички възможни случаи/състояния на даден обект са прихванати от условните конструкции.
* Всички “switch-case”-ове имат имплементиран “default case”.
* Всички условия за прекъсване на цикъл са ясни и достижими.
* В циклите всички индекси, които се достъпват са във валидни граници.
* Всички фрагменти код, които могат да бъдат изпълнени преди тялото на цикъла, са изнесени от него.

## 6. Защитно програмиране

* Всички входни параметри на даден метод са тествани за валидност и пълнота.
* Всички изходящи параметри имат присвоена стойност различна от “null”.
* Всички методи работят върху правилните данни.
* Всички съобщения за грешка са смислени, разбираеми и пълни.

## 7. Качество на кода

* Всички модули са реализирани с подходящо ниво на абстракция.
* Методите имат подходящ брой и тип аргументи.
* Няма излишни функционалности, които не се изискват от заданието.
* Използвани са енумерации (именувани константи) вместо статус кодове или магически числа.
* Всички обекти се използват през интерфейсите им, а не през конкретната им имплементация.

## 8. Тестване на кода

* Компонентните тестове покриват 100% от логиката и поведението на всеки един компонент.
* Компонентните тестове прихващат всички възможни грешни случаи потенциално породени от некоректни условия или невалидни параметри.

## 9. Бързодействие на кода и оптимизации

* Всички изчисления, които се повтарят многократно са избегнати.