Функционално програмиране

Има различни парадигми за писане на код

* Процедурно програмиране – последователност от интструкции, които описват по точен начин как ще се случат нещата. Процедурното програмиране се нарича още **императивно** програмиране – ние казваме на програмата изрично какво да прави чрез различни команди
* Обектно-ориентирано програмиране - моделираме обектите от реалния свят
* Функционално програмиране:
  + Използват се резултатите от изчисляването на някакви математически функции за да се изведе даден резултат
  + **Декларативен** тип програмиране – самите функции не правят никакви действия директно, а само описват как трябва да се случат нещата

Има малка разлика между функции и методи и тя е, че методите са дефинирани в даден клас, а функциите-не

First-class функция – променлива, която сочи към дадена функция, и тази променлива може да се извиква като и се подават дадени параметри (delegates in C#)

Higher-order функции - функции, които приемат други функции като параметри (LINQ extensions in C#), или функции които връщат други фукнции като output

Closure-и – функции, които са дефинирани в други функции. По този начин може да се създават private променливи при функционалните езици

Чрез extension методите се добавят допълнителни функционалности към вече съществуващи типове (.NET-ски и наши типове)

Анонимните типове се използват когато трябва да държим съвкупност от стойности в даден обект и когато сме преценили, че няма нужда да създаваме отделен клас, от който да създзваме този обект

Lambda expression – анонимна функция, която съдържа параметрите, които приема (може и да не приема такива), и израза, който връща

LINQ - съвкупност от методи, които се извикват върху дадена колекция. По този начин се манипулират данните в тази колекция

* The extension methods use the enumeration property in order to do their work (e.g. to determine the count of the collection, LINQ’s Count() method enumerates the collection)

AsParallel() – enables the use of parallel threads

* Improves the performance **when dealing** **with heavy operations** (e.g. iterating 10 000 000 times some collection)
* Can be slower than non-parallel on small number of elements