Pôvodný projekt som dostal v rámci tréningu v práci. Dal som ho do pôvodného stavu a postupne na ňom predvediem zmeny ktoré som vykonal vtedy, a budú smerovať k asynch. behu programu.

Synchrónna verzia:

Cieľom programu je simulovať prípravu raňajok.

* Najprv sa vykonáva funkcia *PourCoffee()* – tá sa vykoná ihneď.
* *FryEggs()* – čas čakania je 6000ms
* *FryBacon()* – čas čakania 6000ms
* *ToastBread()* – čas čakania 3000ms

Pri verzii kde sa jednotlivé úlohy vykonávajú synchrónne je celkový čas niečo cez 15s.



Asynchrónna verzia:

Cieľom vylepšenej verzie je neblokovať vykonávanie v prípade, že akcia vyžaduje čas na dokončenie. Podobne ako v reálnom svete, nevložím vajíčka na panvicu a nečakám kým daná úloha neskončí. Počas čakania sa môžem venovať ostatným úlohám.

Prvý krok v kóde je pozmeniť funkcie aby boli asynchrónne, v jazyku c# je na to kľúčové slovo async. To oznámi že daná funkcie môže odovzdať riadenie. Druhá zmena je zmeniť kľúčové slovo Wait() na await. Kým Wait() tok riadenia zablokuje na daný čas, await odovzdá riadenie späť na volajúceho.

Ďalší krok je upraviť main tak, aby sa úlohy začali vykonávať kongruentne. Na začiatku teda vytvorím handlere a spustím každý task, keď teda prvý narazí na await, vráti riadenie a začne sa ďalśí a tak ďalej. Následne awaitujem jednotlivé tasky.

Vo výpise nižšie môžeme vidieť ako sa jednotlivé úlohy prehadzovali, výpis ktorý končí ‚...‘ sa nachádza pred volaním await. V výsledný čas je 6s. **Oproti pôvodnému to teda je zlepšenie o 9s.**

Riešenie sa zdá 100%. Avšak je tu ešte priestor na zlepšenie. Jednotlivé tasky by sa mohli awaitovať v rovnakom bode. Napríklad pomocou await Task.WhenAny(tasks). Takto to aj lepšie odzrkadľuje reálnu situáciu s varením, keďže necheme aby sa nám vajíčka spálili. To som do výsledného riešenia nezapracoval.

