# ПАРАЛЕЛИЗЪМ, СИНХРОНИЗАЦИЯ, ЕФЕКТИВНОСТ

## Характерен случай, свързан с необходимостта от синхронизация, е множествения достъп до глобални (общи, поделени) променливи/обекти.

Ще се спрем на два проекта от папка ***CPP11.Testbed*** – проект *0004* и *0005*. И в двата проекта се разглежда следната ситуация: създават се 10 паралелни нишки, които инкрементират брояча *safe\_counter*.

[*Проект 0004*](0004) илюстрира два случая на множествен достъп до тази обща променлива:

* без синхронизация на нишките;
* със заключване на множествения достъп чрез обекти от класа *std::mutex* и класа *std::lock\_guard*, които са част от стандартната библиотека *C++11*.

В първия случай се получава некоректен резултат, който при това е и различен при всяко стартиране, докато при втория случай резултатът е гарантирано правилен. За съжаление, заключването на достъпа е свързано със значителни *системни загуби*, които се проявяват със значително увеличаване на времето за изпълнение.

[*Проект 0005*](0005) илюстрира предимствата от използването на атомарни (неделими) обекти от класа *std::atomic*, също част от стандартната библиотека *C++11*. При атомарните обекти библиотечната реализация автоматично осигурява най-добрия начин за синхронизация на достъпа:

* при прости глобални променливи (*int*, *long*, *float*, *double*, …) се използва архитектурната поддръжка (специални машинни инструкции) и *загубите от синхронизация* са почти нулеви;
* при сложни глобални променливи обаче се налага заключване на обекти от класа *std::mutex*.