* Создать на языке Си функцию,
  + Умножающую два вектора. Размер векторов ( входных массивов) **N** элементов типа data\_sc.
    - Дать циклу, реализующему умножение, имя L1
    - Функция возвращает вычисленное значение с помощью оператора return.

Возвращаемое значение должно иметь тип данных data\_sc

* + В файле lab4\_1.h должны быть определены **N=8192** и тип данных data\_sc, имеющий тип int (32 бита).
* Создать на языке Си тест для проверки работы функции. Тест должен обеспечивать
  + Запуск функции 3 раза
  + проверку правильности полученного функцией результата.
* Отладить функцию и тест (при неправильном результате в любом из запусков функции тест должен сообщать об ошибке).
* Создать скрипт автоматизирующий процесс:
  + Создания проекта lab4\_z1,
  + Назначения функции lab4\_z1
  + Подключения файла lab4\_z1.c (папка source ),
  + Подключения файла lab4\_z1\_test.c (папка source),
  + Создания решения
    - sol1 … 4, для которых
      * Микросхема: xa7a12tcsg325-1Q
      * Период тактового сигнала: 6нс, 8нс, 10 нс, 12нс, uncertainty 0.1нс.
      * Си моделирование
      * Синтез
      * С/RTL cosimulation (с опцией полной трассировки)
* Используя средства HLS сравнить полученные решения 1 … 4 (привести Report Comparison) + заполнить таблицу exel
* Выбрать самое производительное решение из sol1 … 4
* Создать (в GUI) два решения
  + - Sol5, для которого
      * для цикла L1 задать директиву **Pipeline**
        + ***set\_directive\_pipeline "lab4\_z1/L1"***
      * Микросхема: xa7a12tcsg325-1Q
      * Период тактового сигнала: ***лучшее решение из sol1*** … ***4***, uncertainty 0.1нс.
      * Синтез
      * С/RTL cosimulation (с опцией полной трассировки)
    - Sol6, для которого
      * для цикла L1 задать директиву **Pipeline и rewind**
        + ***set\_directive\_pipeline -rewind "lab4\_z1/L1"***
      * Микросхема: xa7a12tcsg325-1Q
      * Период тактового сигнала: ***лучшее решение из sol1*** … ***4***, uncertainty 0.1нс.
      * Синтез
      * С/RTL cosimulation (с опцией полной трассировки)
* Используя средства HLS сравнить полученные решения 5 и 6 (привести Report Comparison) + заполнить таблицу exel
* Выбрать самое производительное решение из sol5 и sol6 для сравнения с программной реализацией

Исследование времени выполнения на ПК

* **Используются исходные коды и результаты исследования проведенного ранее.**
* На базе использованного выше Си теста создать отдельный, модернизированный, тест для проверки времени выполнения синтезируемой функции на ПК:
  + добавить в тест операторы измерения **времени выполнения** синтезируемой функции (например, как-то так: <https://solarianprogrammer.com/2019/04/17/c17-programming-measuring-execution-time-delaying-program/> ).
  + Увеличить количество запусков синтезируемой функции до 32. Для каждого запуска измерить время, найти среднее значение и вывести как результат.
  + Точность измерения времени (наносекунды).
  + Провести исследование времени выполнения синтезируемой функции на Вашем ПК
    - Осуществить компиляцию модернизированного теста и запустить его как отдельное приложение
    - В отчете привести:
      * Параметры Вашего ПК: тип процессора, частота работы процессора, объем ОЗУ
      * результаты измерения времени выполнения
* Оформить отчет, который должен включать
  + Задание
  + Раздел с описанием исходного кода функции
  + Раздел с описанием теста
  + Раздел с описание созданного командного файла
  + Раздел с анализом результатов (со снимками экрана с заполненной таблицей и полученным графиком)
    - Анализ и выбор оптимального (критерий максимальная производительность) решения
  + Результаты исследования времени выполнения на ПК и сравнение с аппаратными решениями.
  + Выводы

Архив должен включать всю рабочую папку проекта (включая модернизированный тест и скомпилированное приложение), отчет