Лабораторная работа 1
Вычисление собственных чисел матрицы

Цель работы

Изучить особенности работы с числами с плавающей точкой и массивами в C.

Стандарт языка и требования к работе

C99 и новее. Требования к работам

Описание

Программа должна находить собственные числа квадратной матрицы, состоящей из вещественных значений.

Входной файл в первой строке содержит одно число: размер матрицы N, после чего идут N строк по N чисел. Гарантируется корректность входных данных. N – натуральное число.

Выходной файл должен содержать найденные значения. Формат вывода: `%g`.

	Пример входных данных	Пример выходных данных
1	4 4 1 -2 2 1 2 0 1 -2 0 3 -2 2 1 -2 -1	6.84462 2.26853 1.08436 -2.19752
2	2 0 1.0 -1.0 0	0 +1i 0 -1i

Последним символом в файле должен быть символ перевода строки `'\n'`.

Примечание: ваша программа должна быть способна за разумное время (пара секунд) обрабатывать матрицы размером хотя бы 100x100 (N=100). Тем не менее, вам никто не гарантирует, что на вход в качестве N будет подано значение не больше 100: программа должна быть в принципе работоспособна с любым размером входных данных, умещающихся в оперативной памяти.

Формат аргументов командной строки

Аргументы программе передаются через командную строку:

<имя_входного_файла> <имя_выходного_файла>

Примечание про автотесты на Github

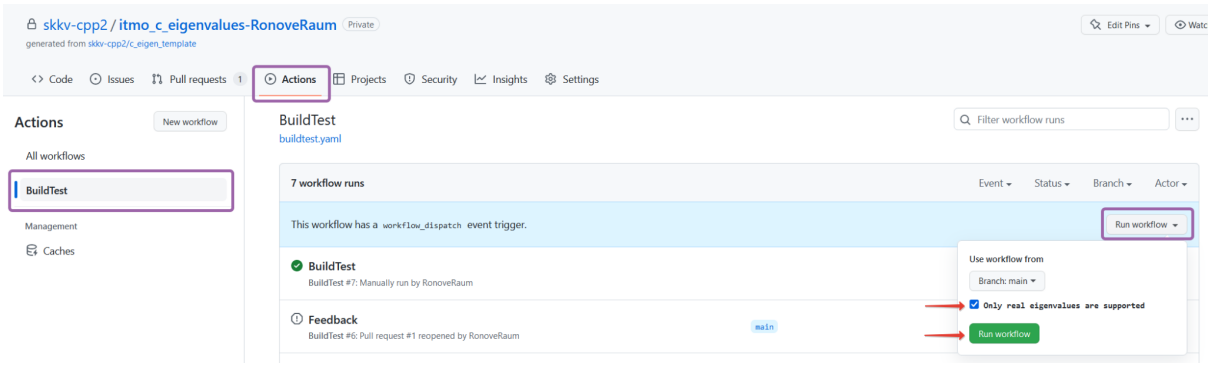
(скоро будет перенесено в памятку курса)

В этой работе в качестве тестовых примеров вам даётся 2 матрицы: с вещественными и комплексными собственными числами.

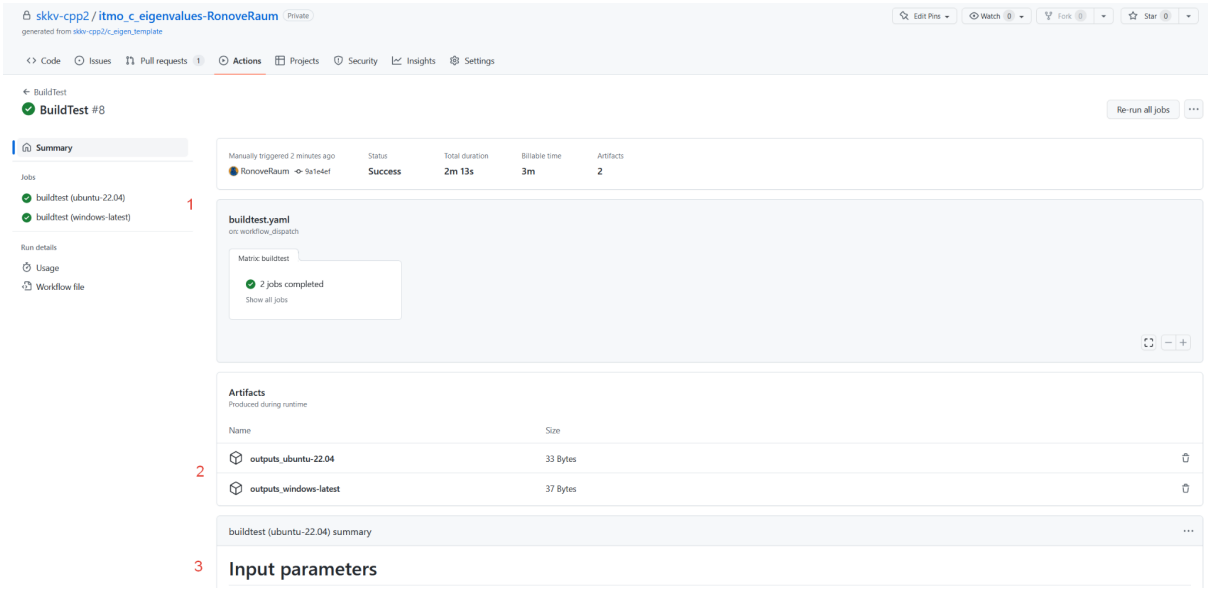
Чтобы у вас была на ранних этапах выполнения работы была возможность проверить работоспособность программы на автотестах на Github только на матрицах с вещественными собственными числами (real eigenvalues), была добавлена опция для запуска автотестов.

Чтобы запустить тесты на Github нужно:

1. Отправить в репозиторий файлы с исходным кодом.
2. Перейти в раздел **Actions**. Там выбрать workflow **BuildTest**, после чего в меню запуска **Run workflow проставить галочку**, что вы хотите провериться не на всех тестах (только первый из примера входных данных) **Only real eigenvalues are supported.** Если галочка не стоит, то будут запущены оба теста.



3. Дождаться окончания запуска. Новый запуск (run) появляется в списке запусков с некоторой задержкой (до 10 секунд), так что наберитесь терпения или press F5.
4. Ознакомиться с результатами запуска. Если всё прошло успешно, то в Summary запуска вы увидите что-то в таком духе (скрин ниже):
   4.1. 1 – запуски (job) на двух разных ОС
   4.2. 2 – полученные с сервера выходные файлы. Если запуск завершился неуспешно или выходные файлы не созданы, то данного “артефакта” может не быть.
   4.3. 3 – Summary по каждому запуску (job). Это отчет, который оформлен для вас в человекочитаемом виде, чтобы вы не мучались и не разбирались в логах.



Внимание №1: проверка на оформление кода **clang-format** является неблокирующей, поэтому удостовериться, что вы всё правильно отформатировали, должны самостоятельно, просмотрев логи. Если форматирование не соответствует ожидаемому, то вам будет сказано, в каких строках кода проблемы.

clang-format OK	clang-format !OK
buildtest (ubuntu-22.04) summary **Input parameters** 'Only real eigenvalues are supported' : True [note] if 'Only real numbers support' == true then only tests with real eigenvalues will be run **clang-format** `Checking formating files: main1.c` `clang-format OK` ✅	buildtest (ubuntu-22.04) summary **Input parameters** 'Only real numbers support' : False [note] if 'Only real numbers support' == true then only tests with real eigenvalues will be ru **clang-format** `Checking formating files: main11.c` `clang-format !OK` ❌ main11.c:6:13: warning: code should be clang-formatted [-Wclang-format-violations] char in1[] = "4\n" ^

Внимание №2: если тесты на Ubuntu не отработали полностью корректно, то на Windows проверка не будет запущена. Поэтому если вы видите, что проверка завершилась неуспешно и вы не понимаете почему, то вместо того, чтобы тратить попытки на Github (которые у вас ограничены), лучше сразу написать проверяющему о своей проблеме.