**Быстрое преобразование Фурье (функция fft)**

1. **Общая постановка задачи**

Требуется реализовать компонент для вычисления дискретного преобразования Фурье (DFT) с использованием алгоритма быстрого преобразования Фурье (FFT). Интерфейс должен поддерживать входные вектора со всеми типами, которые поддерживаются в MATLAB, а именно double, single, int8, int16, int32, uint8, uint16, uint32, logical. В качестве возращаемого вектора должен выстапать вектор, состоящий из комплексных чисел.

1. **Реализуемый алгоритм FFT**

Используется нерекурсивный вариант алгоритма Кули-Тьюки c оптимизацией реверса битов и без использования дополнительной памяти. При вычислении синусов и косинусов внутри алгоритма используется алгоритм CORDIC.

1. **Асимптотика**

Временная сложность:

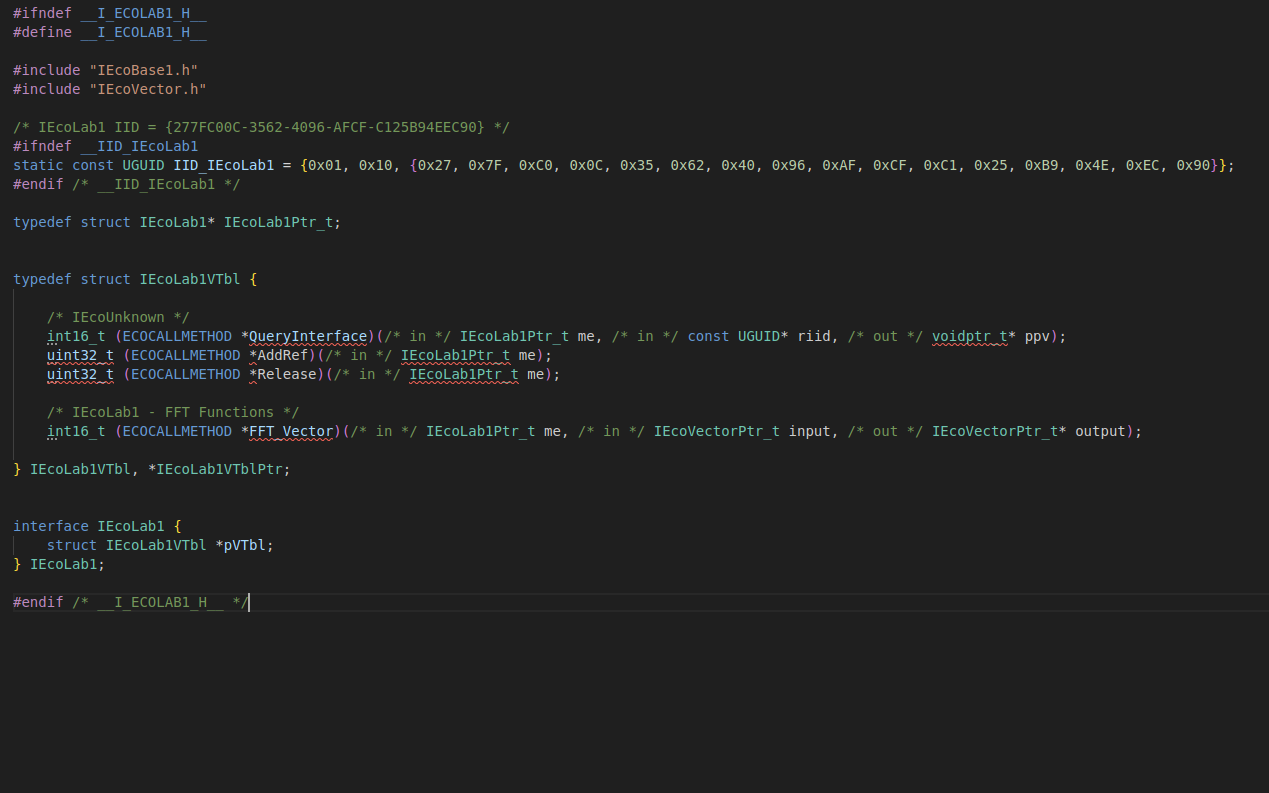
Средняя: O(Nlog⁡N).

Худшая: O(Nlog⁡N).

Сложность по памяти: O(N).

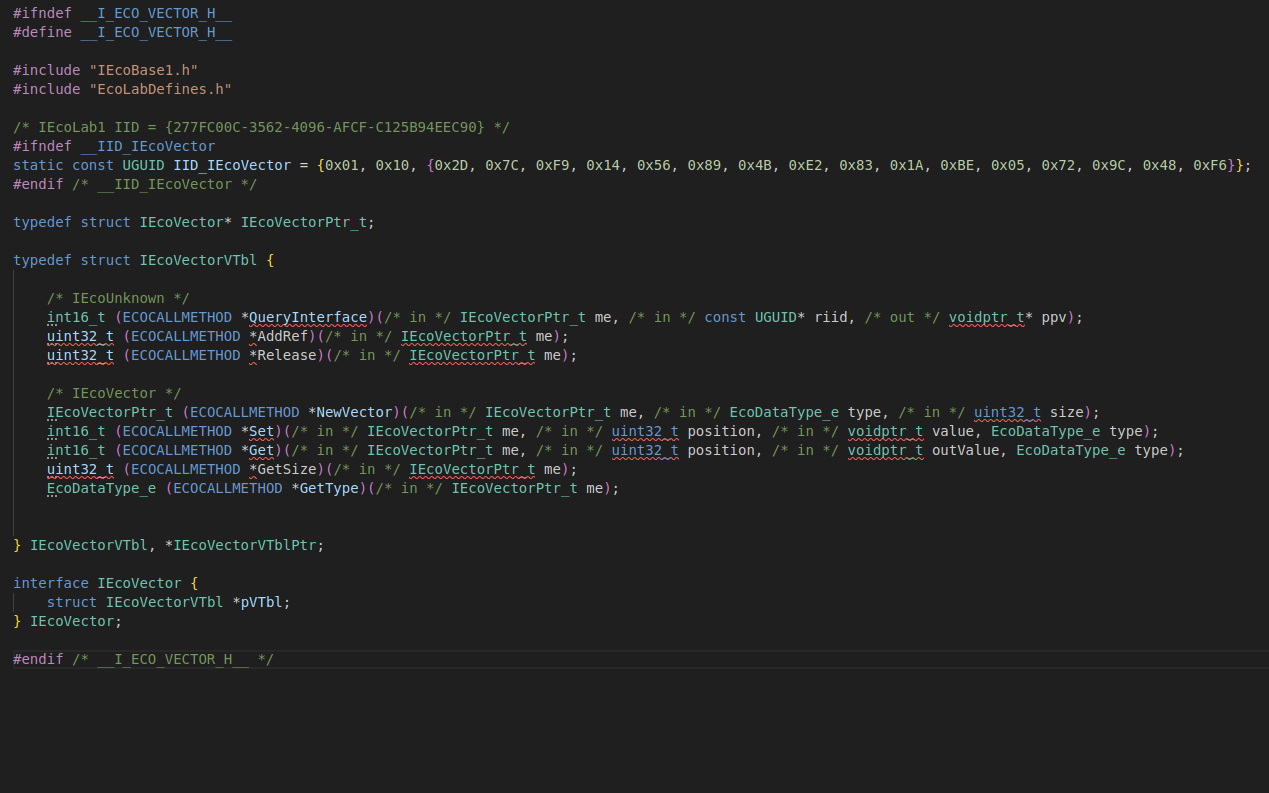
1. **Реализация.**

Для решения задачи были реализованы два интерфейса, один из которых - интерфейс самой лабы:



Единственный метод данного интерфейса, отличный от стандартных - метод FFT\_Vector, который принимает на вход проинициализированный вектор input, и возращает указатель на вектор output через переменную output.

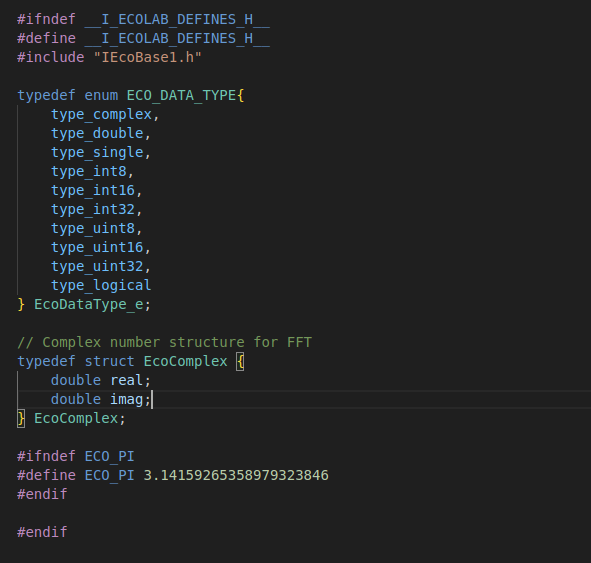
Другой - интерфейс вектора:



У вектора есть следующие методы:

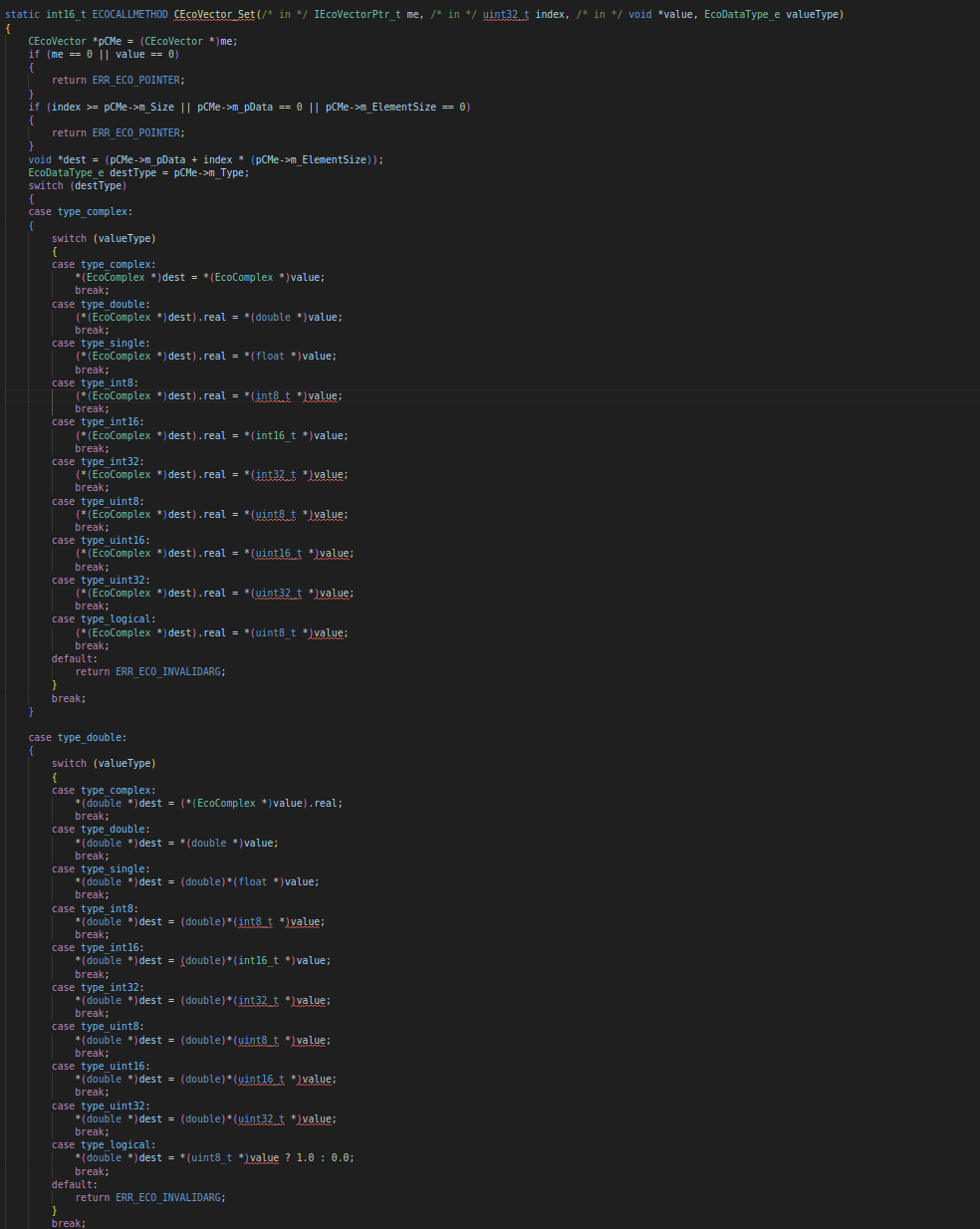
1. NewVector - создаёт новый вектор выбранного типа и размера
2. Set - установить элемент вектора равным value
3. Get получить элемент вектора в outValue
4. GetSize - получить размер вектора
5. GetType - получить тип данных массива

При этом вектор(и, соответственно, функция FFT) поддерживает следующие типы данных:

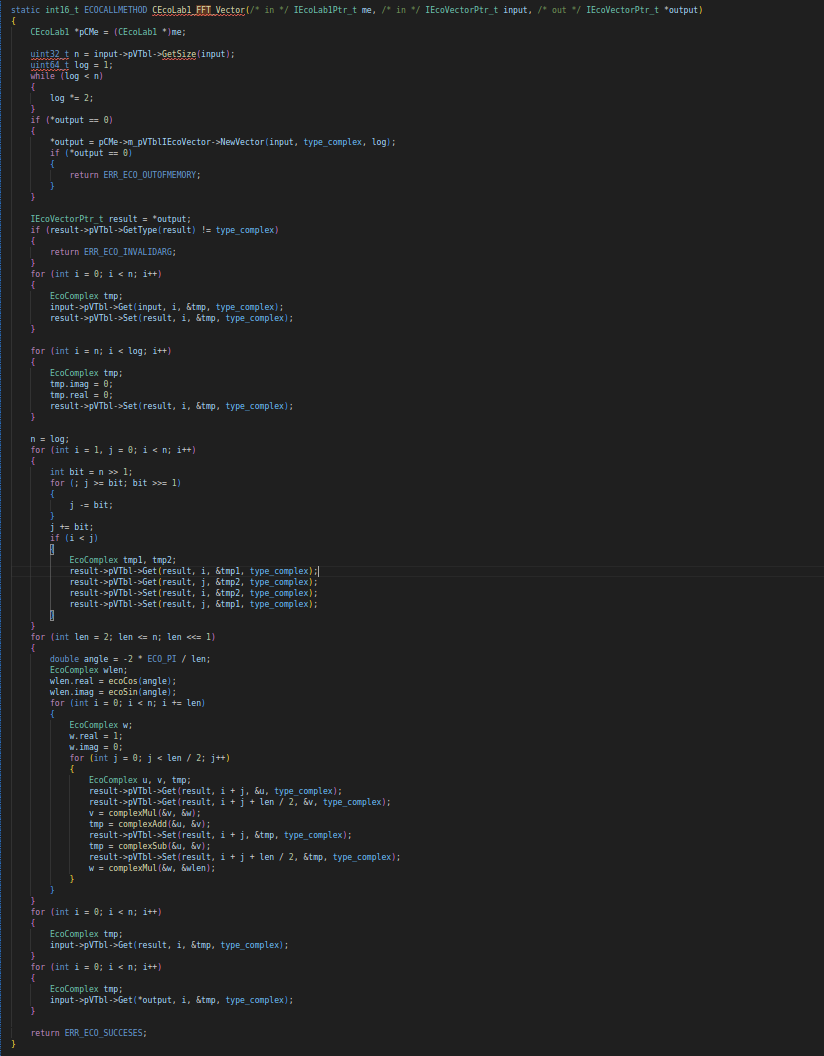


В методах Set и Get происходит безопасная конверсия типов в случае их несовпадения. Если при конверсии возникает ошибка(например происходит переполнение типа) или подаётся неизвестный тип, возращается ошибка ERR\_ECO\_INVALIDARG.

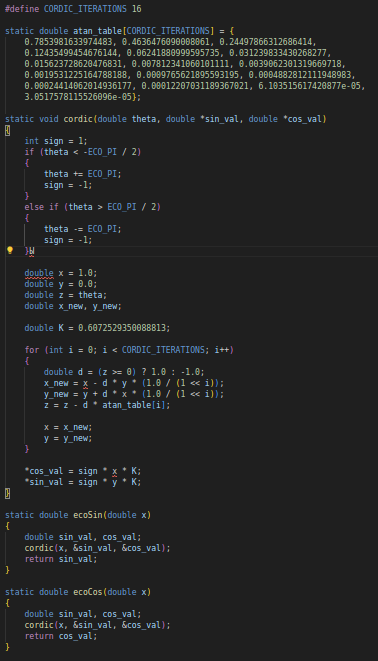
Часть реализации метода Set:



Реализация самого FFT:

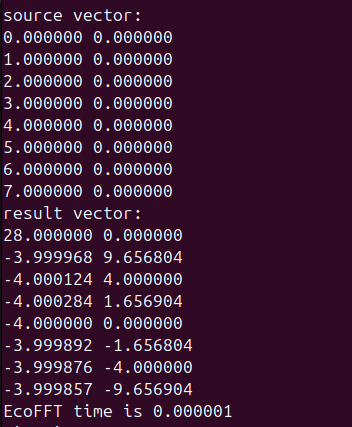


Реализация синуса и косинуса, работающая быстрее, чем обычная реализация через ряды Тейлора:



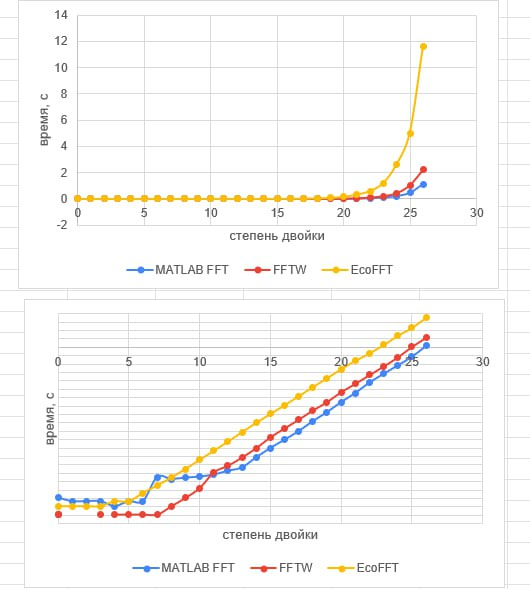
**5. Пример работы**

Результат работы на векторе размера 8:

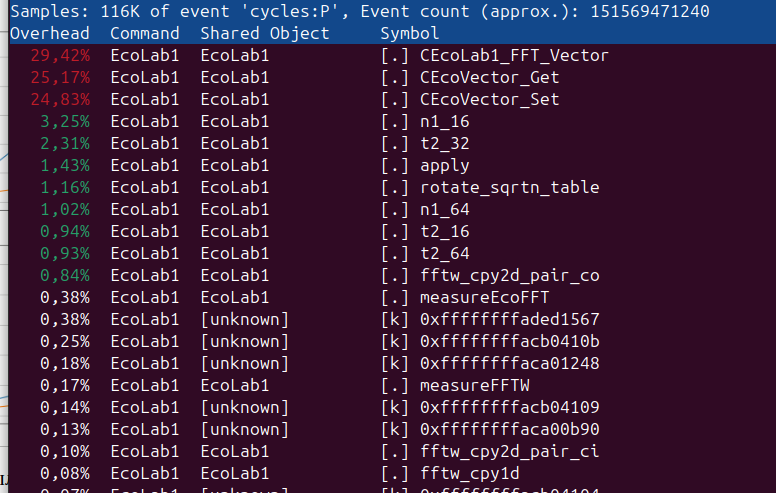


1. **Сравнение с MATLAB(Octave) и fftw**

Для сравнения качества работы алгоритма был написан код с использованием fftw, библиотеки с одной из самых быстрых реализаций fft, не являющейся специфичной для определённого процессора, и проведено сравнение с реализацией MATLAB fft. Так как MATLAB не доступен в России, я пользовался вместо него GNU Octave, некоммерческой реализацией MATLAB.



В целом, моя реализация медленнее fftw в 5 раз и matlab в 10 раз, но эти реализации принимают на вход комплексные числа, а в моей реализации вектора любых типов, и проверка типов вместе с безопасной конверсией замедляет мою реализацию, что видно по репорту perf-а, где ~50% сэмплов падает на set и get:



1. **Написание юнит тестов**

В качестве юнит теста было написано сравнение результатов алгоритма с реализацией из fftw. Для этого я была проведена проверка совпадения элементов результирующих векторов для всех степеней двойки от 0 до 16. В случае отклонения результирующего элемента на 0.01 от результата fftw, юнит тест выводит несовпавший элемент и завершает программу. Для всех тестируемых размеров массива такое не произошло.