Задача 2. BLAS в квадрате

Источник:	базовая I
Имя входного файла:	--
Имя выходного файла:	--
Ограничение по времени:	10 секунд*
Ограничение по памяти:	разумное

В этой задаче требуется вычислить матрицу в квадрате, используя функцию `dgemm` из настоящей библиотеки OpenBLAS. Полный набор файлов этой библиотеки, которые будут доступны при проверке, можно скачать здесь. Для использования следует подключить хедер `cblas.h` с интерфейсом, предназначенным для языка C. Подробную документацию функции `cblas_dgemm` можно посмотреть на сайте Intel здесь. Учтите, что тип `MKL_INT` — это на самом деле просто тип `int`.

Кроме того, необходимо написать собственную реализацию функции `cblas_dgemm` и собрать из неё динамическую библиотеку, которая будет бинарно совместимой с реализацией от OpenBLAS.

Часть 1. Используя библиотеку OpenBLAS, реализуйте в файле `matrixsqr.c` функцию:

```
// Computes R = A * A;
// Here A and R are square matrices N x N.
// Every matrix is stored in row-major layout,
// i.e. A[i*n+j] is the element in i-th row and j-th column.
void MatrixSqr(int n, const double *A, double *R);
```

Пример использования этой функции (содержимое файла `main.c`):

```
#include <assert.h>
void MatrixSqr(int n, const double *A, double *R);
int main() {
    double A[3][3] = {{0,1,2},{2,0,0},{3,0,1}};
    double R[3][3];
    MatrixSqr(3, &A[0][0], &R[0][0]);
    assert(R[0][0] == 8 && R[0][1] == 0 && R[0][2] == 2);
    assert(R[1][0] == 0 && R[1][1] == 2 && R[1][2] == 4);
    assert(R[2][0] == 3 && R[2][1] == 3 && R[2][2] == 7);
    return 0;
}
```

В ходе проверки файл `main.c` будет подменяться на тестовый код жюри, чтобы проверить правильность работы вашей `MatrixSqr`. От вас требуется собрать из `main.c` и `matrixsqr.c` исполняемый файл `checked_solution`.

Часть 2. Напишите собственную простую реализацию функции `cblas_dgemm` с такой же сигнатурой, как у функции из OpenBLAS. Можно заменить enum-типы `CBLAS_LAYOUT` и `CBLAS_TRANSPOSE`, а также тип `MKL_INT`, на тип `int`. Учтите, что от вашей реализации **не** требуется поддерживать все многочисленные настройки, которые есть в настоящем BLAS. Достаточно, чтобы написанный вами код `MatrixSqr` работал с использованием вашей реализации `cblas_dgemm` так же правильно, как с использованием реализации из OpenBLAS.

Функцию следует реализовать в файле `myblas.c`.

При проверке SO будет собираться обычным способом:

```
gcc myblas.c -shared -fPIC -O2 -o myblas.so
```

Учтите, что эта библиотека будет лежать в рабочей директории, так что при сборке исполняемого файла нужно прописать текущую директорию в пути загрузчика.

На проверку требуется отправить:

- `matrixsqr.c` — содержит реализацию функции `MatrixSqr`.
- `myblas.c` — содержит собственную реализацию функции `cblas_dgemm`.
- `c.sh.txt` — скрипт сборки.

Во время сборки и запуска в текущей директории будут доступны:

- `main.c` — файл с тестовым кодом жюри, в котором есть функция `main`.
- Набор файлов библиотеки OpenBLAS (см. выше).

Проверка будет выполняться следующим образом:

1. Будет запущен ваш скрипт сборки (`c.sh.txt`), который должен собрать исполняемый файл `checked_solution`.
2. Будет запущена программа `checked_solution` на наборе тестов ($n \leqslant 1\,500$).
3. Будет собрана DL из `myblas.c`, которая будут переименована в файл `libopenblas.so`) с заменой последнего.
4. Будет снова запущена программа `checked_solution` на наборе тестов ($n \leqslant 700$).

Заметьте, что перед шагом 3 исполняемый файл **не** пересобирается: собранная программа должна отработать так же хорошо с подменённой DL. Правда к вашему решению гораздо слабее требования по скорости: всё-таки, реализация от OpenBLAS работает намного быстрее.