Задача 1. Библиотека линейной алгебры

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

Для выполнения вычислений над векторами и матрицами большого размера обычно используют библиотеку BLAS (Basic Linear Algebra Subprograms). Данная библиотека содержит несколько десятков функций, расфасованных по трём уровням согласно асимптотическому времени работы. Более сложные операции линейной алгебры (вроде решения систем уравнения) обычно реализуют с использованием функций BLAS. Существует множество реализаций библиотеки BLAS: проприетарные и open-source, простые и оптимизированные, для обычного процессора, видеокарты или суперкомпьютера, и так далее. Конечно, большая часть реализаций оптимизируется до предела, чтобы выжать максимальную производительность из конкретной аппаратуры.

В этой задаче предлагается написать простую реализацию некоторых функций BLAS. Сигнатуры функций настоящего BLAS очень громоздкие (например, функция перемножения матриц принимает 13 параметров), поэтому для данной задачи они несколько упрощены. Будем считать, что матрица размера m на n хранится в памяти в row-major формате без пропусков, то есть если указатель на начало матрицы равен `pMatr`, то элемент матрицы в i-ой строке в j-ом столбце лежит в `pMatr[i * n + j]`. Все векторы, матрицы и скаляры в данной задаче содержат вещественные числа типа `double`.

Библиотека должна реализовывать все функции из представленного на следующей странице хедера `myblas.h`.

На проверку требуется отправить три файла: `level1.c`, `level2.c` и `level3.c`. В этих файлах должны быть реализованы функции первого, второго и третьего уровня соответственно. Хедер `myblas.h` с указанным выше содержимым будет доступен в текущей директории.

Кроме того, вы должны научиться собирать из вашего кода статическую библиотеку. Для этого нужно дополнительно файл `c.sh.txt` — скрипт для сборки `myblas.a` компилятором GCC (расширение .txt появилось по историческим причинам, т.к. пересылка скриптов по e-mail может быть запрещена почтовыми сервисы).

Скрипт сборки должен содержать одну или несколько команд, которые нужно запускать в командной строке. Команды будут запускаться по очереди в той же директории, куда будет распакованы/сложены все ваши файлы. Разрешается вызывать программы: `gcc`, `ar` (путь к ним указывать **не** нужно).

Тестирующий код жюри будет корректно вызывать эти функции, все размеры будут положительными и не будут превышать 1 000. Кроме того, гарантируется, что при вызове любой функции все параметры-указатели будет указывать на неперекрывающиеся куски памяти. Общее количество вызовов и размеры будут достаточно малы, чтобы решение с правильной асимптотикой легко укладывалось по времени. Для вашего удоства при обнаружении ошибки в лог будет писаться причина ошибки: имя функции, и иногда даже входные данные.

Содержимое хедера `myblas.h`:

```
#pragma once

//====== уровень 1 ======
// (все векторы длины n)

//скопировать вектор из X в Y
void dcopy(int n, const double *X, double *Y);
//обменять местами содержимое векторов X и Y
void dswap(int n, double *X, double *Y);
//домножить вектор X на коэффициент alpha
void dscal(int n, double alpha, double *X);
//прибавить к вектору Y вектор X, умноженный на коэффициент alpha
void daxpy(int n, double alpha, const double *X, double *Y);
//вычислить скалярное произведение векторов X и Y
double ddot(int n, const double *X, const double *Y);

//====== уровень 2 ======

//вычислить вектор (alpha*A*X + beta*Y) длины m, и записать его в Y
//здесь A -- матрица размера m на n, X -- вектор длины n, а Y -- вектор длины m
void dgemv(
    int m, int n,
    double alpha, const double *A, const double *X,
    double beta, double *Y
);
//вычислить матрицу (alpha*X*Yt + A) и записать её в A
//здесь Yt -- это транспонированный вектор Y, то есть записанный как вектор-строка
// A -- матрица размера m на n, X -- вектор длины m, а Y -- вектор длины n
void dger(
    int m, int n,
    double alpha, const double *X, const double *Y,
    double *A
);

//====== уровень 3 ======

//вычислить матрицу (alpha*A*B + beta*C) и записать её в C
//здесь A -- матрица размера m на k, B -- матрица размера k на n,
// C -- матрица размера m на n
void dgemm(
    int m, int n, int k,
    double alpha, const double *A, const double *B,
    double beta, double *C
);
```