Задача 8. Переменное количество аргументов

Источник:	основная II
Имя входного файла:	--
Имя выходного файла:	--
Ограничение по времени:	3 секунды
Ограничение по памяти:	разумное

Внимание: эта задача проверяется на **emailtester**.

Цель данной задачи — научиться использовать и писать функции с переменным количеством аргументов. От вас требуется реализовать функции, объявленные в хедерах `logger.h` и `pack.h`, в исходных файлах `logger.c` и `pack.c` соответственно. Эти два файла и надо отправить на проверку.

Рекомендуется скачать и посмотреть архив с файлами `logger.h`, `pack.h` и `sample.c`.

Хедер `logger.h` объявляет три функции для записи сообщений в лог:

```
void logSetFile(FILE *file);
void logPrintf(const char *format, ...);
int getLogCallsCount();
```

Подробное описание этих функций содержится в полной версии хедера, которую можно скачать выше по ссылке. Текущий лог-файл и текущее количество успешных вызовов можно хранить в глобальных переменных. В функции `logPrintf` рекомендуется просто перенаправить вызов в соответствующую функцию из `stdio.h`.

Хедер `pack.h` объявляет функцию записи (сериализации) простых данных в байтовый буфер:

```
int pack(char *buffer, const char *format, ...);
```

Подробное описание этой функции содержится в полной версии хедера, которую можно скачать выше по ссылке.

При проверке будут добавлены хедеры и тестирующий код жюри.

Если есть вопросы по требуемому поведению функций в каких-либо случаях, задавайте вопросы.

Пример использования функций можно видеть в этом коде (файл `sample.c`):

```
#include "logger.h"
#include "pack.h"
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int main() {
    logPrintf("Not enabled logging yet\n"); //ignored
    logSetFile(stderr);
    logPrintf("Logging in stderr\n");       //goes to stderr
    logSetFile(stdout);
    logPrintf("Logging enabled!\n");        //goes to stdout

    int five = 5, ten = 10;
    double unit = 1.0;
    const char *hello = "hello";

    logPrintf(
        "Quering number of bytes for: [%d] [%lf] [%s] [%d]\n",
        five, unit, hello, ten
    );
    int bytes = pack(NULL, "%d%lf%s%d", five, unit, hello, ten);
    logPrintf("Allocating buffer of size %d\n", bytes);
    char *buffer = malloc(bytes);
    logPrintf("Packing data into buffer\n");
    int written = pack(buffer, "%d%lf%s%d", five, unit, hello, ten);

    logPrintf("Checking result\n");
    assert(written == bytes && written == 22);
    char correct[22] = {                    //note: assume little-endian
        0x05, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x3F,
        'h', 'e', 'l', 'l', 'o', 0,
        0x0A, 0x00, 0x00, 0x00
    };
    assert(memcmp(buffer, correct, sizeof(correct)) == 0);

    free(buffer);
    logSetFile(0);
    logPrintf("Sample finished\n");         //ignored

    logSetFile(stdout);
    logPrintf("Sample really finished\n");  //printed

    assert(getLogCallsCount() == 7);
    return 0;
}
```