Въпрос **1**

Отговорен

От максимално 12,00

Реализирайте дадените по-долу класове като спазите добрите практики на обектно-ориентирания стил.

За обработка на грешки използвайте изключения. Респективно, следете за хвърлени изключения там където е смислено и можете да направите нещо по въпроса.

Навсякъде, където работите с файлове (отваряне, писане/четене и т.н.) правете проверки за коректност.

Можете да използвате наготово класа `std::string` от `<string>` за работа със символни низове.

Реализирайте клас `Message`, който представя съобщение. Съобщението трябва да има следните атрибути:

- `type` -- вид на съобщението. То може да бъде информативно (info), предупреждение (warning), грешка (error), критична грешка (critical error). За представяне на вида използвайте изброим тип (`enum`).
- `description` -- текст на съобщението.

За класа `Message` дефинирайте подходяща версия на оператора за изход при работа с потоци (`<<`). Тази версия да извежда съобщението във формат:

```
<вид> <текст>
```

където `<вид>` е едно от `INFO`, `WARNING`, `ERROR` или `CRITICAL`, а `<текст>` е текстът на съобщението. Например:

```
CRITICAL: Cannot open input file
```

Обектите от тип `Message` трябва да са immutable, т.е. стойностите за техните атрибути се задават еднократно при създаването на обекта и след това не могат да се променят.

Реализирайте клас `Logger`. Този клас трябва да позволява на потребителя да записва (или да "log-ва") съобщения в текстов файл (т.нар. log file).

Класът трябва предефинира оператора `<<` така, че потребителят да може да записва с него съобщения. Например ако `m` и `n` са съобщения от тип `Message`, а `log` е обект от тип `Logger`, трябва да може да работи следният код:

```
log << m << n;
```

`Logger` обектите трябва да добавят информацията в края на log файла, с който са свързани. Натрупаната до момента информация във файла не трябва да се унищожава или презаписва. Всяко съобщение да се извежда на отделен ред.

Класът трябва да бъде така реализиран, че да може да брои по колко съобщения от всеки вид са били логнати през неговите обекти. Той трябва да може да предоставя тази инфорамация на потребителя. Например трябва да може да се провери колко грешки или колко предупреждения са били записани до момента.

Упътване: използвайте статични член-променливи и член-функции.

Реализирайте клас `Configuration`, който пази глобална конфигурация на приложение.

Класът трябва да бъде синглетон (singleton).

Класът трябва да има два атрибута:

- Файл, от който е била прочетена конфигурацията (символен низ)
- `Logger` обект

Конфигурацията на приложението ще се прочита от текстов файл (т.нар. "конфигурационен файл").

За улеснение, в рамките на задачата считаме, че конфигурационният файл е прост -- това е текстов файл, който съдържа в себе си само един символен низ -- път до файл, в който програмата да log-ва съобщенията си. По-долу са дадени примери за такива файлове:

sample-config.txt

```
c:\my data\log.txt
```

sample-config.txt

```
log.txt
```

Използвайте по-горните класове, за да напишете следната програма:

1. При стартирането си тя може да получи един аргумент от командния ред (т.е. през `argc`/`argv`), път до конфигурационен файл. Ако такъв не бъде указан, да се счита, че пътят до конфигурационния файл е `config.txt`.
2. Зарежда конфигурацията в обект от тип `Configuration`. Ако прочитането на информацията не е успешно, да се изведе съобщение за грешка и програмата да прекрати своето изпълнение.
3. Позволява на потребителя да въвежда колкото поиска съобщения и след това ги записва в `Logger` обекта на конфигурацията.

Програмата трябва да следи за хвърлени от обектите изключения. Ако възникне грешка, поради която не може да се продължи изпълнението и не може да се направи нищо по въпроса, програмата да изведе съобщение за потребителя и да прекрати своето изпълнение. Не трябва да оставяте хвърлени от вас изключения да напускат `main` и да предизвикват терминиране на програмата.