Въпрос **1**

Отговорен

От максимално 12,00

Контролно 2

При решаването на задачата можете да използвате наготово:

- `std::string` от `<string>`
- `std::vector` от `<vector>`

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

Всички операции с файлове да се извършват с помощта на потоците в C++. За улеснение даваме прототипите на някои от функциите за работа с файлове:

```
basic_istream& read(char_type* s, std::streamsize count)
basic_ostream& write(const char_type* s, std::streamsize count);
basic_istream& seekg(pos_type pos);
basic_istream& seekg(off_type off, std::ios_base::seekdir dir);
basic_ostream& seekp(pos_type pos);
basic_ostream& seekp(off_type off, std::ios_base::seekdir dir);
pos_type tellg();
pos_type tellp();
```

В тази задача трябва да напишете проста програма за работа с файлове в двоичен режим. При стартирането си програмата трябва да отвори подаден ѝ от потребителя файл и да изпълнява въведени от него команди.

Всяка команда се състои от една или повече думи, разделени помежду си с произволен брой празни (whitespace) символи. За улеснение ще считаме, че имената на командите и техните аргументи НЕ МОГАТ да съдържат whitespace символи.

Командите, които програмата трябва да поддържа са описани в края на условието на задачата.

Решете следните задачи:

А) Напишете клас `command`, който представя команда.

- Класът трябва да има подходящ конструктор, в който да получава символен низ -- текстът на командата.
- Вътрешно класът трябва да разбие низа на части -- името на командата и нейните аргументи.
- Класът да има функция `std::size_t size() const`, която връща броя на аргументите.
- Класът да предефинира `operator[]` така, че с него да могат да се извличат частите на командата. Операторът ще получава стойност от тип `std::size_t`. Тази стойност указва индекса на дума в командата (т.е. 0 - името на командата, 1 - първият ѝ аргумент и т.н.).

Б) Напишете клас `processor`, който изпълнява команди.

- Класът да има предикат `is_valid`, която получава обект от тип `command` и връща `true` или `false`, в зависимост дали командата е коректна или в нея има грешка.
- Класът да има функция `execute`, която получава команда и я изпълнява.
- Класът да не изпълнява командите директно, нито да работи директно с файла, а вместо това да работи с обект от класа `editor`.
- Класът трябва да прихваща възможни изключения хвърлени от `editor` и когато е нужно да извежда съобщение за грешка.

В) Напишете клас `editor`, който изпълнява операции с файлове.

- Класът да има функции `open`, `close`, с които да може да отваря и затваря файл. Ако отварянето на файл пропадне, да се хвърля изключение. Функциите на класа, чрез които се работи с файл, могат да работят само ако има успешно отворен файл.
- Отварянето на файл трябва да може да се направи и при създаване на обект от тип `editor`. За целта класът да има конструктор, който получава път до файл и го отваря.
- Файлът да се отваря в двоичен режим и да се държи отворен до извикване на close или до унищожаване на обекта от тип `editor`.
- При отварянето на файл, класът да намира размера му и да го запазва в `private` променлива от тип `std::size_t`. За намирането на размера да се използва техниката със `seek`/`tell`, която разглеждахме на лекциите.
- Класът да има функция `size`, която връща размера му като стойност от тип `std::size_t`.

- Класът да има функция `edit(std::size_t offset, std::uint8_t value)`. Тя записва стойността `value` на позиция `offset` спрямо началото на файла. Ако `offset` се намира след края на файла, функцията да не прави нищо, а да хвърля изключение от тип `std::invalid_argument`.
- Класът да има функция `display(std::ostream& out, std::size_t offset, std::size_t limit)`. Функцията извежда, на потока `<out>`, подобно на шестнадесетичен редактор, съдържанието на файла, започвайки от позиция `<offset>`. Извеждането да приключи или когато се изведат точно `<limit>` на брой байта, или се достигне краят на файла. Ако позицията `<offset>` се намира след края на файла, да не се извеждат нищо, а да се хвърли изключение от тип `std::invalid_argument`. За точния формат, в който да се извеждат данните, вижте описанието на командата `SHOW` по-долу.

Г) Свържете така написаните от вас класове в работеща програма.

Програмата ви трябва да получава от командния ред (`argv`/`argc`) път към файл. Тя трябва да отвори файла в двоичен (binary) режим, като за целта използва класа `editor`.

Ако отварянето не успее или пък потребителят не подаде нужния аргумент, да се изведе подходящо съобщение за грешка и да се прекрати изпълнението на програмата.

След това програмата ви трябва да влезе в цикъл, при който:

1. Въвежда се текстът на една команда от стандартния вход.
2. Използва се класът `command`, за да може командата да се разбие на части.
3. Използва се класът `processor`, за да се изпълни командата.

Сами преценете кой е най-подходящият начин да свържете класовете помежду им, кога и как да се създават техни обекти и т.н.

Командите, които програмата трябва да поддържа са описани по-долу. Всяка от тях, освен `EXIT` съответства на функционалност в класа `editor`.

`EXIT`
Затваря файла и излиза от програмата.

`SIZE`
Извежда на екрана размера на файла в брой байтове.

`EDIT <offset> <byte>`
Записва стойността `<byte>` на позиция `<offset>` във файла. Да се извежда текст "OK" или "Fail", в зависимост от това записът е бил успешен. Ако позицията `<offset>` се намира след края на файла, да се изведе съобщение, което уведомява потребителя колко е размерът на файла. Както `<byte>`, така и `<offset>` да се въвеждат като числа в десетичен запис.

`SHOW <offset> <limit>` Извежда на екрана, подобно на шестнадесетичен редактор, съдържанието на файла, започвайки от позиция `<offset>`. Извеждането да приключи или когато се изведат `<limit>` на брой байта, или се достигне края на файла. Ако позицията `<offset>` се намира след края на файла, да не се извеждат байтове, а вместо това да се изведе съобщение, което уведомява потребителя колко е размерът на файла. Както `<offset>`, така и `<limit>` да се въвеждат като числа в десетичен запис.

При извеждането да се използва познатия ви от шестнадесетичните редактори формат. Байтовете да се извеждат един след друг, разделени с интервали. Всеки байт да се изведе в шестнадесетичен запис – число с точно две цифри, при нужда, с водеща нула. След всеки 16 изведени байта, да се извежда нов ред. В началото на всеки от изведените редове да се извежда отстъпа на първия байт в него спрямо началото на файла. В края на задачата е показан нагледен пример за това как трябва да изглежда извеждането.

Упътване: за извеждане в шестнадесетичен режим, използвайте манипулатора `std::hex`. За запълване с водещи нули и точна широчина, използвайте `std::setfill` и `std::setw`. За да можете да работите с тях, включете заглавния файл `<iomanip>`. Например:

```
std::uint32_t data = 0x12345678;
std::cout << std::setfill('0') << std::setw(8) << std::hex << data;
std::uint8_t byte = 0xab;
std::cout << std::setfill('0') << std::setw(2) << std::hex << (unsigned int)byte;
```

По-долу е даден пример за това как би могла да работи програмата след като е била успешно отворена за някакъв примерен файл.

```
> SIZE
100 byte(s)

> SHOW 50 20
00000032 1a 77 65 fd 67 12 98 90 09 09 00 00 1a 12 8a 7d
00000042 23 67 aa 0f

> SHOW 96 20
00000060 56 78 1a bf

> EDIT 97 12
OK

> SHOW 96 20
00000060 56 12 1a bf

> EDIT 1000 20
ERROR: file size is 100.

> EXIT
```

82134-OOP-Assessment2.cpp

◄ Шаблон за второто контролно

Отиди на ...

Блиц тест 1 | структури и обединения ►