Soubor s verzemi

Termín odevzdání:	**17.04.2022 23:59:59**	1210101.507 sec
Pozdní odevzdání s penalizací:	**15.05.2022 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)	
Hodnocení:	**0.0000**	
Max. hodnocení:	**5.0000** (bez bonusů)	
Odevzdaná řešení:	0 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)	
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)	

Úkolem je navrhnout a realizovat třídu `CFile`, která bude simulovat binární soubor.

Požadovaná třída bude splňovat rozhraní podle ukázky níže. Požadavkem jsou čtecí a zápisové operace, nastavení pozice ukazatele a zkrácení souboru. Dále požadujeme, aby si třída dokázala pamatovat verze souboru. V rozhraní existuje metoda, která archivuje stávající obsah souboru. Následně se půjde k této verzi souboru vrátit. Takových verzí obsahu půjde vytvořit mnoho (implicitním limitem je pouze velikost dostupné paměti), k libovolné starší verzi se půjde vrátit (principem "undo"). Předpokládá se, že ukládaná data zapisujete do operační paměti (ne na disk, třída pouze simuluje chování souboru).

Z důvodů zálohování chceme mít možnost vytvářet kopie existujícího instance `CFile`. Kopie budou vznikat jednak kopírujícím konstruktorem a dále i operátorem =. Vzniklé kopie musí být identické nezávislé objekty, tedy operace s jedním z nich nemůže ovlivnit obsah druhého. Na druhou stranu se dá počítat s tím, že změn mezi kopiemi nebude mnoho, tedy některá data mohou kopie sdílet v zájmu šetření místa. Kopírováním vzniká identický objekt, tedy přenesou se i uložené verze obsahu.

Požadované rozhraní třídy:

konstruktor
: implicitní konstruktor vytvoří prázdnou instanci souboru (velikost 0 B, pozice v souboru 0).

destruktor, op= a kopírující konstruktor
: implementujte, pokud automaticky generovaná varianta nevyhovuje,

write (data, len)
: Metoda zapíše daný blok dat (`data`) o délce `len` na aktuální pozici. Aktuální pozice v souboru se po zápisu posune za poslední zapsaný bajt. Metoda `write` přepisuje data (je-li aktuální pozice uvnitř souboru)/rozšiřuje velikost souboru. Návratovou hodnotou je počet zapsaných bajtů.

read (data, len)
: Metoda načte požadovaný počet bajtů (`len`) do pole `data`. Návratovou hodnotou je počet načtených bajtů (může být menší než `len` podle aktuální pozice v souboru). Metoda dále posune aktuální pozici v souboru vpřed o přečtený počet bajtů.

seek (pos)
: metoda přesune aktuální pozici v souboru na pozici `pos`. Pozice se použije pro následné operace čtení/zápisu. Parametr `pos` musí být v rozsahu 0 až délka souboru (obě meze včetně). Návratovou hodnotou je `true` pro úspěch, `false` pro neúspěch (pozice mimo meze).

truncate()
: metoda zkrátí soubor na velikost danou aktuální pozicí v souboru.

fileSize()
: metoda vrátí aktuální velikost souboru v bajtech.

addVersion()
: metoda archivuje aktuální obsah souboru a aktuální pozici v souboru (vytvoří verzi). Tato verze bude uložena v instanci `CFile`.

undoVersion()
: metoda vrátí obsah souboru a aktuální pozici v souboru do stavu, ve kterém byly při odpovídajícím předchozím volání `addVersion`. Vracet se k předchozím verzím lze vícenásobně, dokud existují předchozí archivované verze. Volání `undoVersion` vrátí `true` pro úspěch, `false` pro neúspěch (neexistuje předchozí verze).

Odevzdávejte soubor, který obsahuje implementovanou třídu `CFile`. Třída musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main` (funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce níže).

Třída je testovaná v omezeném prostředí, kde je limitovaná dostupná paměť (dostačuje k uložení seznamu) a je omezena dobou běhu. Pro řešení zcela záměrně nemáte k dispozici STL ani `std::string`. Úloha má procvičit pochopení hluboké a mělké kopie. S využitím STL by tento cíl nebyl naplněn.

Požadované veřejné rozhraní třídy a ukázka použití je v přiloženém archivu.

Poznámky:

- Hodnocení je rozděleno mezi povinné, nepovinné a bonusové testy. Pro zvládnutí povinných testů stačí implementace základní verze kopírování obsahu.
- Pro zvládnutí dalších testů je potřeba efektivně sdílet části dat, aby vznikající kopie zabíraly rozumný objem paměti. Můžete předpokládat, že změny mezi verzemi a změny mezi kopiemi instancí jsou malé. Tedy neměnné části lze sdílet s využitím technik počítaných referencí a copy-on-write.

- V nepovinném testu jsou kopírované instance `CFile` a vytváří se verze obsahu, celkově je ale počet různých verzí obsahu nízký. Tedy stačí uvažovat počítané reference pro celé verze obsahu souborů.
- V bonusovém testu se soubory po zkopírování intenzivně mění a vytváří se mnoho různých verzí souboru. Pro úsporu paměti je tedy potřeba zapojit počítané reference a copy-on-write i pro verze nebo části souborů.
- Řešení této úlohy, které projde povinnými a nepovinnými testy na 100%, může být použito pro code review (pro code review není nutné projít bonusovým testem).

Vzorová data:		**Download**
Odevzdat:	Choose file	**Odevzdat**

☐ **Referenční řešení**