Poštovní server

Termín odevzdání:	**15.04.2018 23:59:59**
Pozdní odevzdání s penalizací:	**01.07.2018 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**3.5000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	15 / 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 realizovat třídu `CMailServer`, která bude simulovat činnost jednoduchého poštovního serveru.

Předpokládáme, že server zprostředkovává zasílání e-mailu uvnitř nějaké organizace. Každý e-mail je reprezentován instancí třídy `CMail`, obsahuje adresu odesílatele, adresu příjemce a tělo zprávy (vše jsou to řetězce). Mail server umí zprávu odeslat a umí zjišťovat obsah odeslané a přijaté pošty (outbox a inbox) pro jednotlivé e-mailové adresy. Seznam odeslané a přijaté pošty je poskytnut pro jednotlivé e-mailové adresy v podobě iterátoru, instance iterátoru nabízí metody, kterými se lze pohybovat v seznamu mailů a lze se dostat k obsahu jednotlivých mailů. Vlastní třída `CMailServer` musí správně zvládat kopírování, její kopie slouží pro archivační a auditní účely.

Úloha je směřována k procvičení pochopení mělké a hluboké kopie objektu. Má části povinné, dobrovolné a bonusové. Pro splnění povinné části úlohy postačuje vytvořit funkční třídu, která bude splňovat požadované rozhraní. Pro zvládnutí nepovinných a bonusových částí je potřeba rozmyslet ukládání dat tak, aby byl přístup k poštovním schránkám rychlý a aby se při kopírování zbytečně neplýtvalo pamětí.

Úloha má procvičit práci s kopírováním objektů. Z tohoto důvodu jsou v úloze potlačené části standardní C++ knihovny, zejména STL a datový typ C++ string. K dispozici je pouze rozhraní knihovny C řetězců (cstring). Pro implementaci může být vhodné si vytvořit vlastní jednoduchou náhradu `std::string`. Můžete se inspirovat řešením z proseminář.

Požadovaná třída `CMail` má následující rozhraní:

`konstruktor (from, to, body)`
: vytvoří instanci e-mailu se složkami from/to/body vyplněnými podle parametrů. Můžete předpokládat, že e-mailové adresy jsou relativně krátké (desítky až stovky znaků) a že tělo zprávy může být relativně dlouhé (i několik megabyte),

`operator ==`
: porovná obsah dvou instancí `CMail`, metoda vrací `true`, pokud jsou instance identické (shodují se všechny složky from, to i obsah e-mailu).

Požadovaná třída `CMailServer` má následující rozhraní:

`implicitní konstruktor`
: vytvoří prázdnou instanci,

`destruktor`
: uvolní prostředky alokované instancí,

`kopírující konstruktor, operator =`
: vytvoří identické kopie instance podle standardních pravidel,

`SendMail ( mail )`
: zašle e-mail předaný v parametrech, efektivně jej zařadí do odpovídajících schránek odesílatele a příjemce. E-mail je vždy zařazen na konec existujícího seznamu. Příjemce ani odesílatele není potřeba zakládat, schránka se automaticky vytvoří po zpracování prvního e-mailu, který obsahuje danou e-mailovou adresu,

`Outbox ( email )`
: zpřístupní poštu odeslanou ze zadané adresy. Návratovou hodnotou je instance `CMailIterator`, která umožní procházet emaily odeslané z adresy `email`. Pokud je zadaná neznámá e-mailová adresa, je výsledkem iterátor pro prázdný seznam e-mailů. Vrácený iterátor musí zachycovat stav mailové schránky v okamžiku, kdy byl vytvořen. Tedy pokud během používání iterátoru dojde ke změně obsahu procházené schránky, tato změna se do hodnot vracených iterátorem nijak nepromítne. Toto chování je demonstrované v ukázkovém běhu např. pro iterátor `i5`.

`Inbox ( email )`
: zpřístupní poštu přijatou na zadanou adresu. Jinak metoda pracuje stejně jako metoda `Outbox`.

emailové adresy
: v úloze mohou být libovolné řetězce, při jejich porovnávání rozlišujeme malá a velká písmena (case sensitive) - v tomto se úloha liší od reálného světa, kde e-mailové adresy mají předepsaný formální tvar a kde se malá a velká písmena zpravidla nerozlišují.

Požadovaná třída `CMailIterator` má následující rozhraní:

`operator bool`
: operátor zjistí, zda iterátor odkazuje na platný e-mail (vrací `true`), nebo zda dosáhl za poslední e-mail v seznamu (tedy e-mail už nelze číst, vrátí `false`),

`operator !`
: funguje stejně jako předešlý operátor, pouze vrací opačnou návratovou hodnotu

`operator *`
: unární operátor * zpřístupní e-mail na aktuální pozici. Návratovou hodnotou je instance `CMail` (případně konstantní reference na `CMail`). Nemusíte řešit situaci, že by se zpřístupnil e-mail za koncem seznamu - testovací prostředí vždy nejprve kontroluje platnost iterátoru a teprve pak případně zpřístupní odkazovaný e-mail.

`operator ++`
: prefixový operátor ++ zajistí přesun iterátoru na další e-mail v seznamu. E-maily jsou iterátorem procházené v pořadí, ve kterém byly odeslané/přijaté. Opakovaným voláním tohoto iterátoru se lze přesunout od prvního e-mailu přijatého/odeslaného zadanou e-mailovou adresou až k poslednímu (pak musí operátor přetypování na `bool` vracet `false`).

`kopírující konstruktor, operator =, destruktor`
: podle způsobu implementace možná nebude postačovat automaticky generovaná varianta. Testovací prostředí iterátory nikde explicitně nekopíruje, ale ke kopírování dochází v okamžiku předávání návratové hodnoty metodami `Inbox` a `Outbox`.

Odevzdávejte soubor, který obsahuje implementované třídy `CMailServer`, `CMailIterator`, `CMail` a další Vaše podpůrné třídy. Třídy 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.

Pokud se rozhodnete řešit i nepovinné a bonusové části úlohy, můžete pro nalezení vhodné reprezentace využít následující pozorování:

- je potřeba rychle vyhledávat podle zadaných adres. Typicky si pamatujeme tisíce různých adres, lineární vyhledávání tedy nemusí být dostatečně výkonné. Nové adresy sice zpočátku vznikají často, ale po určitém naplnění je vznik nové adresy spíše výjimečný.
- V prvním testu paměťové efektivity vzniká relativně málo kopií a mezi kopiemi je relativně dost změn (dá se předpokládat, že většina mailboxů je změněná).
- Druhý test paměťové efektivity vytváří velké množství kopií, kdy mezi kopiemi je provedeno pouze několik málo změn (dá se předpokládat, že většina mailboxů není změněná).
- Třetí test paměťové efektivity zkouší podobné scénáře, navíc předpokládáme, že zprávy zasílané v e-mailech se velmi často opakují (řetězové e-maily, hromadné e-maily).