Leszkovszki Nóra
Neptun-kód: AOM6DL

Naptár osztály

Specifikáció

Feladatleírás:

Készítsen naptár tárolására osztályt! Az osztály legyen képes tetszőleges naptári napot lefoglalni, és le lehessen kérdezni, hogy eddig milyen napok foglaltak, valamint egy adott nap foglalt-e vagy szabad (szabad, ha nem foglalt). Számítsa ki, hogy egy adott naptári nap milyen napra esik. Az osztály legyen képes két naptári nap összehasonlítására, eltelt napok számának kiszámítására! Legyen lehetőség táblázatos formában éves, és havi naptár nyomtatására! Valósítsa meg az összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor átdefiniálásához! Legyen az osztálynak iterátora is! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

Feladatspecifikáció:

A program eseményeket lesz képes rögzíteni naptárszerűen, melyek részleteit szövegfájlokból kapja meg. Ezen eseményekkel különböző műveleteket fog tudni végezni, melyek alább kerülnek részletezésre. Képes lesz a naptárt éves, illetve havi lebontásban szövegfájlba menteni, ami a felhasználó által akár kinyomtatásra is kerülhet.

Egy adott beolvasott eseményhez csak egy dátum tartozhat, azonban több eseményfájlban is szerepelhet ugyanaz a név, ha az azon nevű esemény több napon is megtörténik. (Tehát csak a dátumukban különböznek.)

A felhasználóval a program konzolon keresztül kommunikál, kérdéseket tesz fel, milyen műveletet szeretne végezni, majd a kapott válaszok alapján a kéréseket végrehajtja, és ugyanott, a szabványos kimeneten közli az eredményeket a felhasználóval, illetve ha a kérés magába foglalta, fájlba menti a naptárt nyomtatásra.

Egy esemény a következő attribútumokkal rendelkezik:

- név (tartalmazza a fájlja)
- dátum (tartalmazza a fájlja)
- milyen napra esik a hét napjai közül (a fájlja nem tartalmazza, a program számítja ki)

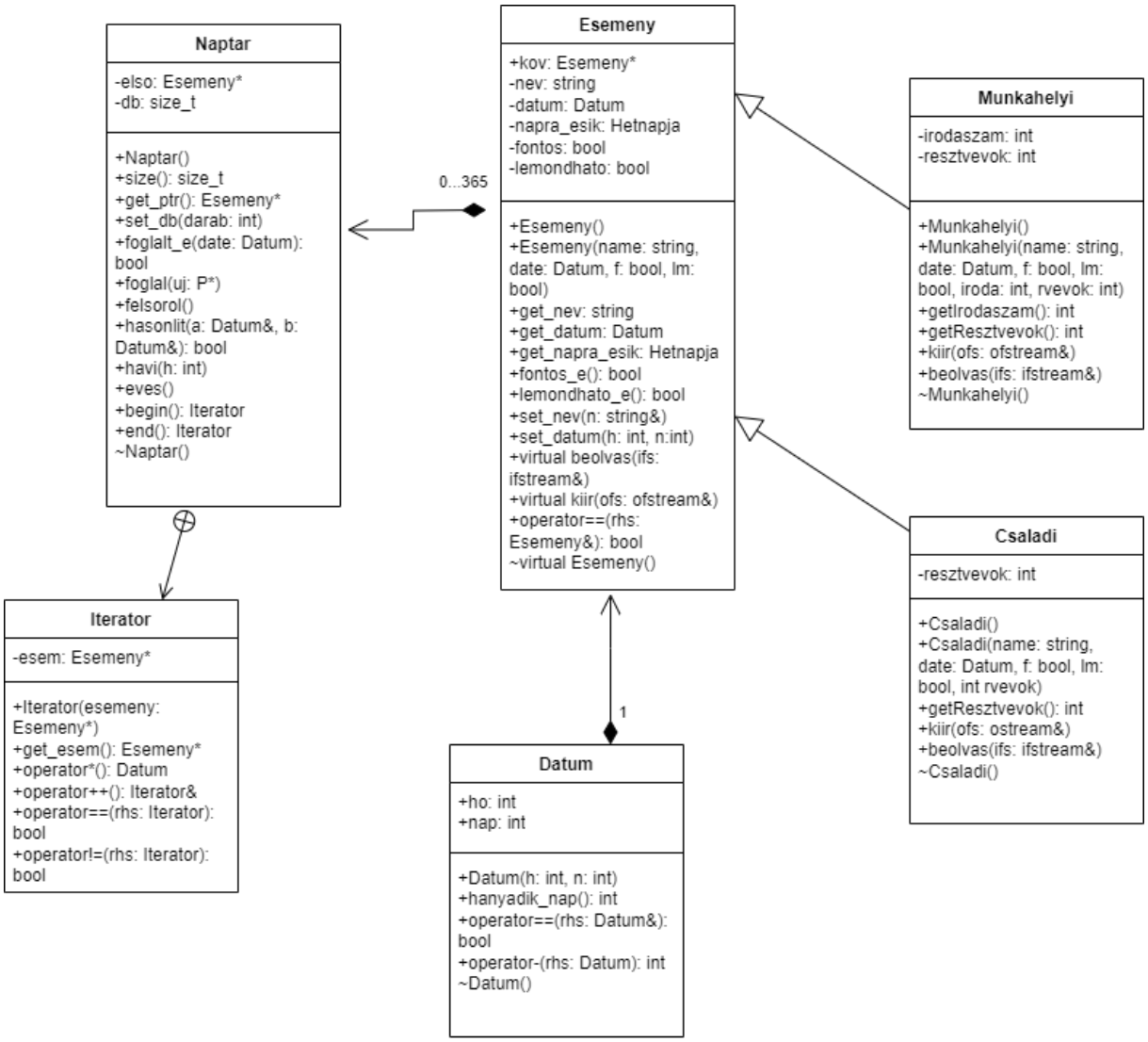
Naptárműveletek:

- *Nap lefoglalása:* a fájljából beolvasva az esemény részleteit, lefoglalja a naptárban az adott napot.
- *Eddig foglalt:* dátumfelsorolást kap a felhasználó azokról a napokról, melyek már le lettek foglalva.
- *Adott nap foglalt-e/szabad-e:* megnézi, egy adott napra volt-e már rögzítve esemény, ha igen, foglaltnak jelzi, ellenben szabadnak. Ebből adódóan egy napon csak legfeljebb egy esemény valósulhat meg.
- *Milyen napra esik:* megmondja egy adott beolvasott eseményről, hogy a hét melyik napjára esik a dátuma szerint.
- *Két nap különbsége:* két dátumról megállapítja, hány nap telik el közöttük.

- *Két nap összehasonlítása:* megállapítja két napról, hogy mindkettő szabad-e, illetve ha nem, ugyanazon megnevezésű esemény szerepel-e rajtuk.
- *Nyomtatás éves/havi lebontásban:* készít egy szöveges fájlt, melyben felsorolja táblázatos formában az egész év/a hónap eseményeit.

Terv

Az osztályok UML diagramja:



Tesztprogram vázlatos leírása:

A tesztprogram fájlokból olvas majd be eseményeket, amikkel elvégzi a különböző osztályműveleteket. Fájlba elmentheti majd a naptárt havi, illetve éves bontásban is. Az inputról beolvasva kapja majd a tesztprogram, melyik tesztesetet kívánja a felhasználó lefuttatni.

A különböző tesztesetek:

1. Esemény hozzáadása a naptárhoz
2. Két dátum között eltelt napok száma
3. Eddig lefoglalt dátumok felsorolása, adott dátum foglaltságának ellenőrzése
4. Két dátum összehasonlítása
5. Naptár mentése nyomtatásra (havi/éves)

Megjegyzés: minden teszteset (kivéve az 1.-t és 2.-at) legalább 3 esemény beolvasásával kezdődik, hogy a tesztet érdemlegesen végre lehessen hajtani.

A program jelentősebb függvényei

Teszt.cpp:

void teszt1(Naptar& naptar):

A kapott „naptar” objektumba felvesz 3 eseményt egy iterátorokkal vezérelt láncolt listába.

void teszt2():

Bekér 2 dátumot a felhasználótól, amikről kiírja a kimenetre, hány nap telik el közöttük.

void teszt3(Naptar& naptar):

Felsorolja a kapott „naptar” objektumban lévő események dátumát, időrendi sorrendben (ugyanis a láncolt listában dátum szerint növekvő sorrendben következnek).

void teszt4(Naptar& naptar):

Felvesz a „naptar” objektumba 3 eseményt, majd ezek után a felhasználótól kapott két dátumról megállapítja, ugyanolyan nevű eseményt tartalmaznak-e. Ha mindkettő nap üres, illetve ha rajtuk szereplő események azonosak névvel rendelkeznek, azonos programnak könyveli el őket a teszt, más esetben különbözőnek.

void teszt5(Naptar& naptar):

A teszt először felvesz 3 eseményt a „naptar”-ba. Ezután a felhasználó által választott formátumban kiírja szöveges fájlba a „naptar” eseményeit dátumuk szerint, illetve egyéb jellemzőiket.

Naptar osztály:

bool foglalt_e(Datum date):

Iterátoros vezérléssel végignézi a Naptár objektum eseményeinek listáját, majd ha a listában a kapott „date” dátummal megegyező dátumút talál, igaz választ ad vissza, vagyis foglaltnak jelzi a kapott dátumot, ellenkező esetben pedig hamis válasszal jelzi, hogy szabad még az a nap.

template <typename P>
void foglal(P* uj):

Ez egy sablonfüggvény, ugyanis a kapott eseményt típusa szerint teszi bele a naptárba. Elsőnek megnézi a kapott esemény dátumát, hogy az a nap nem foglalt-e már. Ha nem volt foglalt, iterátorok segítségével megkeresi sorrendhelyesen az új esemény helyét, és felveszi oda a listába. A végén növeli a tárolt események darabszámát.

void felsorol():

Iterátorok segítségével kiírja a naptárban eddig tárolt események dátumát (növekvő sorrendben).

void havi(int h):

Kiírja egy szöveges fájlba a kapott „h”-adik hónapban tárolt eseményeket dátumukkal es egyéb jellemzőikkel, mindezt iterátorok segítségével.

void eves():

Ugyanaz, mint a havi kiírás, de itt nem csak egy adott hónap, hanem az összes tartalma megjelenik a fájlban.

Iterator begin(), end():

A begin a naptar „elso” nevezetű esemény pointerére képez iterátort, az end pedig a lista vége utánra, ami egy nullpointert jelent.

Iterator osztály (a Naptar osztály belső osztálya):

Datum operator*():

A * operátor az adott iterátor eseményének a dátumát adja vissza, ugyanis a program eszerint rendezi a listát, így az operátornak ez a túlterhelése volt célszerű.

Iterator& operator++():

A következő eseményre lépteti az iterátor eseményét.

Esemeny osztály:

virtual void beolvas(ifstream& ifs):

A kapott ifstream-ről beolvassa az esemény attribútumait.

virtual void kiir(ofstream& ofs) const:

A kapott ofstream-re kiírja az esemény attribútumait.

Munkahelyi és Csaladi osztály:

void kiir(ofstream& ofs) const:

Ugyanúgy működik, mint az Esemeny osztályé.

void beolvas(ifstream& ifs):

Ugyanúgy működik, mint az Esemeny osztályé.

Datum osztály:

int hanyadik_nap():

Visszaadja, hogy az év hányadik napja az adott dátum.

int operator-(Datum rhs):

Visszaadja, hogy az adott dátum objektum és egy masik dátum között hány nap telik el. Ha az eredmény negatív, az annyit jelent, hogy az objektum dátuma korábbi, mint a paraméterként kapott dátum.