HÁZI FELADAT

Programozás alapjai 2.

Specifikáció

Nagy Ádám Balázs

CKTO0H

2025. március 30.

Tartalom

1.Feladatválasztás

Naptár osztály

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!

2.Feladatspecifikáció

2.1Program feladatai

A program célja, hogy egy bővíthető naptár osztály készüljön el. Feladati közé tartoznak a feladatválasztásnál felsoroltak, valamint, hogy különböző típusú események álljanak rendelkezésre a napok lefoglalásához. A naptárnak nincs határa időben, dinamikusan foglal területet az éveknek ha a távoli jövővel, vagy múlttal szeretne valaki műveleteket végezni. A naptárnak részei lesznek az évek, hónapok és a napok. A napok foglalását egy másik osztály végzi el. A naptár osztály, az implicit generálódó függvényeken kívül a következő műveleteket is megvalósítja :

- Létrehozás
- Megszüntetés
- Másolás
- Értékadás
- Más osztálynak üzenés

A felsorolt függvények konstans környezetben is értelmezhető változata is megvalósításra kerül.

2.2Program tesztelése

A felhasználó választhat, hogy melyik funkciót szeretné tesztelni. A feladat utáni számot kell beírnia a felhasználónak. Nap lefoglalás (1), mely napok foglaltak (2), foglalást törölni (3) naptári nap milyen napra esik (4), két nap összehasonlítása (5), két nap között eltelt idő száma (6), naptár kiírása a képernyőre (7). Az azután beírt adatt alapján fog lefutni a megfelelő teszteset. A dátum beírásakor az évek, hónapok és napok elválaszthatóak ponttal, valamint szóközzel. A hónapokat számokkal és betűkkel is ki lehet írni.

Olyan esetben amikor nem megfelelő hónapot vagy napot írnak be, azaz ha 1-nél kisebb, vagy 12-nél vagy az adott hónap napjainak számánál nagyobb, kivétel képződik, aminek elkapása a főprogramban történik. Abban az esetben ha egy foglalt napra szeretnénk új eseményt foglalni,

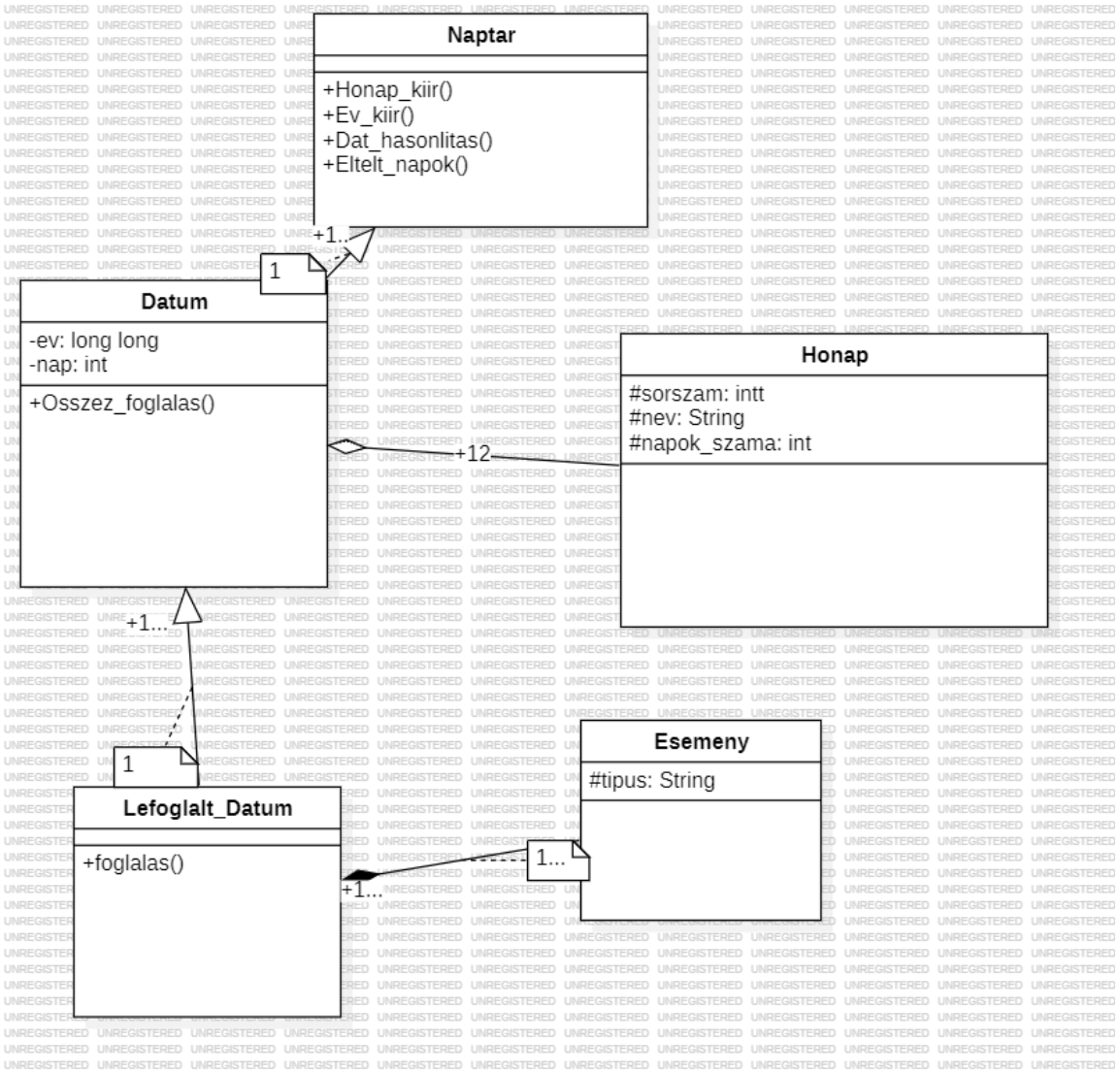
visszakérdez a program, hogy szeretné a felhasználó a korábbi foglalását törölni és újat tenni a helyére.

3.Terv

A feladat egy naptár osztály és egy tesztprogram létrehozását követeli. Ez más osztály segítségével lehet megvalósítani.

3.1.Objektum terv

A projekt főosztálya a naptár osztály. Ez dátumokból áll, a Datum osztály örökli a naptár tulajdonságait és viselkedéseit. A dátumok évek és napok mellett az adott hónapot is tartalmazza. A dátum és hónapok között aggregáció van, mivel a hónapok a dátumon kívül is létezhetnek. A Datum tárolja a foglalt napokat, amiket a Osszez_foglalas függvénnyel lehet kiíratni. A foglalást egy másik osztály végzi, a Lefoglalt_Datum. Közte és a Datum között öröklés áll fent. A lefoglalt dátumok különböző típusúak lehetnek, a típusok neve az Esemeny osztályban van tárolva, amely kompozicióban áll a Lefoglalt_Datum osztállyal.



3.2Algoritmusok

3.2.1Tesztprogram algoritmusai

A tesztprogram a specifikációnak megfelelően a felhasználó inputja alapján teszteli a program következő funkcióit. Azt követően az adatot beolvassa és annak a teszteset ciklusába lép be amelynek megfelel az adat.

3.2.2 Hónap lekérdezés

Ha hónap <0 akkor

Kivétel alulindexelés

Ha hónap>=12 akkor

Kivétel túlindexelés

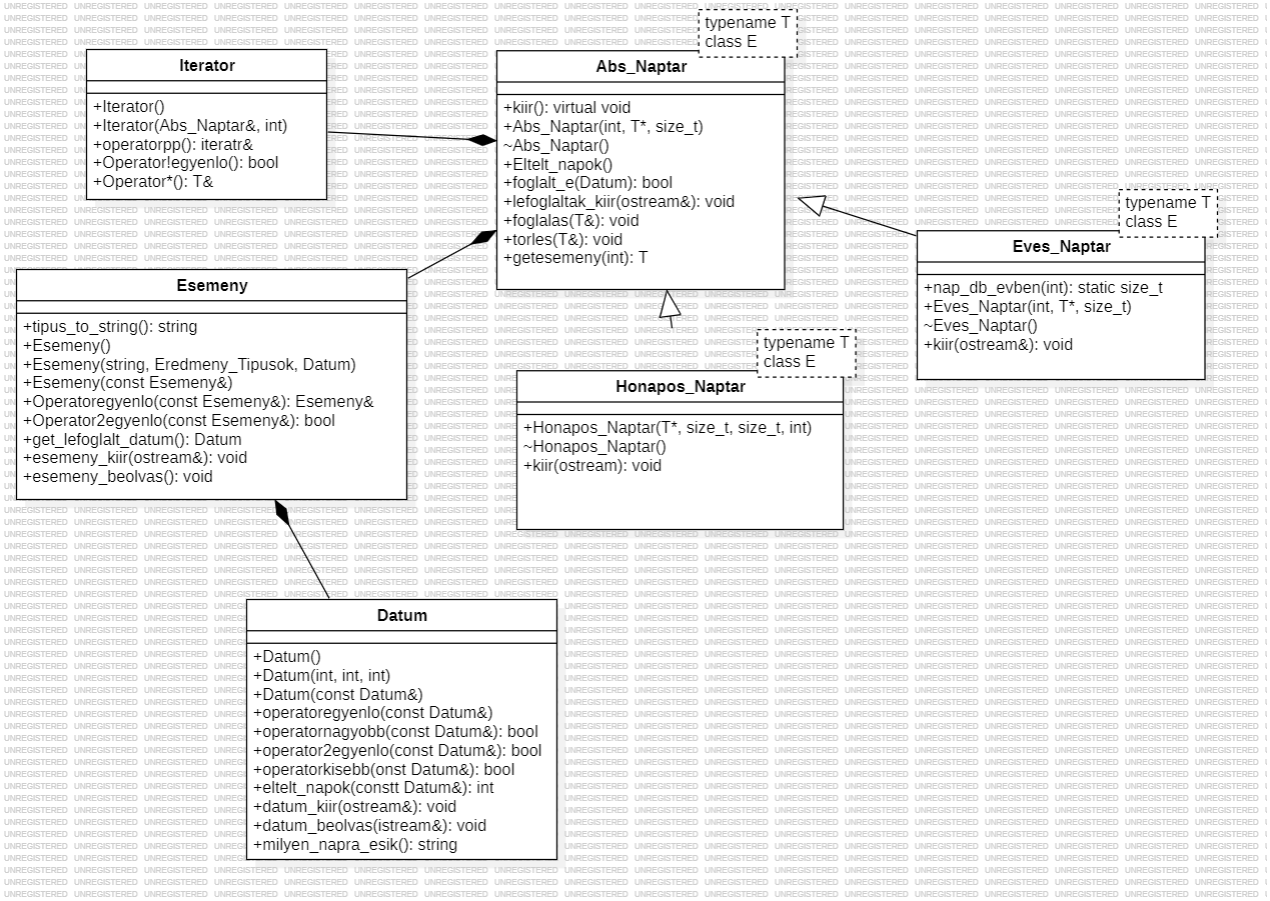
Különben return honapok[honap]

A kivételeket a főprogram kezeli.

3.b Javított terv

3.b.Objektum terv

A projekt alapját az absztrakt `Abs_Naptar` osztály képezi. Ennek tagváltozóit és tagfüggvényeit öröklik a `Honapos_Naptar` és `Eves_Naptar` osztályok. A kiíratás függvénye az alaposztályban virtuális, mivel az örökölt osztályokban eltérő módon kerül megvalósításra.
A T egy típusként (template paraméterként) szerepel, amely osztály típusú (például `Esemeny` vagy `Datum`), és az elemek egy dinamikusan lefoglalt tömbben tárolódnak.
Az E típus a kivételkezeléshez használt osztályt jelöli.
Mivel minden eseménynek van egy dátuma, az `Esemeny` és `Datum` osztályok között kompozíció (összetétel) kapcsolat áll fenn.



4.Megvalósítás

4.1Programzói dokumentáció

A feladat megvalósítása nagy részben megegyezik a tervben bemutatottal. Az osztályok tagfüggvényeinek működése az üzleti modellben tesztelésre kerül. Ennek megvalósítása érdekében a tárgyban megismert "gtest_lite.h" állomány függvényeit használom fel.

Az IO modell megvalósítja a fájlból olvasás valamint fájlba írás műveletek, így az írásra, valamint olvasásra szánt tagfüggvények is bemutatásra kerülnek. A standard bemeneten a felhasználó különböző funkciók közül választhat: eseményt foglalhat, az eddig foglalt eseményeket megtekintheti, eseményt törölhet és naptárakat is megtekinthet, amelyben a lefoglalt napok jelölve vannak, ezen napokon lefoglalt eseményeket is lehetősége van látni.

4.1.1 Az elkészített osztályok bemutatása

A tervtől eltérően a naptár osztályok nem sablonnal működnek, dátumokat magukban nem lehet lefoglalni, mert azzal olyan napok is le lennének foglalva amikor semmi fontos nincs kikötve, hogy történne. A foglalás az Abs_Naptar osztályban történik, és a lefoglalt események növekvő sorrendben vannak tárolva dátum szerint egy dinamikus tömbben. Ehhez a Datum osztáy egyenlőtlenségi operátorai vannak felhasználva. A lefoglalás mellett ugyanebben az osztályban valósul meg egy lefoglalt eseménynek törlése valamint az osztály destruktora által felszabadítva van a lefoglalt memória. A dinamikus tömb bejárásához készült a const_iterator osztály amelyet tartalmaz az Abs_Naptar osztály. Ez fel van használva az események kiiratásánál valamint az ellenőrzésben, hogy egy dátum foglalt-e vagy sem. A void kiir és size_t capacity függvények virtuálisak, más történik a két örökölt osztály megvalósításakor. Az éves naptár kiírásához fel vannak használva az év hónapjainak hónapos naptárként való kirásai.

4.1.2 Elkészített függvények, algoritmusok

A dátum osztályban megvalósított int eltelt_napok függvény két dátum közötti különbséget számolja ki napokban. Ez többféleképpen történhet, ha a két dátum csak a nap terén különbözik egyszerűen a nagyobb dátum napjából ki kell vonni a kisebbet. Ha hónapokban külömböznek, a függvény kivonja a kisebb hónap összes napjaiból a dátum napjait, így megkapva a hónap elkövetkezendő napjait. Ehhez még hozzáadódik a köztes hónapok összes napjainak száma valamint a nagyobb dátumban a hónapban eltelt napok száma. Az évek különbözése esetén is hasonlóan számolható ki a különbség.

Az Abs_Naptar foglalás függvényében a foglalás azon elven valósul meg, hogy az események dátum szerint növekvő sorrendben legyenek eltárolva. Ennek érdekében a függvény addig másolja az új tömbbe az eredeti tömb elemeit amíg nem talál olyan elemet amelyik nagyobb a foglalandó esemény dátumánál. Ebben az esetben megjegyzi ezt a poziciót, a ciklusból kilép és az új esemény bekerül erre a pozícióra. Ezután az új tömbbe mindig az eredeti eggyel kisebb indexű elemek kerülnek be.

A hónapos naptár kiiratásában nagy szerepet játszik a het_kezdo nap függvény ami alapján lehet tudni, hogy milyen a hónap kezdése milyen napra esik, és ez figyelembe van véve a naptár elrendezésében.

A függvények nagy részének működéséről szó esik a függvények felett lévő kommentekben.

4.1.3Felhasznált header fileok, stl tárolók,állományok

A projekt megvalósításához fel voltak használva különböző header fileok. Az <iomanip> a dátumok valamint a naptárak rendezett kiiratásához voltak szükségesek. Ez könnyebben olvashatóvá tette a naptárakat. Stl tárolókból a <string> osztály használata volt megengedett, mást nem használtam. Ebben tároltam a különböző szöveges változókat. Az <fstream> a fájlok kezeléséhez volt szükséges, amelyből van olvasásra, írásra és mindkettőre alkalmas fájl. A <locale> a magyar betűket megfelelően kezelő nyelv beállításához volt szükséges. Az <stdexcept> általak kerülnek a kivételek az outputra. A memóriaszivárgás ellenőrzéséhez a “memtrace.h” állomány van felhasználva, a teszteléshez pedig a “gtest_lite.h”.

4.1.4 Tesztprogram bemutatása

A program a tesztprogramot futtatja ha definiálva van a CPORTA makró. A tesztprogram a dátum osztály tagfüggvényeit teszteli először, majd ezután az őt tartalmazó Esemény osztály funkcióit, ezután az Abs_Naptarban megvalósított függvényeket, végül pedig az ebből az osztályból öröklődő hónapos és éves naptár függvényeit. Különböző eseteket is vizsgál, azt is ellenőrzi, hogy hibás adat esetén érkezett-e kivétel. A dátumok és az események egyenlőségének vizsgálata egyben magába vonja azt is, hogy ezeknek az osztályoknak == operátorai helyesen meg lettek valósítva. Ez igaz a EXPECT_GT, EXPECT_LE... ellenőrzésekre is amiket a dátum osztály egyenlőtlenségi operátorai biztosítanak.

A tesztprogram továbbá ellenőrzi, hogy egy esemény valóban bekerült-e a dinamikus tömbbe foglalás után, valamint, hogy az esemény már nem található meg a tömbben törlés esetén.

4.1.5Fájlkezelés, IO modell

A Beolvaso_Teszt.Txt–ben szerepel egy dátum, egy hónap , egy év, egy n ami azt mutatja hány eseményt szeretnénk lefoglalni és ezután n esemény. Ezek az adatok be vannak olvasva a fájlból majd a Kiiro_Teszt.txt-be beírjuk a kapott dátumot és a lefoglalt eseményeket. Az inputban szereplő hónapnak és évnek megfelelő Naptárakat kiiratjuk, feltüntetve rajtuk a lefoglalt események helyét. Ezzel egyben megmutatjuk a fájlkezelés, valamint az osztályok beolvasó és kiíró tagfüggvényeinek működését.

A foglalt_esemenyek.txt biztosítja, hogy a program lefoglalt eseményei ne vesszenek el. Ezt a fájl olvasásra és írásra is használandó, ezért fstream-ként van deklarálva. Ha a lefoglalt események kiírására szükség van a naptár innen olvassa be az eseményeket, majd ide írja az újonnan foglalt. Törlés esetén töröljük az eseményt a dinamikus tömbből, majd annak tartalmát beirjuk a fájlba, így a fájlban se lesz már jelen a törölt esemény.

A program exceptionbol származó saját Gond_Van kivételosztállyal kezeli a nem szabványos hibákat, ezekhez üzeneteket is társít amit hiba elkapás esetén a kimentre kiír. Az , hogy a Gond_Van kivételeket előbb keresi a program mint más exceptionoket biztosítja, hogy a saját kivételeket ne az általános exception kapja el. Vannak a saját kivételosztályon kívül is kivételek, ezeket elkapja az általános exception, ha pedig egyikre se igaz az azt jelenti, hogy ismertelen a hiba ezért “Nagy baj van.” íródik ki a kimenetre.

4.2 Felhasználói dokumentáció

A program felhasználójának futtatás esetén tud választani, hogy milyen funkcióját a programnak szeretné látni. Ezen funkciók eléréséhez a neki megfelelő kifejezés zárójelében szereplő nagy betűjét kell beírja (pl. törlés-T, esemény foglalás- F, kilépés-K). Hibás betű beírása esetén újrapróbálkozhat a felhasználó. Ha foglalást vagy törlést választ a felhasználó be kell írnia az illető eseményt. Ez először az esemény nevének beírásával történik, majd egy tipust kell beírni. Ez lehet a következők közül: iskolai, globális, magánéleti, munkahelyi vagy egyéb. Ha nem ezek közöl választ a felhasználó a típus szintén egyéb lesz. Utolsónak az esemény dátumát szükséges begépelni. Ez is három lépésben történik. Először az év , majd a hónap és végül az év kerül beolvasásra. Az újévet nem lehet törölni

az események közül, mivel ez minden naptárban le van foglalva első hívás után.

Naptár kiiratás esetén azt kell eldöntenie a felhasználónak, hogy hónapos vagy éves naptárt szeretne kiiratni. A hónapos naptár egy hónapot tartalmaz, az éves pedig értelemszerűen egy évet. Éves naptár esetén a felhasználónak be kell írnia a kívánt évet, hónapos esetén pedig először a hónaphoz tartozó évet, majd magát a hónapot. A lefoglalt események *nap* formában jelennek meg a naptárban. Lehetőség van ezután egy ilyen dátumhoz tartozó eseményt megtekinteni. Ha olyan dátumot ír be a felhasználó ami nincs lefoglalva kivétel keletkezik.

Ha a felhasználó az összes lefoglalt esemény kiírását választja, megjelenik a lefoglalt események száma, ezután pedig a lefoglalt dátumok időrendi sorrendben. Ha a felhasználó ki szeretne lépni, egy funkció beteljesülése után R betűvel végezheti ezt el.

4.3Javított programozói dokumentáció

4.3.1Az elkészített osztályok bemutatása

Datum osztály

Ez az osztály a következőket valósítsa meg:
-Dátumok összehasonlítása
-Dátumok kiíratása, beolvasása
-Dátumok közt eltelt napok kiszámítása
-Egy év ellenőrzése, hogy szökőév-e
-Napok száma egy bizonyos évben, valamint egy bizonyos hónapban.

Esemeny osztály

-Események azonosságának és különbözésének ellenőrzése
-Esemény beolvasása, kiiratása
-Esemény típusának átalakítása stringgé és fordítva.

Abs_Naptar osztály

-Egy esemény lefoglalása
-Egy esemény törlése a foglat eseményekből

-Dátum ellenőrzése, hogy foglalt-e
-Foglalt események kiiratása
-Fájlban található események beolvasása és lefoglalása
-Destruktor, amely felszabadítja a lefoglalt dinamikus területet

Eves_Naptar osztály

-Az adott évben lévő napok száma
-Éves naptár kiíratása

Honapos_Naptar osztály

-Napok száma egy adott hónapban
-Hónapos naptár kiiratása
-Annak kiszámítása, hogy a hónap első napa milyen heti napra esik

4.3.2Az elkészített tagfüggvények bemutatása

Datum osztály

Datum()
-Paraméter nélküli konstruktor. Beállítja a dátumot az idei év első napjára.

Datum(int year, int month, int day)
-Paraméteres konstruktor .
-Ha nem megfelelő a hónap vagy nap paramétere std::out_of_range hibaüzenetet dob.

Datum(const Datum& date)
-Másoló konstruktor.

Datum& operator=(const Datum& date)
-Értékadó operátor.

int getnap() const
-A naptár kiíratásnál szükség van a napra.

bool operator>(const Datum& hasonlitando_date) const
-Két dátum összehasonlítása. Akkor tér vissza igaz értékkel a függvény ha az első dátum a nagyobb.

bool operator==(const Datum& hasonlitando_date) const
-Két dátum közötti egyenlőségnek vizsgálata. Igaz értékkel tér vissza, ha a dátumok egyenlőek, hamissal más esetben.

bool operator<(const Datum& hasonlitando_date) const
-Két dátum közötti eltérésnek vizsgálata.
-Kisebb reláció vizsgálata dátumok között.
- Megvalósításához a korábban megvalósított az "operator>" és "operator=" tagfüggvények is fel vannak használva.

bool operator<=(const Datum& hasonlitando_date) const
-Kisebb vagy egyenlőség vizsgálata dátumok között.
-A kisebb operátor és egyenlőség felhasználásával, ha egyik igaz, igaz értékkel tér vissza, más esetben hamis értékkel.

bool operator>=(const Datum& hasonlitando_date) const
-Nagyobb vagy egyenlőség vizsgálata dátumok között.
-A nagyobb operátor és egyenlőség felhasználásával, ha egyik igaz, igaz értékkel tér vissza, más esetben hamis értékkel.

bool operator!=(const Datum& hasonlitando_date) const
-Két dátum közötti eltérésnek vizsgálata. Az "==" operátor van felhasználva hozzá

static bool szokoev_e(int jelenlegi_ev)
-Vizsgálja, hogy az adott év szökőév-e vagy sem.
- True értékkel tér vissza, ha a dátum éve szökőévre esik, false értékkel más esetben.
-@param jelenlegi_ev- A vizsgálandó év
-statikus, hogy más osztályban is lehetséges legyen dátum objektum nélkül használni.

static int ev_napdb(int jelenlegi_ev)
-Azt vizsgálja egy évben hány nap van.
- A szokoev_e függvényt használja fel, 366-ot térit vissza szökőév esetén, 365-öt ha nincs szökőév.
-@param jelenlegi_ev- A vizsgálandó év

-statikus, hogy más osztályban is lehetséges legyen dátum objektum nélkül használni.

static int honap_napdb(int jelenlegi_honap, int jelenlegi_ev)
-Megkeresi a dátumban hány napból áll az adott hónap. A szökőéveket is számitásba veszi
-A hónapban lévő napok számával tér vissza, egész értékkel
-@param jelenlegi_honap- A vizsgálandó hónap
-@param jelenlegi_ev- A hónaphoz tartozó év amelyről ellenőrizni kell februári hónap esetén, hogy szökőévre esik.
-statikus, hogy más osztályban is lehetséges legyen dátum objektum nélkül használni

int eltelt_napok(const Datum& date2) const
-Két dátum között eltelt nap kiszámítása.
-Az eltelt napok számával tér vissza.

void datum_kiir(std::ostream& os=std::cout) const
-Kiírja a paraméterként kapott dátumot a másik paraméterként megadott kimenetre
-@param os- kimenet

void datum_kiir_beolvasashoz(std::ostream& os=std::cout) const
-A foglalt_esemenyek fileba csak az adatokat írjuk be a könnyebb olvasás érdekében.

void datum_beolvas(std::istream& is = std::cin,std::ostream& os=std::cout
-Dátum beolvasása
-@param is- bemenet

std::string milyen_napra_esik() const
-A függvény azt számolja ki, hogy egy naptári nap milyen napra esik.
-A het_napjai[7] tartalmazza a hét napjainak neveit.
-A függvény felhasználja az eltelt napokat, 2025.04.21-hez képest .
-Az így kapott számnak 7-el való osztási maradéka egy i számot ad meg.
-A het_napjai[i] téríti vissza a megfelelő napot.

inline std::ostream& operator<<(std::ostream& os, const Datum& datum1)
-Dátum kiíratása operator<< segítségével.
-@param os-kimenet, @param datum1- kiírandó dátum.

Esemeny osztály

Esemeny() :megnevezes("újév"), tipus(globalis), idopont(2025, 1, 1)
-Default konstruktor. Az új évet állítja be, 2025.01.01 dátummal, ez egy globális esemény.

Esemeny(const std::string& nev, Esemeny_Tipusok milyen, const Datum& mikor) :megnevezes(nev), tipus(milyen), idopont(mikor)
-Paraméteres konstruktor.

Esemeny(const Esemeny& event)
-Másoló konstruktor

Esemeny& operator=(const Esemeny& event)
-Értékadó operátor
-A Naptár osztályban szükség van a dinamikusan foglalt események eltárolása érdekében.

bool operator==(const Esemeny& hasonlitando_date) const
-Két esemény összehasonlítása.
-Igazzal tér vissza, ha az esemény minden tagváltozója megegyezik, hamissal ellenkező esetben.

bool operator!=(const Esemeny& hasonlitando_date) const
-Két esemény összehasonlítása.
-Az operator== függvényt használja fel a megvalósítás.

std::string tipus_to_string() const
- Az esemény típúsát sztringgé alakítja át. A kiíratáshoz szükséges lépés ez

static Esemeny_Tipusok string_to_tipus(const std::string& tipus_nev)
-Az esemény típus neveit esemény típussá alakítja át. A beolvasáshoz szükséges lépés ez.

const Datum& get_lefoglalt_datum() const
- A naptárban szükség van rá
-A lefoglalt dátum lekérdezése. Erre szükség van, hogy lehessen tudni mely napok vannak lefoglalva a naptárban.
-Az esemény időpontjával tér vissza a függvény.

void esemeny_kiir(std::ostream& os = std::cout) const
-Az esemény kiíratása.
-@param os- kimenet

void esemeny_kiir_beolvasashoz(std::ostream& os=std::cout) const
-A foglalt_esemenyek fileba csak az adatokat írjuk be a könnyebb olvasás érdekében.

void esemeny_beolvas(std::istream& is = std::cin, std::ostream& os = std::cout)
-Az esemény beolvasása.
-@param is- kimenet

inline std::ostream& operator<<(std::ostream& os, const Esemeny& esemeny1)
-Esemény kiíratása operator<< segítségével.
-@param os-kimenet, @param datum1- kiírandó dátum.

Abs_Naptar osztály

Abs_Naptar()
-Az osztály default konstruktora.

~Abs_Naptar()
-Destruktor. A dinamikusan foglalt adatok felszabadításáért felelős.

virtual int capacity() const = 0
-Az összes nap számával tér vissza. Az a szerepe, hogy ez publikusan is látható legyen.
- Virtuális, mivel a napok száma függ a naptár típusától.

virtual void kiir(std::ostream& os = std::cout) const = 0
-A naptár kiíratása. Virtuális mivel más formában írandó ki az éves és a hónapos naptár.
-param os- kimenet

bool foglalt_e(const Datum& vizsgalando_date) const
-Egy dátum ellenőrzése, hogy lefoglalt-e vagy sem.
-A függvény true logikai érékkel tér vissza, ha a dátum már más esemény által foglalt, false értékkel ellenkező esetben.
-@param vizsgalando_date- A dátum amiről szeretnénk megtudni, hogy foglalt-e.

template<class E>
const Esemeny& Datum_Esemenye(const Datum& vizsgalando_date) const
-Egy esemény visszaadása annak dátuma szerint
-A függvény megkeresi azt az eseményt amelynek dátuma megegyezik a paraméterként adottal.
-Abban az esetben ha a paraméterként adott dátumon nincs még lefoglalt esemény saját kivételt dob az E osztályba.
-@param vizsgalando_date- A dátum amire foglalt eseményt meg szeretnénk keresni.

void lefoglaltak_kiir(std::ostream& os=std::cout) const
-A lefoglalt események kiíratása.
- Az események a dátumok időrendi sorrendjében kerülnek kiíratásra
- Az Esemény osztály kiíró függvénye felhasználásra kerül.
-@param os- kimenet

void lefoglaltak_kiir_olvasashoz(std::ostream& os=std::cout) const
//Törlés estén a foglalt_esemenyek fileba csak az adatokat írjuk be a könnyebb olvasás érdekében.

template<class E>
void foglalas(const Esemeny& uj_esemeny)
-Új esemény hozzáadása a lefoglalt események listájához
- Abban az esetben ha ez a dátum már le van foglalva saját kivételt dob.

Az új eseményt úgy szúrja be, hogy az események dátum szerint növekvő sorrendben legyenek.
-Ha az új esemény minden esemény dátumánál nagyobb, akkor a lista végére kerül.
-Más esetben az új esemény be van szúrva két dátum közé majd a további események az eggyel kisebb indexű eseményt kapják meg.
-@param uj_esemeny- A lefoglalt események listájához hozzáadandó esemény.

template<class E>
void torles(Esemeny& torlendo_esemeny)
-Esemény törlése a lefoglalt események listájából
-Ha nem található meg a lefoglalt események között a paraméterként megadott esemény saját kivételt dob.
-Miután megtalálta a függvény a törlendő eseményt, addig a pozícióig az eseményeket lefoglalja.
-A törölt pozíciótól az események elcsusztatása által megkapjuk az új listát.
-A függvénye összehasonlítja a listában lévő eseményeket és egyezés esetén törli

const_iterator begin()
-Iterátor lézrehozása és az első elemre állítása

const_iterator end() const
-Iterátor létrehozása és a az utolsó elem utánra állitása

const_iterator() :p_aktualis(nullptr), p_utolso(nullptr)
-Default konstruktor

const_iterator(const Abs_Naptar& naptar, size_t idx = 0)
-Paraméteres konstruktor

const_iterator& operator++()
-Iterátor preinkremens növelése

const_iterator operator++(int)
-Iterátor posztinkremens növelése

bool operator!=(const const_iterator& masik) const
//Iterátorok összehasonlítása

const Esemeny& operator*() const
-Iterátor indirekciója
-Ha az iterátor a tár végén van std::out of range hibaüzenetet küld.

template<class E>
const Esemeny& getesemeny(size_t i)
-A lista egy elemének visszatérítése sorszáma alapján
-Ha nem megfelelő az index, kivételt dob az E osztályba.

template<class E>
void foglalas_olvasas_file(std::istream& is)
-Az fs-ben szereplő lefoglalt napokat beolvassa és lefoglalja.
-Így valósítható meg, hogy a régebben lefoglalt napok megmaradjanak.

Honapos_Naptar osztály

Honapos_Naptar(int sorszam=1, int ev=2025)
-Az Abs_Naptar által örökölt tagváltozók paraméterekkel, a Honapos_Naptar saját tagváltozói default megvalósított konstruktor.

int capacity() const
-Az összes nap számával tér vissza. Az a szerepe, hogy ez publikusan is látható legyen.

int het_kezdo_nap(const Datum& date1) const
-Megadja azt a napot ahonnan kezdődik az adott hónap
-@param date1- A hónap első napjának dátuma.

void honap_het_kiir(std::ostream& os) const
-A hónap és a hét napjai kiiratása.

void nap_kiir(std::ostream& os, Datum& date1) const
- A napok kiiratása.
-Azok a napok amik le vannak foglalva valamilyen esemény által "*"*nap*"*" jelölést kapnak.

void kiir(std::ostream& os = std::cout) const
-A hónapos naptár kiíratása.
-param os- kimenet

Eves_Naptar osztály

Eves_Naptar(int ev=2025)
-Paraméteres konstruktor. A napok száma az év alapján számítható ki.

int capacity() const
-Az összes nap számával tér vissza. Az a szerepe, hogy ez publikusan is látható legyen.

void kiir(std::ostream& os=std::cout) const
-Az éves naptár kiiratása.
-param os- kimenet

4.3.3 Tesztprogram bemutatása

A program a tesztprogramot futtatja ha definiálva van a CPORTA makró. A következő tesztelések történnek meg:

TEST(Datum,Konstruktorok)
-Dátum konstruktorainak tesztelése.
-Kivételeket is ellenőrzi.
-Az EXPECT_EQ által az "==" operátor is ellenőrizve van. Az EXPECT_NE által pedig a "!=" is.

TEST(Datum,Egyenlotlensegek)
-Dátum reláció jeles operátorok ellenőrzései
-Első elem az a paraméter kell legyen amire igaz az állítás.

TEST(Datum,Napokszama)
-Szökőévvel, hónapok és évek napjainak számával kapcsolatos függvények ellenőrzése.

TEST(Datum,Elteltnap)
-Első négy teszt az eltelt napok számának ellenőrzését biztosítja.
-A következők pedig azt, hogy milyen napra esik egy dátum.

TEST(Esemeny,Konstruktorok)
Esemény konstruktorainak és "==","!=" operátorainak működésének ellenőrzése

TEST(Esemeny,DatumTipus)
-Az esemény osztály Dátum getter függvényének és a sztringet típussá átalakító függvény ellenőrzése.

TEST(Abs_Naptar,Foglalas_Kivetelek)
-Az Abs_Naptar származtatott osztályainak konstruktorainak ellenőrzése .
-Kivétel dobás ellenőrzése.

TEST(Abs_Naptar,Foglalas)
-A foglalás helyes működése.
Lefoglal egy eseményt majd ellenőrzi, hogy az jó helyre került a tömbben

TEST(Abs_Naptar,Torles)
-A törlés funkció helyes működésének tesztelése
-A foglalt_e függvénnyel ellenőrzi, hogy a dátum törölve lett-e.

TEST(Abs_Naptar,EsemenyDatumbol)
-Annak ellenőrzése, hogy a DatumEsemenye függvény helyesen működik
-Ha lefoglalunk egy eseményt, a dátum megadásával meg kell kapni az adott eseményt.

TEST(Eves_Naptar,OsszesNapEv)
-A helyes összes nap működése az éves naptár osztályban

TEST(Honapos_Naptar,OsszesNapHonap)
//A helyes összes nap működése a hónapos naptár osztállyban

TEST(Honapos_Naptar,KezdoNap)
-Annak vizsgálata, hogy helyesen megállapítható, hogy egy hónap kezdete milyen napra esik.

4.3.4Fájlkezelés, IO modell

A Beolvaso_Teszt.Txt–ben szerepel egy dátum, egy hónap, egy év, egy n ami azt mutatja hány eseményt szeretnénk lefoglalni és ezután n esemény. Ezek az adatok be vannak olvasva a fájlból majd a Kiiro_Teszt.txt-be beírjuk a kapott dátumot és a lefoglalt eseményeket. Az inputban szereplő hónapnak és évnek megfelelő Naptárakat kiíratjuk, feltüntetve rajtuk a lefoglalt események helyét. Ezzel egyben megmutatjuk a fájlkezelés, valamint az osztályok beolvasó és kiíró tagfüggvényeinek működését.

A foglalt_esemenyek.txt biztosítja, hogy a program lefoglalt eseményei ne vesszenek el. Ezt a fájl olvasásra és írásra is használandó, ezért fstream-ként van deklarálva. Ha a lefoglalt események kiírására szükség van a naptár innen olvassa be az eseményeket, majd ide írja az újonnan foglalt. Törlés esetén töröljük az eseményt a dinamikus tömbből, majd annak tartalmát berjuk a fájlba, így a fájlban se lesz már jelen a törölt esemény.

A program std::runtime_error származó saját Gond_Van kivételosztállyal kezeli a nem szabványos hibákat, ezekhez üzeneteket is társít amit hiba elkapás esetén a kimentre kiír. Az, hogy a Gond_Van kivételeket előbb keresi a program mint más exceptionoket biztosítja, hogy a saját kivételeket ne az általános exception kapja el. Vannak a saját kivételosztályon kívül is kivételek, ezeket elkapja az általános exception, ha pedig egyikre se igaz az azt jelenti, hogy ismertelen a hiba ezért “Nagy baj van.” íródik ki a kimenetre.

4.3.5Felhasznált header fileok, stl tárolók,állományok

A projekt megvalósításához fel voltak használva különböző header fileok. Az <iomanip> a dátumok valamint a naptárak rendezett kiíratásához voltak szükségesek. Ez könnyebben olvashatóvá tette a naptárakat. Stl tárolókból a <string> osztály használata volt megengedett, mást nem használtam. Ebben tároltam a különböző szöveges változókat. Az <fstream> a fájlok kezeléséhez volt szükséges, amelyből van olvasásra, írásra és mindkettőre alkalmas fájl. A <locale> a magyar betűket megfelelően kezelő nyelv beállításához volt szükséges. Az <stdexcept> általak kerülnek a kivételek az outputra. A memóriaszivárgás ellenőrzéséhez a “memtrace.h” állomány van felhasználva, a teszteléshez pedig a “gtest_lite.h”.

4.3.6 Vátltozatások a pótleadásban

A laborvezető visszajelzése után a következőket megváltoztattam a projektben:

-a programban a nagy objektumok másolása helyett a referenciájukat használom függvények paramétereiben, valamint a függvények visszatéréseként.
-A Gond_van hibakezelési osztály az std::runtime_error-osztályból származik és ennek konstruktorát hívja meg az osztály explicit konstruktoraként.
-A programozói dokumentációt átírtam, hogy strukturáltabb és a programot jobban lefogó legyen.