Koenig Benedek

HÁZI FELADAT

Programozás Alapjai 2

Tartalomjegyzék

Koenig Benedek

A program

A BME EL épületében található sportközpontban lévő rendszer mintájára készülő öltözőszekrény-nyilvántartó program létrehozása a cél. Az épületben, ahol a program működni fog, 2 szinten találhatók öltözők, külön férfi és női, a földszinten 250-250, az emeleten 50-50 öltözőszekrénnyel. A szekrények használatához szükséges egy, a program által generált azonosító kód (string), és ehhez kapcsolódóan név (string) és nem (enum). Az azonosítók és nevek tárolását a programnak egy .txt fájlban kell megtennie.

A program használata

Indításkor a konzolablakon megjelenik a főmenü, melyekből menüpontjaiból a felhasználónak választania kell:

(1) Vendég bejövetel
(2) Szekrény zárás
(3) Szekrény nyitás
(4) Vendég távozás
(5) Szekrény meghatározása
(6) Új vendég felvétele
(7) Kilépés

A felhasználónak a megfelelő számot kell megadnia és entert ütnie. Hibás választás esetén a program erről tájékoztatja a felhasználót és lehetőséget ad az újrapróbálásra.

(1) Vendég bejövetel

A felhasználónak meg kell adnia a bejövő vendég azonosítóját. Ezután a program megkérdezi, listázza-e a szabad szekrényeket. Igen válasz esetén listázza azokat a szekrényeket, amelyeket a vendég lefoglalhat. A felhasználó választ egy szekrényt, annak számának a megadásával (int), melyet a program a vendéghez társít. Szekrényt csak az üresen állóak közül lehet választani, illetve férfiak csak férfi, nők csak női öltözőbe foglalhatnak szekrényt. Sikeres foglalás esetén a program tájékoztatja erről a felhasználót, majd visszatér a főmenübe. Sikertelen foglalás esetén ezt a program közli a felhasználóval, majd lehetőséget ad az újrapróbálásra.

(2) Szekrény zárás

A felhasználónak meg kell adnia a zárandó szekrény számát, majd a szekrényhez rendelt vendég azonosítóját. Jó párosítás esetén a szekrény zárt állapotúvá válik, hibás párosítás esetén a program ezt közli a felhasználóval. Ezután a program visszatér a főmenübe.

(3) Szekrény nyitás

A felhasználónak meg kell adnia a nyitandó szekrény számát, majd a szekrényhez rendelt vendég azonosítóját. Jó párosítás esetén a szekrény nyitott állapotúvá válik, hibás párosítás esetén a program ezt közli a felhasználóval. Ezután a program visszatér a főmenübe.

Koenig Benedek

(4) Vendég távozás

A felhasználónak meg kell adnia a távozó vendég azonosítóját. Ha a szekrény nyitott állapotú, a szekrény tulajdonos nélkülivé (foglalhatóvá) válik. Ha zárt állapotú, a távozás nem engedélyezett. Mindkét esetről a felhasználót szövegesen tájékoztatja a program.

(5) Szekrény meghatározása

A felhasználónak két lehetőség közül kell választania:

(1) Azonosító
(2) Szekrényszám

Az eljárás a főmenüben ismertetetthez hasonló.
Az (1) lehetőség választása esetén a felhasználó beírja egy vendég azonosítóját, a program pedig tájékoztatja a felhasználót, hogy a vendégnek

- van szekrénye, és meg is adja annak számát,
- nincs szekrénye,
- hibás a kódja, vagy nincs ilyen felhasználó.

A (2) lehetőség választása esetén a felhasználó beírja egy szekrény számát és hogy férfi vagy női öltözőben található, a program pedig tájékoztatja a felhasználót, hogy a szekrény

- foglalt és nyitva van,
- foglalt és zárva van,
- nem foglalt.

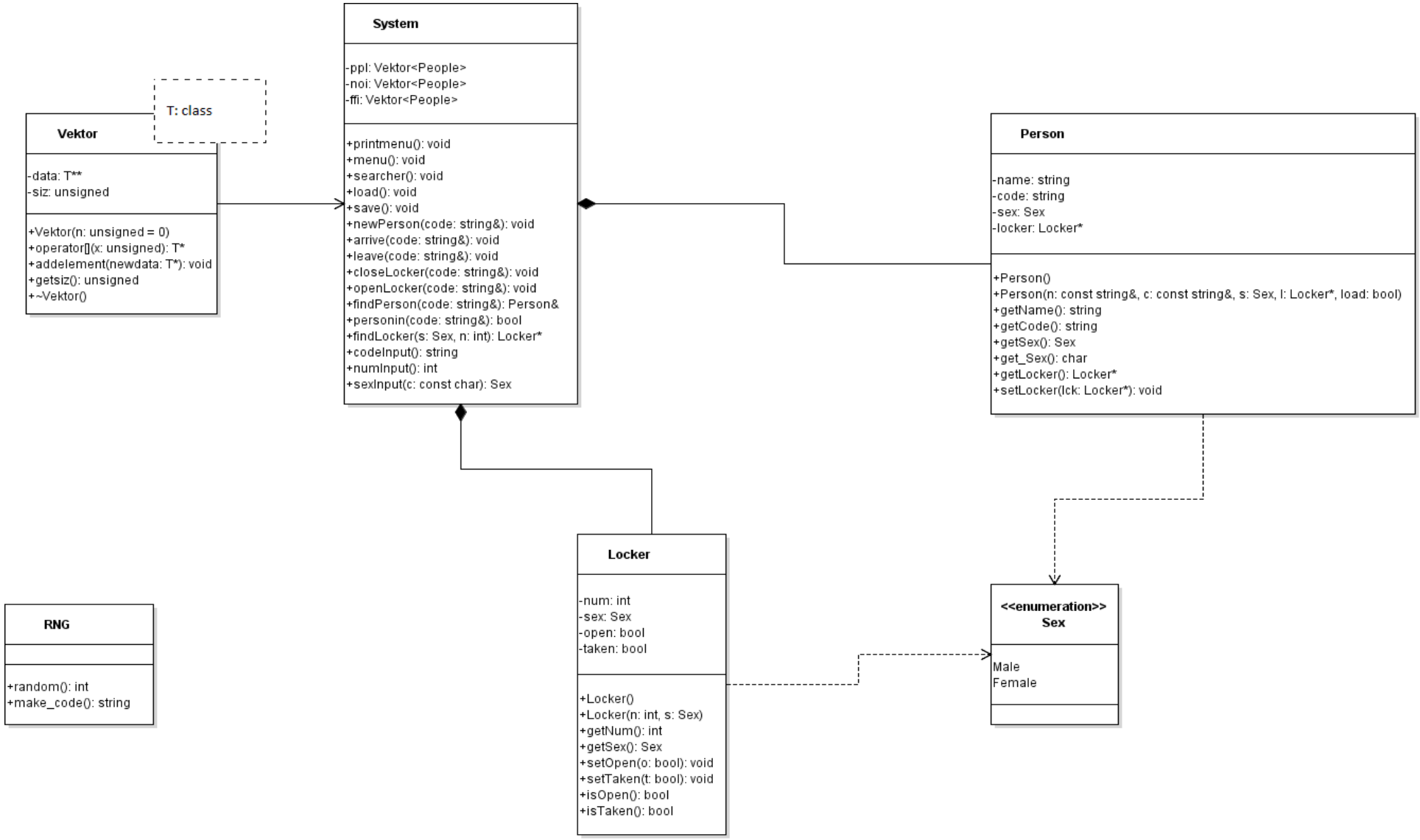
Mindkét esetet követően enter lenyomásával lehet visszatérni a főmenübe.

(6) Új vendég felvétele

A felhasználót a program tájékoztatja, hogy az új felhasználónak milyen adatát kell megadni (név, nem), majd kiírja az újonnan generált azonosítót. Új azonosító generálása esetén ellenőrzi, hogy az adatbázisban szerepel-e a kód, ha igen, addig generál újat, amíg az már nem szerepel. Az azonosító kiírása után visszatér a főmenübe.

Megvalósítás

A megvalósításhoz 4 osztályt, egy osztálysablont és egy enumot használtam fel, melyek kapcsolatát a következő UML diagram szemlélteti:



Koenig Benedek

Makrók, osztályok, tagfüggvények leírása

TESTLEVEL

Ez a makró a tesztüzemmód bekapcsolását teszi lehetővé. 1 érték esetén a főbb függvények kipróbálására szolgáló tesztprogram indul el, 0 érték esetén pedig a normális, specifikációban leírt működésű program.

System

A program irányító osztálya, ennek a főbb függvényei hívják meg a többi osztály tagfüggvényeit, illetve ez kommunikál a felhasználóval.

Adattagok:

ppl: Vektor<Person>

Személyek mutatóit tartalmazó dinamikus tömb.

noi: Vektor<Locker>

Női szekrények mutatóit tartalmazó dinamikus tömb.

ffi: Vektor<Locker>

Férfi szekrények mutatóit tartalmazó dinamikus tömb

Tagfüggvények:

printmenu(): void

Kiírja a menürendszert.

menu(): void

A menürendszer vezérlését irányítja.

searcher(): void

A menü 5. pontját, azaz a keresést vezérli.

load(): void

A system.txt fájl kiolvasásával létrehozza a rendszerben található szekrényeket, az users.txt fájl kiolvasásával pedig az eddig rendszerbe felvett személyeket.

save(): void

Elmenti a rendszert, a system.txt fájlba a szekrényeket, a users.txt fájlba pedig a személyeket.

newPerson(string&): void

Bekéri az új személy létrehozásához szükséges adatokat és azok, valamint a paraméterként kapott kód segítségével létrehoz egy személyt, akit elhelyez a személyek tömbjében.

arrive(string&): void

A menü 1. pontját, azaz egy személy megérkezését és szekrényfoglalását vezérli, akit a paraméterként kapott kód segítségével azonosít.

leave(string&): void

A menü 4. pontját, azaz egy személy távozását vezérli, akit a paraméterként kapott kód segítségével azonosít.

closeLocker(string&): void

A menü 2. pontját valósítja meg, azaz a paraméterként kapott kód segítségével azonosított személy szekrényét bezárja.

openLocker(string&): void

A menü 3. pontját valósítja meg, azaz a paraméterként kapott kód segítségével azonosított személy szekrényét kinyitja.

findPerson(string&): Person&

A paraméterként kapott személyt megkeresi a személyek tömbjében.

personin(string&): bool

Igaz értékkel tér vissza, ha a paraméterként megadott kóddal rendelkező személy megtalálható a személyek tömbjében, hamis értékkel, ha nem.

findLocker(Sex, int): Locker*

Az 1. paraméterként megadott nemű szekrények tömbjében megkeresi a 2. paraméterként megadott számú szekrény pointerét, amelyet visszaad.

codeInput(): string

A felhasználótól bekér egy azonosítókódot, amelyet visszaad.

numInput(): int

A felhasználótól bekér egy szekrényazonosító számot, amelyet visszaad.

sexInput(const char c): Sex

A paraméterként kapott paraméter függvényében, ha az kis vagy nagy m betű Male, különben Female értékkel tér vissza.

Person

Egy személy tárolására alkalmas osztály.

Adattagok:

name: string

A személy neve.

code: string

A személy azonosító kódja.

sex: Sex

A személy neme.

locker: Locker*

A személy szekrényének mutatója, alapesetben null.

Tagfüggvények:

Person()

A személyek üres tömbjének létrehozásához kell.

Koenig Benedek

Person(const string&, const string&, Sex, Locker*, bool)

A paraméterként kapott adatok segítségével létrehoz egy személyt. Az 5. paraméter azért kell, hogy ha a rendszer betöltésekor már sok személy szerepel az adatbázisban, ne írja ki a program feleslegesen a személyek kódját.

GetName(): string

Visszaadja a személy nevét.

GetCode(): string

Visszaadja a személy kódját.

getSex(): Sex

Visszaadja a személy nemét.

get_Sex(): char

Ha a személy férfi M-et, ha nő F-et ad vissza.

GetLocker(): Locker*

Visszatér a személy szekrényének mutatójával.

setLocker(Locker*): void

A paraméterként kapott szekrénymutatóra állítja a személy szekrénymutatóját.

Locker

Adattagok:

num: int

A szekrény száma.

sex: Sex

A szekrény neme.

open: bool

Megadja, hogy nyitva van-e a szekrény.

taken: bool

Megadja, hogy a szekrény foglalt-e.

Tagfüggvények:

Locker()

A szekrények üres tömbjének létrehozásához kell.

Locker(int, Sex)

A kapott paraméterekkel létrehoz egy szekrényt. Alapállapotban a szekrények nyitott állapotúak és szabadok.

getNum(): int

Visszaadja a szekrény számát.

getSex(): Sex

Visszaadja a szekrény nemét.

Koenig Benedek

setOpen(bool): void

A kapott paraméter felhasználásával átállítja a szekrény állapotát nyitottról zártra, vagy zártról nyitottra.

setTaken(bool): void

a kapott paraméter felhasználásával átállítja a szekrény foglaltságát.

isOpen(): bool

Megadja, hogy nyitva van-e a szekrény.

isTaken(): bool

Megadja, hogy foglalt-e a szekrény.

RNG

Az azonosító kódok előállítására szolgáló osztály.

Tagfüggvények:

random(): int

Visszatér egy véletlenszerű karakter ASCII kódjával.

make_code(): string

Előállít egy hattagú véletlen karaktersorozatot.

Vektor

Osztálymutatók tárolására szolgáló osztálysablon.

Adattagok:

data: T**

A mutatók tömbjének mutatója.

siz: unsigned

A tömb jelenlegi mérete.

Tagfüggvények:

Vektor(unsigned = 0)

Dinamikusan létrehoz egy megadott paraméter méretű tömböt.

operator[](unsigned): T*

Visszaadja a tömbnek a megadott paraméter által jelölt helyén található tagját. Végez túlindexelés-vizsgálatot.

addelement(T*): void

Megnöveli a tömb méretét eggyel, és hozzáadja a paraméterként kapott mutatót.

getsiz(): unsigned

Visszaadja a tömb méretét.

~Vektor()

Megsemmisíti a tömböt és a benne található minden adatot.

Koenig Benedek

Sex

Enum, melynek értéke lehet Male (férfi), illetve Female (nő).

Tesztelés

A tesztprogram

A fentebb említett TESTLEVEL makró segítségével lehet futtatni. Először létrehoz egy SYS objektumot a rendszer vezérléséhez, egy szekrényt, egy RNG objektumot, a későbbi kód-előállításhoz, illetve egy új személyt. A tagfüggvények működésének ellenőrzéséhez a gtest_lite nevű tesztkörnyezetet hívja segítségül.

Memóriakezelés tesztje

A memóriakezelés ellenőrzését a laborgyakorlatokon használt MEMTRACE modullal végeztem. Ehhez minden .cpp kiterjesztésű fájlban include-oltam a ”memtrace.h” állományt a standard fejlécállományok és a saját fejlécállományok után, így a saját fejlécállományaimban include-olt standard fejlécek sem keltettek zavart a működésben. Memóriakezelési hibát többször tapasztaltam, a Vektor sablon addelement függvényében fordult elő memóriaszivárgás, de sikerült kijavítani.

Lefedettség

A Jporta rendszer alacsony lefedettséget mutat, ami elsősorban a tesztprogramban nem behívott System-tagfüggvények magas aránya miatt van, melyeket nem teljességükben, hanem az azok által hívott függvényekkel tesztelt a program.