HÁZI FELADAT

Programozás alapjai 2.

Feladatválasztás/feladatspecifikáció

Gutási Ádám
XW53QZ

2022. 04. 03.

TARTALOM

1. Feladat

MVM

Tervezze meg a **Meseországi Villamos Művek (MVM)** nyilvántartási rendszerének egyszerűsített objektummodelljét, majd valósítsa azt meg! A rendszerrel minimum a következő műveleteket kívánjuk elvégezni:

- ügyfél adatinak felvétele
- szolgáltatási szerződés kötése
- szolgáltatási díj előírása (számlázás) (becsült átlagos fogyasztás)
- szolgáltatási díj befizetése
- egyenleg lekérdezése
- fogyasztás bejelentése

A rendszer lehet bővebb funkcionalitású, ezért nagyon fontos, hogy jól határozza meg az objektumokat és azok felelősségét.
Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz **ne** használjon STL tárolót!

2. Feladatspecifikáció

A feladat egy áramszolgáltató nyilvántartási rendszerének egyszerűsített objektummodelljét megvalósítani. A feladat nem specifikálja az adatok tárolásának módját, így a futás alatt dinamikus tömbök használata mellett döntöttem. A teszteléshez használt minta adatokat szövegfájlokba rendezve, az összetartozó adatokat tabulátorral tagolva fogom tárolni. A következő szövegfájlokat fogom megírni a tesztelésekhez:

1. Ügyféladatok (Vevőazonosító[1],*Vezetéknév, Keresztnév, Születési dátum, Lakcím, Telefonszám, e-mail cím, magánszemély/vállalkozás, adószám, legutolsó bejelentett óraállás, egyenleg*)[2]
2. Szolgáltatási szerződések (A főbiztosíték áramerősségét, a fázisok számát, lakossági/vállalati).
3. Fogyasztás bejelentések (induló és zaróállás, vevőazonosító)
4. Előre kiírt számlák (ha az ügyfél elkésik a bejelentéssel, vagy az egyenlege alapján levonás után nincs mit fizetnie).
5. Teljesített számlák
6. Villamos energia tarifák
7. naplófájl (nem tartozik szorosan a teszteléshez)

Az ügyféladatok esetén végrehajtok egy minimális ellenőrzést (például az e-mail cím megfelel-e az e-mail cím szabályos alakjának). Ezt a **regex** könyvtár segítségével oldom meg.

[1] Egy ügyfél létezhet a rendszerben többször is, több azonosítóval (magánszemélyként/vállalkozásként, akár azokban többször is). Egy vevőazonosítóhoz egy mérőóra tartozik.
[2] Két ügyféltípust fog megkülönböztetni a rendszer: magánszemély és vállalkozás. A *vállalkozás* ügyféltípusnál értelemszerűen a megadott személyes adatok a cég kapcsolattartójának adatai. A két kategóriára külön árszabás lesz érvényben.

A program indulásakor megjelenik egy menü, amiből ki lehet választani, milyen műveletet szeretnénk végrehajtani. Az adott művelet végrehajtása után visszatérünk ugyanebbe a menübe, egészen addig, amíg a kilépés opciót nem választjuk.

A teszt megírásához a **gtest_lite** könyvtárat fogom használni, az esetleges memóriaszivárgások felderítésére pedig a **memtrace** könyvtárat.

A fizetendő tartalmazza a fogyasztott áram mennyiségéből, az árszabás alapján számított díjat, a rendszerhasználati díjat, és az ÁFA mértékét (ügyféltípustól változó). Ha az ügyfél túlfizeti számláját (tehát egyenlege a befizetés után pozitív), akkor a következő számlázási időszakban (következő hónapban) a kiállított számla tartalmazza az elszámolás értékét (tehát túlfizetett számlánál levonódik a maximális egyenleg az ügyfél egyenlegéből.)

A saját áramot termelő ügyfelek esetén feltételezzük, hogy a bejelentett mennyiségből már ki van vonva az általuk termelt árammennyiség. Feltételezzük továbbá, hogy ezen ügyfelek mérőórái nem mozdulnak el termelés hatására.

A program képes lesz egy adott számlázási időszak (év hónapjai) összesítésére, illetve kiértékelésére: ha nem érkezett az adott hónapra fogyasztás bejelentés, és sikeres teljesítés, akkor előre kiír számlát a program. Ezeket az eseményeket egy naplófájlban lehet követni, tesztelés és debug céljából. A naplózás a CPORTA flag használatával kikapcsolható.