Lift szimulátor dokumentáció

Programozói dokumentáció:

A program 4 modulból épül fel, és nem használ semmilyen speciális külső könyvtárat, vagy környezetet. A modulok tartalma a következő:

Az első modul a „fileManager.py” fájl, melyben, ahogy a neve is mutatja a fájlok menedzselésének folyamata megy végbe. Megkönnyíti a szimuláció bemenetét feldolgozó program dolgát. Benne három alfüggvény és egy főfüggvény található. Az első alfüggvény bekéri a felhasználó a szimulálni kívánt adatokat, és a megfelelő formátumra alakítja. Az időnél magyarázatra szorul, hogy miért 5 másodperces időközöket választottam, ez egy „becslés” az összes esemény körülbelül 5 másodpercet vesz igénybe (pl.: egy utas beszállása/kiszállása, egy emelet megtétele stb...), ezért úgy veszem hogy 1 „tick” az 5 másodpercet jelent . A második alfüggvény szimplán a bemeneti fájl írásáért felel, legvégül pedig egy külön alfüggvény a vizuális menü megjelenítéséért. Ezeket koordinálja egy rövid főfüggvény, mely addig hívja a függvényeket ameddig kettest nem kap inputként, ebben az esetben elkezdi a szimulációt a létrehozott „szimulacio.txt” szöveges fájl alapján.

A második modulként a grafikus megjelenésért felelős „rajz.py” fájlt említeném. Legelőször is a program vizuális grafikáját a „turtle”, mint beépített könyvtár építi fel. A modul, úgymond „főfüggvénye” a „teljesRajz” függvény, ez értelemszerűen meghívja és kirajzolja azokat az elemeket, melyeket a többi függvény meghatároz. Az érdekesebb függvények a „kijelzoRajz” és „szamRajz” nevű függvények. A „kijelzoRajz” -ot a szimulátor hívja meg minden egyes lift mozgás hatására, mely először letörli, majd kiírja a megfelelő emelet szintjét, annak függvényében melyik liftről beszélünk. A kiírást maga a „szamRajz” függvény végzi. Mindkettő lifthez külön-külön hoztam létre „turtle”-t, hogy a törlés és újraírás a lehető legkönnyebb legyen.

A harmadik modul az egész program szíve, maga a szimuláció algoritmusa. Itt találhatóak az osztályok, melyeket használunk. Az első a „Szemely” osztály, melynek argumentumai a súlya, érkezési ideje, a szint ahonnan indul és a ahova érkezni szeretne, ezenkívül kezdetben inicializált változók, a várakozási idő (, mely az érkezési idő és a liftbe szállás pillanatának idejének különbsége), a utazási idő (, mely az érkezési és várakozási idő összegének a kiszállás pillanatától vett különbsége). A második a „Lift” osztály, kezdeti értékei

a teherbírása, jelenlegi szintje és az iránya. További változók a jelenlegi teher, mely egy személy beszállásakor adódik hozzá, a személyek, akik a liftben tartózkodnak, a cél emeletek listája (ehhez a listához akkor adunk hozzá, ha az adott emelet nem szerepel még a listában, ezt a „lifthezAdas" függvényt dönti el), és a statisztikához szükséges felvett változó, mely a végén megadja hány embert vett fel az adott lift. Kezdetben a „fileManager" meghívásával megtörténik a kívánt szimuláció elindítása. Ezután a program főfüggvénye meghívja a rajz modult, kirajzolódik a grafika és megkezdjük a fájl beolvasását és feldolgozását, majd pedig elindul az algoritmus. Fő eleme egy „while" ciklus, mely addig fut, ameddig mindegyik személyt fel nem dolgoztuk, vagy az idő több nem lesz, mint egy nap (és egy óra azért, hogy minden utast kiszolgáljunk). Három fő részből épül fel az algoritmus: kiszállás, beszállás, mozgatás. Ezeket a függvényeket hívja meg mindaddig ameddig a „while" belépési feltétele igaz.

A kiszállás egyszerűen csak megvizsgálja, hogy azoknak a személyeknek, akik a liftben tartózkodnak megegyezik-e a kiszállási szintjük a jelenlegi lift szinttel. Ha igen, végrehajtja a kiszállást és a személy a kiszolgált tömbhöz adjuk.

A beszállás a program legbonyolultabb része. Végig nézzük a személyek tömböt és ha valamelyik személy érkezési ideje megegyezik a jelenlegi idővel és még nem hívta a liftet, elkezdjük vizsgálni. Először megnézzük melyik lift található hozzá közelebb és ezek hozzá képest melyik irányba mozognak („melyikIrany" függvény), ezután pedig megnézzük az összes lehetőséget, melyet három fő esetre és alesetekre lehet bontani. Ez után mozgatjuk a megfelelő liftet hozzá és beléptetjük a liftbe.

1. Egyik lift sem mozog, a közelebbit mozgatjuk.
2. Mindkét lift mozog.
    a. Ha az első lift közelebb van és az ő irányába mozog.
    b. Ha az első lift közelebb van, de az tőle el mozog.
        i. Ha az első lift közelebb van, de az tőle el mozog, de a második lift az ő irányába mozog.
        ii. Ha az első lift közelebb van, de mindkét lift tőle el mozog. *(Ebben az esetben lép be a mag algoritmus, avagy a „dontes" függvény, mely a szerint dönt, melyik lift tartalmaz kevesebb utast)*
    c. Ha a második lift közelebb van és az ő irányába mozog.
    d. Ha a második lift közelebb van, de az tőle el mozog.

i. Ha a második lift közelebb van, de az tőle el mozog, de az első lift az ő irányába mozog.
ii. Ha a második lift közelebb van, de mindkét lift tőle el mozog. *(Ebben az esetben lép be a mag algoritmus, avagy a „dontes” függvény, mely a szerint dönt, melyik lift tartalmaz kevesebb utast)*

e. Ugyanakkora távolságra van a két lift
i. Az első felé mozog.
ii. A második felé mozog.
iii. Mindkettő tőle el mozog. *(Ebben az esetben lép be a mag algoritmus, avagy a „dontes” függvény, mely a szerint dönt, melyik lift tartalmaz kevesebb utast*

3. Csak az egyik lift mozog, mindig a nem mozgó liftet mozgatjuk.

A mozgatás végig nézi a lifteket és az adott irányokhoz megfelelően mozgatja. Esetekre bontva, ha csak egy utasa van azt kiteszi és megáll, ha több van akkor a következő utas végszintjének irányába mozog és ha nincs utasa és nem is hívták áll. Ezen a ponton hívjuk a kijelző rajzolását.

Majd visszatérve a főfüggvénybe meghívjuk a statisztika modult. Az iv változó melyet látunk az egész modul során, egy „deszinkronizálást” megelőző „flag”, ami megvalósítja, hogy mindkét lift számára ugyanakkor teljen az idő.

Az utolsó modul a „statisztika.py”, megkapva az adott változókat, megfelelő formátumba rendezi és kiírja a „statisztika.txt” fájlba.

Egy plusz modult is betettem, melyet a tesztelés során használtam, a randomizer, mely random előállít egy megadott mennyiségű utast a szimulációhoz.

Felhasználói dokumentáció:

A program feladata liftek vezérlése. A liftekhez adott időközönként emberek érkeznek, akik jelzik úticéljukat (emelet száma). A program algoritmussal irányítja a lifteket, kiszolgálja az utasokat. Az utasok a liftbe beszállva (ha az nincs túlterhelve) eljuthatnak a cél állomásra, ahol aztán kiszállnak.

A bemenet létrehozásában segít a „fileManager.py” modul, de ha saját előre megírt és formázott szimulációt szeretnénk akkor azt is lehet futtatni, csak a program mappájába kell bemásolni a szöveges dokumentumot „szimulacio.txt” néven. Ez után a felhasználónak nincs további teendője, a képernyőn a vizuális megjelenítés hatására nyomon követheti a liftek állapotát, majd a szimuláció lefutása után megtekintheti a „statisztika.txt” fájlban létrehozott statisztikai elemzést, melyet a ugyancsak a program mappájában talál meg.