Programozói dokumentáció

A felhasználótól elvárás :

1. A txt fájlt helyesen töltse ki (a helyes forma

%d,%d\n

%d,%d,%d,%d:%d:%d,%d,%d\n

%d,%d,%d,%d:%d:%d,%d,%d\n

Stb.

2. Ne adjon meg olyan számokat amiket nem lehet teljesíteni

Egyéb elvárás : A célmappában legyen teszt ,és tesztki txt mappa amiből/be olvashat/írhat.

A program működése :

1. A program egy fájlból beolvassa a teszt adatokat ,és bemásolja egy struct tömbbe
2. Rendezi időrendi sorrendbe az adatokat
3. Elindítja a liftszimulátort
4. A liftszimulátor érkezési sorrendben próbálja kiszolgálni az utasokat , de ha valaki útba esik azt kirakja/felszedi
5. A lift 10mp-enként megy egy emeletet ,ha van célja(tehát valaki hívta)
6. Ha minden utast kiszolgált végzett , és az utazás statisztikáit kiírja egy txt fájlba

A projekt felépítése :

Projekt

C fájlok :

1.Main.c

A/ liftmegy

B/beszall

C/leszall

D/rendez

E/vanhely

F/mpidosz

G/statisztika

2.Bekiolvas.c

A/beolvas

B/kiir

3.Celmod.c

A/elozoelem

B/celtorol

C/celbefuz

D/elemcsere

E/helyetcserel

4.Utasmod.c

A/bemasol

B/torol

C/torolliftrol

D/befuz

E/tobb_nyom_rendez

Headers:

Bekiolvas.h

Bekiolvas.c-hez tartozó header

Celmod.h

Celmod.h-hoz tartozó header

Structok.h

Tartalmazza az összes strurctot

Utasmod.h

Utasmod.c-hez tartozó header

ADATSZERKEZETEK, TÍPUSOK :

sructok :

Maxmin

Legtöbbet/kevesebbet várakozó utas

Idok

Utason belül az óra,perc,mp –t egy külön structba raktam mert szorosabban összekapcsolódnak mint a többi

Utas

Szimuláció utasainak az adatait tárolja(honnan , hova megy ,id-je,súlya,ideje(az előző struct) tehát hogy mikor hívja a liftet,mpido(ez az idő mpre átszámolva hogy könnyebb legyen vele számolni)

Cel

Cél ,innen tudjuk meg hogy hova tart a lift

Elemei :

Ertek -> hova tart a lift , *id*->melyik utas az akihez ez az érték tartozik , *mvf* -> megnyom vagy felszál /megkülönbözteti hogy a cél az valakihez megy aki megnyomta a gombot vagy olyat visz a céljához aki rajta van

Tulsok

Suly,fo a túl terhelés adatait egy structba raktározom mert így könnyebb őket értelmezni, mintha egy tömb első és második eleme lennének

Tömbök

a. orak : ide mentem el hogy adott órában hány utas volt
b. Ember vagy tomb : struct utasokból álló tömb ide mentem el a fájlbeolvasásból az utasok adatait(dinamikus tömb)
c. Egyszerre : adott időben a gombot egyszerre megnyomó emberek tömbje

Tömböket az egyszerűségük és a könnyenbejárhatóságuk miatt használtam

Láncoltlisták

irany: az egymást követő célok vannak láncolt listába összefűzve, mert így nem kell mindig átadni mutatóval a méretét a függvényeknek (mivel a célok száma folyamatosan változik)

Varolista: a gombot megnyomó utasok vannak rajta, ha felszálnak a liftre lekerülnek a listáról

Liftenvan : A liften lévő utasokat tárolja (A lift tudja a rajta utazók adatait amíg rajta vannak) , várólistáról ide kerülnek , leszálásról innen törlődnek

FÜGGVÉNYEK :

Lényegesebb függvények :

utas* beolvas(int* scnt,tulsok *tul)

visszatérés : egy utas tömbbel amibe belemásolta a beolvasott adatokat, *paraméter* : *scnt -> hány utas van az int pointerét adom át és azt módosítom , *tul -> a túlterhelés adatatait ebbe mentem el

Feladat : beolvassa az utasok adatait a txt fájlból és bemásolja a megfelelő helyekre

Körülmény: kell lennie txt fájlnak és helyes formában kell szerepelniuk az adatoknak

void kiir(double varatl, int *orak, int emeletsporol, double emeletsporolszazalek, maxmin ai)

Visszatérés : void

Paraméter : statisztikai adatok

Varatl: átlagos várakozási idő

Orak : 24elemű tömb

Emeletsporol : hanz em

Varatl: mennyit vártak átlagosan

Feladat: statisztikai adatok kírása txt fájlba

void statisztika(int varatl,int osszemelet, int meret,const utas* lista,maxmin ia)

Feladat: kiir függvénynek készíti elő a statisztikai adatokat

void liftmegy(utas *tomb, int meret, tulsok tult)

A lift itt halad ő hívja meg a végén a statisztika függvényt

Ő hívja meg a a beszall , leszall függvényeket amik segítségével az utasokat kezeli

Ő módosítja a liftemeletét, ő rakja fel a várólistára az elmeket a befuz függvény segítségével, és ő rakja fel az irányra az új célt a celbefuz függvény segítségével

Pramétere : utasok tömbje és a lift túl terhelés adatai(hogy később átadhassa)

void beszall(utas *varolista[], utas *liftenvan[], int liftemelet, idok ido, cel irany, tulsok tul, int *varatl, maxmin *max,int *varomod)**

Az utasok itt szálnak be a liftbe (és kerülnek le a várólistáról), emellet a cél adatai itt módosulnak a megnyomott gomb helyéről az utas célja helyére

Statiztikai adatokat is számol (pl. Legnagyobb/kisebb várakozó idő össz várakozás)

Paraméterei : liftenvan,liftemlet,irany láncolt listák első eleme ezeket módosítja

Liftemelet : ezzel hasonlítja össze hogy az aktuális elemen van e beszálló

Tulsok tul: ezt adja át a vanhely függvénynek, hogy beállíthassa végértéknek amikor a túlterhelést teszteli (csak akkor lehet felszállni, ha nincs túlterhelve

Ha nem tud felszállni akkor megnézi, hogy az adott utas volt-e a cél, ha igen akkor módosítja a célt az első liften lévőre, hogy legyen hely

void leszall(int liftemelet, utas **liftenvan, idok ido, cel **irany, int* vege, int vegid)

Az utasok itt szálnak le a liftről ,cél adataiból kikerül az adott utas,

Paramétere: liftemelet hogy lássa leszáll e valaki az adott emleeten

Liftenvan, irany láncolt lista ->ezeket módosítja

Vege, vegeid -> megnézi, hogy a leszálló utas idje egyezik e az utolsó utas idjével, ha igen a veget növeli 1el (igy a liftmegy függvény tudni fogja, hogy leszállt az utolsó utas és meg kell állnia)

Kisebb (segéd) függvények:

Main.c-ben

int rendez(const void *a, const void *b)

Qsort segédfüggvény (növekvő sorrendbe rendezik idő szerint az utasokat)

int mpidosz(int o, int p, int m)

Átszámolja az óra:perc:mp –t mpbe

bool vanhely(utas li[], int xtra,tulsok tult)

Megnézi hogy nem lesz e túlterhelve a lift ha felszáll az adott utas

Paraméterek : liftenvan lista , xtra (a felszáló utas súlya) , tulsok tult (a max teherbírás)

Visszatérési értéke true ha vanhely és false ha nincs

Celmod.c-ben

cel* elozoelem(cel* elso, cel* elem)

Megnézi hogy mi az előző utas a cél listában(helyetcserel függvényhez kell)

cel* celtorol(cel* p, int mitid, char c)

Eltávolítja az adott elmet a cél listából

Paraméterek : cél lista , mitid ,c : a mitidvel és a c-vel azonosítja az utast amelyiket törölni kell (azért kell egy char c hogy lássa hogy m ->megnyomott célról van szó vagy f-> felszált célról

cel* celbefuz(cel* elso, utas a, int ce, char c)

A cél listába beszúrja az adott utast és visszatér az új módosított céllistával

Paraméterek: *elso* a lista, a az adott utas, *ce* az utas hova tart (azért kell ezt külön megadni mert a celbefuzt felszállásnál (ahol a hova a fontos) es megnyomásnál (ahol a honnan) is használom

Char *c*-vel mondom meg hogy mvf : megnyom vagy felszáll

void elemcsere(cel **elso, cel mit)

Berakom az adott utast az elso helyre

Utasmod.c-ben

void bemasol(utas *p, utas a)

Az utas adatait másolja sok helyen használom mert érték adásnál nem kell 6 különböző dolgot átmásolni ,csak ezt meghívni

utas* torol(utas* p, int mit)

Utas listáról töröl egy elemet

utas* torolliftrol(utas* p, int mit)

Liften lévő elemek törlésére használatos (itt az utas hovája kell ,törölben a honnanja)

utas *befuz(utas *head, utas a)

Utas listára befűz egy utast

utas* tobb_nyom_rendez(utas tomb[], int meret, int liftemelet)

Ha egyszerre többen nyomták meg ugyan akkor a gombot akkor ez a függvény rendezi őket hogy sorrednben szolgáljha ki őket a lift (először a legközelebbihez megy , ha 2 utas ugyan olyan messze van tőle akkor a kisebb emeleten lévőhöz megy)

A PROGRAM RÉSZLETES MŰKÖDÉSE

BEOLVAS függvénnyel beolvassa az utasokat és bemásolja egy utas struct tömbbe

Qsorttal időszerinti növekvő sorrendbe rendezi

Meghívja a liftmegy függvényt , ami másodpercenként ellenőrzi hogy van e változás egészen addig amíg le nem száll az utolsó utas

Ha a következő utas megnyomási ideje egyezik az aktuális másodperccel akkor megnézi hány ilyen utas van ,berakja őket egy tömbbe és rendezi (tobb_nyom_rendez függvény) majd sorban befűzi őket a várólista listára és a cél irány listára

10 másodpercenként növeli vagy csökkenti a liftemeletet az aktuális céltól függően és lefuttatja a leszall, beszall függvényt

A leszall függvényben: megnézi, hogy a liftenvan listán van e olyan utas, aki odament, ahol épp a lift van, ha van ilyen akkor letörli a liftenvanlistarol a torolliftrol függvény seítségével, és törli a cél irány listáról a celtorol függvénnyel

A beszall függvényben megnézi, hogy a várólistán van e olyan utas, akinek a honnan emelete egyezik az aktuális emelettel, ha van, lefuttatja a vanhely függvényt, ami megmondja, hogy felfér e a liftre, ha igen akkor felrakja a liftenvan listára, leszedi a várólistáról, törli a várólistás célját és berakja a cél listára (celbefuz függvény) a liftenvan célját (nem kicseréli mert érkezési sorrendben halad)

Ezenkívul a max/min várakozást is számolja (ha az aktuális elem többet / vagy kevesebbet várt mint a maximimum, vagy a minimum akkor kicseréli rá)

Ha nincs hely a liften és nem az adott utas a lift aktuális célja, akkor nem csinál semmit, ha ő az aktuális célja akkor az aktuális cél a liften levő első elemre cserélődik (elemcsere függvény) (aki így leszáll, lesz hely a liften és visszamegy az eredeti cél utasért)

Ha a liftmegy végzett az utolsó utassal is, akkor meghívja a statisztika függvényt és átadja neki a gyűjtött adatokat, ami ezután kiszámolja a statisztikákat a gyűjtött adatokból, és meghívja és átadja ezeket a kiir függvénynek, ami kiírja őket egy txt fájlba.