

BENZINKÚT SZIMULÁCIÓ

Készítette: Kiss Endre Farkas

Gyakorlatvezető: Horváth László

Informatikatanári-matematikatanár

- II. szemeszter 2. beadandó –

2004 Május 21.

TARTALOMJEGYZÉK

TARTALOMJEGYZÉK.....	2
FELHASZNÁLÓI ÚTMUTATÓ	3
A program feladata:	3
Követelmények:.....	3
Indítás:	3
A program használata:	4
A képernyő főbb részei:	5
Egy fontos, a program használatával kapcsolatos kérdés, és válasz:	5
Hibalehetőségek	6
FEJLESZTŐI DOKUMENTÁCIÓ.....	7
• A feladat leírása:	7
• Specifikáció:	7
Futtatási környezet:.....	9
Főbb állományok:	9
• Megoldási elv:	10
Tesztelés.....	13
ALGORITMUS:.....	16
A továbbfejlesztés lehetőségei:	24
FORRÁSKÓD	25

FELHASZNÁLÓI ÚTMUTATÓ

A program feladata:

Adott egy benzinkút különböző üzemanyagokat adó kutakkal, melyek előtt egyenként állnak sorba az adott üzemanyagért érkező autósok. Mielőtt valamelyik kúthoz besorolnak, először a benzinkút egy közös beálló sávsában várakozik az összes érkező. Tankolás után a pénztárakhoz mennek, ahol mindegyikük a legrövidebb pénztársorba áll be. A távozás pedig egy közös kimenő sávban történik.

A program véletlen szerű adatokkal modellezi a fenti benzinkút működését, a történésekről statisztikai kimutatásokat készít, valamint lehetőséget biztosít arra, hogy a főbb paramétereket (kutak, illetve pénztárak száma) a felhasználó futásidőben változtathassa.

Követelmények:

A program elfut bármilyen *MS-DOS 6.22*, illetve Win9x kompatibilis rendszer alatt. (pl. *Windows 95, 98, Me, XP*).

Az alkalmazás Windows rendszereket futtatni képes rendszerek esetén nem támaszt extra igényeket gépünkkel szemben, régebbi rendszerek esetén minimális követelményként kell tekintenünk az alábbi konfigurációra:

- Intel 386 CPU
- 2 Mb RAM
- Floppy meghajtó
- VGA vezérlő
- 40 Mb HDD
- 14' monitor

Indítás:

Az iskolai sportjelentkezéseket nyilvántartó program a mellékelt CD lemezen a főkönyvtárban lévő *benzin.exe* fájlal indítható.

A forráskód a *souce* nevű könyvtárban található *benzin.pas* néven.

A program használata:

A program elindítása után egy rövid szöveges ismertetőhöz jutunk, amely ismerteti a program főbb funkcióit, és használatát.

```
Kiss Endre Farkas
Info-Matek 2-ik szemeszter, 2-ik beadandó.

A program feladata az, hogy egy számítógépen modellezze
egy benzinkút működését. Adott egy fő sor, ahol az összes
tankolni vágyó autós érkezik. Ezek aszerint, hogy milyen
jellegű üzemanyagra van szükségük, besorakoznak a megfelelő
terméket adó kúthoz, és várnak arra, hogy az előttiük álló
végezzen a tankolással. Miután tankoltak, a pénztárhoz
fáradnak, ahol abba a sorba állnak be, ahol a legkevesebbet
várnak. Fizetés után pedig mindenki egy közös sorban hagyja
el a benzinkút területét.

Előfordulhat persze, hogy a keresett üzemanyagot adó
kút éppen nincs megnyitva. Ha ez a helyzet, akkor az autós
a bejövő sorból vásárlás nélkül meg tovább a kimenő soron.
Menet közben lehetőség van változtatni a pénztárak, illetve
a nyitva lévő kutak számát a megfelelő billentyű lenyomásával.
Ehhez a szimulációs képernyő alsó részében látható segítség.
A program menet közben összegzi a történéseket, ezt az összeg-
zést szimuláció közben az I betűvel lehet megtekinteni, az
információs részből visszalépni szintén így lehet.
A szimulációt léptetni az X gombbal lehet menet közben.

<ENTER>
```

Ezután már rögtön a program szimulációs képernyőjéhez jutunk.

Ez a képernyő egyszerre szemlélteti az összes várakozó gépkocsi helyzetét, és főbb adatait egyetlen képernyőn.

1. kút 098-as	2. kút 095-ös	3. kút 096-os 10 BMW	4. kút Diesel	BEJÖVETELI SOR 16 096-os Lada 1
1. pénztár	2. pénztár	3. pénztár	4	KIJÁRAT SOR 2
I - info ki/be M - stat mentés	Q - több benzinkút A - kevesebb benzinkút	W - több pénztár S - kevesebb pénztár	5	X - LÉPTETÉS K - KILÉPÉS

A képernyő főbb részei:

1. Bejöveteli sor

Itt jelennek meg egymás alatt a benzinkúthoz érkező autók. Láthatjuk, hogy hány liter üzemanyagot szándékoznak vásárolni, milyen oktánszámút, valamint hogy milyen márkájú a gépkocsi.

2. Kijárat

Miután a benzinkútban minden elintéznivalójával végzett, az összes autós erre a közös sorra kerül, ahonnan a külvilágba (a program hatósugarán kívülre) kerül, vagyis töröljük.

3. Benzinkutak

Itt láthatjuk az egyes benzinkutak adatait. Mindegyik kút csak egy fajta üzemanyagot árusít, hogy melyet azt a kút sorszámozása alatt láthatjuk. Az egyes kutakhoz külön-külön sor tartozik. Látjuk egyrészt, hogy mi a gépkocsi márkája, valamint azt, hogy még hány liter üzemanyagot szeretne az illető tankolni mielőtt a pénztárhoz menne.

4. Pénztárak

Mindegyik pénztár teljesen egyenrangú. Ezt azt jelenti, hogy a kifizetni szándékozó személyek automatikusan abba a sorba állnak be, ahol a legkevesebben várnak.

5. Információs rész

Segít eligazodni a program kezelésében azzal, hogy megmutatja, mely billentyűkkel avatkozhatunk be a szimulációba.

Egy fontos, a program használatával kapcsolatos kérdés, és válasz:

- *Van lehetőség a háttérállomány törlésére a programból?*

Válasz: Nincs, azonban bármikor felülírható az állomány egy újabb mentéssel.

- *Van lehetőség többféle háttérállomány lemezre mentésére?*

Válasz: Nincs, ha mégis ilyen igénye lenne, azt javaslom, hogy másolja át a régi stat.txt-t egy biztonságos helyre, mielőtt újabb állapotot mentene el.

Hibalehetőségek:

Mivel a felhasználó és program közötti kommunikáció szűken meghatározott keretek közé van szorítva, ezért a vezérléssel kapcsolatos hibalehetőség nem ismert.

Egyetlen hiba akkor következik be, ha a véletlen, és a programparaméterek szerencsétlen összejátszása folytán valamelyik sor túlcsordul. Ebben az esetben az alábbi képernyővel találkozunk:



```
HIBA! Valamelyik sor futásidőben túlcsordult!  
Futassa a programot több kúttal, vagy pénztárral!  
<ENTER>
```

FEJLESZTŐI DOKUMENTÁCIÓ

- A feladat leírása:

A feladat egy benzinkút működésének számítógépes szimulációja. Adottak autósok, akik bizonyos fajta benzinre való igényel érkeznek. Adottak kutak, amelyek az igényük alapján hozzájuk sorolt autókat kiszolgálják. Adottak pénztárak, ahol a tankolás után az autósok fizetnek, majd fizetés után egy közös sorban távoznak. A működés közben nyilván kell tartani, hogy mennyi benzin fogyott.

- Specifikáció:

Be: $\text{maxSorHossz} \in \mathbb{Z}$, $\text{Sor} \in (\text{elemszam} \in \mathbb{Z}, \text{elemek} \mathbb{Z}^{\text{maxSorHossz}})$
 $\text{beSor} \in \text{Sor}$, $\text{kutSor} \in \text{Sor}^{\text{maxKut}}$, $\text{penztarSor} \in \text{Sor}^{\text{maxPenztar}}$
 $\text{kiSor} \in \text{Sor}$, $\text{autos} (\text{fajta} \in \mathbb{Z}, \text{igeny} \in \mathbb{Z})^{\text{maxautos}}$

Ki: $\text{tanoltBenzin} \in \mathbb{R}$

Ef: -

Uf:

$\forall j \in [1..\text{maxautos}]:$	Sorba (autos_j , beSor)
$\forall j \in [1..\text{beSor}]:$	Sorba (beSor_j , $\text{kutSor}_{\text{beSor}_j[\text{fajta}]}$) Sorbol (beSor , beSor_j)
$\forall i \in [1..\text{maxKut}]:$	Sorba (kutSor_{i_1} , $\text{penztar}_{\text{minpenztar}}$) Sorbol (kutSor , kutSor_{i_1})
minpenztar $\text{minpenztar} = \forall j \in [1..\text{maxKut}]: \min(\text{kutSor}_j)$	$\text{kutSor}^{\text{maxKut}} \rightarrow \mathbb{R}$
$\forall i \in [1..\text{maxPenztar}]:$	Sorba (penztarSor_{i_1} , kiSor) Sorbol (penztarSor , penztarSor_{i_1})
$\forall j \in [1..\text{kiSor}]:$	Sorbol (kiSor , kiSor_j)

Sorba: elem, Sor \rightarrow Sor, \perp
 (Error $\in \perp = (|Sor| = \text{maxsorhossz})$
 Error = \downarrow :
 (Sor_{elemszam+1} = elem
 Sor.elemszam = Sor.elemszam+1)

Sorbol :Sor \rightarrow Sor, elem, \perp
 (Error $\in \perp = (|Sor| < 1)$
 Error = \downarrow :
 (Sor₁ = elem,
 $\forall j \in [1..|Sor|-1]: \text{sor}_j = \text{sor}_{j-1})$

$$\text{tankoltBenzin} = \sum_{i=1}^{\text{maxautos}} (\text{autos}_i(\text{igeny}))$$

Futtatási környezet:

A program minimális követelményei megegyeznek a Borland Turbo Pascal rendszerkövetelményeivel. Általánosságban egy MS-DOS kompatibilis, legalábbis azt emulálni képes rendszer ajánlható.

Az alkalmazás fejlesztése Windows XP alatt telepített és üzemeltetett Turbo Pascal 7.0-val történt. Mivel ez egységes felületet biztosít, ezért joggal feltételezhető, hogy bármelyik, a Pascal keretrendszer jelen kiadását futtatni képes rendszeren hasonló eredményre juthatunk, ugyanakkor megemlíthető, hogy itthoni rendszeremnél nem állt rendelkezésre a **newdelay** kiegészítő-könyvtár, így ezt nem használtam fel a gyors gépek késleltetéséhez. Bár saját számítógépemem gond nélkül fut a program, elképzelhető, hogy más rendszerben ez gondot jelenthet. A program a CD főkönyvtárából indítható, azonban erősen ajánlott futtatás előtt felmásolni egy írható háttértárolóra, hogy az állományműveletek ne okozzanak gondot.

Főbb állományok:

- **benzin.exe**

Biztonsági mentés: <http://www.mvpp.hu/benzin.exe>

A program indítása történik a segítségével.

- **benzin.pas**

Biztonsági mentés: <http://www.mvpp.hu/benzin.pas>

A program forráskódja.

source nevű könyvtárban.

- **benzin.doc**

Biztonsági mentés: <http://www.mvpp.hu/benzin.doc>

Jelen alkalmazás-dokumentáció.

doc nevű könyvtárban

- Megoldási elv:

A program induláskor elindul az **Informacio** eljárás, amely megjeleníti az információs képernyőt, majd egy gombnyomásra vár. Ezt követően a kezdőértékek nevű eljárás hívódik meg, amely a szimuláció hibamentes futásához szükséges kezdőértékeket állítja be. A benzinkút szimuláció lényegében egy iterációba van foglalva, amely ciklusonként vár a felhasználótól egy gombnyomást. A kutak számát (kut változó), vagy a pénztárak számát (penztar változó) növelő gombra automatikusan megtörténik az új sor megnyitása, azonban a csökkentésnél figyelemmel kell lenni arra, nehogy olyan sort tüntessünk el, ahol még áll valaki. Éppen ezért első lépésként ha a felhasználó csökkenteni akar a sorok számán, akkor az utolsó sort meg kell jelölni. Erre a jelölésre szolgál a kutCsokken, és a penztarCsokken, logikai elemekből álló tömb.

Magát a képernyőn látható adatok kiírását a sorkiir nevű eljárás végzi. Ez először lekérdezi a globálisan deklarált info nevű logikai változó értékét, mely alapján eldönti, hogy a sorok adatait, vagy a statisztikai összegzését írja-e ki a képernyőre (kepernyoreKiir nevű eljárás). Amennyiben a sorokat kell, úgy az első lépésként megjeleníti a kutak indexeit (1. kút, 2. kút, stb...) annak megfelelően, hogy a kut változó szerint mennyi van éppen nyitva, valamint ezek alá azt, hogy milyen terméket lehet ott tankolni. Ezt a kiírást segíti a fajta nevű függvény, amely egy paraméterül kapott egész számnak megfelelően az adott értékhez tartozó üzemanyagfajta szöveges leírását (98-as, Diesel, stb...). Ezután következik a kutakhoz tartozó sorokban várakozó autósor kilistázása. Ez egy ciklus alapján történik, melynek a magja annyiszor ismétlődik, amilyen hosszú a programban a megengedett sorhossz. A ciklusmag minden lefutása egy-egy sort tölt meg a kutak oszlopában olyan módon, hogy a kurzort az éppen aktuális helyre lépteti, majd megnézi, hogy az adott kút elemszáma nem-e alacsonyabb, mint az aktuálisan kiírt sor indexe. Ha nem, akkor kiírja a sorban található aktuális információkat, úgy mint tankolási igény, illetve autómárka. (Ez utóbbit a markaNev nevű eljárás segítségével, amely egy paraméterül kapott számértéknek feleltet meg szöveggént egy konkrét márkanevet.) A bejövő, a kimenő sorok, illetve a pénztársorok képernyőre írása pedig ugyanezen elv alapján történik.

A benzinkút életének a mozgását lényegében a sormenedzser nevű eljárás végzi. Az eljárás első dolga, hogy végig nézi, a kútCsokken, illetve a penztarCsokken logikai tömböket, hogy megnézze, van-e olyan kút, vagy pénztár megjelölve törlésre, ahol már nem áll senki. Ha igen, akkor a kutak, vagy a pénztárak számát ennek megfelelően csökkenti egyel.

Ezek után megvizsgáljuk, hogy van-e várakozó autós a bejövetei sorban (*ememSzam* mezője > 0), ha igen, akkor pedig az üzemanyigényének megfelelő indexű kút nyitva van-e.

Ha nyitva van, akkor további vizsgálatok következnek. Ha nincs tele az adott kút előtti sor (az adott kút sorát átadjuk a teli logikai függvénynek, amely megállapítja, hogy az elemszám van-e akkora, mint a programban megengedett maximális sorhosszúság), és bezárás előtt nincs lejárva, valamint a szerencse is közrejátsszik, akkor akkor az adott autóst áthelyezzük a bejövetei sorból az adott kút sorába. Ezt úgy tesszük meg, hogy az *ujAutos* átmeneti rekordba kieszedjük a *beSor* egy elemét a sorbol eljárással, majd ugyanezt az elemet az adott kútsorhoz adjuk a sorba eljárással.

Ha az üzemanyag igénynek megfelelő indexű kút nincs nyitva, akkor viszont az autós a bejövetei sorból rögtön a kimeneti sorba áll. (Számtén sorbol, és sorba eljárásokkal), majd a sikertelen vásárlást a *nemTankolt* globálisan deklarált számváltozó növelésével jegyezzük fel.

Ezek után következik az utcáról érkező új autós adatainak (márka, tankolási igény) generálása három különböző Random függvény meghívásával. Ha a „mázli faktor” is közrejátsszik, akkor ez az új autós rögtön a bejövetei sorba kerül.

A programban ezt követi az a számlálós ciklus, amelyik végig megy mindegyik nyitott kúton, és megállapítja, hogy az ott álló autósnek melyik pénztárba kell állnia. Elsőnek megvizsgálja, hogy adott kútban áll-e egyáltalán olyas valaki, akinek már nincs több tankolási igénye. Ha igen, akkor az illetőt áthelyezi annak a nyitott pénztárnak a sorába, melynél a legkevesebben állnak. (Ennek a sorszámát az *mpenztar* függvény adja vissza egy egyszerű minimumkiválasztással).

Amennyiben viszont olyan ember áll ott, akinek még van tankolási igénye, akkor az ő igényét csökkenti 5-tel (maximum 0-ra), majd a tankolását feljegyzi az anyag nevű számválotózbba.

Utoljára következik a pénztárak sorát kezelő ciklus, amely végvigmegy az összes nyitott pénztáron, és egy Random mázlifaktor eredményének függvényében (amelyik a pénztáros kiszámíthatatlan lassúságát hivatott modellezni) a sor elején álló fizető embert áthelyezi a kimeneti sorba.

Ebből a kimeneti sorból szintén a szerencsétől függ, hogy mikor hagyhatja el az autós a benzinkút területét. Ha ez megtörténik, akkor a tankolt számváltozó növelésével jegyezzük ezt fel.

Fontos része még a programunknak a hibakezeles nevű eljárás. Ez megnézi, hogy a legutóbbi gombnyomás óta sorműveletek bármelyike tért-e vissza hibával (az error nevű logikai változó értéke igazgá vált-e). Ha igen, akkor meg kell jeleníteni erről egy rövid hibaüzenetet, majd ki kell lépni a programból. Ezt úgy tesszük meg, hogy a gomb változó értékét 'K'-ra állítjuk, mintha a felhasználó legutoljára a K (kilépés) billentyűt nyomta volna meg a klaviatúrán.

Tesztelés

Az alkalmazás helyes működésének tesztelését úgy végeztem, hogy figyeltem a beviteli soron megjelenő adatokat, majd figyeltem, hogy ezek a továbbiakban valóban úgy viselkednek-e, mint ahogy elterveztem.

A tesztelésnél először megjelent a bejövő sorban egy Lada, amelyik 17 litert kívánt tankolni a 98-as benzinből.

```
BEJÖVETELI SOR
17 098-as Lada
```

A következő lépésben meg is jelent a 98-as benzint értékesítő kút sorában.

```
1. kút
098-as
17 Lada
```

Ezzel egyidőben egy BMW jelent meg a bejövő sorban, amelyik 11 litert szeretett volna tankolni szintén a 98-asból.

```
BEJÖVETELI SOR
11 098-as BMW
```

Ezután ő meg is jelent az 1-es (98-as benzint áruló) kút sorában a lada mögött.

Ezzel egyidőben a bejövő sorban a 95-ös benzinből 17 lutert kérő lada jelent meg.

```
1. kút
098-as
17 Lada
11 BMW
```

```
BEJÖVETELI SOR
17 095-ös Lada
```

A következő lépésben csupán annyi történt, hogy a 98-as sornál tankoló Lada az előző állapothoz képest a tartályába eresztett az igényéből már 5 liternyi benzint.

```
1. kút
098-as
12 Lada
11 BMW
```

Ezek után újabb 5 liter benzin tankolása következett, miközben a bővő sorban megjelent egy újabb kocsi, egy a 96-os benzinből 3 liternyit igénylő BMW.

```
1. kút
098-as
7 Lada
11 BMW
```

```
BEJÖVETELI SOR
17 095-ös Lada
3 096-os BMW
```

Az idő teltével a 98-as sor elején álló Lada újabb 5 litert eresztett az autójába, miközben a bejövő sorban álló másik Lada megállt a megfelelő (95-ös benzint árusító) sorba.

1. kút 098-as	2. kút 095-ös	BEJÖVETELI SOR
2 Lada 11 BMW	17 Lada	3 096-os BMW

Ezek után a 98-ast tankoló Lada végre elvégezte a dolgát, a maradék 2 litert is autójába folyatta. Eközben a 95-ös sorban a másik Lada elkezdett tankolni, a bejövő sorban várakozó BMW pedig beállt a 96-os sorba.

1. kút 098-as	2. kút 095-ös	3. kút 096-os
0 Lada 11 BMW	12 Lada	3 BMW

A dolgával végzett Lada tulajdonosa eztután beállt a pénztárhoz, amíg a többiek tovább tankoltak.

1. kút 098-as	2. kút 095-ös	3. kút 096-os	1. pénztár
11 BMW	7 Lada	0 BMW	098-as Lada

Nem sokkal utána mögé állt a 96-os kútnál végzett BMW, a többiek tovább tankoltak.

1. kút 098-as	2. kút 095-ös	3. kút 096-os	1. pénztár
6 BMW	2 Lada		098-as Lada 096-os BMW

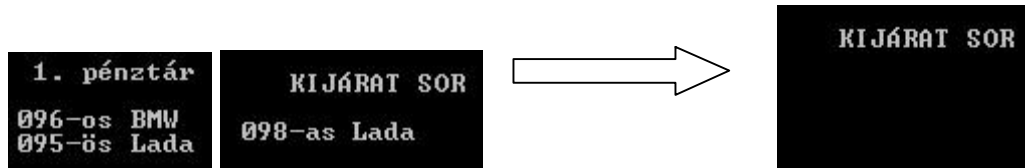
Egy egységnivel később amíg a kutaknál ismét fogyott a benzin, addig a bejövő sorban megjelent egy FIAT, aki 13 litert tankolna a 98-as benzinnél. Az egyetlen nyitott pénztárnál közben a többiek a lassú pénztáros nőre vártak (ott nem történt semmi változás).

1. kút 098-as	2. kút 095-ös	BEJÖVETELI SOR
1 BMW	0 Lada	13 098-as FIAT

Ezután a bejövő FIAT beállt a 98-as sorba, a 95-ös kútnál az imént végzett Lada pedig beállt a pénztárba.

1. kút 098-as	2. kút 095-ös	1. pénztár
1 BMW 13 FIAT		098-as Lada 096-os BMW 095-ös Lada

A következő lépésben a pénztárban végző Lada megjelent a kimeneti sorban, majd egy lépéssel később eltűnt onnan.



Hasonló stílusban hosszasan elemeztem a program működését különböző adatokkal, és miután úgy sem tapasztaltam rendellenességet, hogy különböző módon be is avatkoztam megnet közben a program paramétereinek alakulásába, úgy láttam, hogy kimertően sikerült tesztelnek a benzinkút szimulációm megfelelő működését.

ALGORITMUS:

Megjegyzés: a szokásos sémától eltérően az algoritmusban azért részletezem a képernyőre való írásért felelős utasításokat, mert a technikai megoldás fontos része volt az összes sor

program benzin

```
konstans
    maxKut = 4
    maxPenztar = 3
    maxSorHossz = 7
    maxAutos = 30

típus
    autos = rekord
        igény : egész {mennyit akar tankolni}
        fajta : egész {milyen benzint akar tankolni}
        marka : egész {milyen márkájú az autója}
vége

sor = rekord
    elemek : tömb [1..maxSorHossz] mint autos
    elemSzam : egész
vége

változók:
    penztar, kut, i, k : egész           pénztárak, kutak száma, ciklusváltozók
    kutak : tömb [1..maxKut] mint egész   az egyes kutak fajtái (választék)
    kutSor : tömb [1..maxKut] mint sor     a kutak sorai
    penztarSor : tömb [1..maxPenztar] mint sor   pénztárak sorai
    beSor, kiSor : sor                   a benzinkút be- és kimeneti sora
    ido, anyag, nemTankolt, tankolt : hosszú egész   statisztikai információk
    gomb : char
    ujAutos : autos                     sorelemek átmeneti mozgatásához
    error, info : logikai
    kutCsokken : tömb [1..maxKut] mint logikai   megnyomtuk a kút törlést
    penztarCsokken : tömb [1..maxPenztar] mint logikai   pénztár törlés
```

Eljárás informacio

```
ki: <általános leírás a programról>
gomb: <billentyűzetről beolvasott érték>
Eljárás vége
```

Eljárás kezdőérték

```
penztar := 1
kut := 3
ciklus i:=1-től maxKut-ig
    kutak[i] := i
```



```
        ciklus vége
        info:= hamis
Eljárás vége
```

Eljárás sorba(*ezt: autos; változó helyiSor : sor; változó error : logikai*)

```
    Ha helyiSor.elemszam = maxSorHossz akkor
        hiba:=igaz
    különben
        helyiSor.elemSzam:=helyiSor.elemSzam+1
        helyiSor.elemek[helyiSor.elemSzam]:=ezt
    feltétel vége
Eljárás Vége
```

Eljárás sorbol(*vált. helyiSor:sor; vált. ide:autos; vált. error: logikai*)

```
változó j: egész
    Ha helyiSor.elemSzam < 1 akkor
        hiba := igaz
    különben
        a kiemelt elem a sort ábrázoló tömb első eleme
        ide:= helyiSor.elemek[1]
        ciklus j:=1-től (helyiSor.elemSzam-1)-ig
            a többi elemek előre csúsztatjuk,
            az eddigi második áll az első helyen stb..
            helyiSor.elemek[j]:=helyiSor.elemek[j+1]
        ciklus vége
        helyiSor.elemSzam:=helyiSor.elemSzam-1
    elágazás vége
Eljárás vége
```

Függvény fajta (*sorszam : egész*) : szöveg

```
    Ha sorszam értéke
        1: fajta := '098-as'
        2: fajta := '095-ös'
        3: fajta := '096-os'
        4: fajta := 'Diesel'
    különben
        fajta := <semmi>
    elágazás vége
Függvény vége
```

Függvény markanev (*sorszam : egész*) : szöveg

```
    Ha sorszam értéke
        1: markanev := 'Opel'
        2: markanev := 'FIAT'
        3: markanev := 'BMW '
        4: markanev := 'Lada'
        5: markanev := 'Jeep'
    különben
        markanev := <semmi>
    elágazás vége
Függvény vége
```

Eljárás képernyőreKiir

változó *kutvar*, *penzváltozó*: egész

```
képernyőtörlés
ki ('Eltelt időegység:      ', ido)
ki ('Nyitott kutak:        ', kut, ' darab')
ki ('Nyitott pénztárok:    ', penztar)
ki ('Összes tankolt benzin: ', anyag, ' liter')
ki ('Hiány miatt nem tankolt: ', nemTankolt)
ki ('Fizetett, és távozott: ', tankolt)
ki ('Bejutásra várakozik:  ', beSor.elemSzam)
kutvar:=0
ciklus i:=1-től kut-ig
    kutvar:=kutvar+kutSor[i].elemSzam
ciklus vége
ki ('Kútnál vár, vagy tankol: ', kutVar)
penzvar:=0
ciklus i:=1-től penztar-ig
    penzvar:=penzvar+penztarSor[i].elemszam
ciklus vége
ki ('Pénztárnál várakozik:   ', penzvar)
Eljárás vége
```

Eljárás fajlbaKiir

Változó *kutvar*, *penzváltozó*: egész

fajl : szöveg

```
összeilleszt(fajl,'stat.txt')
felülír(fajl)
képernyőtörlés
ki (fajl, 'Eltelt időegység:      ', ido)
ki (fajl, 'Nyitott kutak:        ', kut, ' darab')
ki (fajl, 'Nyitott pénztárok:    ', penztar)
ki (fajl, 'Összes tankolt benzin: ', anyag, ' liter')
ki (fajl, 'Hiány miatt nem tankolt: ', nemTankolt)
ki (fajl, 'Fizetett, és távozott: ', tankolt)
ki (fajl, 'Bejutásra várakozik:  ', beSor.elemSzam)
kutvar:=0
ciklus i:=1-től kut-ig
    kutvar:=kutvar+kutSor[i].elemSzam
ciklus vége
ki (fajl, 'Kútnál vár, vagy tankol: ', kutVar)
penzvar:=0
ciklus i:=1-től penztar-ig
    penzvar:=penzvar+penztarSor[i].elemszam
ciklus vége
ki (fajl, 'Pénztárnál várakozik:   ', penzvar)
gomb:=<billentyűzetről beolvasott érték>
eljárás vége
```

Eljárás sorkiir

Ha nem(*info*) akkor

ha nem az információs képernyőt kell mutatni, hanem a sorokat

```
képernyőtörlés
ciklus i:=1-től kut-ig
    1.kút, 2. kút, stb
    kurzormozgatásXY ((i*15)-13, 2)
    ki: (i<2 karakterhelyre igazítva>,'. kút')
```

```
Ha kutCsokken[i] akkor
    kurzormozgatásXY((i*15-13),1) ki: ('LEZÁRVA')
    elágazás vége
ciklus vége
ciklus i:=1-től kut-ig
    95-ös, ólommentes, stb
    kurzormozgatásXY ((i*15)-13, 3)
    ki: (fajta(kutak[i])<7 karakterhelyre igazítva>)
ciklus vége

ciklus k:=1-től maxSorHossz-ig
    ciklus i:=1-től kut-ig
        Ha kutSor[i].elemSzam >= k akkor
            3 (liter), BMW, stb...
            kurzormozgatásXY ((i*15-16),4+k)
            ki:(kutSor[i].elemek[k].igeny<5 karakterhelyre igazít>)
            kurzormozgatásXY ((i*15-8), (4+k))
            ki:(markaNev(kutSor[i].elemek[k].marka)<5 karakterhelyre ig.>)
            elágazás vége
        ciklus vége
    ciklus vége

ciklus i:=1-től penztar-ig
    1. pénztár, 2. pénztár, stb...
    kurzormozgatásXY ((i*15-13), 13)
    ki: (i<2 karakterhelyre igazítva>,'. pénztár', ''<9 helyre ig.>)
    Ha penztarCsokken[i] akkor
        kurzormozgatásXY((i*15-13),12) ki ('LEZÁRVA')
        elágazás vége
    ciklus vége

ciklus k:=1-től maxSorHossz-ig
    ciklus i:=1-től penztar-ig
        Ha penztarSor[i].elemSzam >= k akkor
            Diesen BMW stb
            ki: (fajta(penztarSor[i].elemek[k].fajta)<7 helyre ig.>)
            ki: (markaNev(penztarSor[i].elemek[k].marka)<5 helyre ig.>)
            elágazás vége
        ciklus vége
    ciklus vége

kurzormozgatásXY (60,4) ki: ('BEJÖVETELI SOR')
ciklus k:=1-től maxSorHossz-ig
    ha az adott sorban az elemszám alapján még van
    aktuális indexű elem, akkor írjuk ki az adatait
    Ha beSor.elemszam >= k akkor
        kurzormozgatásXY (57,5+k)
        ki: (beSor.elemek[k].igeny<5 karakterhelyre igazítva>)
        ki: (fajta(beSor.elemek[k].fajta)<7 helyre igazítva>)
        ki: (markaNev(beSor.elemek[k].marka)<5 helyre igazítva>)
        elágazás vége
    ciklus vége

kurzormozgatásXY (64,7+maxSorHossz) ki: ('KIJÁRAT SOR')
```

```
ciklus k:=1-től maxSorHossz-ig
  Ha kiSor.elemszam >= k akkor
    kurzormozgatásXY (60,8+maxSorHossz+k)
    ki:  (fajta(kiSor.elemek[k].fajta)<7 helyre igazítva>)
    ki:  (markaNev(kiSor.elemek[k].marka)<5 helyre igazítva>)
  elágazás vége
ciklus vége
különben
```

ha nem a sorokat kell mutatni, hanem az információk képernyőt

```
kepernyoreKiir
elágazás vége
```

legalulra kiírjuk, melyik gombra mi történjen

```
kurzormozgatásXY(18,23)
ki:  ('Q - '<4 karakterhelyre igazítva>, 'több benzinkút      ')
ki:  ('W - '<4 karakterhelyre igazítva>, 'több pénztár       ')
  ki:  ('X - '<4 karakterhelyre igazítva>, 'LÉPTETÉS')
kurzormozgatásXY(18,24)
ki:  ('A - '<4 karakterhelyre igazítva>, 'kevesebb penzinkút  ')
ki:  ('S - '<4 karakterhelyre igazítva>, 'kevesebb pénztár    ')
  ki:  ('K - '<4 karakterhelyre igazítva>, 'KILÉPÉS')
kurzormozgatásXY(1,23)
ki:  ('I - '<4 karakterhelyre igazítva>, 'info ki/be')
kurzormozgatásXY(1,24)
ki:  ('M - '<4 karakterhelyre igazítva>, 'stat mentés')
vége
```

Függvény teli(helyiSor: sor): logikai
Ha helyiSor.elemSzam = maxSorHossz akkor
 teli:=igaz
különben
 teli:=hamis
elágazás vége
Függvény vége

Függvény mPenztar : egész
változój, legkisebb, minindex: egész

minimum kiválasztással a Függvény visszaadja
anmak az aktuálisan nyitott pénztárnak az
indexét, amelyiknél a legkevsebben állnak

```
legkisebb:=6
minindex:=1
ciklus j:=1-től penztar-ig

  Ha legkisebb>penztarSor[j].elemszam akkor
    legkisebb := penztarSor[j].elemszam
    minindex := j
  elágazás vége
ciklus vége
mPenztar := minindex
vége
```

Eljárás sormenedzser

```
ciklus i:=1-től kut-ig
  Ha kutCsokken[i] és (kutSor[i].elemszam<1) akkor
    ha az adott indexű kút törlendő, és nem áll benne autós
      akkor töröljük
        kutCsokken[kut]:=hamis
        kut := kut-1
    elágazás vége
ciklus vége
```

Ha adott pénztár törlendő, és nem állnak benne, akkor töröljük

```
ciklusi:=1-től penztar-ig
  Ha penztarCsokken[i] és (penztarSor[i].elemszam<1) akkor
    penztarCsokken[kut]:=hamis
    penztar := penztar-1
  elágazás vége
ciklus vége
```

Ha (*beSor.elemszam*>0) akkor

ha állnak a bejövetele sorban

i:= *beSor.elemek[1].fajta*

i segédváltozó legyen egyenlő a bejövetele sor elején lévő autós benzín igényét jelző indexszámmal. Ezt a számot a -tölvábbiakban arra használjuk, hogy segítségével meghatározzuk az adott igénynek megfelelő kút indexék -ez a kettő ugyanis a "kezdőterek" eljárás alapján ugyanaz

Ha *i* <= *kut* akkor

ha van nyitva az igénynek megfelelő kút,

vagyis ha a keresett terméket adó kút indexe nem

nagyobb, mint a nyitott kutak száma

i:= (*beSor.elemek[1].fajta*)

Ha nem(*teli(kutSor[i])*) és (Véletlen(2)=0)

és nem(*kutCsokken[i]*) akkor

ha nincs tele az igénynek megfelelő kút,

a szerencse is közre játszik, valamint zárás céljából

nincs lezárva a kút

akkor az autós a megfelelő kút sorába léphet

sorbol (*beSor*, *ujAutos*, *error*)

sorba (*ujAutos*, *kutSor[i]*, *error*)

hogy ne rögtön a csökkent tankolási értéket lássuk

a későbbiekben, ideiglenes megnöveljük 5-tel az

igényt, hogy a sorban a tankolást végző algoritmus

automatizmusa ellenére is a valós igényt lássuk

elsőként

értéknövelés(*kutSor[i].elemek[1].igeny*,5)

elágazás vége

különben

ha nincs az autós igényét kiszolgáló kút nyitva, akkor

automatikusán a kimeneti sorba áll be, a kutakhoz, pénztárakhoz nem

sorbol (*beSor*, *ujAutos*, *error*)

sorba (*ujAutos*, *kiSor*, *error*)

nemTankolt := *nemTankolt*+1

elágazás vége

elágazás vége

az új érkező autós adatainak a generálása

```
ujAutos.igeny := (Véletlen(200) div 10)+1  
ujAutos.fajta := (Véletlen(40) div 10)+1  
ujAutos.marka := (Véletlen(40) div 10)+1
```

ha a véletlen úgy hozza, akkor ő éppen beáll a bementi sorba

```
Ha (Véletlen(2))=0 akkor sorba(ujAutos, beSor, error)
```

Ciklus $i:=1$ -től kut -ig

```
Ha ( $kutSor[i].elemek[1].igeny = 0$ ) és ( $kutSor[i].elemszam>0$ )  
akkor
```

ha adott kútnál áll ember, és már eleget tankolt

```
Ha nem( $penztarCsokken[mPenztar]$ ) akkor
```

valamint a kiszemelt pénztár nincs lezárva

akkor az autós a kúttól átáll a pénztárhoz

```
sorbol ( $kutSor[i]$ ,  $ujAutos$ ,  $error$ )
```

```
sorba ( $ujAutos$ ,  $penztarSor[mPenztar]$ ,  $error$ )
```

elágazás vége

különben

ha az adott kútnál álló embernek még kell tankolnia

```
Ha  $kutSor[i].elemek[1].igeny > 5$  akkor
```

ha az adott kútnál álló ember többet akar tankolni 5 liternél

```
 $kutSor[i].elemek[1].igeny := kutSor[i].elemek[1].igeny - 5$ 
```

```
 $anyag:=anyag+5$ 
```

akkor ebben a fázisban tankol ötöt, összefogyasztás

ennyivel is növekszik

különben

```
 $anyag:=anyag+kutSor[i].elemek[1].igeny$ 
```

```
 $kutSor[i].elemek[1].igeny :=0$ 
```

különben a maradék igényét letankolja

elágazás vége

elágazás vége

ciklus vége

ciklus $i:=1$ -től $penztar$ -ig

```
Ha ( $i=Véletlen(3)$ ) és ( $penztarSor[i].elemSzam>0$ ) akkor
```

ha egy adott pénztárnál állnak emberek, és a pénztáros

munkaidejét helyettesítő mázlifaktor és bejön

akkor az autós az adott pénztártól átáll a kimeneti sorba

```
sorbol ( $penztarSor[i]$ ,  $ujAutos$ ,  $error$ )
```

```
sorba ( $ujAutos$ ,  $kiSor$ ,  $error$ )
```

elágazás vége

ciklus vége

```
Ha ( $Véletlen(2)=0$ ) és ( $kiSor.elemszam>0$ ) akkor
```

ha a kimeneti sorban áll ember, és a forgalmi viszonyokat

helyettesítő mázlifaktor is bejön

akkor az illető kilép a kimeneti sorból,

a sikeresen tankoló emberek száma egyel nő

```
sorbol ( $kiSor$ ,  $ujAutos$ ,  $error$ )
```

```
 $tankolt:=tankolt+1$ 
```

Elágazás vége

vége

Eljárás **hibakezeles**

```
Ha  $error = igaz$  akkor
```

```
 $ki: <hibaüzenet arról, hogy valamelyik sor túlcsordult>$ 
```

```
 $gomb:='K'$ 
```

elágazás vége

Eljárás vége

Főprogram eleje

képernyőtörlés

informacio

kezdoadatok

ismételd

error:= hamis

sorkiir

gomb := <billentyűzetről beolvasott érték>

ido:=*ido*+1

új pénztárat, illetve kutat rögtön nyitunk, de
de a csökkentés előtt egyelőre megjelöljük az adott
sort, hogy azután megvizsgálhassuk csökkentés előtt,
hogy áll-e a sorban valaki:

amennyiben *gomb* értéke

'Q', 'q': Ha *kut*<*maxKut* akkor *kut*:=*kut*+1

'A', 'a': Ha *kut*>1 akkor *kutCsokken*[*kut*]:=igaz

'W', 'w': Ha *penztar*<*maxPenztar* akkor *penztar*:=*penztar*+1

'S', 's': Ha *penztar*>1 akkor

penztarCsokken[*penztar*]:=igaz

'i', 'I': *info*:= nem(*info*)

'm', 'M': *fajlbaKiir*

vége

sormenedzser

hibakezeles

míg nem (*gomb*='K') vagy (*gomb*='k')

end.

A továbbfejlesztés lehetőségei:

A továbbfejlesztés leginkább a grafikai megjelenítés terén rejt magában széles lehetőségeket. Elég sokat javítana esetlegesen a mostani felhasználói felületen, ha a képernyő egyes részeri vastag határolóvonalakkal lennének egymástól elválasztva. Szintén sokat dobna a megjelenítésen, ha a pascal graph unit-jának használatával megvalósítanánk egyrajta szemléletes grafikus ábtázolást. (Például apró, mozgó téglalapot jelképeznék az autókat, amelyek mozgását a sorokon belül, illetve a sorok közt egyszerűbb animációval lehetne igazán élvezetessé tenni).

A program statisztikai kimutatások készítésére sokkal jobban lenne használható, ha a jelenleginél szélesebb skálán lehetne variálni a szimuláció egyes paramétereit, például ha meg lehetne egyenként határozni, hogy melyik benzinkút milyen üzemanyagot használhasson. Így ha úgy itélnénk, hogy aránytalanul sok a 98-as benzint tankolók száma, és ezt már nem bírja az egy darab kút, akkor valamelyik társát át tudnánk a szimulációban állítani arra, hogy a továbbiakban kettejük között oszoljon meg ezen benzin árusításának a feladata.

FORRÁSKÓD

{Készítette: Kiss Endre Farkas
Informatikatanár-Matematikatanár
Második szemeszter, második beadandó
Egy benzinkút szimulációja sorokkal}

```
program benzin;  
uses crt;  
const  
  maxKut = 4;  
  maxPenztar = 3;  
  maxSorHossz = 7;  
  maxAutos = 30;  
  
type  
  autos = record  
    igény : byte; {mennyit akar tankolni}  
    fajta : byte; {milyen benzint akar tankolni}  
    marka : byte; {milyen márkájú az autója}  
  end;  
  
  sor = record  
    elemek : array [1..maxSorHossz] of autos;  
    elemSzam : byte;  
  end;  
  
var penztar, kut, i, k : byte;           {pénztárak, kutak száma, ciklusváltozók}  
    kutak : array [1..maxKut] of byte;   {az egyes kutak fajtái (választék)}  
    kutSor : array [1..maxKut] of sor;    {a kutak sorai}  
    penztarSor : array [1..maxPenztar] of sor; {pénztárak sorai}  
    beSor, kiSor : sor;                  {a benzinkút be -és kimeneti sora}  
    ido, anyag, nemTankolt, tankolt : integer; {információk}  
    gomb : char;  
    ujAutos : autos;                     {sorelemek átmeneti mozgathatóságához}  
    error, info : boolean;  
    kutCsokken : array [1..maxKut] of boolean; {megnyomtuk a kút törlést}  
    penztarCsokken : array [1..maxPenztar] of boolean; {pénztár törlés}
```

```
procedure informacio;  
begin  
  clrscr;  
  writeln ('Kiss Endre Farkas':30);  
  writeln ('Info-Matek 2-ik szemeszter, 2-ik beadandó.');
```

writeln;
writeln ('A program feladata az, hogy hogy számítógépen modellezze');
writeln ('egy benzinkút működését. Adott egy fő sor, ahol az összes');
writeln ('tankolni vágyó autós érkezik. Ezek aszerint, hogy milyen');
writeln ('jellegű üzemanyagra van szükségük, besorakoznak a megfelelő');
writeln ('terméket adó kúthoz, és várnak arra, hogy az előttük álló');
writeln ('végezzen a tankolással. Miután tankoltak, a pénztárhoz');
writeln ('fáradnak, ahol abba a sorba állnak be, ahol a legkevesebben');
writeln ('várnak. Fizetés után pedig mindenki egy közös sorban hagyja');
writeln ('el a benzinkút területét.');

writeln ('Előfordulhat persze, hogy a keresett üzemanyagot adó');
writeln ('kút éppen nincs megnyitva. Ha ez a helyzet, akkor az autós');
writeln ('a bejövő sorból vásárlás nélkül meg tovább a kimenő soron.');

writeln ('Menet közben lehetőség van változtatni a pénztárak, illetve');
writeln ('a nyitva lévő kutak számát a megfelelő billentyű lenyomásával.');

writeln ('Ehhez a szimulációs képernyő alsó részében látható segítség.');

writeln ('A program menet közben összegzi a történéseket, ezt az összeg-');

writeln ('zést szimuláció közben az I betűvel lehet megtekinteni, az');
writeln ('információs részből visszalépni szintén így lehet.');

writeln ('A szimulációt léptetni az X gombbal lehet menet közben.');

writeln;
writeln ('<ENTER>':25);
gomb:=readkey;
end;

```
procedure kezdortek;  
begin  
  penztar := 1;  
  kut := 3;  
  for i:=1 to maxKut do  
    begin  
      kutak[i]:=i;  
    end;  
  info:= false;  
end;
```

```
procedure sorba(ezt: autos; var helyiSor : sor; var hiba : boolean);
begin
  if helyiSor.elemszam = maxSorHossz then
  begin
    hiba:=true;
  end else
  begin
    helyiSor.elemSzam:=helyiSor.elemSzam+1;
    helyiSor.elemek[helyiSor.elemSzam]:=ezt;
  end;
end;
```

```
procedure sorbol(var helyiSor : sor; var ide: autos; var hiba: boolean);
var j: byte;
begin
  if helyiSor.elemSzam < 1 then
  begin
    hiba := true;
  end else
  begin
    {a kiemelt elem a sort árbázoó tömb első elemee}
    ide:= helyiSor.elemek[1];
    for j:=1 to (helyiSor.elemSzam-1) do
    begin
      {a többi elemek előre csúsztatjuk,
      az eddigi második áll az első helyen stb..}
      helyiSor.elemek[j]:=helyiSor.elemek[j+1];
    end;
    helyiSor.elemSzam:=helyiSor.elemSzam-1;
  end;
end;
```

```
function fajta (sorszam : byte) : string;
begin
  case sorszam of
    1: fajta := '098-as';
    2: fajta := '095-ös';
    3: fajta := '096-os';
    4: fajta := 'Diesel';
    else
      fajta := "";
  end;
end;
```

```
function markanev (sorszam : byte) : string;
begin

    case sorszam of
        1: markanev := 'Opel';
        2: markanev := 'FIAT';
        3: markanev := 'BMW ';
        4: markanev := 'Lada';
        5: markanev := 'Jeep';
        else
            markanev := '';
    end;

end;

procedure kepernyoreKiir;
var kutvar, penzvar : byte;
begin
    clrscr;
    writeln ('Eltelt időegység:      ', ido);
    writeln ('Nyitott kutak:              ', kut, ' darab');
    writeln ('Nyitott pénztárak:            ', penztar);
    writeln ('Összes tankolt benzin:        ', anyag, ' liter');
    writeln ('Hiány miatt nem tankolt:      ', nemTankolt);
    writeln ('Fizetett, és távozott:        ', tankolt);
    writeln;
    writeln('      - VÁRAKOZÓK -');
    writeln;
    writeln ('Bejutásra várakozik:         ', beSor.elemSzam);
    kutvar:=0;
    for i:=1 to kut do
    begin
        kutvar:=kutvar+kutSor[i].elemSzam;
    end;
    writeln ('Kútnál vár, vagy tankol: ', kutVar);
    penzvar:=0;
    for i:=1 to penztar do
    begin
        penzvar:=penzvar+penztarSor[i].elemszam;
    end;
    writeln ('Pénztárnál várakozik:      ', penzvar);
end;
```

```
procedure fajlbaKiir;
var kutvar, penzvar : byte;
fajl : text;
begin
  assign(fajl,'stat.txt');
  rewrite(fajl);
  clrscr;
  writeln (fajl, 'Eltelt időegység:      ', ido);
  writeln (fajl, 'Nyitott kutak:          ', kut, ' darab');
  writeln (fajl, 'Nyitott pénztárak:      ', penztar);
  writeln (fajl, 'Összes tankolt benzin:  ', anyag, ' liter');
  writeln (fajl, 'Hiány miatt nem tankolt: ', nemTankolt);
  writeln (fajl, 'Fizetett, és távozott:  ', tankolt);
  writeln (fajl, '');
  writeln(fajl, '      - VÁRAKOZÓK -');
  writeln;
  writeln (fajl, 'Bejutásra várakozik:   ', beSor.elemSzam);
  kutvar:=0;
  for i:=1 to kut do
  begin
    kutvar:=kutvar+kutSor[i].elemSzam;
  end;
  writeln (fajl, 'Kútnál vár, vagy tankol: ', kutVar);
  penzvar:=0;
  for i:=1 to penztar do
  begin
    penzvar:=penzvar+penztarSor[i].elemszam;
  end;
  writeln (fajl, 'Pénztárnál várakozik:   ', penzvar);
  clrscr;
  gotoXY(12,12); writeln('Statistika sikeresen lemezre mentve stat.txt néven. ');
  gotoXY(35,14); writeln('<ENTER>');
  gomb:=readkey;
end;
```

```
procedure sorkiir;
begin
  if not(info) then
  {ha nem az információs képernyőt kell mutatni, hanem a sorokat}
  begin
    clrscr;
    for i:=1 to kut do
    begin
      {1.kút, 2. kút, stb}
      gotoXY ((i*15)-13, 2); write (i:2, '. kút');
      if kutCsokken[i] then
      begin
        gotoXY((i*15-13),1); write ('LEZÁRVA')
      end;
    end;
  end;
```

```
for i:=1 to kut do
begin
    {95-ös, ólommentes, stb}
    gotoXY ((i*15)-13, 3);write (fajta(kutak[i]):7);
end;
writeln; writeln;

for k:=1 to maxSorHossz do
begin

    for i:=1 to kut do
    begin
        if kutSor[i].elemSzam >= k then
        begin
            {3 (liter), BMW, stb...}
            gotoXY ((i*15-16),4+k);
            write (kutSor[i].elemek[k].igeny:5);
            gotoXY ((i*15-8),(4+k));
            write (markaNev(kutSor[i].elemek[k].marka):5);
        end;
    end;
    writeln;
end;

writeln;

for i:=1 to penztar do
begin
    {1. pénztár, 2. pénztár, stb...}
    gotoXY ((i*15-13), 13); write (i:2, '. pénztár', ":9);
    if penztarCsokken[i] then
    begin
        gotoXY((i*15-13),12); writeln ('LEZÁRVA');
    end;
end;

writeln; writeln;

for k:=1 to maxSorHossz do
begin
    for i:=1 to penztar do
    begin
        if penztarSor[i].elemSzam >= k then
        begin
            {Diesen BMW stb}
            write (fajta(penztarSor[i].elemek[k].fajta):7);
            write (markaNev(penztarSor[i].elemek[k].marka):5);
        end;
    end;
end;
```

```
end;

gotoXY (60,4); write ('BEJÖVETELI SOR');
for k:=1 to maxSorHossz do
begin
  {ha az adott sorban az elemszám alapján még van
  aktuális indexű elem, akkor írjuk ki az adatait}
  if beSor.elemszam >= k then
  begin
    gotoXY (57,5+k);
    write (beSor.elemek[k].igeny:5);
    write (fajta(beSor.elemek[k].fajta):7);
    write (markaNev(beSor.elemek[k].marka):5);
  end;
end;

gotoXY (64,7+maxSorHossz); write ('KIJÁRAT SOR');
for k:=1 to maxSorHossz do
begin
  if kiSor.elemszam >= k then
  begin
    gotoXY (60,8+maxSorHossz+k);
    write (fajta(kiSor.elemek[k].fajta):7);
    write (markaNev(kiSor.elemek[k].marka):5);
  end;
end;
end else begin
{ha nem a sorokat kell mutatni, hanem az információs képernyőt}
kepernyoreKiir;
end;
{legalulra kiírjuk, melyik gombra mi történjen}
gotoXY(18,23);
write ('Q - ':4, 'több benzinkút ');
write ('W - ':4, 'több pénztár '); write ('X - ':4, 'LÉPTETÉS');
gotoXY(18,24);
write ('A - ':4, 'kevesebb penzinkút ');
write ('S - ':4, 'kevesebb pénztár '); write ('K - ':4, 'KILÉPÉS');
gotoXY(1,23);
write ('I - ':4, 'info ki/be');
gotoXY(1,24);
write ('M - ':4, 'stat mentés');

end;
```

```
function teli(helyiSor: sor): boolean;  
begin  
  if helyiSor.elemSzam = maxSorHossz then  
    begin  
      teli:=true;  
    end else  
    begin  
      teli:=false;  
    end;  
end;
```

```
function mPenztar : byte;  
var j, legkisebb, minindex: byte;  
begin  
  {minimum kiválasztással a függvény visszaadja  
  annak az aktuálisan nyitott pénztárnak az  
  indexét, amelyiknél a legkevsebben állnak}  
  legkisebb:=6;  
  minindex:=1;  
  for j:=1 to penztar do  
    begin  
      if legkisebb>penztarSor[j].elemszam then  
        begin  
          legkisebb := penztarSor[j].elemszam;  
          minindex := j;  
        end;  
    end;  
  mPenztar := minindex;  
end;
```

```
procedure sormenedzser;
```

```
begin  
  for i:=1 to kut do  
    begin  
      if kutCsokken[i] and (kutSor[i].elemszam<1) then  
        {ha az adott indexű kút törlendő, és nem áll benne autós}  
        begin  
          {akkor töröljük}  
          kutCsokken[kut]:=false;  
          kut := kut-1;  
        end;  
    end;  
  end;
```



```
{Ha adott pénztár törlendő, és nem állnak benne, akkor töröljük}
for i:=1 to penztar do
begin
  if penztarCsokken[i] and (penztarSor[i].elemszam<1) then
  begin
    penztarCsokken[kut]:=false;
    penztar := penztar-1;
  end;
end;

Randomize;

if (beSor.elemszam>0) then
{ha állnak a bejövetei sorban}
begin
  i:= beSor.elemek[1].fajta;
  {i segédváltozó legyen egyenlő a bejövetei sor elején lévő
  autós benzin igényét jelző indexszámmal. Ezt a számot a
  továbbiakban arra használjuk, hogy segítségével meghatározzuk
  az adott igénynek megfelelő kút indexék -ez a kettő ugyanis
  a "kezdőérték" eljárás alapján ugyanaz}
  if i <= kut then
  {ha van nyitva az igénynek megfelelő kút,
  vagyis ha a keresett terméket adó kút indexe nem
  nagyobb, mint a nyitott kutak száma}
  begin
    i:= (beSor.elemek[1].fajta);
    if not(teli(kutSor[i])) and (Random(2)=0) and not(kutCsokken[i]) then
    {ha nincs tele az igénynek megfelelő kút,
    a szerencse is közre játszik, valamint zárás céljából
    nincs lezárva a kút}
    begin
      {akkor az autós a megfelelő kút sorába léphet}
      sorbol (beSor, ujAutos, error);
      sorba (ujAutos, kutSor[i], error);
      {hogya ne rögtön a csökkent tankolási értéket lássuk
      a későbbiekben, ideiglenes megnöveljük 5-tel az igényt,
      hogy a sorban a tankolást végző algoritmus automatizmusa
      ellenére is a valós igényt lássuk elsőként}
      inc(kutSor[i].elemek[1].igeny,5);
    end;
  end else begin
  {ha nincs az autós igényét kiszolgáló kút nyitva, akkor
  automatikusan a kimeneti sorba áll be, a kutakhoz, pénztárakhoz
  nem}
  sorbol (beSor, ujAutos, error);
  sorba (ujAutos, kiSor, error);
  nemTankolt := nemTankolt+1;
end;
end;
```

```
{az új érkező autós adatainak a generálása}
ujAutos.igeny := (Random(200) div 10)+1;
ujAutos.fajta := (Random(40) div 10)+1;
ujAutos.marka := (Random(40) div 10)+1;

{ha a véletlen úgy hozza, akkor ő éppen beáll a bementi sorba}
if (Random(2))=0 then sorba(ujAutos, beSor, error);

for i:=1 to kut do
begin
  if (kutSor[i].elemek[1].igeny = 0) and (kutSor[i].elemszam>0) then
  begin
    {ha adott kútnál áll ember, és már eleget tankolt}
    if not(penztarCsokken[mPenztar]) then
      {valamint a kiszemelt pénztár nincsen lezárva}
      begin
        {akkor az autós a kúttól átáll a pénztárhoz}
        sorbol (kutSor[i], ujAutos, error);
        sorba (ujAutos, penztarSor[mPenztar], error);
      end;
    end else begin
      {ha az adott kútnál álló embernek még kell tankolnia}
      if kutSor[i].elemek[1].igeny > 5 then
        {ha az adott kútnál álló ember többet akar tankolni 5 liternél}
        begin
          kutSor[i].elemek[1].igeny := kutSor[i].elemek[1].igeny - 5;
          anyag:=anyag+5;
          {akkor ebben a fázisban tankol ötöt, összefogyasztás
          ennyivel is növekszik}
        end else
        begin
          anyag:=anyag+kutSor[i].elemek[1].igeny;
          kutSor[i].elemek[1].igeny :=0;
          {különben a maradék igényét letankolja}
        end;
      end;
    end;
  end;
end;
for i:=1 to penztar do
begin
  if (i=Random(3)) and (penztarSor[i].elemSzam>0) then
    {ha egy adott pénztárnál állnak emberek, és a pénztáros
    munkaidejét helyettesítő mázlifaktor és bejön}
    begin
      {akkor az autós az adott pénztártól átáll
      a kimeneti sorba}
      sorbol (penztarSor[i], ujAutos, error);
      sorba (ujAutos, kiSor, error);
    end;
  end;
end;
```

```
if (Random(2)=0) and (kiSor.elemszam>0) then
  {ha a kimeneti sorban áll ember, és a forgalmi viszonyokat
  helyettesítő mázlifaktor is bejön}
  begin
    {akkor az illető kilép a kimeneti sorból,
    a sikeresen tankoló emberek száma egyel nő}
    sorbol (kiSor, ujAutos, error);
    tankolt:=tankolt+1;
  end;
end;
```

```
procedure hibakezeles;
begin
  if error = true then
    begin
      clrscr;
      gotoXY(12,12);
      writeln('HIBA! Valamelyik sor futásidőben túlsordult!');
      gotoXY(9,13);
      writeln('Futassa a programot több kúttal, vagy pénztárral!');
      gotoXY(30,15); writeln('<ENTER>');
      gomb:=readkey;
      gomb:='K'
    end;
end;
```

```
begin
  clrscr;
  informacio;
  kezdoertek;
  repeat
    error:= false;
    sorkiir;
    gomb := readkey;
    ido:=ido+1;
```

case gomb of

{új pénztárat, illetve kutat rögtön nyitunk, de
de a csökkentés előtt egyelőre megjelöljük az adott
sort, hogy azután megvizsgálhassuk csökkentés előtt,
hogy áll-e a sorban valaki}

'Q', 'q': if kut<maxKut then kut:=kut+1;
'A', 'a': if kut>1 then kutCsokken[kut]:=true;
'W', 'w': if penztar<maxPenztar then penztar:=penztar+1;
'S', 's': if penztar>1 then penztarCsokken[penztar]:=true;
'i', 'I': info:= not(info);
'm', 'M': fajlbaKiir;

end;

sormenedzser;

hibakezeles;

until (gomb='K') or (gomb='k');

end.

2004-05-21