Az előző részben kitűzött feladatok megoldása

Feladatok: 1. Az alábbi feladatok legfeljebb 10×10-es méretű, egész típusú számokat tartalmazó tömbre vonatkoznak.

a) Szubrutin készítendő, amely m soros, n oszlopos mátrixban ($2 \leq m \leq 10$, és $2 \leq n \leq 10$) fölcseréli az i-edik sor elemeit a j-edik sor elemeivel ahol $i \neq j$, $1 \leq i \leq m$ és $1 \leq j \leq n$. A szubrutin átveszi a mátrixot, az i és j számokat és az átvett mátrixot módosítva adja vissza. (Az eredeti szövegben n és m jelentése fel volt cserélve.)

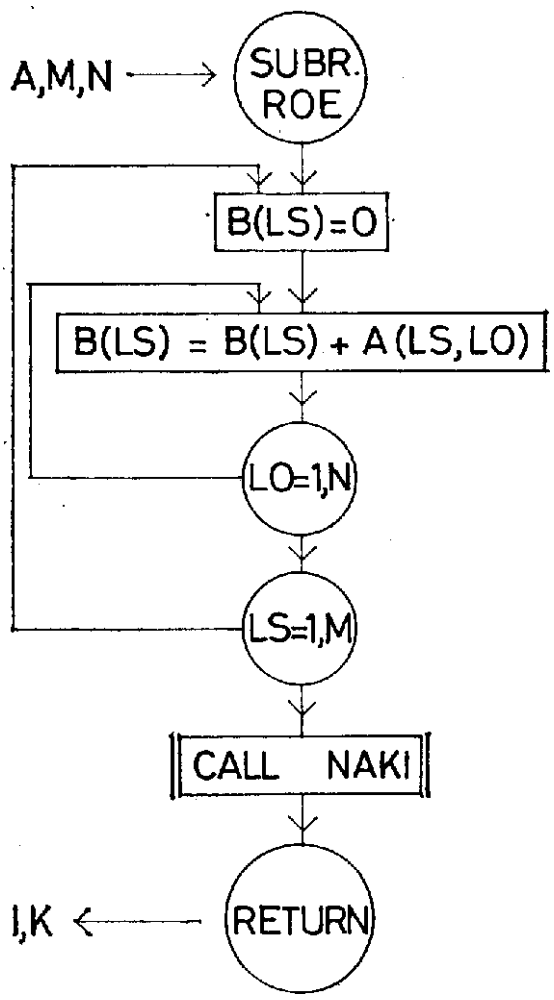
b) A fentihez hasonló megkötéssel szubrutin készítendő oszlopcserére.

c) Program készítendő, amely felcseréli a legnagyobb sorösszegű sort a legkisebbel, valamint a legnagyobb oszlopösszegű oszlopot, a legkisebbel. Nyomtatandó az eredeti mátrix, a cserélt sorú és (az eredetihez mérten) cserélt oszlopú mátrix.

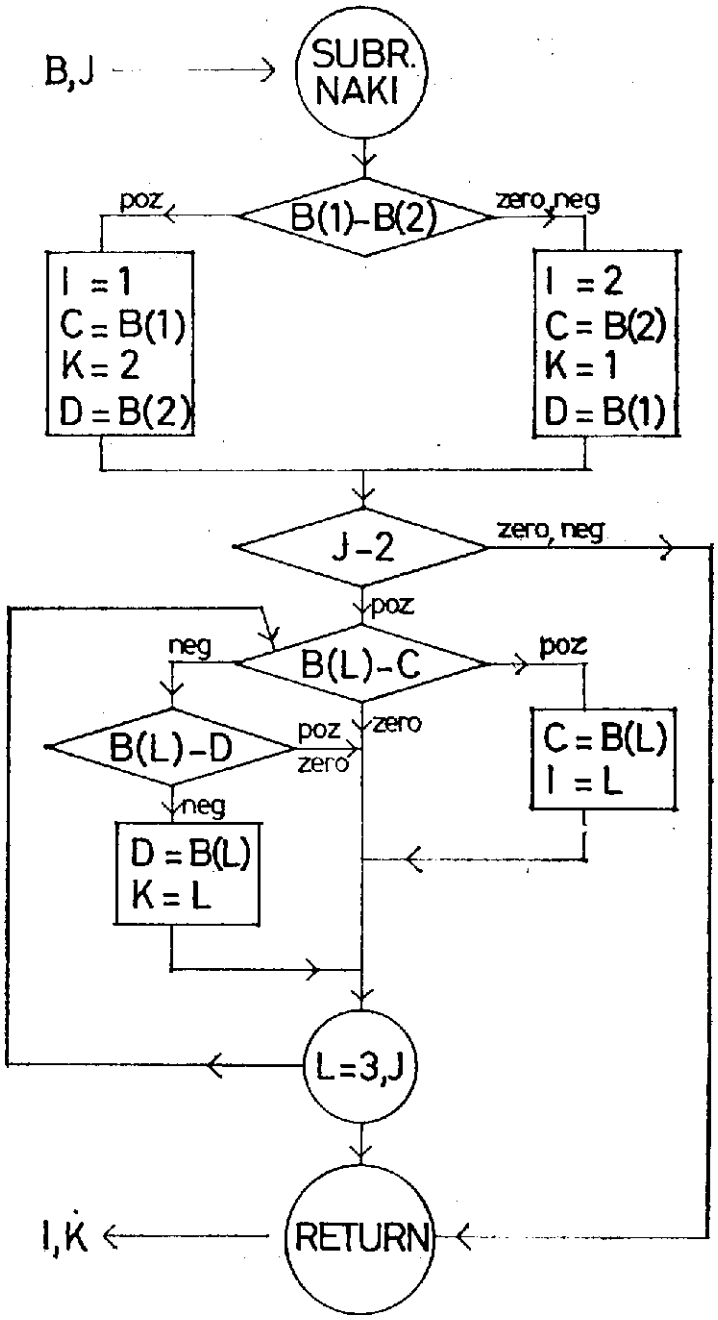
A 2. feladat megoldását nem közöljük.

Megoldás: Az alábbiakban bemutatjuk a program főszegmensét, a sorösszegét számító (ROE), a legnagyobb és legkisebb elemen kereső (NAKI), és a sorcserélő (SORCS) szubrutinokat. Nem közöljük az oszlopösszeget számító (POE) az oszlopot cserélő (OSZCS) és a tömböt kiíró (MAWR2) szubrutinokat.

Az 1.a) és 1.b) ábrán a ROE és NAKI szubrutinok blokkdiagramjai láthatók.



1a. ábra


B,J
SUBR.
NAKI
poz
B(1)-B(2)
zero,neg
I = 1
C = B(1)
K = 2
D = B(2)
I = 2
C = B(2)
K = 1
D = B(1)
J-2
zero, neg
poz
neg
B(L)-C
poz
B(L)-D
poz
zero
zero
neg
C = B(L)
I = L
D = B(L)
K = L
L=3,J
I,K
RETURN

1b. ábra

```
      |   |  MASTER PR11
      |   |  INTEGER A(10,10)
      |   |  DO 2 I = 1,10
    2 |   |  READ(1,3)(A(1J)J = 1,10)
      |   |  READ(1,1)M,N
      |   |  CALL MAWR2(A,M,N)
      |   |  CALL ROE(A,M,N,IS,KS)
      |   |  CALL SORCS(A,M,N,IS,KS)
      |   |  CALL MAWR2(A,M,N)
      |   |  CALL SORCS(A,M,N,IS,KS)
      |   |  CALL POE(A,M,N,IO,KO)
      |   |  CALL OSZCS(A,M,N,IO,KO)
      |   |  CALL MAWR2(A,M,N)
    1 |   |  FORMAT(2(2X,12))
    3 |   |  FORMAT(1018)
      |   |  STOP
      |   |  END
    6 |   |  SUBROUTINE ROE(A,M,N,I,K)
    2 |   |  INTEGER A(M,N),B(10)
      |   |  DO 1 LS = 1,M
      |   |  B(LS) = 0
      |   |  DO 2 LO = 1,N
    2 |   |  B(LS) = B(LS) + A(LS,LO)
    1 |   |  CONTINUE
      |   |  CALL NAKI(B,M,I,K)
      |   |  RETURN
      |   |  END
      |   |  SUBROUTINE NAKI(B,J,I,K)
      |   |  INTEGER B(J),C,D
      |   |  IF(B(1) - B(2))3,0,0
      |   |  I = 1
      |   |  C = B(1)
      |   |  K = 2
      |   |  D = B(2)
      |   |  GO TO 4
    3 |   |  I = 2
      |   |  C = B(2)
      |   |  K = 1
      |   |  D = B(1)
    4 |   |  IF(J - 2)10,10,0
      |   |  DO 5 L = 3,J
      |   |  IF(B(L) - C)6,5,0
      |   |  C = B(L)
      |   |  I = L
      |   |  GO TO 5
    6 |   |  IF(B(L) - D)0,5,5
      |   |  D = B(L)
      |   |  K = L
    5 |   |  CONTINUE
   10 |   |  RETURN
      |   |  END
      |   |  SUBROUTINE SORCS(A,M,N,I,K)
      |   |  INTEGER A(M,N),B(10)
      |   |  DO 7 L = 1,N
      |   |  B(L) = A(I,L)
      |   |  A(I,L) = A(K,L)
    7 |   |  A(K,L) = B(L)
      |   |  RETURN
      |   |  END
```

```
SUBROUTINE OSZCS(A, M, N, I, K)
.
.
.
END
SUBROUTINE POE(A, M, N, I, K)
.
.
.
END
SUBROUITNE MAWR2(A, M, N)
.
.
.
END
FINISH
```

Megjegyzések:

- A főszegmensben először a 10×10-es tömböt olvassuk be, majd az m és n indexeket, melyekről feltesszük, hogy az előírt korlátokon belül vannak. Az $m \times n$-es részmátrix a 10×10-es mátrixban a bal felső sarok felé tömörítve helyezkedik el.

- A főszegmensben a cserélt sorú mátrix nyomtatása után újra cseréltetjük a sorokat, hogy visszaálljon az eredeti mátrix.

- Figyeljük meg az A tömb átadását a ROE és a SORCS szubrutinokban. A-val együtt M és N indexeket is átadjuk és a szubrutinokban levő deklarációkban egész változók állnak az indexek helyén. Ilyenkor dinamikus indexhatárról beszélünk, ami lényegében az átadott tömb méretének változtathatóságát jelenti adott korlátokon belül. Ez csakis SUBROUTINE és FUNCTION szegmensekben fordulhat elő és csakis úgy,

- ha a szubrutin nyitó utasításának zárójelpárja közt szerepel a tömbazonosító és az indexéül használt (egy vagy több) egész változó;

- ha a dinamikus indexeknek adott számértékek nem nagyobbak a hívószegmensben deklarált tömb számértékű indexeinél.

6. Elsőfokú, többismeretlenes (inhomogén) egyenletrendszer megoldására szolgáló program

6.1. Az eliminációs módszer

Elsőfokú, többismeretlenes egyenletrendszer gyökeinek meghatározására számos, a számítástechnikában is alkalmazható módszert ismerünk. Pl.: a közismert Cramer-szabályon alapuló megoldás is programozható, de számítógépidő-felhasználás szempontjából gazdaságtalan. A gyakorlatilag hasznosítható eljárások felsorolásával, ismertetésével nem foglalkozhatunk. Az alábbiakban egy közismert, esetenként jól használható számítógépes eljárást mutatunk be, mely az un. **Gauss-féle eliminációs módszeren** alapul. Az eljárást egy numerikus példán tesszük érthetővé. Az egyenleteket $E_{m,n}$-nel fogjuk jelölni. Az egyenletrendszert ún. „menetek"-ben alakítjuk át, a kiindulási egyenletre $m = 0$, az első menet után kapott egyenletrendszerre $m = 1$ stb. Egy adott meneten belül az egyenleteket az n indexszel sorszámozzuk. Pl.: $E_{3,5}$ jelenti a harmadik menet után kapott egyenletrendszerben az ötödik egyenletet. Numerikus példánk, a megjelölt egyenletekkel az alábbi:

$$\begin{aligned} E_{0,1}: &\quad 3x_1 + 4x_2 + 3x_3 = \quad 1, \\ E_{0,2}: &\quad 2x_1 \ - \ x_2 \ - x_3 = \quad 6, \\ E_{0,3}: &\quad x_1 + 3x_2 + 2x_3 = -1. \end{aligned}$$

Az egyes menetekben az átalakításokat is jelölni fogjuk.

Ha k konstans, akkor $k \cdot E_{i,j}$ az egyenlet mindkét oldalán álló kifejezésnek k-val való szorzását, $E_{i,j} + E_{k,1}$ a két egyenlet megegyező oldalain álló kifejezéseinek összegezését jelenti stb. Az első menet után kapott egyenletek:

$$\begin{aligned} E_{1,1} = E_{0,1}: &\qquad 3x_1 + 4x_2 + 3x_3 = 1, \\ E_{1,2} = \frac{2}{3} \cdot E_{0,1} - E_{0,2}: &\qquad \frac{11}{3}x_2 + 3x_3 = -\frac{16}{3}, \\ E_{1,3} = \frac{1}{3} \cdot E_{0,1} - E_{0,3}: &\qquad -\frac{5}{3}x_2 - x_3 = \frac{4}{3}. \end{aligned}$$

Példánk második, egyben utolsó menete:

$$\begin{aligned}
E_{2,1} = E_{1,1}: &\qquad 3x_1 + 4x_2 + \quad 3x_3 = 1,\\
E_{2,2} = E_{1,2}: &\qquad \frac{11}{3}x_2 + \quad 3x_3 = -\frac{16}{3},\\
E_{2,3} = -\frac{5}{3}\cdot\frac{3}{11}E_{1,2} - E_{1,3}: &\qquad -\frac{4}{11}x_3 = \frac{36}{33}.
\end{aligned}$$

Az eljárás eddigi szakaszában (két menetben) a három ismeretlenből kiküszöböltünk (eliminéltunk) kettőt. Ez az eljárásnak ún. eliminációs része. Az eljárás következő szakaszában az egyenletekben alulról felfelé végighaladva kifejezzük az ismeretleneket: $E_{2,3}$-ból x_3-at, $E_{2,2}$-ből x_2-t, $E_{2,1}$-ből x_1-et. Ez az ún. visszahelyettesítő rész. Eredményül az $x_1 = 2$, $x_2 = 1$ és $x_3 = -3$ gyököket kapjuk. Már az eddigiekből is nyilvánvaló, hogy a menetek száma az egyenletek számánál eggyel kevesebb.

Feladat:

Program készítendő, amely a példaként hozott egyenletrendszer együtthatóit kártyáról beolvassa, és az egyenletet eliminációs módszerrel megoldja. A program álljon két szubrutinból, az első az eliminációt, a második a visszahelyettesítést hajtsa végre. Nyomtatandók egymás alatti sorokban az adott egyenletrendszer egyenletei és néhány sorral ezek alatt az eredményül kapott gyökök. Az egyenletek együtthatói három kártyán helyezkednek el, egy kártyán $15 - 15$ karakterszélességű mezőkben három tizedesjegy pontossággal egy egyenlet együtthatói.