Tutorial & Laboratory

Programming & Data Structure: CS11001/19001

Section - 4/D

DO NOT POWER ON THE MACHINE

Department of Computer Science and Engineering I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (**20.03.2014**)

Download

Download the file date200314.pdf from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

Assignment XIV

Download `int main()` and write the following C functions to complete the program. [Marks: $5 + 5 + 5 + 5 + 5$]

Task I

The function `void readMat(int n, double a[][ORD])` reads a square matrix of size $n \leq ORD$ in a 2-D array pointed by `a`.
The function `void writeMat(int n, double a[][ORD])` prints the matrix in the 2-D array of `a`.
[Marks: 5]

Task II

The function `void copyMat(int n, double b[][ORD], double a[][ORD])` copies the $n \times n$ matrix of `a` to the 2-D array of `b` $(b \leftarrow a)$.
The function `void initInvMat(int n, double aInv[][ORD])` initialises the 2-D array of `c` by the identity matrix of size n. [Marks: 5]

Task III

The function `void gaussElim(int n, double a[][ORD], double aInv[][ORD])` applies elementary row-operations to transform the matrix of `a` to an upper-triangular matrix. Same row-operations are also applied on the matrix of `aInv`. [Marks: 5]

Gauss Elimination (expl.)

$$a = \begin{bmatrix} 1 & 0 & 5 \\ 2 & 1 & 6 \\ 3 & 4 & 0 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gauss Elimination (expl.)

The elementary row-operations are,

$\text{Row}_1 \leftarrow \text{Row}_1 + (-2) \times \text{Row}_0$.

$\text{Row}_2 \leftarrow \text{Row}_2 + (-3) \times \text{Row}_0$.

The multipliers are obtained by $-\frac{a_{10}}{a_{00}}$ and $-\frac{a_{20}}{a_{00}}$

Gauss Elimination (expl.)

Transformed matrices are:

$$a = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -4 \\ 0 & 4 & -15 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

Gauss Elimination (expl.)

$\text{Row}_2 \leftarrow \text{Row}_2 + (-4) \times \text{Row}_1$.

The multiplier is $-\frac{a_{21}}{a_{11}}$

$$a = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 5 & -4 & 1 \end{bmatrix}$$

Gauss Elimination (expl.)

You do not have to take care of the situation where the position of pivotal element has a zero. So no permutation of rows is necessary. Following is an example of such situation.

Gauss Elimination (expl.)

$$a = \begin{bmatrix} 1 & 0.5 & 5 \\ 2 & 1 & 6 \\ 3 & 4 & 0 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gauss Elimination (expl.)

$\text{Row}_1 \leftarrow \text{Row}_1 + (-2) \times \text{Row}_0.$

$\text{Row}_2 \leftarrow \text{Row}_2 + (-3) \times \text{Row}_0.$

$$a = \begin{bmatrix} 1 & 0.5 & 5 \\ 0 & 0 & -4 \\ 0 & 4 & -15 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

It is necessary to permute Row_1 and Row_2.

Task IV

The function `void jordanElim(int n, double a[][ORD], double aInv[][ORD])` applies elementary row-operations to transform the matrix of `a` to a diagonal matrix. Same row-operations are also applied on the matrix of `aInv`. [Marks: 5]

After Gauss Elimination

$$a = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}, \quad aInv = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 5 & -4 & 1 \end{bmatrix}$$

Jordan Elimination (expl.)

$\text{Row}_0 \leftarrow \text{Row}_0 + (-5) \times \text{Row}_2$.
$\text{Row}_1 \leftarrow \text{Row}_1 + (4) \times \text{Row}_2$.
The multipliers are $-\frac{a_{02}}{a_{22}}$ and $-\frac{a_{12}}{a_{22}}$.

Jordan Elimination (expl.)

$$a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad aInv = \begin{bmatrix} -24 & 20 & -5 \\ 18 & -15 & 4 \\ 5 & -4 & 1 \end{bmatrix}$$

It happens to be the case that a_{01} is already 0, but the program will perform the following step.

Jordan Elimination (expl.)

$\text{Row}_0 \leftarrow \text{Row}_0 + (0) \times \text{Row}_0$.

The multiplier is $-\frac{a_{01}}{a_{11}}$.

$$a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad aInv = \begin{bmatrix} -24 & 20 & -5 \\ 18 & -15 & 4 \\ 5 & -4 & 1 \end{bmatrix}$$

Task V

It is well known that elementary row operations do not change the determinant of the matrix. Write the function `double det(int n, double a [][ORD])` computes and returns the determinant of the matrix of `a`.

Task V (cont.)

Write the function `void makeIdentity(int n, double a[][ORD], double aInv[][ORD])` that transforms the matrix pairs by dividing every row $i = 0, \cdots, n-1$ by the diagonal element a_{ii}. This makes the matrix of `a` an identity matrix and the matrix of `aInv`, the inverse of the original matrix. [Marks: 5]

Task VI

Write the function `void multMat(int n, double a[][ORD], double b[][ORD], double c[][ORD])` that computes $c \leftarrow a \times b$, where both `a` and `b` are square matrices of size n. [Marks: 5]

Input

```
4
5 -3 1 2
1 -3 2 10
-4 1 -7 -2
-3 -1 -2 -5
```

Output

```
The matrix is:
  5.00  -3.00   1.00   2.00
  1.00  -3.00   2.00  10.00
 -4.00   1.00  -7.00  -2.00
 -3.00  -1.00  -2.00  -5.00
```

Output

```
The augmented matrix is:
  1.00   0.00   0.00   0.00
  0.00   1.00   0.00   0.00
  0.00   0.00   1.00   0.00
  0.00   0.00   0.00   1.00
```

Output

```
The matrix after Gauss Elimination:
  5.00  -3.00   1.00   2.00
 -0.00  -2.40   1.80   9.60
  0.00  -0.00  -7.25  -6.00
 -0.00   0.00   0.00 -12.10
```

Output

```
Augmented matrix:
  1.00   0.00   0.00   0.00
 -0.20   1.00   0.00   0.00
  0.92  -0.58   1.00   0.00
  0.39  -0.89  -0.48   1.00
```

Output

```
The matrix after Jordan Elimination:
  5.00  -0.00  -0.00   0.00
 -0.00  -2.40   0.00  -0.00
  0.00  -0.00  -7.25  -0.00
 -0.00   0.00   0.00 -12.10
```

Output

```
Augmented matrix:
  0.80  -0.49   0.19  -0.74
  0.29   0.26  -0.08   0.67
  0.72  -0.14   1.24  -0.50
  0.39  -0.89  -0.48   1.00
```

Output

```
The determinant is: -1053.00
```

Output

```
The Inverse matrix:
  0.16  -0.10   0.04  -0.15
 -0.12  -0.11   0.03  -0.28
 -0.10   0.02  -0.17   0.07
 -0.03   0.07   0.04  -0.08
```

Output

```
The Product is I:
  1.00   0.00  -0.00   0.00
 -0.00   1.00   0.00  -0.00
  0.00  -0.00   1.00   0.00
 -0.00  -0.00   0.00   1.00
```

Submission by ftp

```
$ ftp 10.5.17.186
Connected to 10.5.17.186.
220---------- Welcome to Pure-FTPd ----
220-You are user number 1 of 50 allowed.
220-Local time is now 07:54. ..... 21.
220-IPv6 connections ........
220 ... disconnected .. inactivity.
Name (10.5.17.186:...): pds
```

Submission by ftp

```
331 User pds OK. Password required
Password: pds04
230-User pds has group access to:  pds
230 OK. Current restricted directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd assignment14
250 OK. Current directory is /assignment14
```

Submission by ftp

```
ftp> put D0614.c
local: D0614.c remote: D0614.c
200 PORT command successful
150 Connecting to port 47093
226-File successfully transferred
226 0.001 seconds .. 39.00 Kbytes ..
27 bytes sent in 0.00 secs (1098.6 kB/s)
ftp> bye
21-Goodbye. .....
221 Logout.
$
```