SM2-21st Computer Lab 9

Arrays

10 November 2017, Friday, 6:45pm
Computer Lab.

Unzip Lab9.zip. Please prepare your work before the actual lab session.

1. Matrix
(http://en.wikipedia.org/wiki/Matrix_multiplication)

If **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix}$$

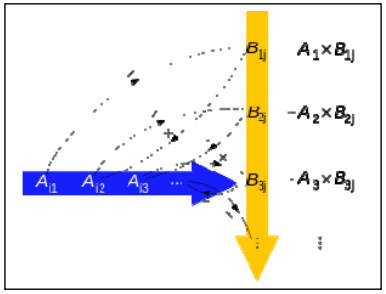
the **matrix product AB** (denoted without multiplication signs or dots) is defined to be the $n \times p$ matrix

$$\mathbf{AB} = \begin{pmatrix} (\mathbf{AB})_{11} & (\mathbf{AB})_{12} & \cdots & (\mathbf{AB})_{1p} \\ (\mathbf{AB})_{21} & (\mathbf{AB})_{22} & \cdots & (\mathbf{AB})_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{AB})_{n1} & (\mathbf{AB})_{n2} & \cdots & (\mathbf{AB})_{np} \end{pmatrix}$$

where each *i*, *j* entry is given by multiplying the entries A_{ik} (across row *i* of **A**) by the entries B_{kj} (down column *j* of **B**), for $k = 1, 2, ..., m$, and summing the results over *k*:

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} .$$

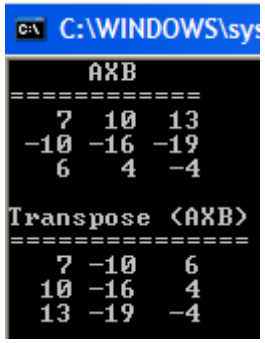
Arithmetic process of multiplying numbers (solid lines) in row *i* in matrix **A** and column *j* in matrix **B**, *then* adding the terms (dashed lines) to obtain entry *ij* in the final matrix.



The **transpose** of a square matrix **A** is another matrix created by reflecting **A** over its main diagonal (which runs from top-left to bottom-right). For example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

The contents of a 3x4 matrix A, and, the contents of a 4x3 matrix B, are stored in two integer arrays in matrix.c (unzip from lab9.zip). Complete the program to display the contents of AxB and $(AxB)^T$ on the screen as follows:



2. A word is a palindrome if its sequence of letters may be read the same way in either forward or reverse direction. For examples, level, madam, radar, civic, rotator are a few of such interesting words. Write a full program in C language to read a word of not more than 20 characters from the keyboard and check if the word is a palindrome. The program runs repeatedly until the enter key is pressed without entering any word. A session of the program execution is as follows:

```
Enter a word of not more than 20 characters: Madam
"Madam" is not a palindrome.

Enter a word of not more than 20 characters: madam
"madam" is a palindrome.

Enter a word of not more than 20 characters: W
"W" is a palindrome.

Enter a word of not more than 20 characters: abbcddedcbba
"abbcddedcbba" is not a palindrome.

Enter a word of not more than 20 characters:
Press any key to continue . . .
```

3. The text file (unzip from lab9.zip) has the contents shown in **words.inf**. The number of words and the longest word length in the text file are

unknown but you can assume that there are no more than 25 words in the text file and each word contains no more than 15 characters. Write a complete C program to sort the words in lexicographical order.

2-D array should be used in your program. You should use the strcmp function in string.h. The syntax for the **strcmp** function is:

int strcmp(const char *s1, const char *s2);

s1 is an array to compare.
s2 is an array to compare.

The return value of the **strcmp** function is as follows:

0: s1 and s2 are equal

Negative integer: The stopping character in s1 is less than the stopping character in s2

Positive integer: The stopping character in s1 is greater than the stopping character in s2

The sorted output is displayed on the screen and should be as follows:

Screen Output

```
a
word
is
a
palindrome
if
its
sequence
of
letters
may
be
read
the
same
way
in
either
forward
or
reverse
direction
```

words.inf

Use debugger whenever in doubt!