CS1100 – Introduction to Programming

Trimester 3, April – June 2021

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lecture 18

Character arrays and standard library support

- Character arrays or strings occur very often.
- C provides a standard library string.h
- exposes several useful functions:
  - strlen
  - strcmp
  - strcpy
  - strstr

Some Standard Library Functions: Strlen

Standard C function to compute length of a string.

Some Standard Library Functions: Strlen

Standard C function to compute length of a string.

```
#include <stdio.h>
#include <string.h>

int main( )
{
    int len;
    char array[20]="fresh2refresh.com" ;

    len = strlen(array) ;

    printf ( "string length  = %d \n" , len ) ;
    return 0;
}
```

Some Standard Library Functions: Strcmp

Standard C function to compare two strings

- Returns zero if the two strings are same.
- If length of string1 less than string2, it returns negative value. If length of string1 greater than string2, it returns positive value.

```
#include <stdio.h>
#include <string.h>
int main( )
{
   char str1[ ] = "fresh" ;
   char str2[ ] = "refresh" ;
   int i, j, k ;
   i = strcmp ( str1, "fresh" ) ;
   j = strcmp ( str1, str2 ) ;
   k = strcmp ( str1, "f" ) ;
   printf ( "\n%d %d %d\n", i, j, k ) ;
   return 0;
}
```

Some Standard Library Functions: Strcpy

Standard C function to copy source string to target

```
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[ ] = "Hello" ;
   char target[20]= "All" ;
   printf ( "\nsource string = %s", source ) ;
   printf ( "\ntarget string = %s", target ) ;
   strcpy ( target, source ) ;
   printf ( "\ntarget string after strcpy( ) = %s\n", target);
   return 0;
}
```

Some Standard Library Functions: Strstr

Standard C function to search for one string in another: returns "pointer" to the first occurrence of the string in a given string

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a test string for testing";
  char *p;
  p = strstr (string,"test");
  if(p)
  {
    printf("string found\n" );
    printf ("First occurrence of string \"test\" in \"%s\" is \"%s\"\n",string, p);
  }
  else printf("string not found\n" );
   return 0;
}
```

Output

string found

First occurrence of string "test" in "This is a test string for testing" is "test string for testing"

Some Standard Library Functions: Getchar

```
#include <stdio.h>
int main(){
    char name[30], ch;
    int i=0;
    printf("Enter name: ");
    while(ch!='\n')    // terminates if user hit enter
    {
        ch=getchar();
        if (ch != '\n')
         {
         name[i]=ch;
         i++;
       }
    }
    name[i]='\0';       // inserting null character at end
    printf("Name: %s",name);
    return 0;
}
```

Matrix Operations : Addition

- Write a program to add two matrices M1 and M2

Matrix Operations : Addition

- Write a program to add two matrices M1 and M2

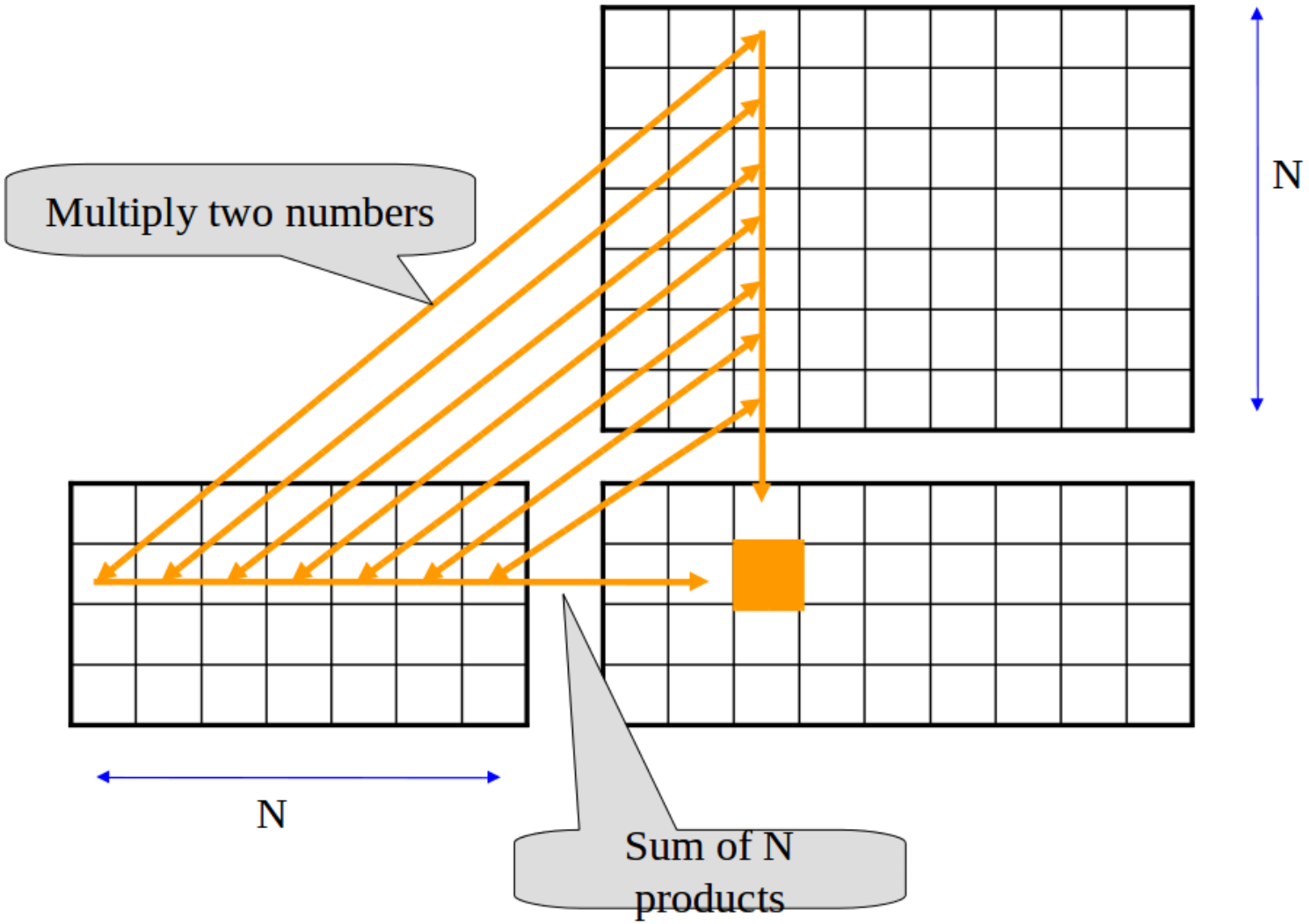
```
#include<stdio.h>
main() {

    /* Assume N1 and N2 are defined as const int */

    int A[N1][N2];
    int B[N1][N2];
    /*initialize M1, M2 suitably */
    int C[N1][N2];
    int i, j;

    for (i = 0; i<N1; i++) {
        for (j = 0; j<N2; j++) {
            A[i][j] = B[i][j] + C[i][j];
        }
    }
}
```

Matrix Operations : Multiplication

- Write a program to multiply matrices A and B

Matrix Operations : Multiplication

- Write a program to multiply matrices A and B



Program to Multiply Matrices

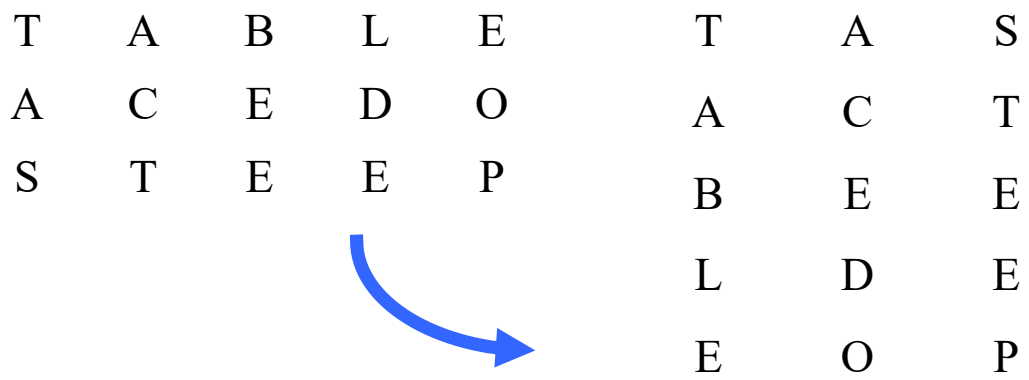
```
#include <stdio.h>
int main() {
  int m, n, p, q, c, d, k, sum = 0;
  int first[10][10], second[10][10], multiply[10][10];
  printf("Enter number of rows and columns of first matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter elements of first matrix\n");
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      scanf("%d", &first[c][d]);
 printf("Enter number of rows and columns of second matrix\n");
  scanf("%d%d", &p, &q);
 if (n != p)
    printf("The multiplication isn't possible.\n");
  else  {
    printf("Enter elements of second matrix\n");
    for (c = 0; c < p; c++)
      for (d = 0; d < q; d++)
        scanf("%d", &second[c][d]);
    for (c = 0; c < m; c++) {
      for (d = 0; d < q; d++) {
        for (k = 0; k < p; k++) {
          sum = sum + first[c][k]*second[k][d];        }
    multiply[c][d] = sum;
    sum = 0;    }
    }
    printf("Product of the matrices:\n");
    for (c = 0; c < m; c++) {
      for (d = 0; d < q; d++)
        printf("%d\t", multiply[c][d]);  printf("\n");  } }
  return 0; }
```

Matrix Operations : Transpose

- Write a program to compute transpose of a matrix

Matrix Operations : Transpose

- Write a program to compute transpose of a matrix



Computing Matrix Transpose

```
#include "stdio.h"
int main() {
  const int N1=10;
  int  A[N1][N1];
  int n1;
  int temp;
  int i, j;
  printf ("Please input the size of a matrix:\n");
  scanf ("%d", &n1);
  printf ("input the elements for matrix A\n");

    for (i = 0; i < n1; i++)
      for (j = 0; j < n1; j++)
scanf("%d", &A[i][j]);

    for (i = 0; i < n1; i++)
      for (j = 0; j < n1; j++) {
if (j < i) {
  temp = A[i][j];
  A[i][j] = A[j][i];
  A[j][i] = temp;
}
}
  /* fill in your code here */
    for (i = 0; i < n1; i++) {
      for (j = 0; j < n1; j++)
printf ("%d ", A[i][j]);
      printf("\n");
    }

    return 0;
}
```

Character grids

- Given a character grid, and a string s, print the indices of the rows and columns of grid that contain s.

---

c	a	t	t	y
c	c	s	e	p
e	s	c	e	l
s	e	e	s	e
h	a	p	s	a

Character grids

- Given a character grid, and a string s, print the indices of the rows and columns of grid that contain s.

---

c	a	t	t	y
c	c	s	e	p
e	s	c	e	l
s	e	e	s	e
h	a	p	s	a

- Which rows and columns contain cse?

Fun With Words: Palindrome Square

- A word can be read either horizontally or vertically in every row/column

Fun With Words: Palindrome Square

- A word can be read either horizontally or vertically in every row/column

S	T	E	P
T	I	M	E
E	M	I	T
P	E	T	S

R	A	T	S
A	B	U	T
T	U	B	A
S	T	A	R

Reversing an Array

```
#include <stdio.h>
int main()
{
  int n, c, d;

  printf("Enter the number of elements in array\n");
  scanf("%d", &n);
  int a[n];
  int b[n];
  printf("Enter array elements\n");

  for (c = 0; c < n ; c++)
    scanf("%d", &a[c]);

  // Copying elements into array b starting from the end of array a

  for (c = n - 1, d = 0; c >= 0; c--, d++)
    b[d] = a[c];

  // Copying reversed array into the original, we are modifying the original array.

  for (c = 0; c < n; c++)
    a[c] = b[c];

  printf("The array after reversal:\n");

  for (c = 0; c < n; c++)
    printf("%d\n", a[c]);

  return 0;
}
```

Reversing an Array In Place

- Swap the first element with last
- Second with second last
- and so on ...

How about the following code?

```
for (i=0; i<n; i++)
        swap (a, i, n-1-i);

void swap (char a[ ], int i, int j){
char c;
c = a[i];
a[i]=a[j];
a[j]=c;
}
```

Reversing an Array In Place