2-D Arrays in C

Palash Dey
Department of Computer Science & Engg.
Indian Institute of Technology
Kharagpur

Slides credit: Prof. Indranil Sen Gupta

Two Dimensional Arrays

- We have seen that an array variable can store a list of values.
- Many applications require us to store a table of values.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Student 1	75	82	90	65	76
Student 2	68	75	80	70	72
Student 3	88	74	85	76	80
Student 4	50	65	68	40	70

Contd.

- The table contains a total of 20 values, five in each line.
  - The table can be regarded as a matrix consisting of *four rows* and *five columns*.
- C allows us to define such tables of items by using *two-dimensional* arrays.

Declaring 2-D Arrays

- **General form:**

```
type array_name[row_size][column_size];
```

- **Examples:**

```
int      marks[4][5];
float    sales[12][25];
double   matrix[100][100];
```

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use two indices.
  - First indicates row, second indicates column.
  - Both the indices should be expressions that evaluate to *integer values*.
- Examples:

```
x[m][n] = 0;
c[i][k] += a[i][j] * b[j][k];
val = sqrt (a[j*3][k]);
```

How is a 2-D array is stored in memory?

- Starting from a given memory location, the elements are stored *row-wise* in consecutive memory locations.

  x: starting address of the array in memory

  c: number of columns

  k: number of bytes allocated per array element

Element `a[i][j]` :: allocated memory location at address `x+(i*c+j)*k`

a[0][0] a[0][1] a[0]2] a[0][3]	a[1][0] a[1][1] a[1][2] a[1][3]	a[2][0] a[2][1] a[2][2] a[2][3]
Row 0	**Row 1**	**Row 2**

How to read the elements of a 2-D array of size *nrow*$\times$*ncol*?

- By reading them one element at a time

```
for (i=0; i<nrow; i++)
   for (j=0; j<ncol; j++)
      scanf ("%d", &a[i][j]);
```

- The ampersand (&) is necessary.
- The elements can be entered all in one line or in different lines.

How to print the elements of a 2-D array?

- By printing them one element at a time.

```
for (i=0; i<nrow; i++)
   for (j=0; j<ncol; j++)
     printf  ("\n %d", a[i][j]);
```

  - The elements are printed one per line.

```
for (i=0; i<nrow; i++)
  for  (j=0; j<ncol; j++)
     printf  ("%d", a[i][j]);
```

  - The elements are all printed on the same line.

Contd.

```
for (i=0; i<nrow; i++)
  {
     printf ("\n");
     for (j=0; j<ncol; j++)
       printf ("%d   ", a[i][j]);
   }
```

- The elements are printed nicely in matrix form.

- How to print two matrices side by side?

- **Printing two matrices A and B of sizes m×n each side by side.**

```
for (i=0; i<m; i++)
    {
           printf ("\n");
           for (j=0; j<n; j++)
                  printf ("%d   ", A[i][j]);
           printf ("        ");
           for (j=0; j<n; j++)
                  printf ("%d   ", B[i][j]);
    }
```

Example: Matrix Addition

```
#include <stdio.h>

main()
{
    int a[100][100], b[100][100],
        c[100][100], p, q, m, n;

    scanf ("%d %d", &m, &n);

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

    for (p=0; p<m; p++)
      for (q=0; q<n; q++)
        scanf ("%d", &b[p][q]);
```

```
  for (p=0; p<m; p++)
    for (q=0; q<n; q++)

      c[p]q] = a[p][q] + b[p][q];

  for (p=0; p<m; p++)
  {
      printf ("\n");
      for (q=0; q<n; q++)
        printf ("%d   ", c[p][q]);
  }
}
```

Passing 2-D Arrays

- Similar to that for 1-D arrays.
  - The array contents are not copied into the function.
  - Rather, the address of the first element is passed.
- For calculating the address of an element in a 2-D array, we need:
  - The starting address of the array in memory.
  - Number of bytes per element.
  - Number of columns in the array.
- The above three pieces of information must be known to the function.

Example Usage

```
#include  <stdio.h>

main()
{
  int a[15][25],b[15]25];
     :
     :
  add (a, b, 15, 25);
     :
}
```

```
void add (int x[][25],
int y[][25], int rows,
int cols)
{
       :
}
```

We can also write

int x[15][25], y[15][25];

Example: Transpose of a matrix

```
#include <stdio.h>

void transpose (int x[][3],
int n)
{
   int  p, q, t;

   for (p=0; p<n; p++)
     for (q=0; q<n; q++)
      {
         t = x[p][q];
         x[p][q] = x[q][p];
         x[q][p] = t;
       }
}
```

```
main()
{
  int a[3][3], p, q;

  for (p=0; p<3; p++)
    for (q=0; q<3; q++)
      scanf ("%d", &a[p][q]);
  transpose (a, 3);
  for (p=0; p<3; p++)
  {
    printf ("\n");
    for (q=0; q<3; q++)
       printf ("%d  ", a[p][q]);
  }
}
```

Is the function correct?

```
10  20   30

40  50   60

70   80  90
```

$$\Downarrow$$

```
10  20   30

40  50   60

70  80   90
```

The Correct Version

```
void transpose (x, n)
int x[][3], n;
{
   int  p, q, t;

   for (p=0; p<n; p++)
     for  (q=p; q<n; q++)
       {
         t = x[p][q];
         x[p][q] = x[q][p];
         x[q][p] = t;
       }
}
```

$$\begin{array}{ccc} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{array}$$

$$\Downarrow$$

$$\begin{array}{ccc} 10 & 40 & 70 \\ 20 & 50 & 80 \\ 30 & 60 & 90 \end{array}$$

Some Exercise Problems to Try Out

1. A shop stores n different types of items. Given the number of items of each type sold during a given month, and the corresponding unit prices, compute the total monthly sales.
2. Multiple two matrices of orders mxn and nxp respectively.