2-D Arrays in C

Prof. Indranil Sen Gupta

Dept. of Computer Science & Engg.
Indian Institute of Technology
Kharagpur

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.

<table>
<thead>
<tr>
<th>Student</th>
<th>Subject 1</th>
<th>Subject 2</th>
<th>Subject 3</th>
<th>Subject 4</th>
<th>Subject 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>82</td>
<td>90</td>
<td>65</td>
<td>76</td>
</tr>
<tr>
<td>2</td>
<td>68</td>
<td>75</td>
<td>80</td>
<td>70</td>
<td>72</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>74</td>
<td>85</td>
<td>76</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>65</td>
<td>68</td>
<td>40</td>
<td>70</td>
</tr>
</tbody>
</table>
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 which evaluate to integer values.

• Examples:
  \[ x[m][n] = 0; \]
  \[ c[i][k] += a[i][j] \times b[j][k]; \]
  \[ val = \sqrt{a[j*3][k]}; \]

How is a 2-D array 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 \( x + (i\times c + j) \times k \)

| \( a[0][0] \) | \( a[0][1] \) | \( a[0][2] \) | \( a[0][3] \) | \( a[1][0] \) | \( a[1][1] \) | \( a[1][2] \) | \( a[1][3] \) | \( a[2][0] \) | \( a[2][1] \) | \( a[2][2] \) | \( a[2][3] \) |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 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

```c
for (i=0; i<nrow; i++)
    for (j=0; j<ncol; j++)
        scanf ("%f", &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.

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

– The elements are printed one per line.

```c
for (i=0; i<nrow; i++)
    for (j=0; j<ncol; j++)
        printf ("%f", 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 ("%f ", 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 ("%f ", A[i][j]);
    printf (" ");
    for (j=0; j<n; j++)
        printf ("%f ", B[i][j]);
}
Example: Matrix Addition

```c
#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++)
        for (q=0; q<n; q++)
            printf ("%f   ", 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

```c
#include <stdio.h>

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

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

We can also write
int x[15][25], y[15][25];

---

Example: Transpose of a matrix

```c
#include <stdio.h>

void transpose (x, n)
    int x[][3], 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]);
            printf ("\n");
        }
    }
```
Is the function correct?

\[
\begin{array}{cccc}
10 & 20 & 30 \\
40 & 50 & 60 & \\
70 & 80 & 90 & \\
\hline
10 & 20 & 30 \\
40 & 50 & 60 & \\
70 & 80 & 90 & \\
\end{array}
\]

The Correct Version

```c
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;
        }
}
```
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.