ESC101: Introduction to Computing

Multi dimensional Arrays

Example calls for marginals

```
void marginals(double mat[ ][6], int nrows);
void main() {
      double mat[9][6];
     /* read the first 8 rows into mat */
     marginals(mat,8);
}
```

✓

```
void marginals(double mat[ ][6], int nrows);
void main() {
     double mat[9][6];
    /* read 9 rows into mat */
     marginals(mat,10);
}
```

UNSAFE

✗

The 10th row of mat[9][6] is not defined. So we may get a segmentation fault when marginals() processes the 10th row, i.e., i becomes 9.

As with 1 dim arrays, allocate your array and stay within the limits allocated.

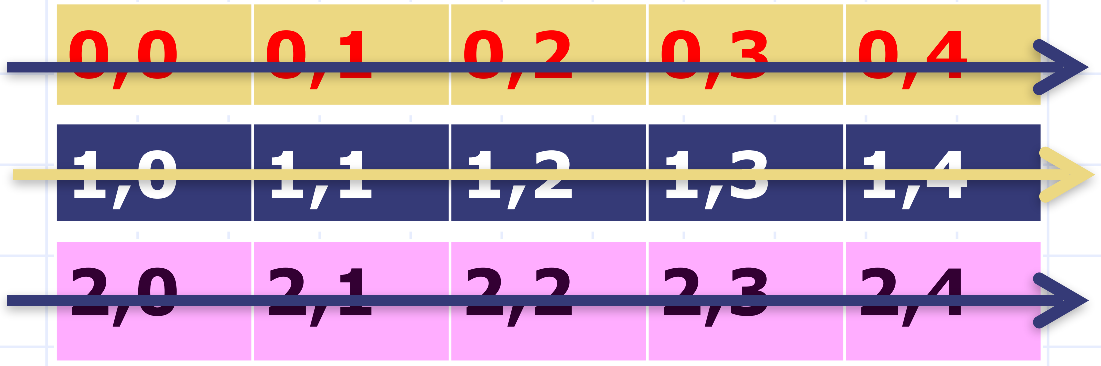
Number of columns

Why is the number of columns required?

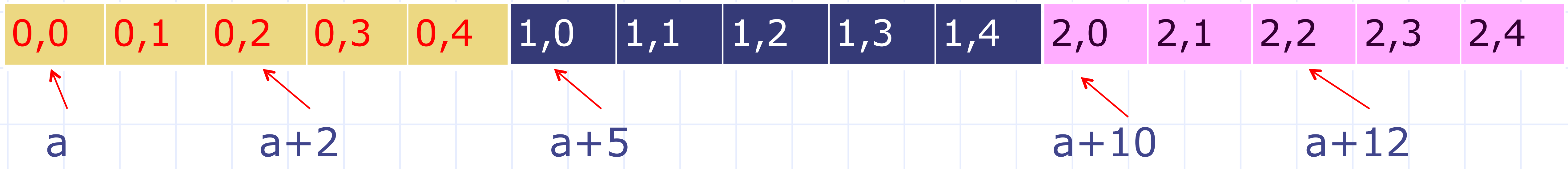
- The memory of a computer is a 1D array!
- 2D (or >2D) arrays are "flattened" into 1D to be stored in memory
- In C (and most other languages), arrays are flattened using Row-Major order
  - In case of 2D arrays, knowledge of number of columns is required to figure out where the next row starts.
  - Last n-1 dimensions required for nD arrays

Row Major Layout

mat[3][5]



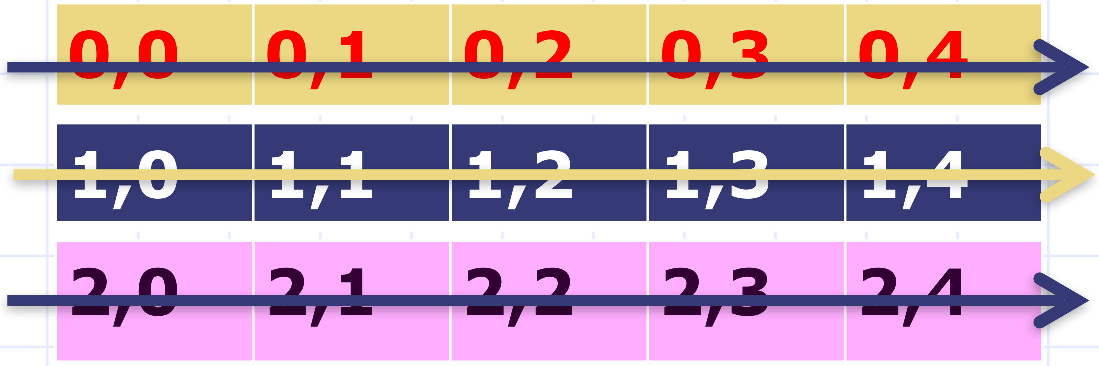
Layout of mat[3][5] in memory



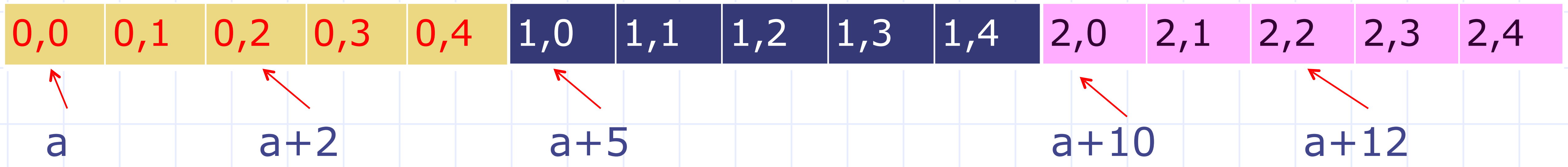
- **for a 2D array declared as mat[M][N], cell [i][j] is stored in memory at location i*N + j from start of mat.**
- **for k-D array arr[N_1][N_2]…[N_k], cell [i_1][i_2]…[i_k] will be stored at location**
  $i_k + N_k*(i_{k-1} + N_{k-1}*(i_{k-2} + (\ldots + N_2*i_1) \ldots))$

Row Major Layout

mat[3][5]



Layout of mat[3][5] in memory



- **About C implementation:** a = *mat
- *mat = mat[0], *(mat+1) = mat[1], *(mat+2) = mat[2],...... Each of which stores the reference to the corresponding row.

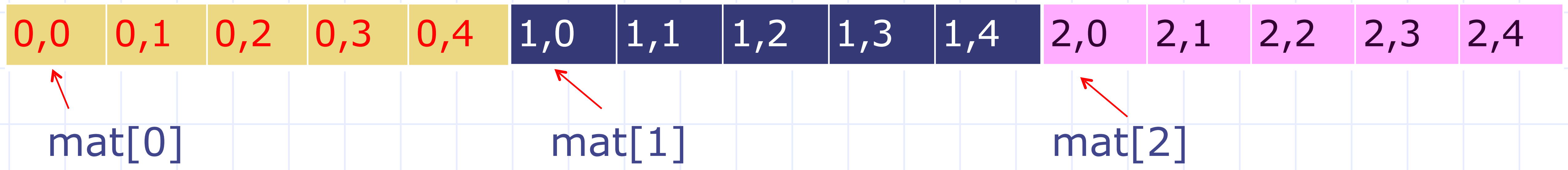
•That is, **mat** POINTS to the beginning of the array that stores the references to each of the rows.

Row Major Layout

mat[3][5]

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4

Layout of mat[3][5] in memory



Array of Strings

- 2D array of char.
- Recall
  - Strings are character arrays that end with a '\0'
  - To display a string we can use printf with the %s placeholder.
  - To input a string we can use scanf with %s. Only reads non-whitespace characters.

Array of Strings

- Write a program that reads and displays the name of few cities of India

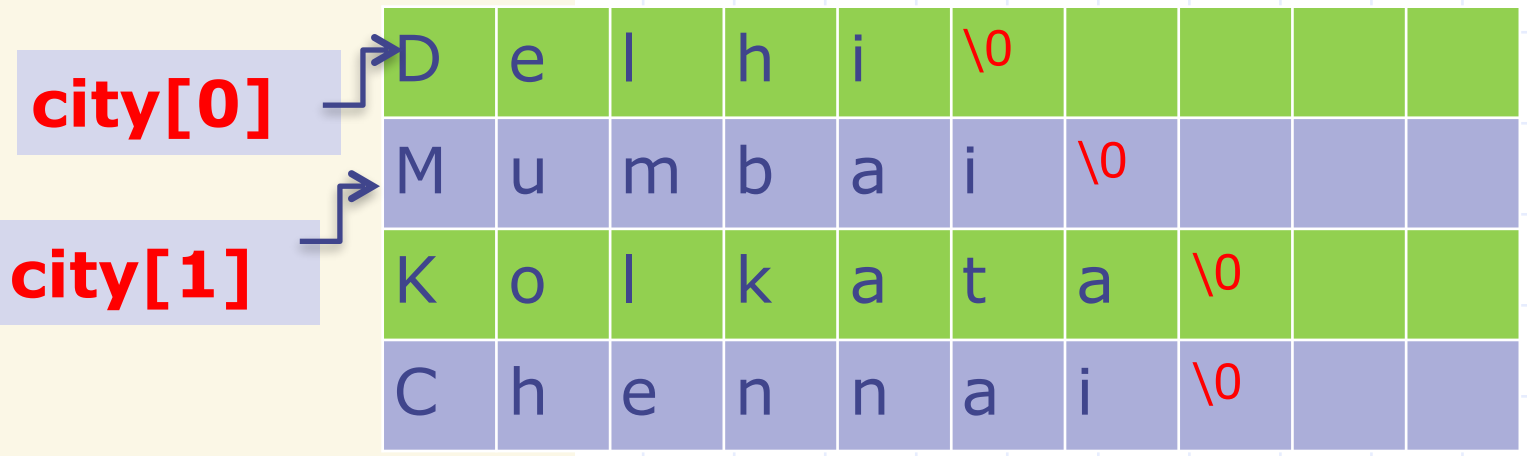
```
#define NCITY 4
#define LENCITY 10

int main(){
   char city[NCITY][LENCITY];
   int i;

   for (i=0; i<NCITY; i++){
      scanf("%s", city[i]);
   }

   for (i=0; i<NCITY; i++){
      printf("%d %s\n", i, city[i]);
   }
   return 0;
}
```

INPUT
```
Delhi
Mumbai
Kolkata
Chennai
```



D	e	l	h	i	\0				
M	u	m	b	a	i	\0			
K	o	l	k	a	t	a	\0		
C	h	e	n	n	a	i	\0		

OUTPUT
```
0 Delhi
1 Mumbai
2 Kolkata
3 Chennai
```

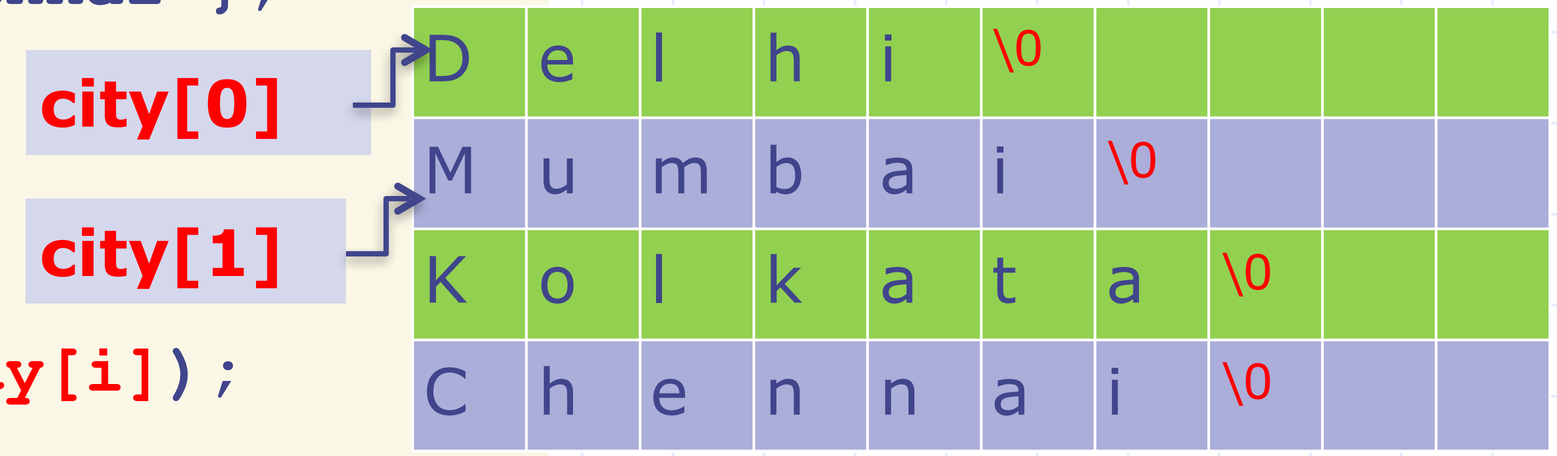
Array of Strings

List initialization is also allowed

```
#define NCITY 4
#define LENCITY 10

int main(){
  char city[][LENCITY] = {"Delhi",
   "Mumbai", "Kolkata", "Chennai"};
  int i;


  for (i=0; i<NCITY; i++){
    printf("%d %s\n", i, city[i]);
  }
  return 0;
}
```



D	e	l	h	i	\0				
M	u	m	b	a	i	\0			
K	o	l	k	a	t	a	\0		
C	h	e	n	n	a	i	\0		

OUTPUT

```
0 Delhi
1 Mumbai
2 Kolkata
3 Chennai
```

Practice Problem

- We are provided with list of 5 names. Sort them in chronological order.
- Input : Harpreet Shivam Bhuvesh Amlan Nishant
- Output:

Amlan

Bhuvesh

Harpreet

Nishant

Shivam

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

void swap( char s1[100], char s2[100]);
void sort( char names[5][100] );

int main()
{
   char names[5][100];
   for(int i=0; i<5; i++)
      scanf("%s",names[i] );
   sort( names );
   for(int i=0; i<5; i++)
      printf("%s\n",names[i]);
   return 0;
}
```

```c
void swap( char s1[100], char s2[100])
{
    char str[100];
    strcpy( str, s1);
    strcpy( s1, s2 );
    strcpy( s2, str );
}
void sort( char names[5][100] )
{
    for(int i=0; i<5; i++)
    {
        for(int j=i+1; j<5; j++)
        {
            if(strcmp(names[i],names[j])>0)
                swap(names[i], names[j]);
        }
    }
    return;
}
```

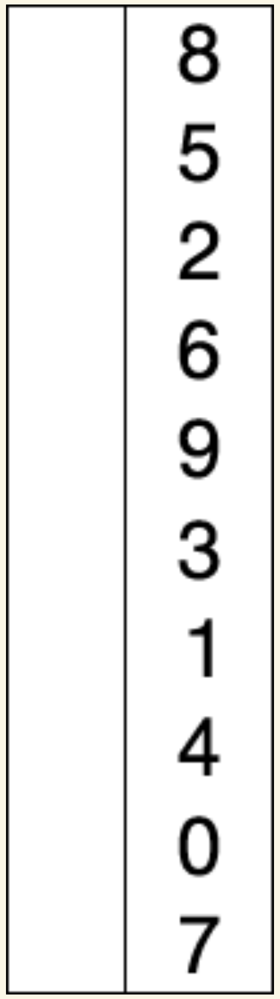


Fig Source: Wikipedi

Practice Problem

- Each course given as a string.
- Each course has with it its pre-requsite course listed (NULL if no pre-requisite)
- Input: List of 5 courses with its pre-requisite
- Output: A sequence of courses to be followed (if CS201 and CS210 both are possible, CS201 should be output before CS210)

```
Input
ESC101 NULL
CS210 ESC101
CS345 CS210
CS340 CS201
CS201 ESC101

Output
ESC10 CS201 CS210 CS340 CS345
```

```
void swap( char s1[100], char p1[100], char s2[100], char
p2[100])
{
   char str[100];
   strcpy( str, s1);
   strcpy( s1, s2 );
   strcpy( s2, str );
   strcpy( str, p1);
   strcpy( p1, p2 );
   strcpy( p2, str );
}
void sort_courses( char courses[5][100], char prereq[5]
[100] )
{
   for(int i=0; i<5; i++)
   {
      for(int j=i+1; j<5; j++)
      {
         if(strcmp(courses[i],courses[j])>0)
            swap(courses[i], prereq[i], courses[j],
prereq[j]);
      }
   }
}
```

```
void order_courses( char course[5][100], char prereq[5]
[100])
{
   char str[100]="NULL";
   int cnt=1;
   //looping over prereq with i
   for( int i=1; i<5; i++)
   {
      //looping over courses to check if i is a prereq
      for(int j=0; j<5; j++)
      {
         if( strcmp(prereq[j],str) == 0 )
            seq[j] = cnt++;
      }
      for(int j=0; j<5; j++)
         if(seq[j] == i)
            strcpy(str, course[j]);
   }
}
```

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

int seq[5] = {0};
void swap( char s1[100], char p1[100], char s2[100], char
p2[100]);
void sort_courses( char crs[5][100], char prq[5][100] );
void order_courses( char crs[5][100], char prq[5][100]);
int main()
{
   char course[5][100];
   char prereq[5][100];

   for(int i=0; i<5; i++)
      scanf("%s %s",course[i], prereq[i] );

   sort_courses( course, prereq );
   order_courses( course, prereq );
    for(int i=1; i<=5; i++)
      for(int j=0; j<5; j++)
         if(seq[j] == i)
            printf("%s\n",course[j]);
    return 0;
}
```