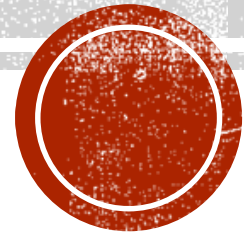


**ARRAY**

**MAN**



# CONSIDER A REAL LIFE PROBLEM

- Write a program that
  - Takes the number of students  $N$  as input
  - Takes the CT marks of  $N$  students
  - Calculates the average, minimum and maximum marks of  $N$  students
- We do not know the number of students  $N$  when coding
- $N$  can be very large
- Let's think for a while...



# ARRAY

- A collection/list of variables of the same type
- Accessed through a common name
- Declaration
  - `type var_name[size];`
    - The size must be an integer constant greater than zero
  - `int myarray[20];`
  - `int m[10], a[5];`
- Today we will deal with int, float, double
- Later char inshaAllah



# ARRAY

- Accessed by indexing
  - Known as subscript
  - Can be any valid **integer** expression
  - Begin at 0
  - `myarray[1]` : 2<sup>nd</sup> element
- Array elements are stored in contiguous memory location
  - The lowest address corresponds to the first element



# INITIALIZATION

- `int n[6]={48, 53, 26,71, 9, 12};`
- `float n[]={3.1, -5, 2.5, 17.4, 29};`
  - Array dimension optional

48	53	26	71	9	12
----	----	----	----	---	----

3.1	-5	2.5	17.4	29
-----	----	-----	------	----



# MEMORY LAYOUT

```
int a[5];
```

```
for(int i=0; i<5; i++) a[i]=i;
```

- After the declaration  
20 bytes get reserved in memory
  - Each integer 4 bytes long

	<b>a[0]</b>	<b>a[1]</b>	<b>a[2]</b>	<b>a[3]</b>	<b>a[4]</b>
<b>a</b>	0	1	2	3	4
	4002	4006	4010	4014	4018



# SOME IMPORTANT NOTES

- Array element Can be used anywhere a variable/constant can  

```
int a[5];  
for(int i=0; i<5; i++) scanf("%d", &a[i]);
```
- C compiler does not perform any bound checking on array index
- It is programmer's duty
- Data entered with a subscript exceeding the array size will lead to unpredictable results
- Program may crash  

```
int a[5];  
for(int i=0; i<5; i++) a[i]=i;  
printf("%d\n", a[10]);//a[5]
```



# ARRAY COPY

- It is not possible to assign one entire array to other array

```
int a1[5], a2[5];
```

```
a1=a2; //not possible
```

- Need to copy explicitly

```
for(int i=0; i<5; i++) a1[i]=a2[i];
```





# THE SIEVE OF ERATOSTHENES

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61



# BUBBLE SORT

6 5 3 1 8 7 2 4



# SELECTION SORT

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7



# MULTIDIMENSIONAL ARRAY

- Arrays of two or more dimension
- `int count[10][12];`
- 2-d array
  - Actually an array of one dimensional arrays
  - Easy to think in Row, column format, matrix
  - Usually accessed a row at a time from left to right

	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]				
2					
3	a[3][0]				a[3][4]

Row subscript

Column subscript



# MULTIDIMENSIONAL ARRAY

```
#include<stdio.h>

int main()
{
    int td[4][5];
    int i, j;
    for(i=0; i<4; i++)
        for(j=0; j<5; j++)
            td[i][j]=i*j;
```

```
for(i=0; i<4; i++)
{
    for(j=0; j<5; j++)
        printf("%d ", td[i][j]);
    printf("\n");
}
return 0;
}
```

**Output:**

```
0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12
```



# MULTIDIMENSIONAL ARRAY

- Initialization:

```
int sqr[3][3] ={  
    1,2,3,  
    4,5,6,  
    7,8,9  
};
```

	Col no. 0	Col no. 1	Col no. 2
Row no. 0	1	2	3
Row no. 1	4	5	6
Row no. 2	7	8	9

- Initialization:

- Specify all but the leftmost dimension

```
int sqr[][3] ={  
    1,2,3,  
    4,5,6,  
    7,8,9  
};
```



# MULTIDIMENSIONAL ARRAY

- Initialization:

```
int sqr[3][3] ={  
    {1,2,3},  
    {4,5,6},  
    {7,8,9}  
};  
int sqr[3][3] ={1,2,3,4,5,6,7,8,9};  
int sqr[][3] ={1,2,3,4,5,6,7,8,9};
```



# MULTIDIMENSIONAL ARRAY

- Initialization:

```
int sqr[3][] = {1,2,3,4,5,6,7,8,9};
```

```
int sqr[][] = {1,2,3,4,5,6,7,8,9};
```

- This would **never** work





# MEMORY LAYOUT

- Arrangement of 2-D array in memory
- Memory doesn't contain row and columns
- Elements are stored in one continuous chain

S[0][0]	S[0][1]	S[0][2]	S[1][0]	S[1][1]	S[1][2]	S[2][0]	S[2][1]	S[2][2]
1	2	3	4	5	6	7	8	9
5002	5006	5010	5014	5018	5022	5026	5030	5034



# MULTIDIMENSIONAL ARRAY

```
int th[4][2][7]
```



# MULTIDIMENSIONAL ARRAY

```
int arr[3][3][2]={  
    {  
        {2,4},  
        {7,8},  
        {4,9}  
    },  
    {  
        {7,6},  
        {5,1},  
        {3,4}  
    },  
    {  
        {2,3},  
        {7,2},  
        {9,4}  
    }  
}
```



# THANKS TO

- Johra Muhammad Moosa
  - Lecturer
  - Department of Computer Science & Engineering
  - Bangladesh University of Engineering & Technology

