Working with fixed memory locations

Organised & Supported by **RuggedBOARD**

Agenda

- Arrays
- Array Declaration & Initialisation
- Printing Array Elements
- Array Length
- 2D- Arrays
- 2D-Array Declaration & Accessing Elements

Arrays

Def : Set of similar data elements has shared a common name called an Array.

Syn: varname [size]

Here, size must be a +ve integer only.

size represent maximum size of an Array.

Types of Arrays:

There are two types of arrays.

Those are,

i.Single Dimensional Array [1 - D array]

ii.Multi Dimensional Array [n - D array]

Single dimensional array :

A variable name followed by a single pair of square brackets called 'Single dimensional array'.

Syn: Datatype varname [size];

Ex: int a[5];

Here, 'a' is a variable name and 5 is the maximum size of that array.

That means 'a' can holds 5 integer values.

If a variable is declared as int data type that array is called 'integer array'.

Similarly, 'float array'.

But character array is called a 'string'

Multi dimensional array:

A variable name followed by more than one pair of square brackets ([ ]) called 'Multi dimensional array'.

Syn: Datatype varname [size1] [size2] [size3] ...[size n];

2 - D Array :A variable name followed by two pair of square brackets ([ ]) called '2 - D array'.

Syn: Datatype varname[size1][size2];

Int a[3][2];

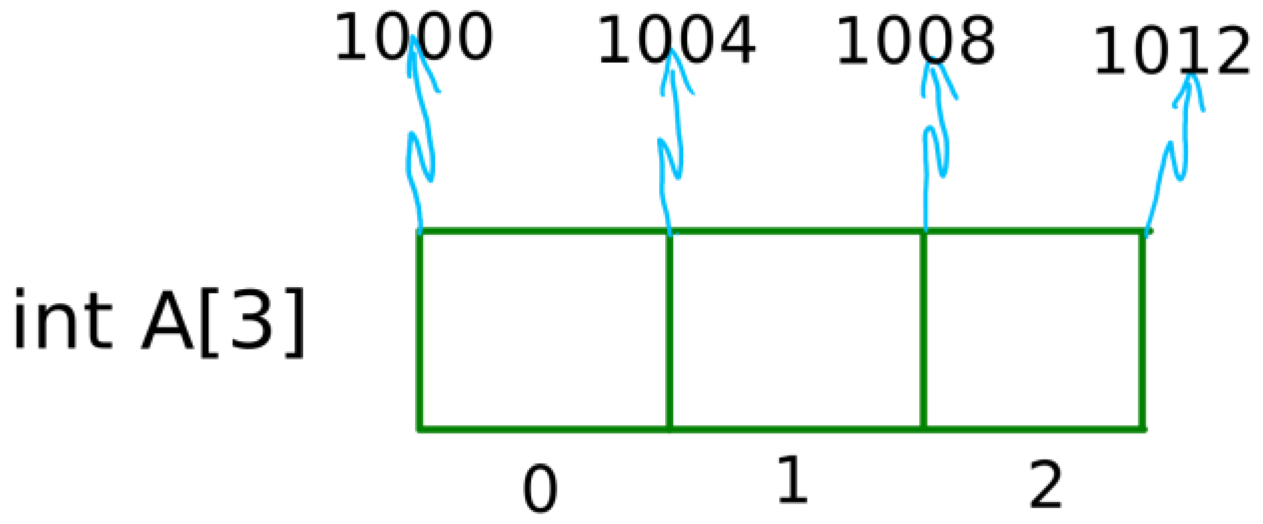
Here, 3 --> rows and 2 --> columns.

Arrays

Arrays are structures that hold multiple variables of the same data type.
An array from integer type holds integer values.

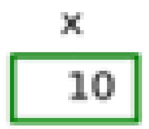
```
int scores[10];
```

- The array "scores" contains an array of 10 integer values.
- We can use each member of array by specifying its index value.
- Members of above array are scores[0],...,scores[9] and we can work with these variables like other variables

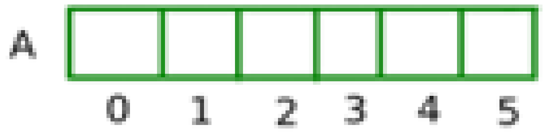


Array Declaration & Initialisation

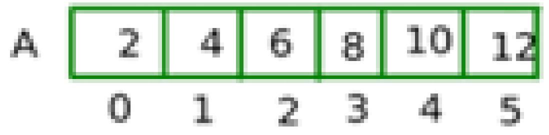
```
int x = 10;
```



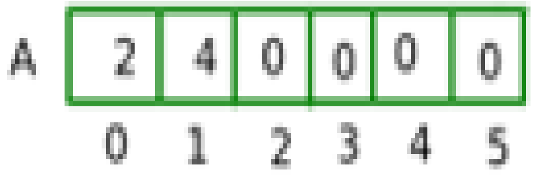
```
int A[6];
```



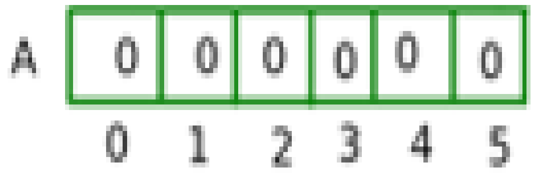
```
int A[6] = {2,4,6,8,10,12};
```



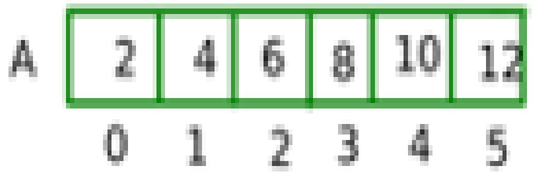
```
int A[6] = {2,4,};
```



```
int A[6] = {0};
```



```
int A[] = {2,4,6,8,10,12};
```



Printing Array Elements

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int A[6]={2,4,6,8,10,12};
        int *p;
        int i;
        for(i=0;i<6;i++)
        {
                printf("%d ",A[i]);
        }
        printf("\n");
        return 0;
}
```

Character strings are arrays of characters.
Each member of array contains one of characters in the string.

```
#include<stdio.h>
Int main()
{
   char name[20];
    printf("Enter your name : ");
    scanf("%s",name);
    printf("Hello, %s , how are you ?\n",name);
    return 0;
}
```

Enter your name : Brian

If user enters "Brian" then the first member of array will contain 'B' , second cell will contain 'r' and so on.

Note: type name by leaving space and check.

Make a Note of it

Accessing Array element :

```
printf("%d ",A[i]);
printf("%d",i[A]);
printf("%d",*(A+i));
```

Printing Array Elements

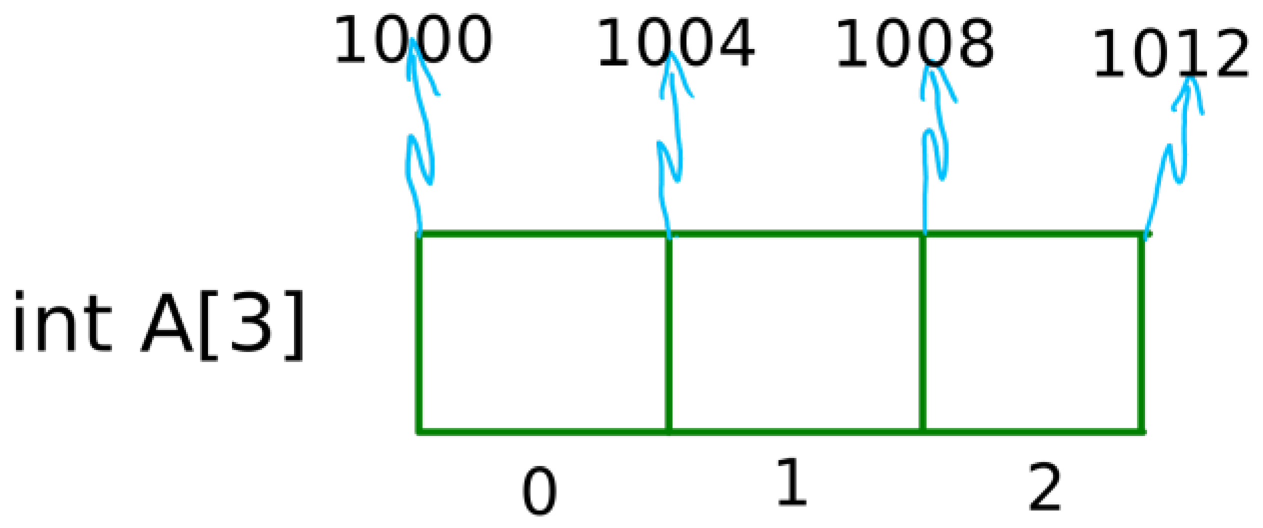
```
#include<stdio.h>
int main()
{
	char name[20];
	printf("Enter your name : ");
	scanf("%[^\n]s",name);
	printf("Hello, %s , how are you ?\n",name);
	return 0;
}
```

```
#include<stdio.h>

int main()
{
	char arr[4] = {'a','b','c','\0'};//if no '\0' need to use  loop to print

	printf("arr[] is %s\n",arr);
	return 0;
}
```

Array Length



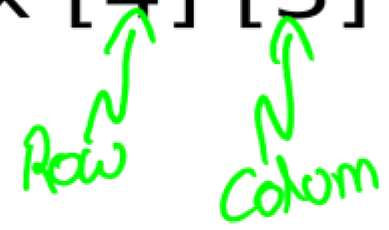
$$
\begin{aligned}
\text{len} &= \text{sizeof}(A)/\text{sizeof}(\text{int}) \\
&= 3*4/4 \\
&= 3
\end{aligned}
$$

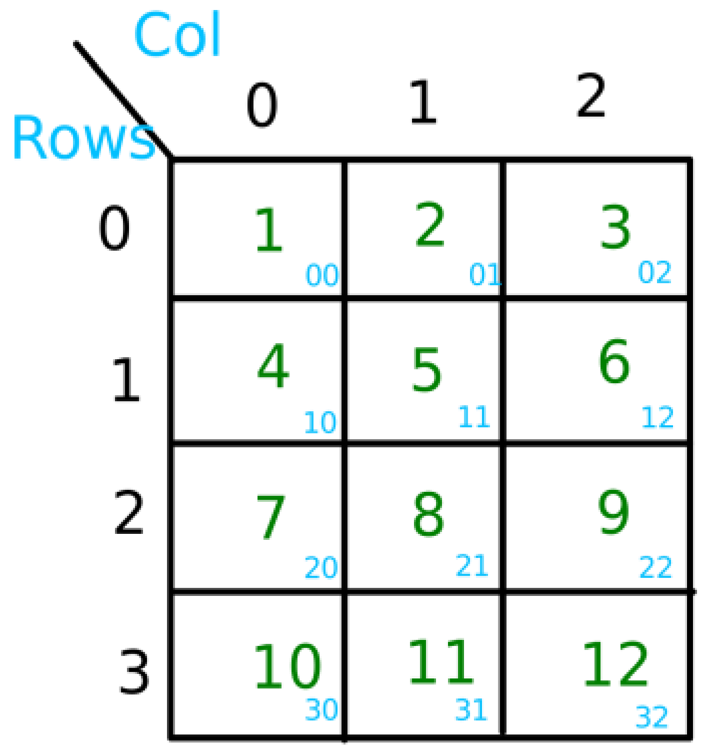
```
#include<stdio.h>

int main()
{
        int arr[10];
        int len = sizeof(arr)/sizeof(int);
        for(int i =0;i<len;i++)
                arr[i] = i*2;
        for(int i = 0;i<len;i++)
                printf("arr[%d]  is %d \n",i,arr[i]);
        return 0;
}
```

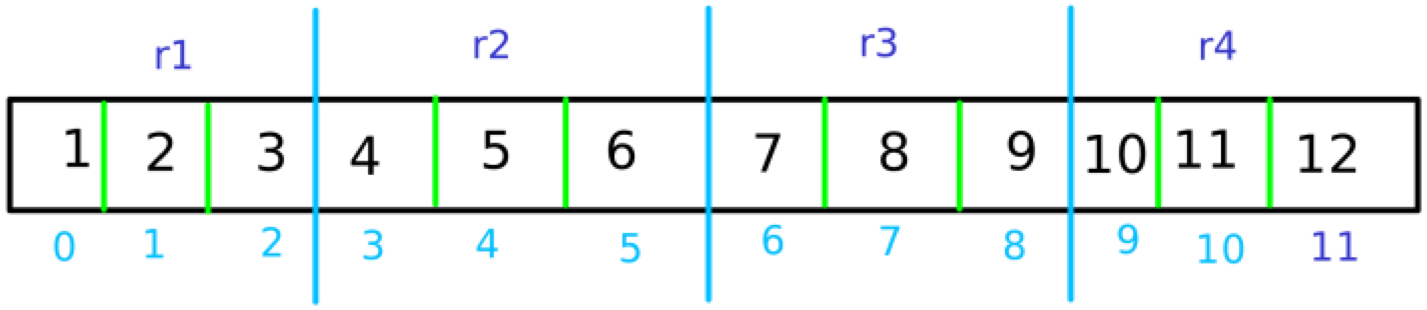
2D-Arrays

int matrix [4] [3] = {1,2,3,4,5,6,7,8,9,10,11,12}





Compiler stores data in below form for 2D array



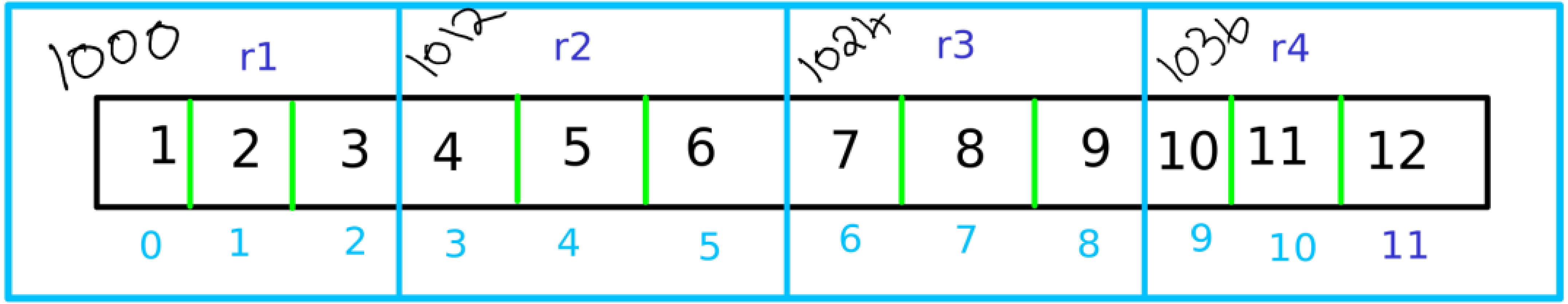
2D-Array Declaration & Accessing Elements

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int A[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};
        for(int i=0;i<4;i++)
        {
                for(int j=0;j<3;j++)
                {
                        printf("%d ",A[i][j]);
                }
                printf("\n");
        }
        return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int A[3][4]={{1,2,3,4},{2,4,6,8},{1,3,5,7}};
        for(int i=0;i<3;i++)
        {
                for(int j=0;j<4;j++)
                {
                        printf("%d ",A[i][j]);
                }
                printf("\n");
        }
        return 0;
}
```

2D-Array Declaration & Accessing Elements

matrix[4][3]



```
printf("%p",(matrix+0));     --> 1000
printf("%p",*(matrix+0));    --> 1000
printf("%p",*(*(matrix+0)); --> 1
```

General way to print 2D-Array

```
printf("%d",*(*(matrix+i)+j)); --> prints value
printf("%p",(*(matrix+i)+j))); --> prints address
```

2D-Array Declaration & Accessing Elements

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
	int A[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};
	for(int i=0;i<4;i++)
	{
		for(int j=0;j<3;j++)
		{
			printf("%d ",A[i][j]);
			printf("%p ",(*(A+i)+j));
		}
		printf("\n");
	}
	return 0;
}
```

Thank YOU