

# Arrays



Collection of similar data types stored at contiguous memory locations

# Syntax

```
int marks[3];
```

```
char name[10];
```

```
float price[2];
```



# Input & Output

```
scanf("%d", &marks[0]);
```

```
printf("%d", marks[0]);
```



# Inititalization of Array

```
int marks[ ] = {97, 98, 89};
```

```
int marks[ 3 ] = {97, 98, 89};
```



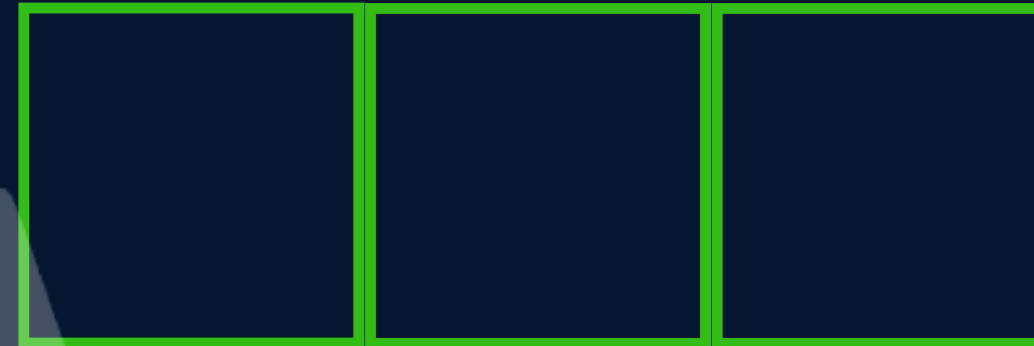
Memory Reserved :

# Pointer Arithmetic

Pointer can be incremented  
& decremented

## CASE 1

```
int age = 22;  
int *ptr = &age;  
ptr++;
```



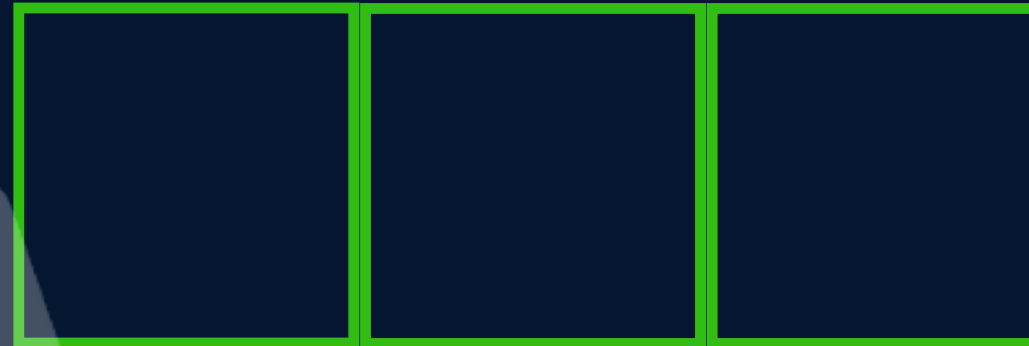
# Pointer Arithmetic

## CASE 2

```
float price = 20.00;  
float *ptr = &price;  
ptr++;
```

## CASE 3

```
char star = '*';  
char *ptr = &star;  
ptr++;
```



# Pointer Arithmetic

- We can also subtract one pointer from another
- We can also compare 2 pointers

# Array is a Pointer

```
int *ptr = &arr[0];
```

```
int *ptr = arr;
```

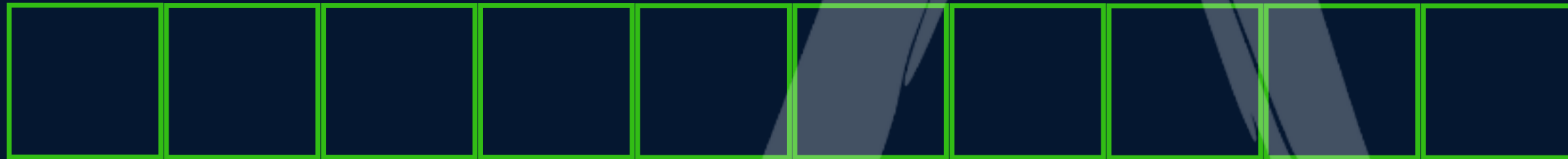




# Traverse an Array

```
int aadhar[10];
```

```
int *ptr = &aadhar[0];
```



# Arrays as Function Argument

//Function Declaration

void **printNumbers** (int arr[], int n)

OR

void **printNumbers** (int \*arr, int n)

//Function Call

**printNumbers**(arr, n);

# Multidimensional Arrays

## 2 D Arrays

```
int arr[ ][ ] = { {1, 2}, {3, 4} }; //Declare
```

//Access

```
arr[0][0]
```

```
arr[0][1]
```

```
arr[1][0]
```

```
arr[1][1]
```