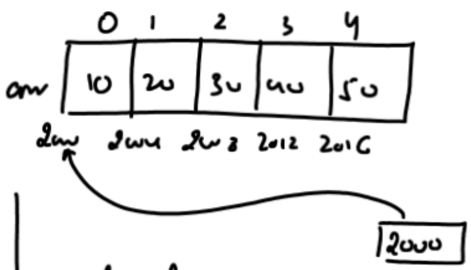
# Passing Array As Argument To Function

```
void display(int * );
int main( )
{
    int arr[5];
    int i;
    for(i=0; i<5; i++)
    {
        printf("Enter no:");
        scanf("%d", &arr[i]);
    }
    display( arr );
    return 0;
}
```

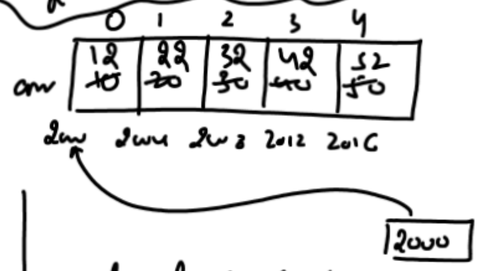


```
void display(int *p)
{
    int i;
    for(i=0; i<5; i++)
        printf("\n %d", *(p+i));
}
```

Arrays are always passed to fn using pass By ref

```
void display(int * );
int main( )
{
    int arr[5];
    int i;
    for(i=0; i<5; i++)
    {
        printf("Enter no:");
        scanf("%d", &arr[i]);
    }
    display( arr );
    for(i=0; i<5; i++)
        printf("\n %d", arr[i]);
    return 0;
}
```

```
12
22
32
42
52
```



```
void display(int *p)
{
    int i;
    for(i=0; i<5; i++)
    {
        printf("\n %d", *(p+i));
        *(p+i) = *(p+i)+2;
    }
}
```

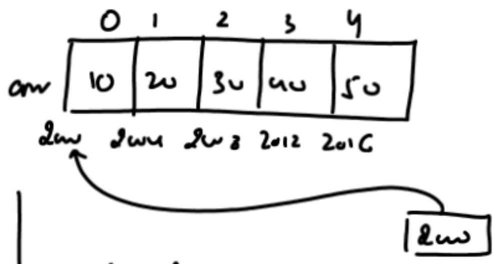
```
10
20
30
40
50
```

## 2nd Way of Passing Array As Argument To Function

```
void display(int[ ]);
int main( )
{
   int arr[5];

   int i;

   for(i=0; i<5; i++)
   {
      printf("Roll no:");
      scanf("%d", &arr[i]);
   }
display(arr);
retn 0;
}
```



```
void display(int brr[5])
{
   int i;
   for(i=0; i<5; i++)
      printf("\n %d", brr[i]);
}
```

## Dynamic Memory Allocation

?

Run Time Memory Allocation

$$\begin{tikzcd} & \text{Memy Alloc} \arrow[dl] \arrow[dr] & \\ \boxed{\text{static or compile time}} & & \boxed{\text{dynamic or run time memy alloc}} \end{tikzcd}$$

```
int a;
int arr[10];
char name[10];
```

Allocating memy at run time acc to user's requirement

## How C supports DMA?

① <u>Pre-defined fn</u> : malloc( )

② <u>Header File</u> : alloc.h → TC

stdlib.h, malloc.h } → MinGW

③ <u>Decl / Prototype</u> : $\boxed{\underbrace{\text{void*}}_{\text{Return Type}}\ \text{malloc}(\overbrace{\text{size\_t}}^{\text{Argument}});}$

## What is size_t ?

## What is typedef ? → A keyword which allows us to provide alternate names to existing data type names

Syntax: typedef old_data_type new name;

For ex: typedef int number;

typedef float decimal;

$\boxed{\text{typedef unsigned int size\_t;}}$

This typedef statement is mentioned in some selected header files:

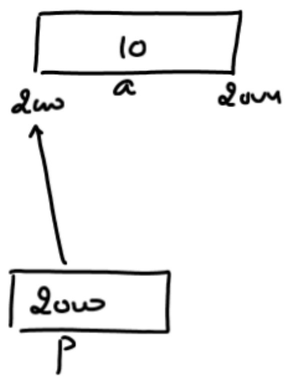
① string.h
② alloc.h
③ stddio.h
④ stdlib.h
⑧ malloc.h

What is a void * ?

A pointer which can point to any variable irrespective of its data type

```
int a = 10;
char b = 'x';
float c = 1.7;
void *p;
p = &a; ✓
p = &b; ✓
p = &c; ✓
```

```
int a = 10;
int *p;
p = &a;
printf("%d", *p); ✓
p++; ✓
```



# PROBLEM WITH void *

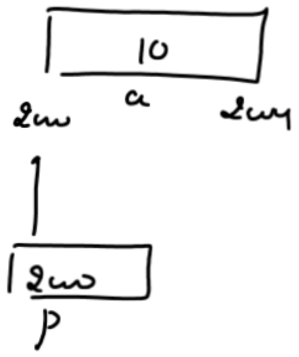
```
int  a = 10;

void * p;

p = &a;

X printf("%d", *p);

printf("%d", *((int *)p));
p++; X
```



We cannot easily dereference or include a void *

```
void *   malloc  ( size_t );
```

Returns the base address of the dynamic array in the form of void *

Size of the dynamic array in number of bytes.

## Example

✓ void *p;

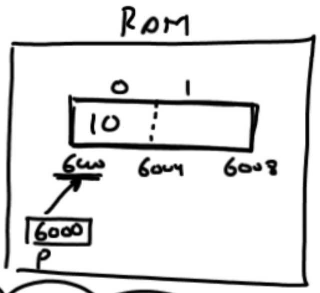
✓ p = malloc( 8 );

X *p = 10;

X p++;

Error:

We cannot de-reference or inc/dec a void* in normal way



✓ int *p;

✓ p = (int *) malloc ( 8 );

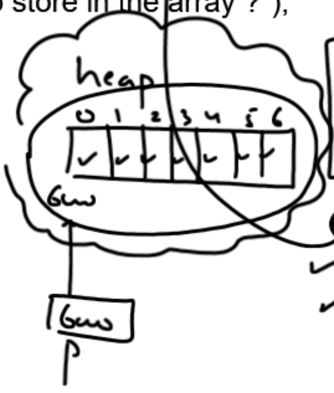
✓ *p = 10;

.
.
.

WAP to create a DYNAMIC INTEGER ARRAY of 'n' elements where 'n' should be taken from the user . Now ask the user to input values in this array and finally display all the array values along with their SUM and AVERAGE

```
✓ #include <stdio.h>
✓ #include <malloc.h>        NULL ⟶ 0
  int main()
  {
✓  int n,i,*p,sum=0;
   printf("How many values you want to store in the array ?");
  scanf("%d",&n); ⟶ 7
  p=(int *)malloc(n*sizeof(int));
 if(p==NULL)
  {
    printf("Insufficient memory");
    return 1;  ⟵ FAILURE
 }
```

```
for(i=0;i<n;i++)
{
 printf("Enter no:");
 scanf("%d",p+i);
}
for(i=0;i<n;i++)
{
 printf("\n%d",*(p+i));
 sum=sum+*(p+i);
}
free(p);
✓ printf("Sum is %d",sum);
✓ printf("\nAvg is %f",(float)
sum/n);
return 0;
}
```


heap
0 1 2 3 4 5 6
6ww
6ww
p