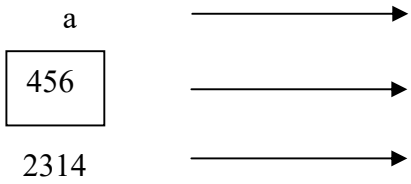
Pointer Introduction

int a;




```
Void main()
{
        int a;
printf("\n Value of a = %d",a);
}
```

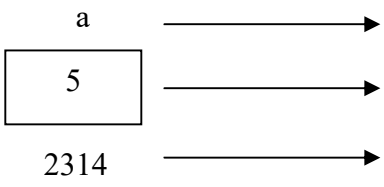
Value of a = 456 (garbage value)

```
Void main()
{
        int a;
printf("\n Value of a = %d",a);
 printf("\n Address of a = %u",&a);
}
```

Value of a = 456 (garbage value)
Address of a = 2314

%u is denote by the address of any variable which is must be & (address of varable)

int a=5;




```
Void main()
{
        int a=5;
 printf("\n Value of a = %d",a);
printf("\n Address of a = %u",&a);
 printf("\n Value of a = %d",*(&a));
```

Value of a = 5
Address of a = 2314
Value of a = 5

```
}
```

```
int a=5;
int *ptr;
```


```
prt=&a
```

```
Void main()
{
        int a=5;
        int *ptr;
        ptr=&a;
printf("\n Value of a = %d",a);
 printf("\n Address of a = %u",&a);
printf("\n Value of a = %d",*(&a));
printf("\n Value of a = %d",*ptr);
}
```

```
Value of a =  5
Address of a =  2314
Value of a =  5
Value of a =  5
```

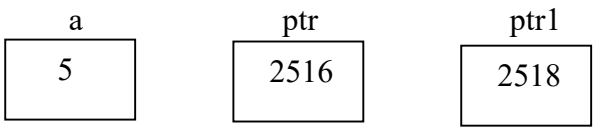
Pointer is the variable which store the address of the same data type of next varable
A *pointer* is a variable that holds a memory address
Pointer Declaration
Syntex datatype *pointer varaible
Example int *ptr
Pointer Assigne
ptr=&a
Where is ptr is pointer type variable and a is simple variable and both are same data type

```
int a=5;
int *ptr;
ptr=&a
```

Pointer initialize
Pointer assigne

```
int a=5;

int *ptr;
int **ptr1;
ptr=&a;
```



ptr1=&ptr;	

void main()
{
int a=5;
int *ptr;
int **ptr1;
clrscr();
printf("\n Value of a = %d",a);
ptr=&a;
printf("\n Value of a = %d",*ptr);
ptr1=&ptr;
printf("\n Value of a = %d",**ptr1);
}
	Value of a =5
	
	Value of a =5
	
	Value of a =5

Call by Value & Call by refference

In function call
when the value is pass in call function is called the 'call by value' and when the address is pass in function in call function is called 'call by refference'

	Call by Value	Call by refference
Prototype	void swap(int a);	void swap(int *a);
Call	swap(a)	swap(&a)

Define	void swap(int a)
	{
	statement;
	}
		void swap(int *a)
		{
		statement;
		}

Concept	In this method the 'value of' in the calling function is copied in the corresponding formal arguments of the called function.	In call by address , the function is allowed access to the actual memory location of the argument

Effect	The changes made to the formal arguments in the called function have no effect on the values of actual arguments in the calling function.	The changes made to the formal arguments in the called function have effect on the values of actual arguments in the calling function.

CALL BY VALUE

1. Write a C program to calculate the area of circle given radius of circle

```
#include<stdio.h>
void funarea(int );
void main()
{
        int r;
        float a;
        clrscr();
        printf("\n Enter the Radius = ?");
        scanf("%d",&r);
        funarea(r);
getch();
}
void funarea(int r)
{
        float a;
        a=3.14*r*r;
printf("Area of Circle = %f",a);
}
```

CALL BY REFFRENCE

2. Write a C program to calculate the area of circle given radius of circle using call by reference

```
#include<stdio.h>
void funarea(float *p, int r );
void main()
{
        int r;
        float a;
        clrscr();
`       printf("\n Enter the Radius = ?");
```

```
            scanf("%d",&r);
            funarea(&a, r);
        printf("Area of Circle = %f",a);
        getch();
        }
        void funarea(float *a , int r)
        {
            *a=3.14*r*r;
        }
```

3. Write a C program implement a call by value and call by reference with an example.

CALL BY VALUE

```
#include<stdio.h>
#include<conio.h>
void swap(int a, int b);
void main()
{
    int a,b;
    clrscr();
    printf("\n Enter two Number ");
    scanf("%d%d",&a,&b);
    printf("\n A=%d B=%d",a,b);
    swap(a,b);
    printf("\n A=%d B=%d",a,b);
    getch();
}
void swap(int a, int b)
{
    int c;
    c=a;
    a=b;
    b=c;
    printf("\n A=%d B=%d");
}
```

CALL BY REFFERENCE

```
Void swap(int *a, int *b);
Void main()
{
        int a,b;
        clrscr();
        printf("\n Enter two Number ");
        scanf("%d%d",&a,&b);
        printf("\n A=%d B=%d",a,b);
        swap(&a,&b);
        printf("\n A=%d B=%d",a,b);
        getch();
}
void swap(int *a, int *b)
{
        int c;
        c=*a;
        *a=*b;
        *b=c;

}
```