# Pointers asses:

1 Write a program to print the address of a variable , its value ,its size, try different executions and note the address printed. Assign the address of the variable to a pointer variable.Print the size of pointer and its indirected value (apply * operator)

```
#include<stdio.h>
void main(){
int var=30;
int *ptr=&var;
printf("value of var:%d\n",var);
printf("address ofvariable:%d\n",ptr);
printf("size of var:%lu\n",sizeof(var));
printf("size of pointer:%lu\n",sizeof(ptr));
}
```

Output:

```
value of var:30
address ofvariable:-1151500812
size of var:4
size of pointer:8
```

2 Write a program to add two variables using their pointers.

```
#include<stdio.h>
void main(){
int a=30,b=20;
int *ptr1=&a;
int *ptr2=&b;
int c=*ptr1+*ptr2;
printf("addition of a and b :%d",c);
}
```

Output:

```
addition of a and b :50
```

3 Write a program to take input for two character variables using pointers and find out which character has higher ascii value.

```
#include<stdio.h>
int main(){
char ch1,ch2;
printf("enter ch1:");
scanf(" %c",&ch1);
printf("enter ch2:");
scanf(" %c",&ch2);
char *ptr1=&ch1;
char *ptr2=&ch2;
if(*ptr1>*ptr2){
printf("the ch1 is grater than ch2");
}
else if(*ptr1<*ptr2){
printf("the ch2 is gratere than ch1");
```

```
}
else{
printf("both are same");
}
return 0;
}
```

output :
enter ch1:a
enter ch2:a
both are same

4.Declare 3 integer variables. Declare an integer pointer. Assign the address of each variable to the pointer in succession and print the value of the variable using indirection operator on the pointer.

```
#include<stdio.h>
int main(){
int a=20,b=30,c=40;
int *ptr;
ptr=&a;
printf("value of a using pointer:%d\n",*ptr);
ptr=&b;
printf("value of b using pointer:%d\n",*ptr);
ptr=&c;
printf("value of c using pointer:%d\n",*ptr);
return 0;
}
```

Output:
value of a using pointer:20
value of b using pointer:30
value of c using pointer:40

5 Declare 3 pointer variables of integer type, and an integer variable. Assign the address of integer variable to the 3 pointers. Print the variable value using indirection on each pointer variable. Change the variable value directly and check the values of each of the pointers using indirection. Change the variable value , using each pointer and print the variable value and indirected values of all pointers every time you change.

```
#include<stdio.h>
int main(){
int num=100;
int *p1,*p2,*p3;
p1=&num;
p2=&num;
p3=&num;
printf("initial values of pointrs:\n");
printf("value using p1:%d\n",*p1);
printf("value using p2:%d\n",*p2);
```

```
printf("value using p3:%d\n",*p3);
*p1=200;
printf("num:%d\n",num);
printf("value of p2:%d\n",*p2);
printf("value of p3:%d\n",*p3);
*p2=300;
printf("num:%d\n",num);
printf("value of p1:%d\n",*p1);
printf("value of p3:%d\n",*p3);
*p3=400;
printf("num:%d\n",num);
printf("value of p1:%d\n",*p1);
printf("value of p2:%d\n",*p2);
return 0;
}
output:initial values of pointrs:
value using p1:100
value using p2:100
value using p3:100
num:200
value of p2:200
value of p3:200
num:300
value of p1:300
value of p3:300
num:400
value of p1:400
value of p2:400
```

6 declare different pointers with different data types. Print the sizes of the pointer variables using sizeof operator. Why do u think all of them are giving same size irrespective of the data type ?

```
#include<stdio.h>
void main(){
char name='A';
int id=2;
float marks=88.00;
char *ptr1=&name;
int *ptr2=&id;
float *ptr3=&marks;
printf("name:%c\n",*ptr1);
printf("id:%d\n",*ptr2);
printf("marks:%f\n",*ptr3);
printf("size of ptr1:%lu\n",sizeof(name));
printf("size of ptr2:%lu\n",sizeof(id));
printf("size of ptr3:%lu\n",sizeof(marks));
}
output:
name:A
id:2
```

marks:88.000000
size of ptr1:1
size of ptr2:4
size of ptr3:8

7 Write a program to find the biggest of three numbers using pointers that point to those numbers.

```
#include<stdio.h>
void main(){
int a,b,c;
printf("enter the a value:");
scanf("%d",&a);
int *p1=&a;
printf("enter the b value:");
scanf("%d",&b);
int *p2=&b;
printf("enter the c value:");
scanf("%d",&c);
int *p3=&c;
int *biggest=p1;
if(*biggest<*p2){
biggest=p2;
}
if(*biggest<*p3){
biggest=p3;
}
printf("the biggest value is :%d",*biggest);
}
```

output:

```
`enter the a value:5
enter the b value:8
enter the c value:2
the biggest value is :8
```

8 Take three input integers x,y and z. Write a program to rotate their values such that, x has the value of y, y has the value of z and z has the value of x. Do this using pointers that point to x,y and z.

```
#include<stdio.h>
void main(){
int x,y,z;
printf("enter x value:");
scanf("%d",&x);
int *p1=&x;
printf("enter y value:");
scanf("%d",&y);
int *p2=&y;
```

```
printf("enter z value:");
scanf("%d",&z);
int *p3=&z;
int temp=*p1;
*p1=*p2;
*p2=*p3;
*p3=temp;
printf("\nafter rotation:\n");
printf("x:%d\n",*p1);
printf("y:%d\n",*p2);
printf("z:%d\n",*p3);
}
```

output:
enter x value:1
enter y value:2
enter z value:3

after rotation:
x:2
y:3
z:1

9 Declare an integer array of size 10 and initialize it to some values. Print the addresses of each element of the array using a pointer. using indirection operator , print the value stored in each element of the array.

```
#include<stdio.h>
int main(){
int arr[10];
printf("enter array elemts:");
for(int i=0;i<10;i++){
scanf("%d",&arr[i]);
}
int *ptr=&arr[10];
printf("element\taddress\tvalue\n");
for(int i=0;i<10;i++){
printf("arr[%d]\t%p\t%d\n",i,(ptr+i),arr[i]);
}
return 0;
}
```

output:
enter array elemts:1
2
3
4
5
6
7
8
9
10

element address value
arr[0] 0x7fff82aadfb8 1
arr[1] 0x7fff82aadfbc 2
arr[2] 0x7fff82aadfc0 3
arr[3] 0x7fff82aadfc4 4
arr[4] 0x7fff82aadfc8 5
arr[5] 0x7fff82aadfcc 6
arr[6] 0x7fff82aadfd0 7
arr[7] 0x7fff82aadfd4 8
arr[8] 0x7fff82aadfd8 9
arr[9] 0x7fff82aadfdc 10

## Call by value and call by reference:

1) Write a program to swap two numbers using Call by Value and Call by Reference

```
#include<stdio.h>
int swapbyvalue(int *a,int *b){
int temp=*a;
*a=*b;
*b=temp;
printf("inside swapbyvalue a:%d,b:%d\n",*a,*b);
}
int main(){
int x=10,y=20;
printf("before swaping x=%d,y:%d\n",x,y);
swapbyvalue(&x,&y);
printf("after swaping x=%d,y:%d\n",x,y);
return 0;
}
```

output:
before swaping x=10,y:20
inside swapbyvalue a:20,b:10
after swaping x=20,y:10

2) Note down the differences between Call by Value and Call by Reference and when to use what.

Call by Value
In Call by Value, when a function is called, a copy of the actual argument (i.e., the value stored in the variable) is passed to the function. This means any changes made to the parameter inside the function do not affect the original variable in the calling function.

Call by Reference
In Call by Reference, the function receives the memory address (reference) of the actual argument. Thus, the function works with the original variable directly. Any changes made to the parameter inside the function reflect on the original variable

3) Write a function that can rotate the values of three variables. print the results in the main function

```
#include <stdio.h>
void rotate(int *a, int *b, int *c) {
int temp = *a;
*a = *c;
*c = *b;
*b = temp;
}

int main() {
int x = 10, y = 20, z = 30;

printf("Before rotation:\n");
printf("x = %d, y = %d, z = %d\n", x, y, z);

rotate(&x, &y, &z);

printf("After rotation:\n");
printf("x = %d, y = %d, z = %d\n", x, y, z);

return 0;
}
```

output:
Before rotation:
x = 10, y = 20, z = 30
After rotation:
x = 30, y = 10, z = 20

4) Write a function that can take two integers as input, and gives 5 outputs : addition, subtraction, multiplication, quotient and reminder of those two numbers. Print the outputs in the main function.

```
#include<stdio.h>
void fun(int a,int b){
int sum=a+b;
int sub=a-b;
int mul=a*b;
int quofi=a/b;
int remen=a%b;
printf("addition=%d\n",sum);
printf("subtraction=%d\n",sub);
printf("multiplication=%d\n",mul);
printf("quofient=%d\n",quofi);
printf("remender=%d\n",remen);
}
int main(){
fun(6,3);
return 0;
}
```

output:
addition=9
subtraction=3
multiplication=18
quofient=2
remender=0

5) "Write a function that communicates with main using a single static variable without taking any input arguments.
Everytime function returns something using the static variable,after using it, main sends another input using the same variable and calls the function again.
eg., print the square of each number of an array :
for each number of the array :
call the function
main gets the static variable address as return value from function.
main puts the array element in static variable.
in the function :
create static variable.
if static variable value is not zero, print its square.
function sends static variable address back to main."

```
#include <stdio.h>
int* communicate() {
static int value = 0;
if (value != 0) {
printf("Square: %d\n", value * value);
value = 0;
}
return &value;
}
int main() {
int arr[] = {2, 4, 6, 8, 10};
int size = sizeof(arr) / sizeof(arr[0]);
for (int i = 0; i < size; i++) {
int* ptr = communicate();
*ptr = arr[i];
communicate();
}
return 0;
}
```

output:
Square: 4
Square: 16
Square: 36
Square: 64
Square: 100

6) Write two source files, main.c and swap.c. The main function initializes a two-element array of ints, and then calls the swap function to swap the pair.

```
#include <stdio.h>
#include "swap.h"
```

```
int main() {
int nums[2] = {5, 10};

printf("Before swap: nums[0] = %d, nums[1] = %d\n", nums[0], nums[1]);

swap(&nums[0], &nums[1]);

printf("After swap: nums[0] = %d, nums[1] = %d\n", nums[0], nums[1]);

return 0;
}
file2:
#include "swap.h"
void swap(int* a, int* b) {
int temp = *a;
*a = *b;
*b = temp;
}
output:

Before swap: nums[0] = 5, nums[1] = 10
After swap: nums[0] = 10, nums[1] = 5
```