3. Pointers.

(Marks:-18)

Q.1 Introduction of Pointer.

Ans.

In a C Pointer is a variable that points to or reference a memory location in which data is store.

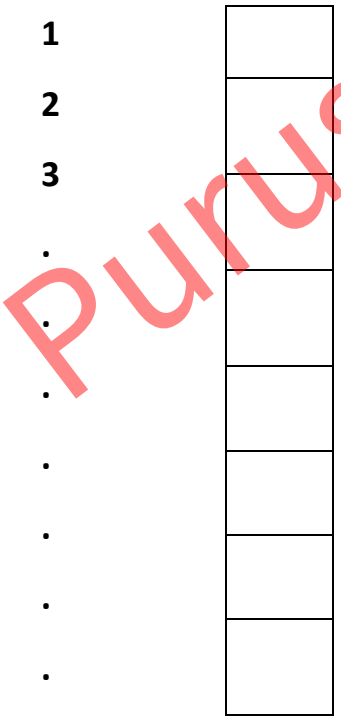
- ☞ Each memory self in the computer has an address that can be used to access that location so as a pointer variable point to a memory location we can access & change the contain of this memory location via pointer.

Q.2. Understanding Pointer.

Ans.

The computer's memory is a sequential collection of storage cell.

Address Memory cell



Syntax: -

data type * Variable name;

Ex.:-

int a=150;

memory location 5000.

Q.3. Accessing the address of a variable.

Ans.

- ☞ The actual location of a variable in the memory is system dependent.
- ☞ The address of a variable isn't known to us immediately.

Q.4.How can being then determine the address of a variable?

Ans.

- ☞ This can be done with the help of the operator & available in C.
- ☞ We have already seen the use of address operator in the scanf().

Ex.

```
main()
{
    int a=8;
    printf("%d",a);
    printf("%u",&a);
 getch();
}
```

Output:-

a=8

4566

Ex.:-

```
main()
{
  char a=2;
  printf("%c",a);
  printf("%u",&a);
 getch();
}
```

Ex 19 :-

```
{
  char q;
  int x;
  float p;
  a='A';
  x=125;
  p=10.25;

 printf("%c is stored at address of %u", a,&a);
 printf("%d is stored at address of %u", x,&x);
 printf("%f is stored at address of %u", p,&p);

getch();
}
```

Q.5. Declaring pointer variable: -

Ans.

- ☞ In a C every variable must be declare for its type.
- ☞ Since pointer variable contain addresses that belong to a separate data type they must be declare as pointer before we use them.

Syntax: -

Data type * variable name;

Ex.

int * p;

- ❖ **Rules of pt.name :-**

1. The asterisk (*) tells that variable pt.name is a pointer variable.
2. Pt.name needs a memory location.
3. Pt.name points to a variable of data type (variable).

Q.6. Initialization of pointer variable:-

Ans.

- ☞ The process of accessing the address of variable to a pointer variable it's called initialization.
- ☞ Once the pointer variable has been declaring we can use the assignment operator to the initialization variable.

Ex.:-

int x;

int * p; // pointer variable declaration

p=*x; initialization.

Or.

we can also combine the initialization with the initialization with the declaration.

Ex.: -

int *p=&x;

☞ We could also define a pointers variable with an initial value null or zero.

Ex.:-

Int *p=&x;

☞ We could also define a pointer variable with an initial value null or zero.

Ex.

int *p=NULL

int *p=0

} **Viva.**

Q.7.Accessing a variable through it's pointer.

Ans.

☞ Pointer has been assign the address of a variable. The question remains as to how to access the value of the variable using pointers.

☞ We can use * operator in the pointer it's called indirection operator.

Ex.

```
main()
{
    int x,y;
    int *ptr;
    x=10;
    ptr=8x;
    y=*ptr;
     printf("%d" x is stored at %u,x,&x);
```

```
        printf("%d" y is stored at %u,y,&y);
    getch();
}
```

Q.8. Pointer Expression.

Ans.

☞ Pointer variable can be uses in expression.

Ex.

int x,y,*p_1,*p_2;

x=P_1 * P_2;

Or.

X=(*P_1) * (*P_2);

Sum=Sum + *P_2;

*P_2= 10+ *P_2;

☞ *P_1 and *P_2 are properly declare an initialize pointers then the these statement are valid.

Ex.

P++;

☞ In addition the arithmetic operation discussed & pointers can also be compared using the relation operators.

Ex.

P_1> P_2;

P_1==P_2;

P_1!=P_2

☞ We may not use pointer in division or multiplications.

Ex.

$P_1/3$;

P_1/ P_2;

$P_1 * P_2$

} Are Not allowed.

☞ This statement are not allowed semiyearly two pointer cannot added.

Ex.

$P_1 + P_2$;

It is illegal.

Ex.

(ca * c- (*P_2))) / (*P_1)) * +10;

Or

C*-*P_2 / *P_1 + 10;

Q.9. Pointer increment & scale factor:-

Ans.

☞ We have seen that pointer can be increment like as

$P_1 = P_1 + 1$;

$P_2 = P_2 + 2$;

P++;

☞ When we increment a pointers it's value is increased by the length of the data type that is point to the length call scale factor.

Ex.

The IBMPC the length of various data type are as follow.

Data Type	Length
Character	1 Byte
int	2 Byte
Float	4 Byte
Long int	4 Byte
Double	8 Byte

☞ The number of bytes use to stored various data type depends on the system & can be found making use of size operator.

Ex.

char * my char;

my char++;

		++	
	My char		

int *P: P++

Notes :-

System like Pentium use four bytes for storing integers & (2) bytes for short integers.

Q.10. pointer & Array: -

Ans.

When an array is declaring the complies a base address & sufficient amount of storage to contain all the elements of the array in memory location.

Ex.

int X [5]= {1,2,3,4,5}

index	x[0]	x[1]	x[2]	x[3]	x[4]
value	1	2	3	4	5
Address	1000	1002	1004	1006	1008

- ☞ The name X is define as a constant pointer pointing to the first element X [50] and therefore the value of x is 1000 therefore the value of x is 1000 the location where x[0] is store.

Ex.

X= & * [0] =1000

- ☞ We declare P as an integer pointer then we can make the pointer P to point the array X.

```
int X , *p;'
p=&x;
is equal
  int x[5], p;
p=8 *[0];
```

Ex.:-

P-20:-

```
main()
{
  int I, sum=0;
  int a[5]*p;
  clrscr();
  printf("\n enter the num= ");
  for(i=1;i<=5,i++)
  {
     scanf("%d",&a[i]);
  }
```

```
for(i=1;1<=5;i++)
 p=&a[a];
{
    sum=sum + *(p+i);
}
    printf( “Sum is=%d”,sum);l
getch();
}
```

Output: -

1

2

3

4

5

Sum=15

Q.11.Pointer & character string: -

Ans.

We declare character array initialize as follow.

Ex.

```
char str[5]=”good”;
```

The complier automatically insert the null character “ \o” at the and of the string.

We can create string using pointer variable of type char.

Ex.

```
char * str[s]=”good”;
```

g	o	o	d	\o

Ex.

```
printf("%s",str);
puts (str);
```

Q.12.Pointer as function argument:-

Ans.

- ☞ We pass address to a function the parameters receiving the address should be pointers.
- ☞ The process of calling a function using pointers to pass the address of variable is known as "call by reference".
- ☞ The function which is call by reference can change the value of the variable use in the call.

Ex.

```
C=add (*a,*b);
C=add(&a,&b);
```

Ex.

```
(swap)
void exchange (int *,int*);
main()
{
    int x,y;
    x=100, y=200;
    exchange (&x, &y);
    printf("x=%d, y=%d" x,y);
```

```
}
exchange (int*a, int *b)
{
        int= tmp;
       tmp=*a;
       *a= *b;
       *b=tmp;
}
```

Output:-

x=200

y=100

Q.13.Pointer & structure.

Ans.

We know than the name of an array stand for the address of 0 length.

☞ We use this notation:-

(* ptr).name ← member

For ex.

(*st1).name

☞ The parenthesis around *ptr are necessary because the members operator has a higher presidency then operator *.

Ex.

```
struct name
{
    int a;
}
```

```
main()
  {
      struct name *ptr;
      clrscr()
      printf("Enter the no:- ");
      scanf("%d",&(*ptr).a);
      printf("%d",(*ptr).a);
  }
```

Output: -

Enter the no: -10

a=10