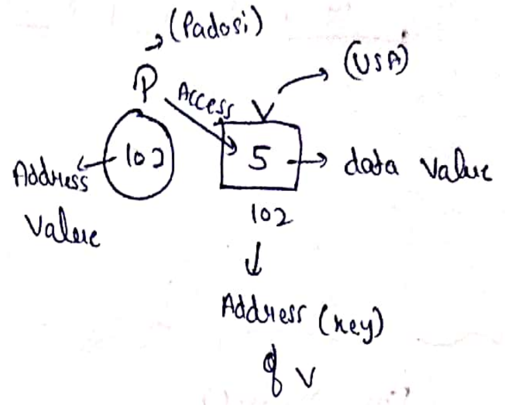
★ Pointers :-

Pointers are special variables which can store variable. We can access and modify the value of a variable of which it contains the address.

```
int main()
{
    int v = 5;
    int * p ; // Declaration of pointer
                 variable
int*p=&v;
    p = &v ; // storing address
               of v in p

    printf(" v = %d \n", v); // 5
    printf("\n &v = %d", &v) ; // 102
    printf("\n p = %d", p) ; // 102
    printf("\n *p = %d ", *p) ; // 5 (Access)
                            ③②
                          ↙   ↘
                     Value at → 102
                          ③

    ~~v~~ 9;

    *p = 9; // modify → modification
    printf("\n v = %d ", v) ; //9

    printf("\n %d ", *&*&v); //9

    printf("\n %d", *&*p); //9
```

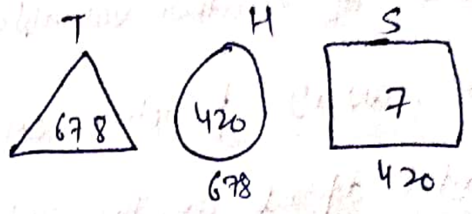
We have to process the pointers by moving from Right to Left.



$\boxed{v == *p}$

★ **Pointers to Pointers** :- A pointer that can store address of another pointer is known as pointer to pointer. It is declared multiple indirection operator (*) also called de-referencing operator.

```
int main ( )
{
    int S = 7;
    int * H = &S;
   int ** T ; // Declaration of pointer to
                pointer
     T = &H;
    printf ("\n S = %d", S); // 7
     printf ("\n T = %d", T); // 678
      printf ("\n * T = %d", * T); // 420
```



$\boxed{**T == S}$

$\boxed{*H == S}$

Value at 678

```
  printf ("\n ** T = %d", **T); // 7 (Access)
```

Value at 420

```
        ** T = 9; // modification

          printf (" *H = %d ", * H); // 9
   }
```

Note :- Except - (minus), no arithmetic operation can be performed on 2 addresses.

Home Work

Q- Write a short note on pointers.

A- The Pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture. However, in 32-bit architecture the size of a pointer is 2 byte.

The pointer in C language can be declared using * (asterisk symbol). It is also known as indirection pointer used to dereference a pointer.

```
int*   a; // Pointer to int
char*  c; // Pointer to char
float* b; // Pointer to float
```

Q- Program for Area of Rectangle using pointers.

A-
```
int main ( )
{  int L, B, A;
   printf ("Enter Length & Breadth : \n");
   scanf ("%d %d", &L, &B);
   int *P = &L, *R = &B;
   A = (*P) * (*R);

   printf (" Area = %d", A);
}
```

Q- Write a program for Simple interest using pointers.

A-
```
int main ()
{
    int P, R, T, S

    printf (" Enter principal value, Rate of interest , time: \n")
    Scanf ("%d %d %d", &P, &R, &T);

    int *RM= &P , *Q = &R , *C = &T ;
    S = (*RM) * (*Q) * (*C) /100;
  printf (" Simple interest = %d", s);
 printf (" Total amount = %d", S+(*MR));
}
```

Short Question

①
```
int main ()    // Write error if (any)
{
    int a= 10; Ptr = &a;
    printf ("%d", *Ptr);
}
```

Ans. :- 10

②
```
int main()
{
    int a = 10;
    float * ptr;
    v = &a;
    printf ("%f", * ptr);
}
```

Error :- v is undeclared , ∴ &a → undeclared

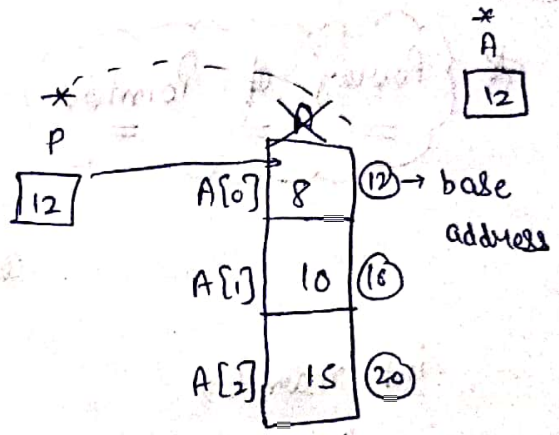
③
```
int main ()
{
    int a = 10, b = 20;
    printf ("%u", &a + &b);
}
```

Error :- Sum of address Can't be operated

④ // Find Error
```
int main ()
{
    int x = 2, y = 8, z, * p1 = &x , * p2 = &y
    z = *p1 + * & *p2;
    printf ("%d", *&z);
}
```

Z = 10

★ **Pointer to Array** :- A pointer containing base address of array is known as pointer to array. Using pointer to array we can access and modify the whole array elements. Eg:- address of 0th location is known as base address of array.



```
int main ()
{
    int N, i, *P;
    printf ("Enter N:\n");
    Scanf ("%d", &N); // 3
   int A[N] ; // creation of array.
   for (i=0 ; i<N; i++)
    {
      printf ("Enter value at A[%d] : \n", i);
      Scanf ("%d", & A[i]);
    }
      P= &A[0] ; // Storing Base address of array.
   printf (" \n output\n");
   for (i=0 ; i<N ; i++)
    {
      printf ("%d\n", A[i]); // 8 10 15
      printf ("%d\n", *(P+i));
      printf ("%d\n", *(i+P));
      printf ("%d\n", *P++);
                      ①  ②
```

*(P+i) ⇒ 8
*(12+0) ⇒ 8 (12) → adress
*(12+1) ⇒ 10 (16)
*(12+2) ⇒ 15 (20)

```
printf ("%d \n", * (A+i));
printf ("%d \n", * (i+A));
printf ("%d \n", i[A]);
printf ("%d \n", P[i]);
}
```

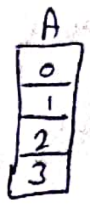
★ **Array of Pointers** :- An array Containing address of multiple values is known as array of pointers.

```
int main ()
{ int A=5, B=6, C=8, i
  int * P[3];
    P[0] = &A;
    P[1] = &B;
    P [2] = &C;
   printf ("output \n");
   for (i=0 ; i<3 ; i++)
    {
      printf ("\n Address = %d and Value = %d \n", P[i], * P[i]);
        Sum = Sum + * P[i];
    }
    printf (" \n Sum = %d", Sum);
}
```

Home work

Q- Program for greatest element in array Using pointers to array.

A -
```
int main()
{  int N, i, *P;  int max;
   printf ("Enter value of N: \n");
   Scanf ("%d", &N);
   int A[N];
   for (i=0 ;i<N; i++)
   {
     printf (" Enter value at A [%d] :\n",i);
     Scanf ("%d", & A[i]);
   }
   P = & A[0]  ;   max = A[0];
   for (i=0 ;i<N ;i++)
    {
      if (max < *(P+i))
         max = *(P+i);
    }
   printf (" Greatest element = %d\n", max);
}
```



Q- Write a program for Searching an element using Pointer to Array.

A-
```
int main ()
{
    int N, i, *P;
    printf ("Enter the value of N: \n");
    scanf ("%d", &N);
    int A [N].
    for (i=0 ; i<N; i++)
    {
        printf (" Enter the value at A [%d] : \n", i);
        scanf (" %d", A[i]);
    }
    P = &A [0];
    printf (" Enter element want to search : \n");
    scanf (" %d", & wanted).
    for (i=0 ;i< N; i++)
    {
        if (wanted == *(P+i))
        {printf (" Searched element : %d \n", *(P+i));
        printf (" Searched at Location : %d \n", i);
        }

    }
}
```

Q- what is Pointer arithmetic.

A - Pointer Variables are also known as address data types because they are used to store the address of another Variable. The address is the memory location that is assigned to the variable. It doesn't Store any value. Hence, there are only few operations that are allowed to perform on pointers in C language. The operations are slightly different from the ones that we generally use for mathematical Calculations. The operations are:

(i) Increment | Decrement of a pointer.

(ii) Addition of integer to a pointer.

(iii) Subtraction of integer to a pointer.

(iv) Subtracting two pointers of Same type.

Q - what is Void Pointer. Explain with Example.

A - A void pointer is a pointer that has no associated data type with it. A void pointer Can hold address of any type and Can be typecasted to any type.

Eg :-
```
# include < stdio.h>
    int main ()
    {
        int a = 10;
        Void * Ptr = &a;
        Printf ("%d", * Ptr);
        return = 0;
    }
```

Scanned with CamScanner