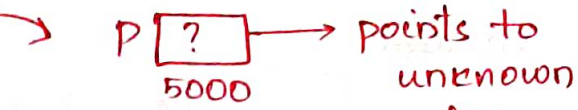
ex) int q; Initialization of ptr var:-

q=5

int *p;

p=&q;



int *p=, &q; ✓

float a,b;

int x,*p;

p=&a; ✗

int x,*p=&x ✓

int *p = &x, x ✗

Initialization with Null:-

int *p = NULL;

int *p = 0;

Flexible:-

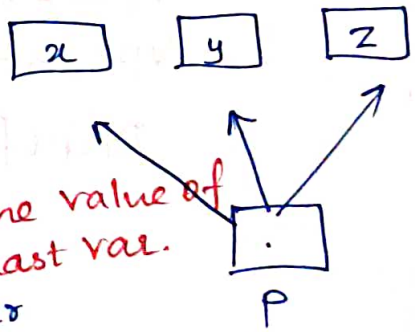
int x,y,z,*p;

p=&x;

p=&y;

p=&z; → always have the value of last var.

same pointer points to dif vars

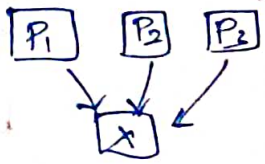


int x;

int *p1=&x;

int *p2=&x;

int *p3=&x;



Same var pointed by many pointers.

with the exception of NULL & 0, no other constant value is assigned to a pointer variable.

```
int *p = 5360;
```
✗ wrong.

NULL pointer

```
main()
{
    int *p;
    p = NULL;
    Print p;
}
```

O/P: 0.

Void pointer

Points to variable of any datatype.

```
main()
{
    int a = 5;
    double b = 3.14;
    void *vp;
    vp = &a;
    vp = &b;
    Printf ("a = %d", ~~((int *)vp)~~ *(int*)vp);
    Printf ("b = %d", ~~((double *)vp)~~ *(double*)vp);
}
```

O/P:

a = 5

b = 3.14

Pointer arithmetic

1) Assignment of ptrs to the same type of variable.

2) Add/sub a pointer and an integer.

3) Subtracting/comparing 2 pointers that point to the elements of an array.

4) Inc/Dec the ptr that point to the elements of an array.

5) Assigning the value zero to the ptr & comparing zero with the pointer.

Invalid oprns:-

1) Addn of 2 ptrs.

2) multiplication & division } of pointers with numbers.

Examples:-

```
y = *p1 * *p2;        (*p1) x (*p2)
sum = sum + *p1;
z = 5* - *p2/*p1;     (5 * (-(*p2)))/(*p1)
*p2 = *p2 +10
```

blank space → or treated as comments.

p1+4 → add int

p2-2 → sub int

p2-p1 → pointers to same array, gives the no. of elements between p1 & p2

```
p1++;
-p2;
sum += *p2;
```
} Valid.

Relational opts can be used.

Pointing to same array.

```
p1 > p2,
p1 == p2
p1 != p2.
```
✓

pointer variable can be compared with zero.

↓

unrelated variables makes no sense.

Comparisons are used in arrays & strings.

but,

```
p1/p2,
p1 * p2,
p1/3
p1 + p2
```
} all illegal.

ex1):

```
a = 12;
b = 4;
int a,b, *p1, *p2, x,y,z;
p1 = &a;
p2 = &b;
x = *p1 **p2 - 6;
y = 4* - *p2/*p1 + 10;
Print a,b
Print x,y
*p2 = *p2 + 3;
```

a=12, b=4

x=42, y=9

$x = 12 \times 4 - b$

$x = 42$

$y = 4x - 4/12$

$y = -1 + 10$

$y = 9$

```
*p1 = *p2 - 5 ;
z = *p1 * *p2 - 6
Print a,b        a = 2, b = 7
Print z          z = 8
```

ex 2):-

```
main()
{
  int i = 3, *x;
  float j = 1.5, *y;
  char k = 'c', *z;
  x = &i;
  y = &j;
  z = &k;
  Print x,y,z
  x++;
  y++
  z++,
  Print x,y,z.
```

Take i/p

Property of a pointer

Incrementing a ptr, it points to the immediately next loc of its type.

65524, 65520, 65519

x++ → x + sizeof(int) * 1

65526, 65524, 65520

ex 3)

```
main()
{
  int arr[] = {10, 20, 30, 40, 50, 67, 56, 74};
  int i = 4, *j, *k, *p, *q;
  j = &i;
  j = j + 9;
  k = &i;
```

```
K=K-3;
x= &arr[1];
y= &arr[5];
Printf ("%d", y-x);
j= &arr[4];
K= (arr+4);
if (j==K)
    Printf  2 pointers point to the
            same loc.
                ✓
else
    The 2 pointers point to different
        locations.
```

Arrays & pointers:-

Arrays - Stored in contiguous memory locations.

Pointer when incremented always point to the next loc. of its type.

```
main()
{   int num[] = {24,34,12,44,56,17}
    int i, *ptr;
    ptr = &num[0]; | ptr = num;
for (i=0; i<=5; i++)
  {  Printf ("%u %d", ptr, *ptr);  *(ptr+i)
                                   , *(num+i)
     ptr++;                        ,*(i+num),
                                    num[i],
                                    i[num]);
  }
}
```