# Pointers in C

- Pointer is variable, which holds memory address of another variable, then first variable is said to points to the second variable.

**Pointer Operators:-**

- **& - Address of or Direction or referencing operator.**

  Address of Operator (&) returns the memory location address allocated to the variable.

  **Syntax:-**

  & variable_name;

  **Example:-**

  &i;

- *** - Value at address or Indirection** or **dereferencing pointer operator**

  Value at address (*) operator returns the value stored inside memory location. '*' is used to declare pointer variable.

  **Syntax:-**

  *memory_location;

  **Example:-**

  *(&i);

- Suppose we have variable declaration

  **int i=5;** ← Normal Variable
  **int *p** ← Pointer Variable

- Then,

  **p=&i; /*Assigning address of variable i to pointer p*/**

  then following three things will take place, for each variable in memory.

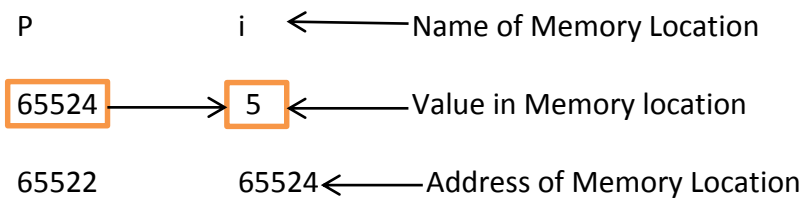



**Fig:- Memory allocation for variable**

- In above fig, p is pointer variable (contains address of i) & that points to variable i.

## Advantages of Pointers

1. Pointers provides the means by which function can modify their calling arguments (actual parameter)
2. Supports dynamic allocations.
3. Can improve efficiency of certain routines.
4. Can be used for destructive work.

## Uses of Pointer:-

1. Accessing Array element.
2. Passing arguments to functions.
3. Creating data structures eg. Link list, tree.
4. Dynamic memory allocation.(DMA).

## Pointer Variables Declaration:

- Like normal variable declaration we can declare pointer variable using value at address operator(*)

  **Syntax:-**

  ```
  data_type *variable_name;
  ```

  **Example:-**

  ```
  int *ptr;
  float *ptr2;
  char *ch;
  ```

## Initialization or Assigning values to Pointer variable or

## What information pointer variable contains?

- Pointer may be initialized to zero, NULL or an address(if we know).
- Initializing pointer to zero is equivalent to initializing a pointer to NULL.
- NULL is symbolic constant available in “stdio.h”.
- Example:-
  1. int *p=0;
  2. int *p=NULL;
  3. int *p=&i;

     In example 3 ‘i’ is normal variable & address of i is assign to p.

## Programming Example to show use of pointer:

```
#include<stdio.h>
main( ){
        int i=5;          <-- Declaration of normal variable
        int *p;           <-- Declaration of pointer variable.
        p=&i;             <-- Assigning address to pointer variable.
        printf("%u\n",p); <-- Display value of P.
        printf("%d",*p);  <-- Display value at address, see fig 2.
}
```

Output:-

65524

5

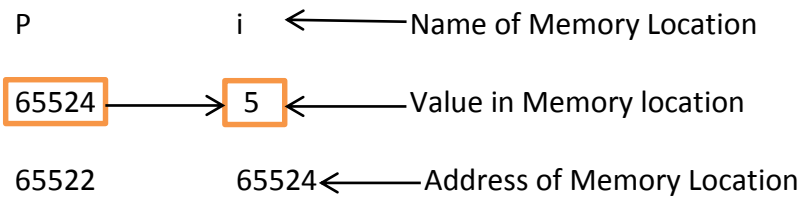



**Fig 2:- Memory allocation for i & p**

## Pointer Arithmetic:-

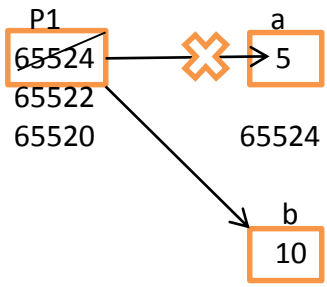
1. **Integer constant can be added or subtracted from pointer.**
   - We can add and/or subtract Integer constant from pointer, depending on its base type(char 1byte, int 2byte, float 4 byte & double 8byte)
   - Example:- In following example we are subtracting integer constant from pointer.

```
#include<stdio.h>
main(){
int a=5,b=10;
int *p1;
p1=&a;          <-- Pointer Points to 65524 that is a=5.
printf( "%d\n",*p1);
p1=p1-1;        <-- Now, Pointer Points to 65522 that is b=10
printf("%d",*p1);
}
```

**Output:-**

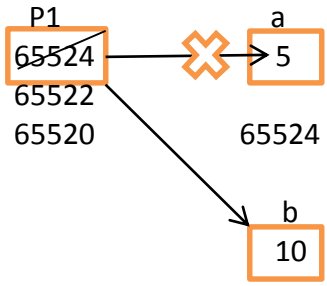
**5**

**10**



2. **Pointer can be incremented or decremented.**
   - We can increment and/or decrement value of pointer, depending on size of base type of the pointer.
   - Example:- In following example we are decrementing pointer value, as base type is int it will be decremented by 2.

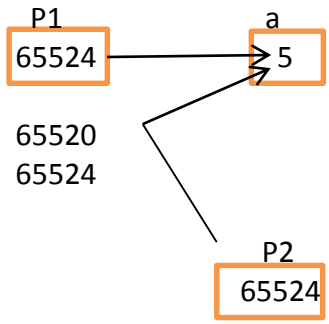
```
#include<stdio.h>
main(){
int a=5,b=10;
int *p1;                          Pointer Points to
p1=&a;                            65524
printf( "%d\n",*p1);              that is a=5.
p1--;
printf("%d",*p1);                 Now, Pointer Points to
}                                 65522                    65522
                                  that is b=10
```




**Output:-**
**5**
**10**

3. **Pointer assignment**
   - We can assign one pointer value to another pointer value, called as pointer assignment.
   - Example:-In following example we are assigning value of p1 to p2, then p1 & p2 points same memory location.

```
#include<stdio.h>
main(){                                                  65520
int a=5,;                         Pointer Points to 65524 65524
int *p1,*p2;                      That is a=5.
p1=&a;
p2=p1                             Pointer Assignment,
printf( "%d\n",*p1);              Now both pointers
printf("%d",*p2);                 points to same memory
}                                 location.              65522
```




**Output:-**
**5**
**5**

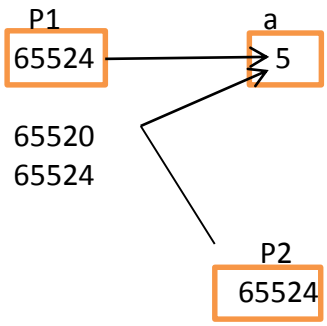
4. **One pointer can be subtracted from another pointer.**
   - We can subtract one pointer from another pointer.
   - In following example, as both pointers are pointing to same memory location subtraction will be zero.

```
#include<stdio.h>
main(){
int a=5;
int *p1,*p2;
p1=&a;
p2=p1

printf("%d",p1-p2);
}

Output:-
0
```

Pointer Points to 65524
That is a=5.

Pointer Assignment,
Now both pointers
points to same memory
location.

Subtracting two
pointers



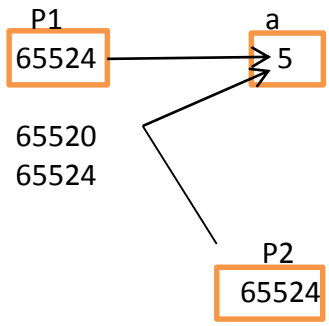
5. **Pointers can be compared.**
   - Two pointers can be compared using relational & logical operators like(<,>,<=,>=,==,!=)

```
#include<stdio.h>
main(){
int a=5;
int *p1,*p2;
p1=&a;
p2=p1
if(p1==p2)
printf(" same");
else
printf(" not same");
}
```

Pointer Points to 65524

Pointer Assignment,

Comparing two pointers




**Output:-**
**Same**

6. **We cannot add two pointers.**
   - Two pointers cannot be added.
7. **Cannot be multiplied or divided by integer constant.**
   - Pointer cannot be multiplied and/or divided by integer constant.
   - Two pointers cannot be multiplied or divided.

**Pointer to Pointer (Chain of a Pointer).**

- One pointer variable contains address of another pointer variable.
- **General syntax is,**

  Data_type **pinter_to_pointer_var_name;
- **Example:-**

  int **p;
- **Example:-**

| | | |
|---|---|---|
| int a; | - | Normal/ordinary variable |
| int *p | - | pointer variable |
| int **p | - | pointer to pointer |
| int ***p | - | pointer to pointer to pointer |

**/* Programming Example of how to work with Pointer to Pointer*/**

```
main(){
        int i=5,*j,**k;
        j=&i;           //pointer contains address of variable.
        k=&j;           //pointer contains address of another pointer variable.
        printf("i=%d",i);
        printf("i=%d",*j);
        printf("i=%d",**k);
}
```

Output:-

5

5

5

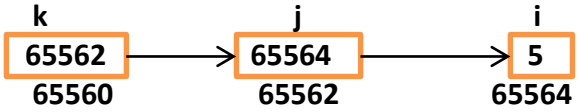



**Fig:- Memory allocation for pointer( like chain)**

## Pointers & Array (Pointers to Array)

- Pointer also can points to array variable.
- Suppose we have following array declaration, then in memory, memory for array will be allocated as shown in following fig.

  **int a[5] ={45,67,87,55,90}**
- "a" is array variable name and it contains starting index memory address i.e. of a[0].

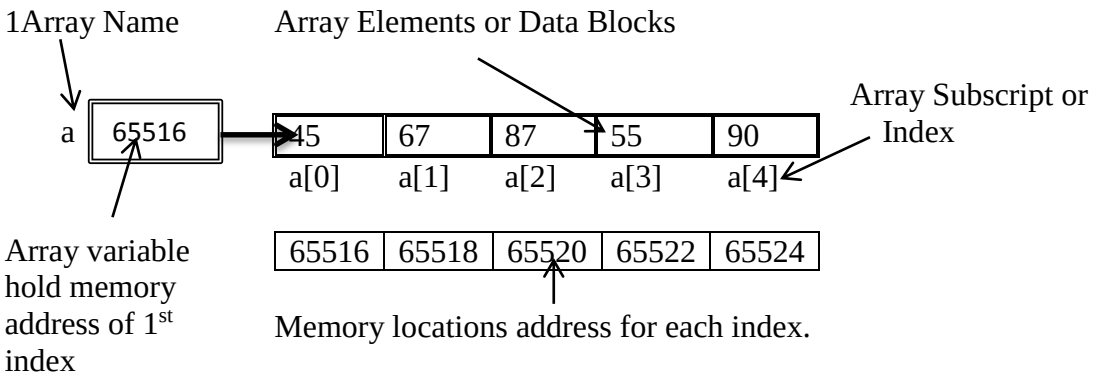
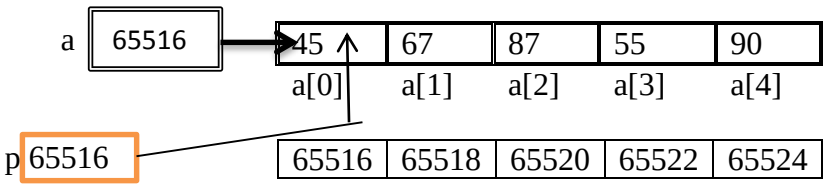



Fig :- Memory allocation for array variable.

- If we declare a pointer "**p**" then we can make pointer "p" may point to the array as follows,
  - p=a this is equivalent to p=&a[0]
  - Now both "a" & "p" points to first element of array.




- Now, to access every element of array "a", we can increment value of 'p' or 'a' as follows, to point to remaining elements of array.
  - p=65516
  - p+1=65518
  - p+2=65520
  - p+3=65522
  - p+4=65524.
  - a=65516
  - a+1=65518
  - a+2=65520
  - a+3=65522
  - a+4=65524.
- There are different ways to access array elements as per discussed below, and out of them whenever we use 3rd & 4th types, they are efficient & faster to access.

- Generally to access array element we write it in for loop as follows

```
1. for(i=0;i<5;i++)
       printf("%5d",a[i]);
```

- We can also write as

```
2. for(i=0;i<5;i++)
       printf("%5d",i[a]);
```

- There is another way to access array element that is

```
3. for(i=0;i<5;i++)
       printf("%5d",*(a+i))
```

- If we want to access array element using pointer, then we have

```
4. for(i=0;i<5;i++){
       printf("%5d",*p);
       p++;
   }
```

- a[i], i[a] & *(a+i) is equivalent to ***(base_address + value of (i).)**

**/* programming example to work with array using pointer*/**

1. **Write a program to dereference a pointer to an array.**

```
#include<stdio.h>
main( ){
int a[10]={87,55,33,27,65,40,32,94,11,19};
int I;
int *p;
p=a[];
for(i=0;i<10;i++){
    printf("%5d",*p);
    p++;
}
}
```

Assigning address of array 'a' to pointer that is **referencing pointer to array.**

**Dereferencing a pointer to array.**

2. **Write a c program to sort an array of 10 integer elements in descending order using pointers.**

```
#include<stdio.h>
main( ){
int a[10],i,j,temp;
int *p;
printf("Enter 10 elements\n");
```

```
for(i=0;i<10;i++)
        scanf("%d",a[i]);
p=a;
/* now sorting array in descending order*/
for(i=0;i<9;i++){
    for(j=0;j<9-i;j++){
          if(*(p+j)<*(p+j+1)){
           temp=*(p+j);
           *(p+j)=*(p+j+1);
           *(p+j+1)=temp;
           }
     }
}
printf(" Sorted Array is \n");
for(i=0;i<10;i++){
    printf("%5d",*p)
    p++;
}
}
```

3. **Write a program to copy contents of one int array to another int array using pointer.**

```
#include<stdio.h>
main()[
int a[10],b[10];
int i, *p1, *p2;
printf("Enter array A elements");
for(i=0;i<10;i++)
   scanf("%d",&a[i]);
p1=a;                /* referencing pointer p1 to array 'a' */
p2=b;                /* referencing pointer p1 to array 'a' */
for(i=0;i<10;i++){
   *p2=*p1;       /* copying one array to another array */
    p1++;
    p2++;
}
printf("Resulting array  B  is \n");
p2=b;
for(i=0;i<10;i++){
   printf("%5d",*p2);
   p2++;
}

}
```

## Pointers & Strings

- Pointer also can points to strings.
- Suppose we have following string declaration, then in memory, memory for char array will be allocated as shown in fig.

  **char a[ ] = "nilu"**
- "a" is char array variable and it contains starting index memory address i.e.a[0].

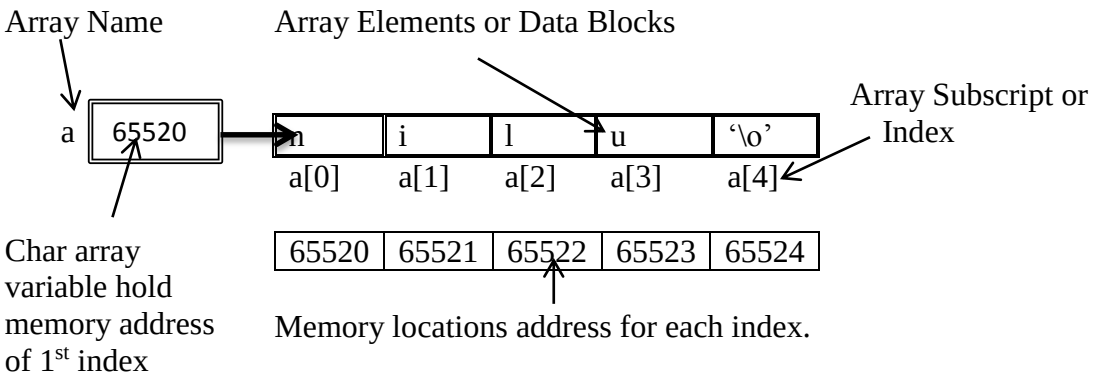
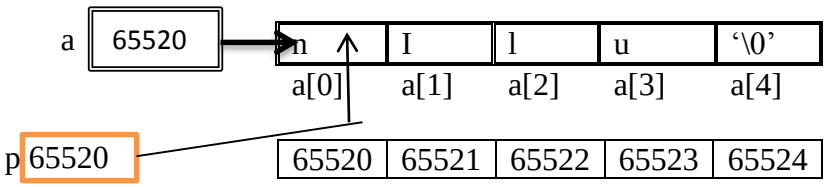



Fig :- Memory allocation for array variable.

- If we declare a pointer "**p**" then we can make pointer "p" may point to the string as follows,
  - p=a this is equivalent to p=&a[0]
  - Now both "a" & "p" points to first element of string.




- Now, to access every element of string "a", we can increment value of 'p' or 'a' as follows, to point to remaining elements of array.
  - p=65520
  - p+1=65521
  - p+2=65522
  - p+3=65523
  - p+4=65524.
  - a=65520
  - a+1=65521
  - a+2=65522
  - a+3=65523
  - a+4=65524.

- Generally to access to access we use printf() with %s.
  printf("%s",a);

- If we want to access string using pointer, then we have

```
1. for(i=0;i<5;i++){
        printf("%c",*p);
        p++;
}
```

## /* Programming Example to work with pointer & work */

**2. Write a program to print contents of given string using pointer.**

```
#include<stdio.h>
main(){
char name[ ]="Amit";
char *p;
p=name;
while(*p!='\0'){
printf("%c",*p);
p++;
}
}
```

**3. Write program to print length of string using pointer.**

```
#include<stdio.h>
main( ){
char str[10];
char *p;
int l=0;
p=str;
while(*p!='\0'){
    l++;
    p++;
}
printf("length of string is %d",l);
}
```

4. **Write a program to print the count of occurrence of any particular character of a string given as:- abc[ ]="programming with c", using pointer**.

```
#include<stdio.h>
main(){
char abc[ ]="programming with c";
char ch,*p;
int cnt=0;
printf("Enter character to see its occurrence\n");
```

```
ch=getchar();
p=abc;
while(*p!='\0'){
        if(ch==*p)
        cnt++;
p++;
}
printf("%c occurred   %d no. of times\n",ch,cnt);
}
```

5. **Write a program to copy the contents of one string to another string & print both string using pointer.**

```
#include<stdio.h>
main( ){
char A[10],B[10];
char *p1,*p2;
printf("Enter one string\n");
scanf("%s",A);
p1=A;
p2=B;
while(*p1!='\0'){
          p2=p1;                  //copying one string to another string.
          p1++;
          p2++;
}

p1=A;
printf("Contents of String A is :");
while(*p1! = '\0'){
           printf("%c",*p1);          //Displaying contents of A…
           p1++;
}
p2=B
printf("Contents of string B is:");
while(*p2!= '\0'){
           printf("%c",*p2);       //Displaying contents of B….
           p2++;
}
}
```

6. **Write a c program to initialize a string as " I am proud to be Indian " and print it in reverse order using pointer.**

```
#include<stdio.h>
main(){
char A[ ]= "I am proud to be Indian ";
char *p;
int l=0;
p=A;
while(*p!= '\0'){
        p++;
        l++;
}
p--;
printf("Reverse String is\n");
while(l>0){
         printf("%c",*p)
         p--;
}
}
```

7. **Write a c program using pointer to reverse the characters of a string using pointer.**

```
#include<stdio.h>
main(){
char A[10], *p;
int l=0;
printf("Enter one string\n");
scanf("%s",A);
p=A;
while(*p!= '\0'){
        p++;
        l++;
}
p--;
printf("Reverse String is\n");
while(l>0){
         printf("%c",*p)
         p--;
}
}
```

**8. Write a program in c to concatenate following two string using pointer. Str1="we are learning", str2="programming in c". or Write a program to concatenate two given string using pointer.**

```
#include<stdio.h>
main( ){
char str1[ ]= "we are learning";
char str2[ ]= "programming in c";
char *p1,*p2;
p1=str1;
p2=str2;
while(*p1!= '\0'){
        p1++;
}
// Concatenating of str2 to str1…
while(*p2!= '\0'){
        *p1=*p2:
         p1++;
         p2++;
}
*p1= '\0';
p1=str1;
printf("String after Concatenations is\n");
while(*p1!= '\0'){
         printf("%c",*p1);
         p1;
}
}
```

**9. Write a program to using pointer to concatenate two string.**

```
#include<stdio.h>
main( ){
char str1[20], str2[20];
char *p1,*p2;
printf("Enter string1:");
scanf("%s",str1);
printf("Enter string2:");
scanf("%s",str2);
p1=str1;
p2=str2;
while(*p1!= '\0'){
        p1++;
}
// Concatenating of str2 to str1…
while(*p2!= '\0'){
```

```
            *p1=*p2:
             p1++;
             p2++;
}
*p1= '\0';
p1=str1;
printf("String after Concatenations is\n");
while(*p1!= '\0'){
            printf("%c",*p1);
            p1;
}
}
```