



**CHANDIGARH  
UNIVERSITY**

*Discover. Learn. Empower.*

## **POINTERS**

**Presented by**

**Er. Jasleen Kaur  
Assistant Professor  
Applied Science(CSE)  
Chandigarh University  
Gharuan (Mohali).**

# Pointer

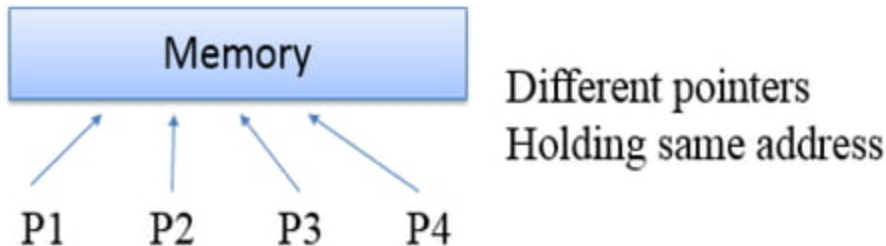
- A variable that holds a memory address.
- This address is the location of another object in the memory.
- Pointer as an address indicates where to find an object.
- ✓ Not all pointers actually contain an address example NULL pointer.
- ✓ Value of NULL pointer is 0.

- Pointer can have three kinds of content in it
  - 1) The address of an object, which can be dereferenced.
  - 2) A NULL pointer.
  - 3) Invalid content, which does not point to an object.

(If p does not hold a valid value, it can crash the program)

- If p is a pointer to integer, then
  - `Int *p`

- ✓ It is possible in some environments to have multiple pointer values with different representations that point to same location in memory.



- ✓ But make sure if the memory is deleted using delete or if original variable goes out of scope.

# Declaring pointer

Data-type \*name;

- \* is a unary operator, also called as indirection operator.
- Data-type is the type of object which the pointer is pointing.
- Any type of pointer can point to anywhere in the memory.
- \* is used to declare a pointer and also to dereference a pointer.

➤ When you write `int *`,



compiler assumes that any address that it holds points to an integer type.

➤ `m = &count;`

it means memory address of count variable is stored into m.

& is unary operator that returns the memory address.

i.e. & (orally called as ampersand) is returning the address.

➤ so it means m receives the address of count.

✓ Suppose, count uses memory  
Address 2000 to store its value 100.  
so,  $m = \&\text{count}$  means  $m$  has 2000  
address.

2000      count=100

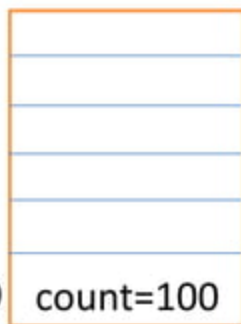
✓  $q = *m$

it returns the value at address  $m$ .

value at address 2000 is 100.

so,  $q$  will return value 100.

i.e.  $q$  receives the value at address  $m$ .



## Address-of operator(&)

- It is used to reference the memory address of a variable.
- When we declare a variable, 3 things happen
  - Computer memory is set aside for variable
  - Variable name is linked to that location in memory
  - Value of variable is placed into the memory that was set aside.



➤ `Int *ptr;`

declaring variable ptr which holds the value at address of int type

➤ `int val =1;`

assigning int the literal value of 1

➤ `ptr=&val;`

dereference and get value at address stored in ptr

➤ `int deref =*ptr`

`printf(“%d\n”, deref);`

Output will be 1

# Pointer Conversions

- One type of pointer can be converted to another type of pointer.
- ```
int main() {  
    double x=100.1, y;  
    int *p;  
    p= (int *) &x; //explicit type conversion  
    y= *p;  
}
```

## Generic Pointer

- ✓ void \* pointer is called as generic pointer.
- ✓ Can't convert void \* pointer to another pointer and vice-versa.
- ✓ void \* pointer can be assigned to any other type of pointer.
- ✓ void \* is used to specify a pointer whose base type is unknown.
- ✓ It is capable of receiving any type of pointer argument without reporting any type of mismatch.

# Pointer Arithmetic

- There are only two arithmetic operations that can be used on pointers
  - Addition
  - Subtraction
- To understand this concept, lets p1 be an integer pointer with value 2000 address.
  - int is of 2 bytes
  - After expression `p1++`;
  - P1 contains address 2002 not 2001.

- Each time p1 is incremented, it will point to next integer.
- The same is true for decrement.
  - for p1--;
  - Causes value of p1 to be 1998.
- Each time a pointer is incremented, it points to the memory location of the next element of its base type.
- If decremented, then it points to previous element location.
- P1=p1+12; makes p1 points to 12<sup>th</sup> element of p1 type.

# Arithmetic Rules

- You cannot multiply or divide pointers.
- You cannot add or subtract two pointers.
- You cannot apply bitwise operators to them.
- You cannot add or subtract type float or double to or from pointers.

# Pointer Comparison

- You can compare two pointers in a relational expression, example:  
if(p<q)  
printf("p points to lower memory than q \n");
- Pointer comparison are useful only when two pointers point to a common object such as an array.

# Benefits of pointer

- Pointers are used in situations when passing actual values is difficult or not desired.
- To return more than one value from a function.
- They increase the execution speed.
- The pointer are more efficient in handling the data types .
- Pointers reduce the length and complexity of a program.



- The use of a pointer array to character string results in saving of data.
- To allocate memory and access it( Dynamic memory Allocation).
- Implementing linked lists, trees graphs and many other data structure.

# How to get address of a function

```
/*A program to get address of a function */  
#include<stdio.h>  
void main()  
{  
    void show(); /* usual way of invoking a function */  
    printf(" The address of show function is=%u", show);  
}  
void show()  
{  
    printf(" \welcome to HPES!!")  
}
```

# Uses of pointer to function

- Pointers are certainly awkward and off-putting and thus this feature of pointer is used for invoking a function
- There are several possible uses :
  - (a) In writing memory resident program.
  - (b) In writing viruses, or vaccines to remove the viruses.
  - (c) In developing COM/DCOM component
  - (d) In VC++ programming to connect events to function calls.

```

#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Address of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}

```

Variable name ----->



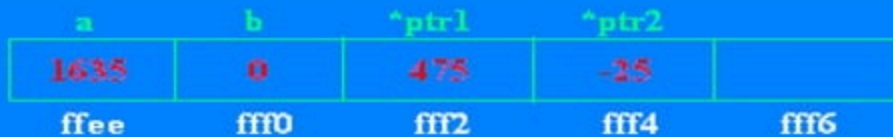
Address ----->



**Garbage values**

Garbage values are the data that was in the memory spots before the variable declared.

```
#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Adress of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}
```





```

#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Address of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}

```

|       | a    | b    | *ptr1 | *ptr2 |       |
|-------|------|------|-------|-------|-------|
| ..... | 10   | 20   | 475   | -25   | ..... |
|       | fff6 | fff0 | fff2  | fff4  | fff6  |

```

Enter two numbers
10
20

```

```

#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Address of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}

```

| a     | b    | *ptr1 | *ptr2 |      |
|-------|------|-------|-------|------|
| 10    | 20   | 475   | -25   |      |
| fffef | fff0 | fff2  | fff4  | fff6 |

Enter two numbers

10

20

Given numbers are 10 and 20

```

#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Address of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}

```

| a    | b    | *ptr1 | *ptr2 |      |
|------|------|-------|-------|------|
| 10   | 20   | fff0  | fff0  |      |
| fff0 | fff0 | fff2  | fff4  | fff6 |

Enter two numbers

10

20

Given numbers are 10 and 20



```

#include<conio.h>
#include<stdio.h>
int main()
{
    int *ptr1,*ptr2,a,b;
    clrscr();
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    printf("Given numbers are %d and %d\n",a,b);
    ptr1=&a;
    ptr2=&b;
    printf("Address of a is %x and that of b is %x\n",ptr1,ptr2);
    printf("Sum of %d and %d is %d\n",a,b,*ptr1+*ptr2);
    getch();
    return 0;
}

```



```

Enter two numbers
10
20
Given numbers are 10 and 20
Address of a is ffee and that of b is fff0
Sum of 10 and 20 is 30

```

Thank you