# C Programing

# Pointer

- Pointer is a variable that can store the memory address of another variable.
- The Pointer is a special Derived  data type in c language.
- System provides address always in unsigned int format.
- Size of pointer to any type of location is equals unsigned int of respective compiler.
  - 32 Bit Compiler Size Of Pointer is : 4 Byte
  - 64 Bit Compiler Size Of Pointer is : 8 Byte
- We can declare pointer till n level of indirection.

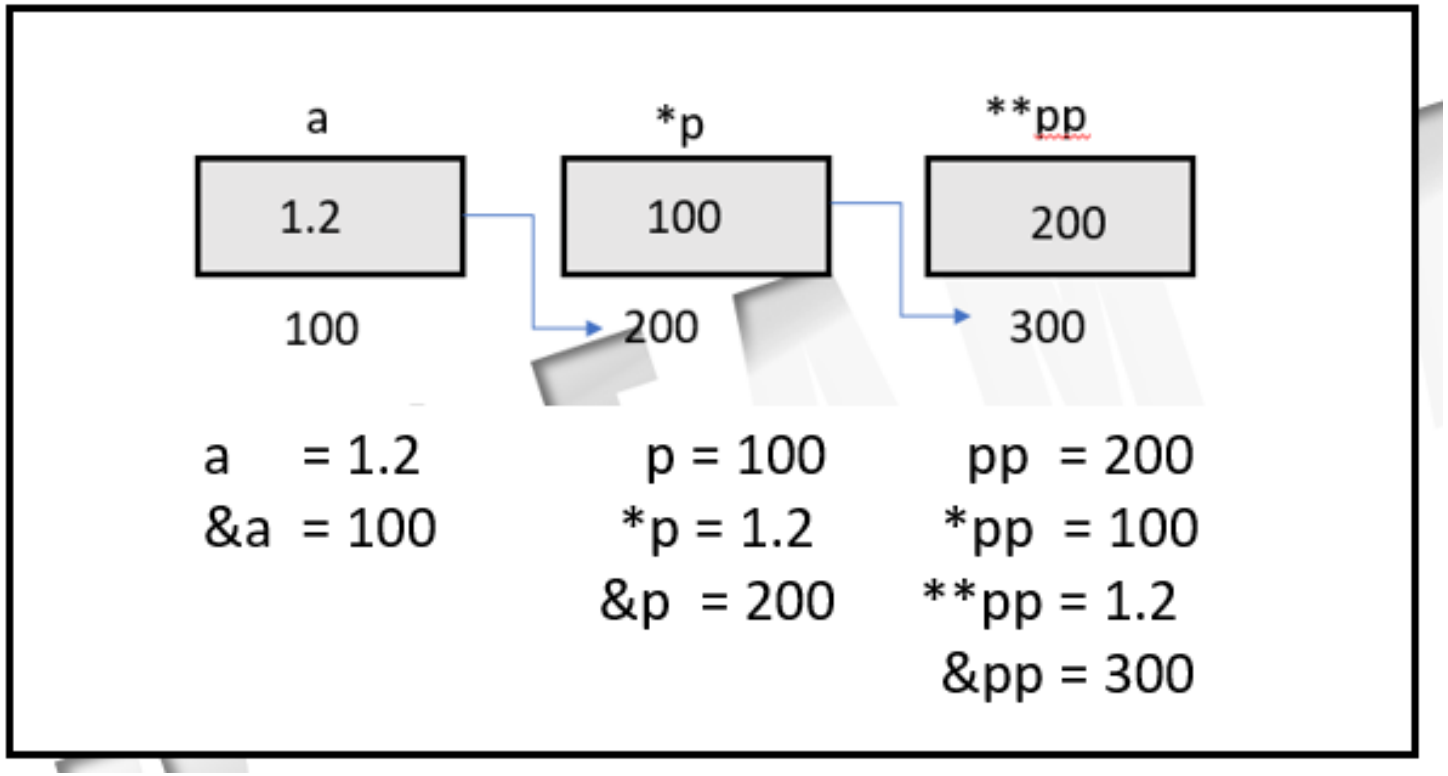
- Operators used when you deal with pointers
  - * : Value at operator / dereferencing operator / indirection operator
  - & : Address Operator / reference operator / direction

# Pointer -Syntax

- Pointer syntax:
  - Declaration:
    - double d ;
    - double *p;
  - Initialization:
    - p = &d;
  - Dereferencing:
    - printf("%lf\n", *p)

# Pointer -Syntax

```
int main()
{
	double a = 1.2;
	double *p = &a;
	double **pp = &p;
	printf("%lf\n", a);
	printf("%lf\n", *p);
	printf("%lf\n", **pp);
	return 0;
}
```




- Pointer to pointer stores address of some pointer variable.
- Level of indirection: Number of dereference operator to retrieve value.

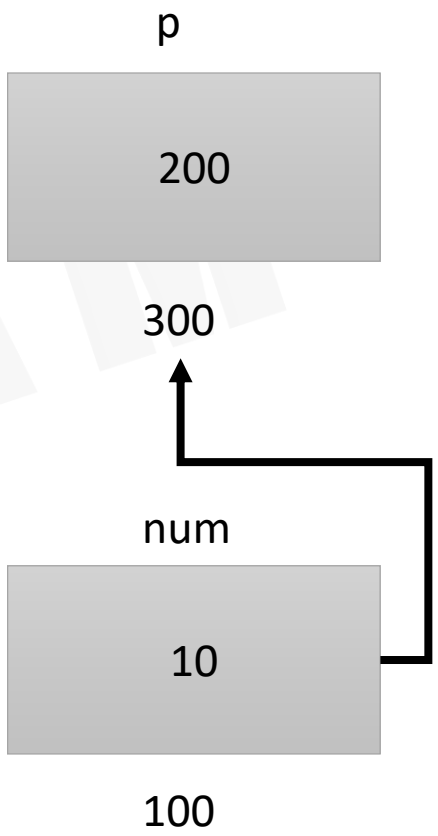
# Passing arguments: Call by value vs Call by address/reference

- Call by value
  - Formal argument is of same type as of actual argument.
  - Actual argument is copied into formal argument.
  - Any change in formal argument does not reflect in actual argument.
  - Creating copy of argument need more space as well as time (for bigger types).
  - Most of data types can be passed by value – primitive & user defined types.
- Call by address
  - Formal argument is of pointer type (of actual argument type).
  - Address of actual argument is collected in formal argument.
  - Actual argument can be modified using formal argument.
  - To collect address only need pointer. Pointer size is same irrespective of data type.
  - Array and Functions can be passed by address only.

# Call by address/reference

```
void accept_number (int *p)
{
        printf("Specify number \n");
        scanf("%d",p);
} p = 200     *p = 10


 int main()
{
                int num;
                accept_number(&num);
                printf("Number = %d",num);
}
 num = 10      &num = 200
```



## Wild Pointer:

- int *p; // wild pointer a pointer which is not initialised at the time of declaration

## NULL Pointer:

- int *p = NULL; // pointer which is pointing to nothing

# Pointer - Scale factor

- Size of data type of pointer is known as Scale factor.
- Scale factor defines number of bytes to be read/written while dereferencing the pointer.
- Scale factor of different pointers
  - Pointer to primitive types: char*, short*, int*, long*, float*, double*
  - Pointer to pointer: char**, short**, int**, long**, float**, double**, void**
  - Pointer to struct/union.
  - Pointer to enum

# Pointer - Scale factor , Arithmetic

- Scale factor plays significant role in pointer arithmetic.
- n locations ahead from current location
- ptr + n = ptr + n * scale factor of ptr
- n locations behind from current location
- ptr - n = ptr - n * scale factor of ptr
- number of locations in between
- ptr1 – ptr2 = (ptr1 – ptr2) / scale factor of ptr1

# Pointer Arithmetic

- Possible Operations:
  - We can add or subtract integer constant from pointer (address). No doubt we can apply increment / decrement operator on pointer(address). One operand as pointer(address) another operand as integer constant.
  - We can subtract two pointers (Both operands as address type )

- Not Possible Operations:
  - We can not add two pointers (addresses) (Both operands as address type).
  - Multiplication/division on two pointers (addresses) is meaning less.


# Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com