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.

Pointer

- Pointer is a variable that stores address of some memory location.
- Internally it is unsigned integer (it is memory address).
- In C, pointer is a special data type.
- It is not compatible with unsigned int.
- Pointer is derived data type (based on primitive data type).
    - To store address of int, we have int pointer.
    - To store address of char, we have char pointer, ...
- Size of pointer variable is always same, irrespective of its data type (as it stores only the address).
- Pointer syntax:
    - Declaration:
        - double *p;
    - Initialization:
        - p = &d;
    - Dereferencing:
        - printf("%lf\n", *p);
- Reference operator - &
    - Also called as direction operator.
    - Read as “address of”.
- Dereference operator - *
    - Also called as indirection operator.
    - Read as “value at”.

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* - 1 bytes
  - short* - 2 bytes
  - int* - 4 bytes
  - long* - 8 bytes
  - float* - 4 bytes
  - double* - 8 bytes
- Pointer to pointer:
  - char**, short**, int**, long**, float**, double**, void** - 8 bytes
- Pointer to struct/union.
  - depends on size of struct/union
- Pointer to enum.
  - 4 bytes as enums are integers only