C Language Pointers recapitulation

Pointer Concept

- Each variable is assigned a particular memory location referenced by its address.
- For example, when the variable i is declared as
- int i = 20;
- i becomes a named location having an address (say 1000) in memory holding an integer value of 20

Pointer Definition

- In C, it is possible to manipulate a variable either by its name, or by its address. The address of a variable can be stored in another variable (**called a pointer variable**), and the variable can be accessed through this pointer variable.
- **A pointer can therefore be defined as a variable that holds the address of another variable.**
- A Pointer variable is associated with the type of the value it is pointing to.
- Thus Pointer is a derived type.

Pointer – Declaration & initialization

- Pointers can be declared and initialized as follows:
- int i, *ip;
- i = 20;
- ip = NULL;
- ip = &i;
- & is the referencing operator returning the address of a variable

Pointer dereferencing

- * is the dereferencing operator which returns the value pointed to by a pointer
- Thus one can write
- j = *ip;
- j = *ip + 1;
- *ip = 10;
- ip1 = ip2;

Arrays and Pointers

- In the declaration
- int arr[10];
- **the name arr of the array refers to the starting address of the area that gets allocated for storing the elements of the array, i.e. address of arr[0], i.e., &arr[0]**
- **Thus arr is a <u>constant</u> pointer, pointing to arr[0]**
- **(arr + 1) points to arr[1], i.e., (arr + 1) is same as &arr[1], and so on**
- **In other words, *(arr + 1) means arr[1], and so on**

Pointer Arithmetic

- If ip is a pointer variable, ++ip, ip++, --ip, ip–- and ip + n, and ip – n (n an integer) are valid expressions
- ip++ means
- **new value of ip = old value of ip + size of data type associated with ip**
- **ip * n and ip / n are not valid**
- **If ip1 and ip2 point to two different elements of an array, ip1>ip2, ip1<ip2, etc. are meaningful**

Pointers and 2-dimensional arrays

- What is meant by

- int p[3][5] = {

```
{ 2, 4, 6, 8, 10},
{ 3, 6, 9, 12, 15},
{ 5, 10, 15, 20, 25}
};
```

- What are the values of *(*p), *(*p+1), *(*(p+1)), *(*(p+1)+1), *(*(p+1)+1)+1 ?

Pointer to Pointer

- The address of a variable can be stored in another pointer variable, as discussed earlier

- Similarly, the address of a pointer variable can be stored in another variable; referencing and dereferencing can be done upto any level of nesting

- int i, *ip, **ip2p;
- i = 20;
- ip = &i;
- ip2p = &ip;

Strings and pointers

- What is meant by
- char *s = "abcdefgh";
- char st[20] = "Akash Chopra" ;
- A string is a <u>sequence</u> of characters terminated by a NULL character '\0'

Pointers as arguments of functions

- What is the difference between

```
void swap (int x, int y)
   {  int temp = x;
      x=  y;
      y=temp;
   }
```

- And

```
void swap (int *x, int *y)
   {   int temp=*x;
       *x=*y;
       *y=temp;
   }
```

Revision requirement

- Void pointers and byte pointers
- Structs, its variants and self-referntial structures
- Pointers to functions
- Function pointers as parameters of functions