Programming for Problem Solving, ES-CS291

Tutorial on Pointers

In the computer memory, every stored data item must occupy one or more contiguous memory cells. Whenever we declare a variable, the system allocates memory location to hold the value of the variable, e.g.

```
int  x = 10;
```

By the above declaration, compiler allocates a location for the integer variable 'x', and assign a value 10 in that location. Suppose that the address location is 12350. During execution of the program, the system always associates the name 'x' with the address 12350 and the value 10 can be accessed by using either the name 'x' or the address 12350.

Since memory addresses are simply numbers, they can be assigned to some variables which can be stored in memory. Such variables that hold memory addresses are called pointers. It is a variable; its value is also stored in some memory location. If we want to assign the address of 'x' to a variable 'ptr', then 'ptr' is said to point to the variable 'x'. Pointer is just a variable whose value is the address of another variable. General form:

```
data_type *pointer_name;
```

The asterisk (*) tells that the variable is a pointer variable.

```
int *ptr ;

ptr = &x ;  // pointer initialization
```

Here ***'ptr'*** is declared as a pointer variable by using ***'*'*** preceded to the variable name. Pointer variables must always point to a data item of the same type. Once a pointer has been assigned the address of a variable, the value of the variable can be accessed using the indirection operator (*).

Example:

```
#include<stdio.h>
main()
{
        int a=5;
        int *ptr;
        ptr = &a;
        printf ("\n a=%d",a);
        printf ("\n address of a=%u",&a);
        printf ("\n ptr=%u",ptr);
        printf ("\n address of  ptr=%u",&ptr);
        printf ("\n ptr points to the value %d",*ptr);
}
```

Output

```
a=5
address of a=789563
ptr=789563
address of ptr=798698
ptr points to the value 5
```