Lecture 2

Array & Pointer

ARRAYS

In this Lecture, we will try to develop understanding of some of the relatively complex concepts.

The following are explained in this lecture with examples:

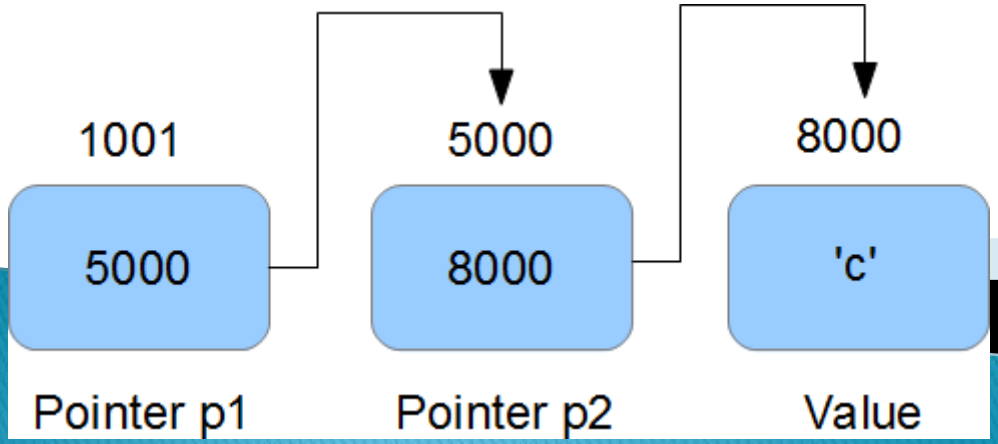
- Pointer to pointer with an example
- Array of pointers with an example
- Pointer to functions with an example

1. C Pointer to Pointer

Till now we have used or learned pointer to a data type like character, integer etc. But in this section we will learn about pointers pointing to pointers.

As the definition of pointer says that its a special variable that can store the address of an other variable. Then the other variable can very well be a pointer. This means that its perfectly legal for a pointer to be pointing to another pointer.

Lets suppose we have a pointer ‘p1’ that points to yet another pointer ‘p2’ that points to a character ‘ch’. In memory, the three variables can be visualized as :



So we can see that in memory, pointer p1 holds the address of pointer p2. Pointer p2 holds the address of character ‘ch’.
So ‘p2’ is pointer to character ‘ch’, while ‘p1’ is pointer to ‘p2’ or we can also say that ‘p2’ is a pointer to pointer to character ‘ch’.

Now, in code ‘p2’ can be declared as :

```
char *p2 = &ch;
```

But ‘p1’ is declared as :

```
char **p1 = &p2;
```

So we see that ‘p1’ is a double pointer (ie pointer to a pointer to a character) and hence the two *s in declaration.

Now,

- *‘p1’ is the address of ‘p2’ ie 5000*
- *‘*p1’ is the value held by ‘p2’ ie 8000*
- *‘**p1’ is the value at 8000 ie ‘c’*

Example for Pointer to Pointer

```
#include<stdio.h>
int main(void)
 {
    char **ptr = NULL;

    char *p = NULL;

    char c = 'd';

     p = &c;
    ptr = &p;

            printf("\n c = [%c]\n",c);
            printf("\n *p = [%c]\n",*p);
            printf("\n **ptr = [%c]\n",**ptr);
 return 0;
}
```

output :

```
 c =    [d]
*p =    [d]
**ptr =  [d]
```

2. C Array of Pointers

Just like array of integers or characters, there can be array of pointers too.

An array of pointers can be declared as :

<type> *<name>[<number-of-elements];

For example :

char *ptr[3];

The above line declares an array of three character pointers.

Pointers(Example)

```
#include<stdio.h>
 int main(void)
 {
            char *p1 = "Himanshu";
            char *p2 = "Arora";
            char *p3 = "India";

            char *arr[3];

             arr[0] = p1;
            arr[1] = p2;
             arr[2] = p3;

                        printf("\n p1 = [%s] \n",p1);
                        printf("\n p2 = [%s] \n",p2);
                        printf("\n p3 = [%s] \n",p3);

                        printf("\n arr[0] = [%s] \n",arr[0]);
                        printf("\n arr[1] = [%s] \n",arr[1]);
                        printf("\n arr[2] = [%s] \n",arr[2]);
return 0;
}
```

Pointers(Example)

In this example,

- *we took three pointers pointing to three strings.*
- *Then we declared an array that can contain three pointers*.
- *We assigned the pointers ‘p1’, ‘p2’ and ‘p3’ to the 0,1 and 2 index of array.*

Let’s see the output :

p1 = [Himanshu]
p2 = [Arora]
p3 = [India]
arr[0] = [Himanshu]
arr[1] = [Arora]
arr[2] = [India]

5. C Function Pointers

Just like pointer to characters, integers etc, we can have pointers to functions.

A function pointer can be declared as :

<return type of function> (*<name of pointer>) (type of function arguments)

For example :

int (*fptr)(int, int)

The above line declares a function pointer ‘fptr’ that can point to a function whose return type is ‘int’ and takes two integers as arguments.

(Example)

```
#include<stdio.h>
 int func (int a, int b)
 {
          printf("\n a = %d\n",a);
           printf("\n b = %d\n",b);

 return 0;
 }
int main(void)
{
          int(*fptr)(int,int);     // Function pointer
          fptr = func;               // Assign address to function pointer

          func(2,3);
          fptr(2,3);

 return 0;
 }
```

3. C Function Pointers (Example)

In the above example,

- we defined a function ‘func’ that takes two integers as inputs and returns an integer.
- In the main() function, we declare a function pointer ‘fptr’ and then assign value to it.
- Note that, name of the function can be treated as starting address of the function so we can assign the address of function to function pointer using function’s name.

Lets see the output :

a = 2

b = 3

a = 2

b = 3

So from the output we see that calling the function through function pointer produces the same output as calling the function from its name.