Chapter 7: Pointers in c

✓ Pointers introduction :

- As the name suggests, the pointers are used to point to something. We have studied so far variables of type int, char, array etc. They hold the value of Pointer doesn't hold any particular value. It *points to a variable* and thereby of that type. We know that variable is a named memory location. So point to memory locations
- Pointer are variables that are used to stored the address of another variable .
- Pointer are used frequently in c ,as they have a number of useful application following uses of pointer

-connect data structured in memory(lists,chains ,queues,trees,graphs)
-allows functions to modify its arguments(actual parameter /call by reference)
-the concepts of dynamic memory is implemented
-concise and efficient array access
-effecently pass strcture to functions.

- Pointer unlike other variables do not stored values as stated they store the address of other variables of same data type.
- We declare a pointer using '*'(value at address) notation
- Syntax

Data_type * pointer_varible;

- Example

int *p;

This statement indicates that *p is a pointer of type integer. It means address contained in p is an int.p is pointer name ,can be used as a pointer to point to any of the variable of type int)

- This declaration tells the C compiler to reserve memory for pointer *variable 'p'.*
- When you execute following statement

p=&i; // i is integer type variable

address of i is put as value for p. Pictorially it can be expressed as

Variable	Memory	Address
i	4	1234
P	3001	5001

Variable 'P' stores a member 1234 which in address of 'i'.
If p points to i. we can obtain the value of I directly from p; the expression *p evaluates to value of i. this evaluation is called 'dereferencing the pointer' p. and the symbol * is called the dereference operator.
Execute following instruction

printf ("%d",*p);

it will print value stored at memory location stored in variable 'p' .

Program to illustrate pointer

```
#include<stdio.h>
#include<conio.h>
{
int i = 5;
 printf(" \n Address of i=%d",  &i) ;
printf(" \n value of i=%d" , i) ;
}
/***output*******
Address of i=1234
value of i=5                */
```

Explanation: In 1st printf statement '&' is used . it is known as address operator.The expression &i returns the address of the variable i, which in the case happens to be 1234, since 1234 represent an address.The other pointer operator in C is '*', called value at address operator. It gives the value stored at a particular address. The value at address operator is also known as Indirection operator. Lets see another program to understand this concept.

➢ **Program to illustrate Pointer**

```
#include <stdio.h>
#include<conio.h>
void main( )
{int i = 5 ;
clrscr( );
printf(" \n Address of i = %d " , &i) ;
printf(" \n Value of i =%d" , i );
printf(" \n Value of i =%d "  ,*(&i)) ;
getch( );
}
```

```
Address of i = 1234
Value of i =  5
Value of i =  5
```

✓ Pointer variable :

The expression &i gives the address of the variable i.This expression can be collected in a variable

j =&i ;

j=it is a variable which contains the address of variable i.

since j is a variable the compiler must provide it space in the memory.

You can see that i is holding value 5 and address of i is 1234 , which in turn hold by variable j. and address of j is 5678.

But we have to declared j in program.since j is variable contains the address of i,it is declared as

int *j;

This declaration tells the compiler that j will be used to store the address of an integer value. Or we can say that 'j' is points to an integer'.

You know that '*' stands for value at address. Hence above statement means, the value at the address contained in j is an int.

➢ **Program to illustrate Pointer variable**

```
#include <stdio.h>
#include<conio.h>
 void main( )
{  clrscr( );
   int i = 5 ;
   int * j ;              // Declaration of Pointer Variable
   j = &i ;          // Assigning address of variable i to variable j
   printf(" \n Address of i = %d ",  &i) ;
   printf(" \n Address of i =%d  " , j) ;
   printf(" \n Value of i =%d  ", i) ;
   printf(" \n Value of i =%d  " ,*(&i)) ;
   printf(" \n Value of i = %d " ,*j) ;
   getch( );  }
```

```
Address of i =1234
Address of i =1234
Value of i =  5
Value of i =  5
Value of i =  5
```

At a Glance…..

- '&' is called an address operator which gives address of variable.
- '*' is called as value at address , which gives the value stored at a particular address.

- Pointer are variables which contain addresses.
- And since address are always whole numbers, pointer would always contain whole numbers.

Some declarations are:

Declaration	Explanation
int * j;	'i' is going to contain the address of a integer value.
char * ch;	ch is going to contain the address of a char value
float * f;	f is going to contain the address of a floating point value

✓ Pointer to pointer:

- Pointer we know is a variable which contains address of another variable .now this variable itself might be another pointer thus we now have a pointer which contains another pointer address.

	Variable	Memory	Address
	i	4	3001
pointer	P	3001	5001
pointer to pointer	pp	5001	7000

Example :

```
#include <stdio.h>
#include<conio.h>
void main( )
{   int i=5,*p,**pp;
    clrscr( );
    p=&i;
    pp=&p;
    printf("Value of i :%d ",i);
    printf("\nValue of i : %d ",*p);
    printf("\nValue of i : %d ",*(&i));
    printf("\nValue of i : %d ",**pp);
    printf("\nValue of i : %d ",i);
    printf("\nAddress of i : %d \n",&i);
    printf("\nAddress of i : %d ",p);
    printf("\nValue of p : %d \n",p);
    printf("\nValue of p : %d ",*(&p));
    printf("\nValue of p : %d ",*pp);
    printf("\nAddress of p : %d ",&p);
    printf("\nAddress of p : %d \n",pp);
    printf("\nValue of pp : %d ",pp);
    printf("\nAddress of pp : %d "&pp);
    getch( ); }
```

```
Value of i : 5
Value of i : 5
Value of i : 5
Value of i : 5
Value of i : 5

Address of i : 0x8fb7fff4
Address of i : 0x8fb7fff4

Value of p : 0x8fb7fff4
Value of p : 0x8fb7fff4
Value of p : 0x8fb7fff4

Address of p : 0x8fb7fff2
Address of p : 0x8fb7fff2

Value of pp : 0x8fb7fff2

Address of pp : 0x8fb7fff0
```

Explanation of program:

Observed how the 3 variables have been declared

int i=5,*p.**pp;

Here, i is a variable of type integer, p is a pointer to an integer, and pp is a pointer an integer pointer. So p contains the address of 'i' and pp contains the address of 'P'

UNIVERSITY PAPERS QUESTIONS :

What is pointer? how will you declare and initialize a pointer? **(VIMP)**

WAP to swap two integers using pointers and using reference variables. **(Dec 2014)**