```c
#include <stdio.h>

int main(void)
{
    //variable declarations
    int num;
    int *ptr = NULL;
    int **pptr = NULL; //Declaration Method 1 :- **pptr Is A Variable Of type 'int'

    //code
    num = 10;

    printf("\n\n");

    printf(" ****** BEFORE ptr = &num ******\n\n");
    printf("Value Of 'num'            = %d\n\n", num);
    printf("Address Of 'num'          = %p\n\n", &num);
    printf("Value At Address Of 'num' = %d\n\n", *(&num));

    //Assigning address of variable 'num' to pointer variable 'ptr'
    //'ptr' now contains address of 'num'...hence, 'ptr' is SAME as '&num'
    ptr = &num;

    printf("\n\n");

    printf(" ****** AFTER ptr = &num ******\n\n");
    printf("Value Of 'num'            = %d\n\n", num);
    printf("Address Of 'num'          = %p\n\n", ptr);
    printf("Value At Address Of 'num' = %d\n\n", *ptr);

    // Assigning address of variable 'ptr' to pointer-to-pointer variable 'pptr'
    // 'pptr' now contains the address of 'ptr' which in turn contains the address of 'num'
    // Hence, 'pptr' is SAME as '&ptr'
    // 'ptr' is SAME as '&num'
    // Hence, pptr = &ptr = &(&num)
    // If ptr = &num and *ptr = *(&num) = value at address of 'num'
    // Then, pptr = &ptr and *pptr = *(&ptr) = ptr = value at address of 'ptr' i.e: 'ptr' i.e : address of 'num'
    // Then, **pptr = **(&ptr) = *(*(&ptr)) = *ptr = *(&num) = num = 10
    // Hence, num = *(&num) = *ptr = *(*pptr) = **pptr

    pptr = &ptr;

    printf("\n\n");

    printf(" ****** AFTER pptr = &ptr ******\n\n");
    printf("Value Of 'num'                          = %d\n\n", num);
    printf("Address Of 'num' (ptr)                  = %p\n\n", ptr);
    printf("Address Of 'ptr'  (pptr)                = %p\n\n", pptr);
    printf("Value At Address Of 'ptr' (*pptr)       = %p\n\n", *pptr);
    printf("Value At Address Of 'num' (*ptr) (*pptr) = %d\n\n", **pptr);

    return(0);
```

```
53  }
54
```