**Batch: B2 Roll No.: 121**

**Experiment / assignment / tutorial No. 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:**  Virtual Lab experiment on Call by reference. |

**AIM:** Virtual Lab experiment on Call by reference

<http://cse02-iiith.vlabs.ac.in/>  
[[https://cse02-iiith.vlabs.ac.in/exp/pointers/simulation.html](http://cse02-iiith.vlabs.ac.in/exp8/simulation/CallByReferencePointers/index.html)](https://cse02-iiith.vlabs.ac.in/exp/pointers/simulation.html)  
Program to swap two number without using third variable using Call by reference.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO4: Design modular programs using functions and demonstrate the concept of pointers and file handling.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.
4. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

The **call by reference** method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call.

In the program, a function called swap ( ) is used in which the address is used to access the actual argument.

**Algorithm:**

Step 1: Start

Step 2: In the main() function, declare integer variables A and B.

Step 3: Initialize the variables A and B.

Step 4: Display the initial values of A and B.

Step 5: Define a function swap() with void return type.

Step 5.1: Pass \*Pa and \*Pb as pointer arguments to the function.

Step 5.2: Declare an integer variable temp.

Step 5.3: The following processing is performed:

Step 5.3.1: temp 🡨 \*Pa

Step 5.3.2: \*Pa 🡨 \*Pb

Step 5.3.3: \*Pb 🡨 temp

Step 6: In the main() function,

Step 6.1: The swap() function is called. Formal arguments &A and &B (i.e., the address of A and B) are passed to this function.

Step 6.2: The swapped values of A and B are displayed.

Step 7: Stop

**Implementation details:**

Code for the program:

#include<stdio.h>

void main(){

  int A = 5, B = 9;

  printf('Value of A is %d\n',A);

  printf('Value of B is %d\n',B);

  swap( &A , &B );

  printf('Value of A after swapping is %d\n',A);

  printf('Value of B after swapping is %d\n',B);

}

void swap( int \*Pa , int \*Pb){

  int temp = \*Pa;

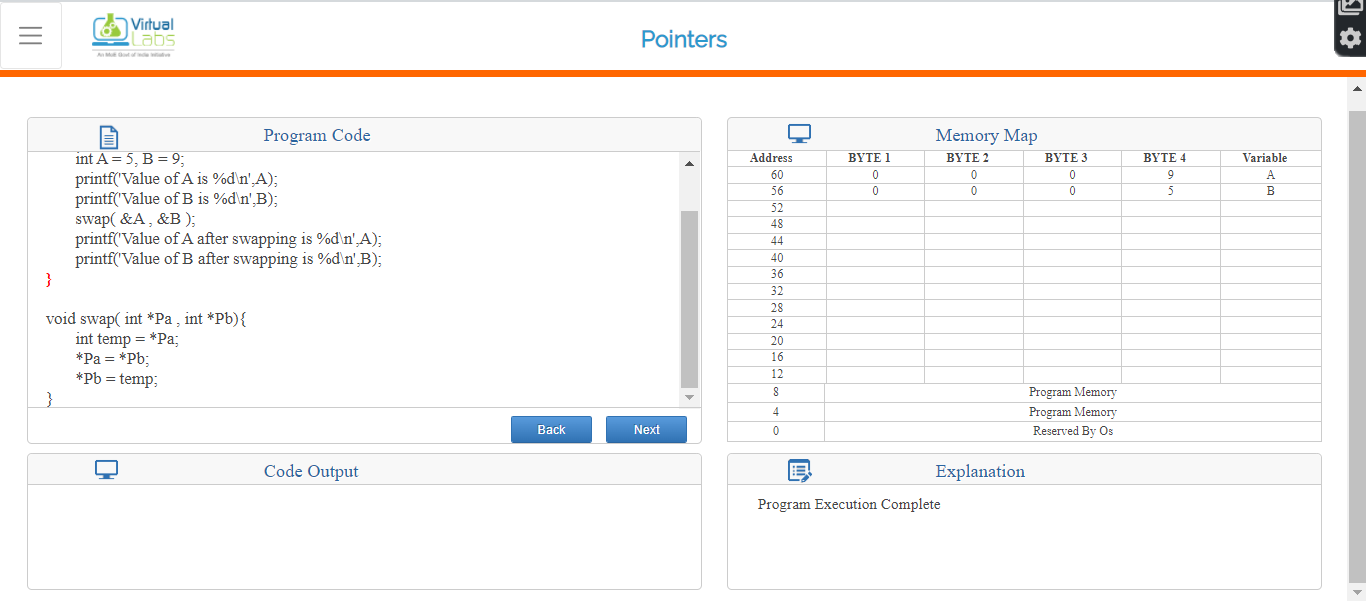
  \*Pa = \*Pb;

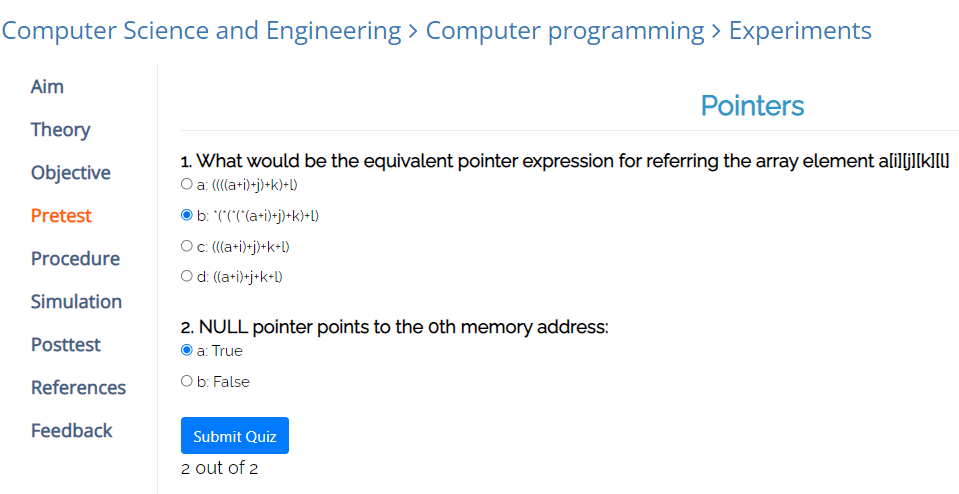
  \*Pb = temp;

}

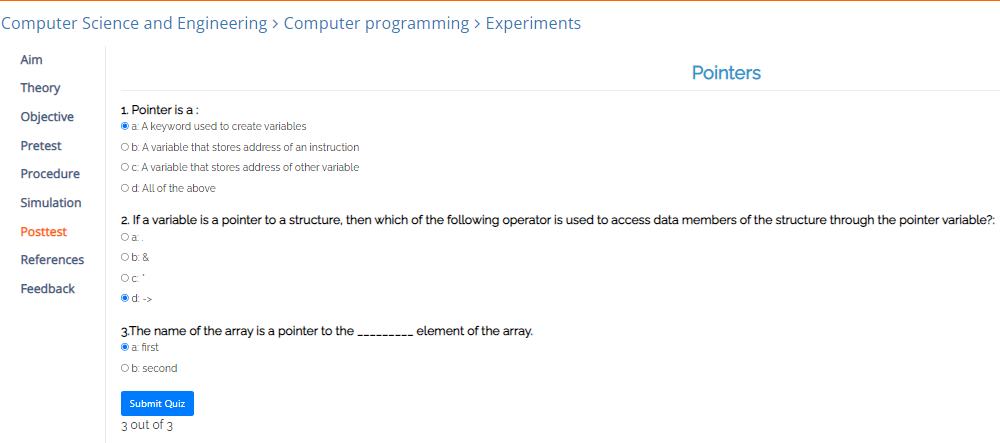
**Output(s):**

(Attach screenshots of output of the Program Code implemented in Virtual Lab and Quiz attempted)

Output of Program



Pretest



Posttest

**Conclusion:**

Thus, in this experiment, modular programs have been designed and the concept of pointers and file handling has been shown. By the use of functions, it has been possible to swap the values of two variables without requiring a third variable in the main() function. Thus, the use of functions makes programming much easier as now the operation to be performed in order to swap the values of two variables is to just call the swapping function and pass the address of those two variables as arguments. This can be performed any number of times without requiring the user to write code each time, as coding only once for the function is sufficient. Thus, it has been shown that the use of functions in a program makes the program more efficient, shorter and easier to debug.

**Post Lab Descriptive Questions**

1. **Difference between call by value and call by reference in C.**

**Ans.**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Call by Value | Call by Reference |
| 1. | While calling a function, the values of variables are passed to it. | While calling a function, the address of variables are passed to it. |
| 2. | The value of each variable in the calling function is passed to each corresponding dummy variable in the called function. | The address of each variable in the calling function is passed to each corresponding dummy variable in the called function. |
| 3. | The changes made in the value of the dummy variables in the called function do not get reflected in the actual variables of the calling function. | The changes made in the value of the dummy variables in the called functions get reflected in the actual variables of the calling function. |
| 4. | The value of the actual variables cannot be changes by the process of calling a function. | The value of the actual variables can be changes by the process of calling a function. |
| 5. | The procedure of passing values to variables is simple and straightforward. | Pointer variables need to be defined in order to store the address of the variables. |

1. **What would be the output of the following program:**

main( )

{

float a = 13.5 ;

float \*b, \*c ;

b = &a ; /\* suppose address of a is 1006 \*/

c = b ;

printf ( "\n%u %u %u", &a, b, c ) ;

printf ( "\n%f %f %f %f %f", a, \*(&a), \*&a, \*b, \*c ) ;

}

Output:

1006 1006 1006

13.500000 13.500000 13.500000 13.500000 13.500000

**Date: 20-01-2022 Signature of faculty in-charge**