

| **TITLE:**  Write a program in C to demonstrate use of a pointer. |
| --- |

**AIM:** 1) Write a program that calculates and prints the transpose of a given matrix using pointers(use function to find transpose of a matrix).

2) Write a file copy program in C that copies a file into another.

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

**Expected OUTCOME of Experiment:**

Apply concepts of pointers in dynamic memory allocation and file handling(CO5).

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

**Books/ Journals/ Websites referred:**

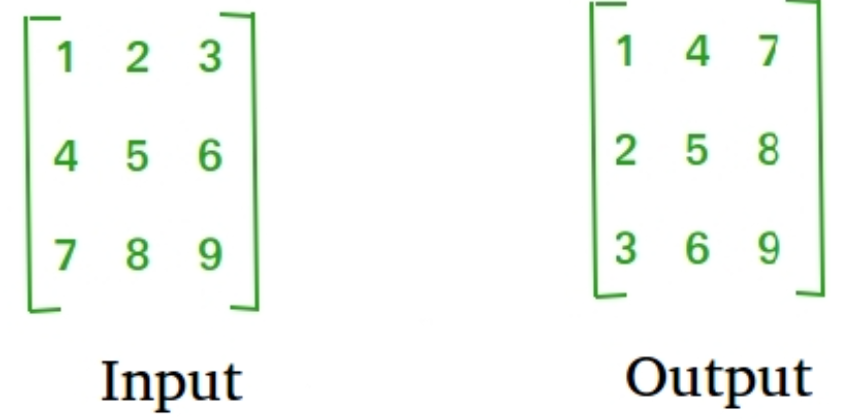
1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

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

**Problem Definition:**

1. The program allows the user to input a matrix, dynamically allocates memory for the matrix and its transpose, calculates and prints the transpose of the matrix using pointers, and then frees the dynamically allocated memory(Use function to find the transpose of a matrix).

For example



1. The program copies the contents of a source file to the destination file, character by character.

**Algorithm:**

**1)**

* Here's the algorithm for the provided code:
* Include Header Files: Include necessary header files such as stdio.h and stdlib.h.
* Function Definition - transpose:
* Define a function named transpose which takes three parameters: a pointer to a pointer to int (matrix), rows, and cols.
* Declare local variables i, j, and a pointer to a pointer to int transpose.
* Allocate memory for transpose dynamically to hold the transposed matrix, with the number of rows equal to the number of columns and vice versa.
* Perform nested loops to transpose the matrix:
* Iterate over each element of the original matrix.
* Assign the value of the original matrix to the corresponding position in the transposed matrix.
* Print the transposed matrix.
* Free the memory allocated for transpose.
* Main Function:
* Define the main function.
* Declare variables i, j, rows, cols, and a pointer to a pointer to int matrix.
* Allocate memory for matrix dynamically based on the user input for the number of rows and columns.
* Read the elements of the matrix from the user.
* Call the transpose function with the matrix, rows, and cols.
* Free the memory allocated for matrix.
* Return 0 to indicate successful execution.
* Input and Output:
* Prompt the user to enter the number of rows and columns of the matrix.
* Read the elements of the matrix from the user.
* Print the transposed matrix.
* Memory Management:
* Allocate memory dynamically for the matrix and the transposed matrix using malloc.
* Free the allocated memory after usage to avoid memory leaks.
* The algorithm aims to transpose a given matrix. It dynamically allocates memory for both the original and transposed matrices, transposes the original matrix, prints the transposed matrix, and then deallocates the memory.

**Implementation details:**

**1)**

#include <stdio.h>

#include <stdlib.h>

void transpose(int \*\*matrix, int rows, int cols)

{

int i, j;

int \*\*transpose;

transpose = (int \*\*)malloc(cols \* sizeof(int \*));

for (i = 0; i < cols; i++)

{

transpose[i] = (int \*)malloc(rows \* sizeof(int));

}

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

transpose[j][i] = matrix[i][j];

}

}

printf("Transpose of the matrix:\n");

for (i = 0; i < cols; i++)

{

for (j = 0; j < rows; j++)

{

printf("%d ", transpose[i][j]);

}

printf("\n");

}

for (i = 0; i < cols; i++)

{

free(transpose[i]);

}

free(transpose);

}

int main()

{

int i, j, rows, cols;

int \*\*matrix;

printf("Aksh Maheshwari 16010123032\n");

printf("Enter the number of rows and columns of the matrix: ");

scanf("%d %d", &rows, &cols);

matrix = (int \*\*)malloc(rows \* sizeof(int \*));

for (i = 0; i < rows; i++)

{

matrix[i] = (int \*)malloc(cols \* sizeof(int));

}

printf("Enter the elements of the matrix:\n");

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

scanf("%d", &matrix[i][j]);

}

}

transpose(matrix, rows, cols);

for (i = 0; i < rows; i++)

{

free(matrix[i]);

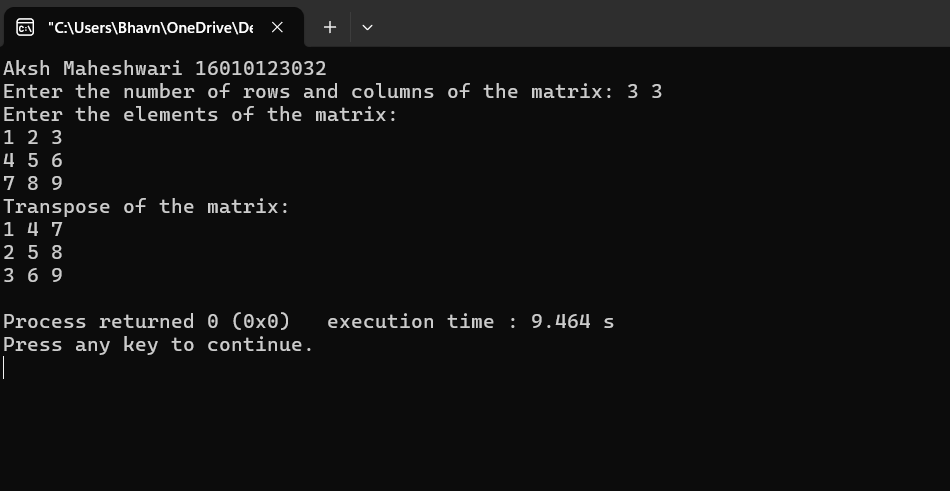
}

free(matrix);

return 0;

}

**Output(s):**

****

**Conclusion:**

**We learnt about pointers and file handling in C and their various applications.**

**Post Lab Descriptive Questions**

* WAP to accept a string from the user and calculate the length of a given string using pointers.

#include <stdio.h>

int string\_length(char \*str)

{

char \*ptr = str;

int length = 0;

while (\*ptr != '\0')

{

length++;

ptr++;

}

return length;

}

int main()

{

printf("Aksh Maheshwari 16010123032\n");

char str[100];

printf("Enter a string: ");

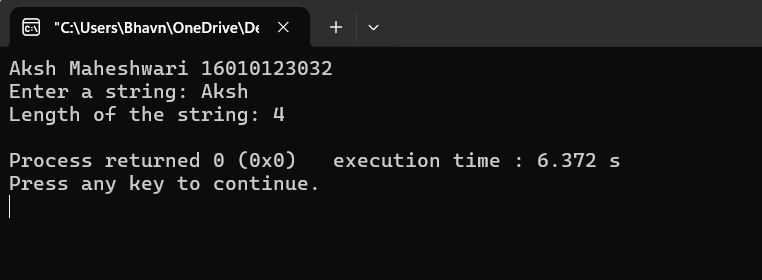
scanf("%s", str);

int length = string\_length(str);

printf("Length of the string: %d\n", length);

return 0;

}



* WAP to count the number of characters and number of lines in a file.

#include <stdio.h>

int main()

{

FILE \*fp;

char filename[100];

char ch;

int char\_count = 0;

int line\_count = 0;

printf("Enter the filename: ");

scanf("%s", filename);

fp = fopen(filename, "r");

if (fp == NULL)

{

printf("Error opening the file.\n");

return 1;

}

while ((ch = fgetc(fp)) != EOF)

{

if (ch != '\n')

{

char\_count++;

}

else

{

line\_count++;

}

}

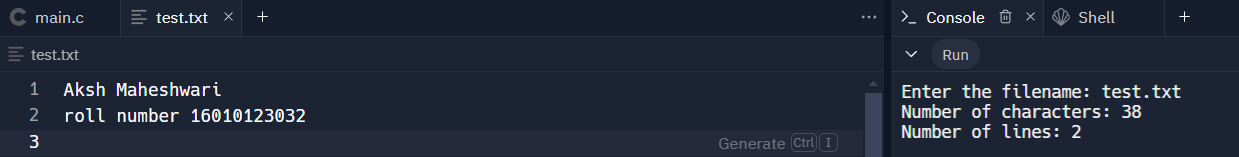
fclose(fp);

printf("Number of characters: %d\n", char\_count);

printf("Number of lines: %d\n", line\_count);

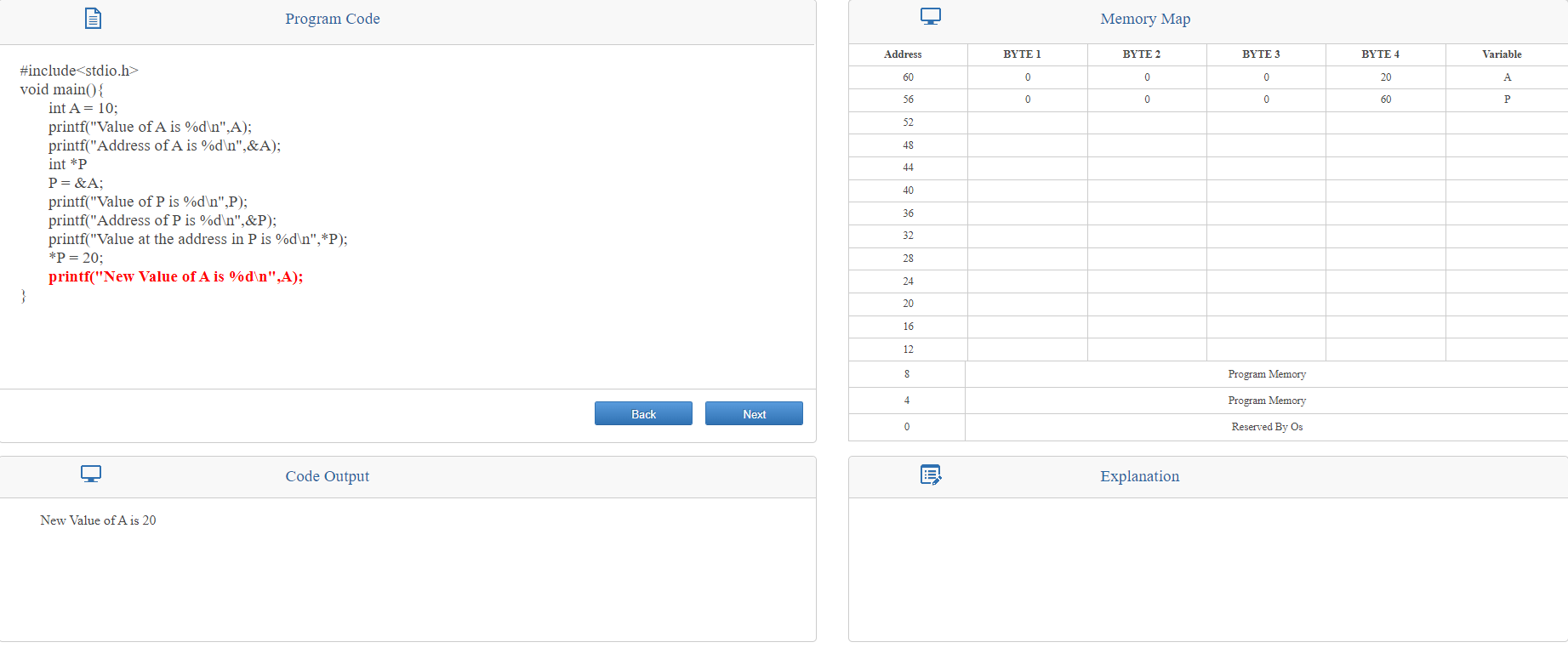
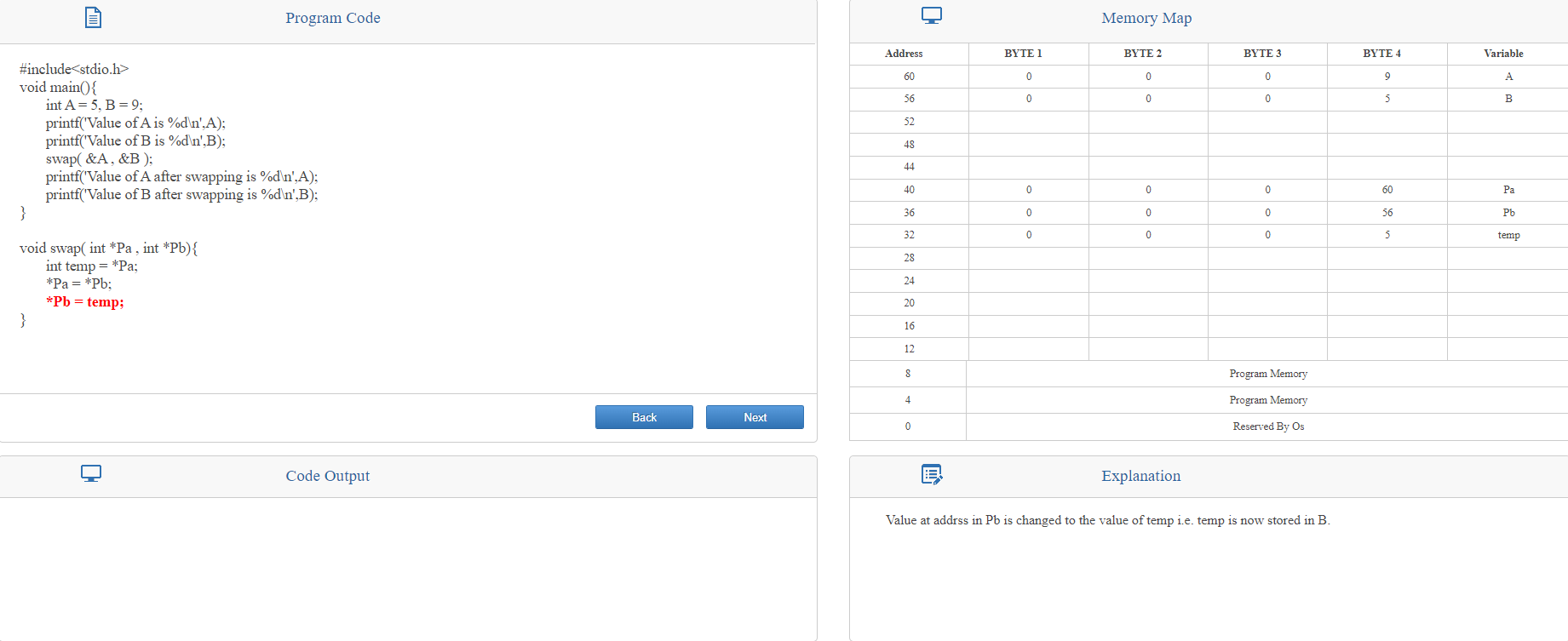
return 0;

}



* Virtual Lab for pointers

<https://cse02-iiith.vlabs.ac.in/exp/pointers/simulation.html>



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**