

Batch: C1-2 Roll No.: 16010123036

Experiment / assignment / tutorial No. 8

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

Signature of the Staff In-charge with date

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Input

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Output

- 2) The program copies the contents of a source file to the destination file, character by character.

Algorithm:

1)

1. Start

2. Declare a function ``transposeMatrix`` that takes four parameters: ``matrix`` (pointer to int), ``transpose`` (pointer to int), ``rows`` (int), and ``cols`` (int):

a. Iterate through each element of the matrix using nested loops:

i. Outer loop iterates over rows (from 0 to rows-1).

ii. Inner loop iterates over columns (from 0 to cols-1).

iii. Transpose the matrix by swapping elements between the original matrix and the transpose matrix using pointer arithmetic.

3. In the main function:

- a. Declare variables ``rows`` and ``cols`` to store the number of rows and columns respectively.
 - b. Prompt the user to enter the number of rows and read it into ``rows``.
 - c. Prompt the user to enter the number of columns and read it into ``cols``.
 - d. Allocate memory dynamically for the original matrix and the transpose matrix using ``malloc``.
 - e. Prompt the user to enter the elements of the matrix row-wise and store them in the original matrix.
 - f. Call the ``transposeMatrix`` function to transpose the matrix.
 - g. Print the transpose of the matrix.
 - h. Free the dynamically allocated memory for both the original matrix and the transpose matrix using ``free``.
4. End

2)

1. Start
2. Include the necessary header file ``stdio.h``.
3. Declare the ``main`` function.
4. Declare two file pointers ``sourceFile`` and ``destinationFile``.
5. Declare a character variable ``ch``.
6. Open the source file "source.txt" in read mode using ``fopen`` and assign it to ``sourceFile``.
 - a. Check if the file pointer ``sourceFile`` is NULL:
 - i. If true, print an error message "Error opening source file."
 - ii. Return 1 to indicate an error.
7. Open the destination file "destination.txt" in write mode using ``fopen`` and assign it to ``destinationFile``.

- a. Check if the file pointer ``destinationFile`` is NULL:
 - i. If true, print an error message "Error opening destination file."
 - ii. Close the source file using ``fclose(sourceFile)``.
 - iii. Return 1 to indicate an error.
8. Enter a while loop to read each character from the source file until the end of the file (EOF):
 - a. Read a character from the source file using ``fgetc`` and store it in the variable ``ch``.
 - b. Check if the read character is not equal to EOF:
 - i. If true, write the character to the destination file using ``fputc(ch, destinationFile)``.
9. Print "File copied successfully." indicating successful file copying.
10. Close both the source and destination files using ``fclose``.
11. Return 0 to indicate successful execution.
12. End.

Implementation details:

1)

```
#include <stdio.h>
#include <stdlib.h>

void transposeMatrix(int *matrix, int *transpose, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            *(transpose + j*rows + i) = *(matrix + i*cols + j);
        }
    }
}

int main() {
    int rows, cols;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int *matrix = (int *)malloc(rows * cols * sizeof(int));
    int *transpose = (int *)malloc(rows * cols * sizeof(int));

    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", matrix + i*cols + j);
        }
    }

    transposeMatrix(matrix, transpose, rows, cols);

    printf("Transpose of the matrix:\n");
    for (int i = 0; i < cols; i++) {
```

```
        for (int j = 0; j < rows; j++) {
            printf("%d ", *(transpose + i*rows + j));
        }
        printf("\n");
    }

    free(matrix);
    free(transpose);

    return 0;
}
```

2)

```
#include <stdio.h>

int main() {
    FILE *sourceFile, *destinationFile;
    char ch;

    sourceFile = fopen("source.txt", "r");
    if (sourceFile == NULL) {
        printf("Error opening source file.\n");
        return 1;
    }
    destinationFile = fopen("destination.txt", "w");
    if (destinationFile == NULL) {
        printf("Error opening destination file.\n");
        fclose(sourceFile);
        return 1;
    }

    while ((ch = fgetc(sourceFile)) != EOF) {
        fputc(ch, destinationFile);
    }

    printf("File copied successfully.\n");

    fclose(sourceFile);
}
```

```
fclose(destinationFile);  
  
return 0;  
}
```

Output(s):

1)

```
Enter the number of rows: 3  
Enter the number of columns: 3  
Enter the elements of the matrix:  
1 2 3  
4 5 6  
7 8 9  
Transpose of the matrix:  
1 4 7  
2 5 8  
3 6 9
```

2)

Conclusion:

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 main() {  
    char str[100];
```

```
char *ptr = str;
int length = 0;

printf("Enter a string: ");
scanf("%s", str);

while (*ptr != '\0') {
    length++;
    ptr++;
}

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 ch;
    int num_chars = 0, num_lines = 0;

    fp = fopen("input.txt", "r");
    if (fp == NULL) {
        printf("Error opening file\n");
        return 1;
    }

    while ((ch = fgetc(fp)) != EOF) {
        num_chars++;
        if (ch == '\n') {
            num_lines++;
        }
    }

    printf("Number of characters: %d\n", num_chars);
}
```

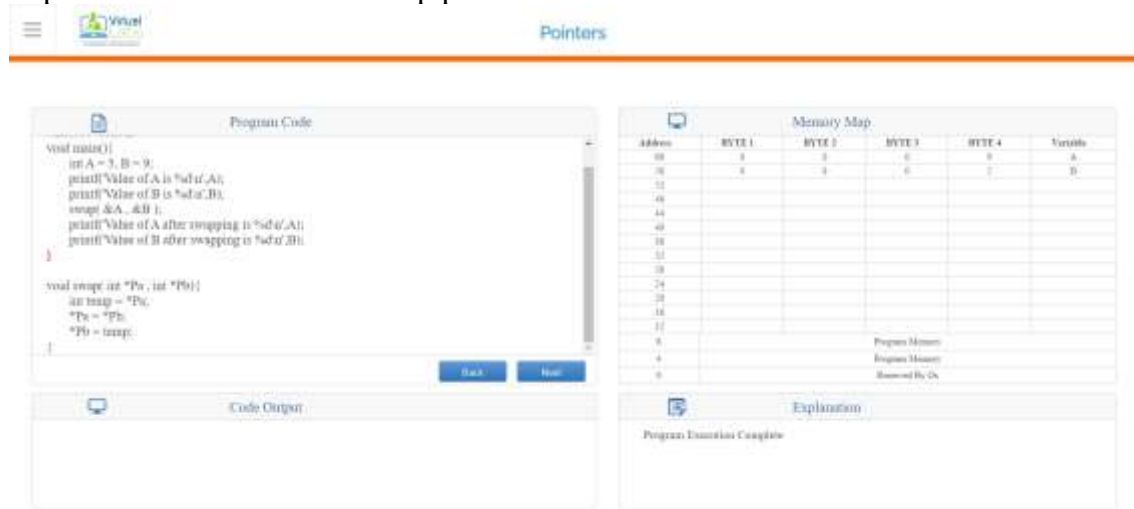


```
printf("Number of lines: %d\n", num_lines + 1);

fclose(fp);

return 0;
}
```

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

 **Pointers**

Program Code

```
void main()
{
    int A = 3, 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 *P1, int *P2)
{
    int temp = *P1;
    *P1 = *P2;
    *P2 = temp;
}
```

Code Output

Memory Map

Address	BYTE 1	BYTE 2	BYTE 3	BYTE 4	Variable
20	3	9	0	0	A
24	9	3	0	0	B
32					
40					
44					
48					
52					
56					
60					
64					
68					
72					
76					
80					
84					
88					
92					
96					
100					
104					
108					
112					
116					
120					
124					
128					
132					
136					
140					
144					
148					
152					
156					
160					
164					
168					
172					
176					
180					
184					
188					
192					
196					
200					
204					
208					
212					
216					
220					
224					
228					
232					
236					
240					
244					
248					
252					
256					
260					
264					
268					
272					
276					
280					
284					
288					
292					
296					
300					
304					
308					
312					
316					
320					
324					
328					
332					
336					
340					
344					
348					
352					
356					
360					
364					
368					
372					
376					
380					
384					
388					
392					
396					
400					
404					
408					
412					
416					
420					
424					
428					
432					
436					
440					
444					
448					
452					
456					
460					
464					
468					
472					
476					
480					
484					
488					
492					
496					
500					
504					
508					
512					
516					
520					
524					
528					
532					
536					
540					
544					
548					
552					
556					
560					
564					
568					
572					
576					
580					
584					
588					
592					
596					
600					
604					
608					
612					
616					
620					
624					
628					
632					
636					
640					
644					
648					
652					
656					
660					
664					
668					
672					
676					
680					
684					
688					
692					
696					
700					
704					
708					
712					
716					
720					
724					
728					
732					
736					
740					
744					
748					
752					
756					
760					
764					
768					
772					
776					
780					
784					
788					
792					
796					
800					
804					
808					
812					
816					
820					
824					
828					
832					
836					
840					
844					
848					
852					
856					
860					
864					
868					
872					
876					
880					
884					
888					
892					
896					
900					
904					
908					
912					
916					
920					
924					
928					
932					
936					
940					
944					
948					
952					
956					
960					
964					
968					
972					
976					
980					
984					
988					
992					
996					
1000					

Explanation

Program execution complete.

Program Code

```
#include <stdio.h>
void main()
{
    int A = 10;
    printf("Value of A is %d\n", A);
    printf("Address of A is %d\n", &A);
    int *P;
    P = &A;
    printf("Value of P is %d\n", *P);
    printf("Address of P is %d\n", &P);
    printf("Value at the address in P is %d\n", *P);
    *P = 20;
    printf("New Value of A is %d\n", A);
}
```

Code Output

Memory Map

Address	BYTE 1	BYTE 2	BYTE 3	BYTE 4	Variable
20	10	0	0	0	A
24	10	0	0	0	B
32					
40					
44					
48					
52					
56					
60					
64					
68					
72					
76					
80					
84					
88					
92					
96					
100					
104					
108					
112					
116					
120					
124					
128					
132					
136					
140					
144					
148					
152					
156					
160					
164					
168					
172					
176					
180					
184					
188					
192					
196					
200					
204					
208					
212					
216					
220					
224					
228					
232					
236					
240					
244					
248					
252					
256					
260					
264					
268					
272					
276					
280					
284					
288					
292					
296					
300					
304					
308					
312					
316					
320					
324					
328					
332					
336					
340					
344					
348					
352					
356					
360					
364					
368					
372					
376					
380					
384					
388					
392					
396					
400					
404					
408					
412					
416					



K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Science and Humanities



Date: _____

Signature of faculty in-charge