

Batch: C1_1 Roll No.: 16010123012

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:

Q1.

1. Input the number of rows (r) and columns (c) for the matrix.
2. Create a 2D array 'matrix' with dimensions r x c to store the original matrix.
3. Create a 2D array 'tran' with dimensions c x r to store the transpose of the matrix.
4. Input the elements of the matrix from the user.
5. Print the original matrix.
6. Perform the transpose operation:
 - a. Iterate over each element (i, j) of the original matrix: Assign the value of matrix[i][j] to tran[j][i].
7. Print the transpose matrix.

Q2

1. Print the name and roll number using `printf`.
2. Declare two file pointers `fptr1` and `fptr2`.
3. Define two character arrays `PIC` and `PIC1` to store file names.
4. Open the file `PIC` for writing using `fopen` and store the file pointer in `fptr1`.
5. Write the string "C_Programming" into the file `PIC` using `fprintf`.
6. Close the file `PIC` using `fclose`.
7. Declare a character array `line` to store lines read from the file.

8. Reopen the file `PIC` for reading using `fopen` and store the file pointer in `fptr1`.
9. Open the file `PIC1` for writing using `fopen` and store the file pointer in `fptr2`.
10. Read each line from the file `PIC` using `fgets` into the `line` array until the end of the file is reached.
11. Write each line read from `PIC` into the file `PIC1` using `fprintf`.
12. Close both files `PIC` and `PIC1` using `fclose`.

Implementation details:

Q1.a Without dynamic allocation

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

```
void transpose(int *matrix,int *tran,int r,int c)
```

```
{  
    int i,j;  
    for(i=0;i<r;i++){  
        for(j=0;j<c;j++){  
            *(tran+j*r+i)=*(matrix+i*c+j);  
        }  
    }  
}
```

```
int main(void)
```

```
{  
    int i,j,r,c;  
    printf("Aaryan Sharma\n");  
    printf("16010123012\n");  
    printf("Enter rows for matrix:");  
    scanf("%d",&r);  
    printf("Enter columns for matrix:");  
    scanf("%d",&c);  
    printf("Enter elements of the matrix:");  
    int matrix[r][c];  
    int tran[c][r];  
    for(i=0;i<r;i++){  
        {  
            for(j=0;j<c;j++){  
                {  
                    scanf("%d",&matrix[i][j]);  
                }  
            }  
        }  
    }  
    printf("Matrix :\n");  
    for(i=0;i<r;i++){  
        {  
            for(j=0;j<c;j++){  
                {
```

```
printf("%d ",matrix[i][j]);
}
if(j== c-1);
{
printf("\n");
}
}
transpose(&matrix[0][0],&tran[0][0],r,c);
printf("Transpose Matrix :\n");
for(i=0;i<c;i++)
{
for(j=0;j<r;j++)
{
printf("%d ",tran[i][j]);
}
if(j== r-1);
{
printf("\n");
}
}
}
```

b. With dynamic memory allocation

```
#include<stdio.h>
#include<stdlib.h>
double** memory(int rows, int cols) {
double** matrix = (double**)malloc(rows * sizeof(double*));
if (matrix == NULL) {
printf("Memory allocation failed\n");
exit(1);
}
for (int i = 0; i < rows; i++) {
matrix[i] = (double*)malloc(cols * sizeof(double));
if (matrix[i] == NULL) {
printf("Memory allocation failed\n");
exit(1);
}
}
return matrix;
}
void freememory(double** matrix, int rows) {
for (int i = 0; i < rows; i++) {
free(matrix[i]);
}
}
```

```
    free(matrix);
}
void Transpose(double** matrix, double** transpose, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }
}
void Matrix(double** matrix, int rows, int cols) {
    printf("Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%.0f\t", matrix[i][j]);
        }
        printf("\n");
    }
}
int main() {
    printf("Aaryan Sharma\n");
    printf("16010123012\n");
    int rows, cols;
    printf("Enter the number of rows of the matrix: ");
    if (scanf("%d", &rows) != 1 || rows <= 0) {
        printf("Invalid input. Please enter a positive integer for the number of rows.\n");
        return 1;
    }
    printf("Enter the number of columns of the matrix: ");
    if (scanf("%d", &cols) != 1 || cols <= 0) {
        printf("Invalid input. Please enter a positive integer for the number of\n");
        printf("columns.\n");
        return 1;
    }
    double** matrix = memory(rows, cols);
    printf("Enter the elements of the matrix:");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (scanf("%lf", &matrix[i][j]) != 1) {
                printf("Invalid input. Please enter a valid number.\n");
                freememory(matrix, rows);
                return 1;
            }
        }
    }
}
```

```
double** transpose = memory(cols, rows);
Transpose(matrix, transpose, rows, cols);
Matrix(matrix, rows, cols);
printf("\n");
Matrix(transpose, cols, rows);
freememory(matrix, rows);
printf("Original matrix memory has been freed\n");
freememory(transpose, cols);
printf("Transpose matrix memory has been freed\n");
return 0;
}
```

Q2

```
#include<stdio.h>
int main(){
    printf("Aaryan Sharma\n");
    printf("16010123012\n");
    FILE *fptr1;
    FILE *fptr2;
    char PIC[]="PIC.txt";
    char PIC1[]="PIC1.txt";
    fptr1=fopen(PIC,"w");
    fprintf(fptr1,"C_Programming");
    fclose(fptr1);
    char line[100];
    fptr1=fopen(PIC,"r");
    fptr2=fopen(PIC1,"w");
    while (fgets(line,100,fptr1)!=NULL){
        fprintf(fptr2,"%s",line);
    }
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

Output(s):

Q1.a

```
"C:\Users\Admin\Documents\PIC_C1_1-12\exp 8.1.exe"
Aaryan Sharma
16010123012
Enter rows for matrix:3
Enter columns for matrix:3
Enter elements of the matrix:1
2
3
4
5
6
7
8
9
Matrix :
1 2 3
4 5 6
7 8 9
Transpose Matrix :
1 4 7
2 5 8
3 6 9

Process returned 3 (0x3)   execution time : 9.343 s
Press any key to continue.
```

```
"C:\Users\Admin\Documents\PIC_C1_1-12\exp 8.1.exe"
Aaryan Sharma
16010123012
Enter rows for matrix:2
Enter columns for matrix:2
Enter elements of the matrix:1
2
3
4
Matrix :
1 2
3 4
Transpose Matrix :
1 3
2 4

Process returned 2 (0x2)   execution time : 6.195 s
Press any key to continue.
```

Q1.b

```
Aaryan Sharma
16010123012
Enter the number of rows of the matrix: 2
Enter the number of columns of the matrix: 2
Enter the elements of the matrix:1 2 3 4
Matrix:
1  2
3  4

Matrix:
1  3
2  4
Original matrix memory has been freed
Transpose matrix memory has been freed

=== Code Execution Successful ===
```

```
Aaryan Sharma
16010123012
Enter the number of rows of the matrix: 3
Enter the number of columns of the matrix: 3
Enter the elements of the matrix:1 2 3 4 5 6 7 8 9
Matrix:
1  2  3
4  5  6
7  8  9

Matrix:
1  4  7
2  5  8
3  6  9
Original matrix memory has been freed
Transpose matrix memory has been freed

=== Code Execution Successful ===
```


Q2

```
"C:\Users\Admin\Documents\PIC_C1_1-12\exp 8.2.exe"

Aaryan Sharma
16010123012

Process returned 0 (0x0)   execution time : 0.285 s
Press any key to continue.
```

```
PIC - Notepad
File Edit Format View Help
C_Programming
```

```
PIC1 - Notepad
File Edit Format View Help
C_Programming
```

Conclusion:

We have successfully completed the experiment and learned about dynamic memory allocation and file handling.

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 slen(char *st){
    int l=0;
    while(*st){
        l++;
        st++;
    }
    return l;
}
int main() {
    printf("Aaryan Sharma\n");
    printf("16010123012\n");
    char string[99];
    printf("Enter String: ");
    scanf("%s",string);
```

```
printf("Length of string is: %d",slen(string));  
return 0;  
}
```

```
Aaryan Sharma  
16010123012  
Enter String: aaryan  
Length of string is: 6  
Process returned 0 (0x0)    execution time : 2.061 s
```

```
Aaryan Sharma  
16010123012  
Enter String: Cprogramming  
Length of string is: 12  
Process returned 0 (0x0)    execution time : 5.263 s
```

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

```
#include<stdio.h>  
int main()  
{  
printf("Aaryan Sharma\n");  
printf("16010123012\n");  
FILE *fptr;  
char ch;  
int character=0,lines=0;  
fptr=fopen("file8.txt","w");  
fprintf(fptr,"Aaryan Sharma\n");  
fprintf(fptr,"12\n");  
  
fclose(fptr);  
fptr=fopen("file8.txt","r");  
  
if(fptr==NULL){  
printf("Error in file creation");  
}  
ch=fgetc(fptr);  
while(ch!=EOF){  
character++;  
if(ch=='\n'){  
lines++;  
}  
}
```

```

ch=fgetc(fptr);
}
printf("Number of characters: %d\nNumber of lines: %d",character,lines);
fclose(fptr);
return 0;
}

```

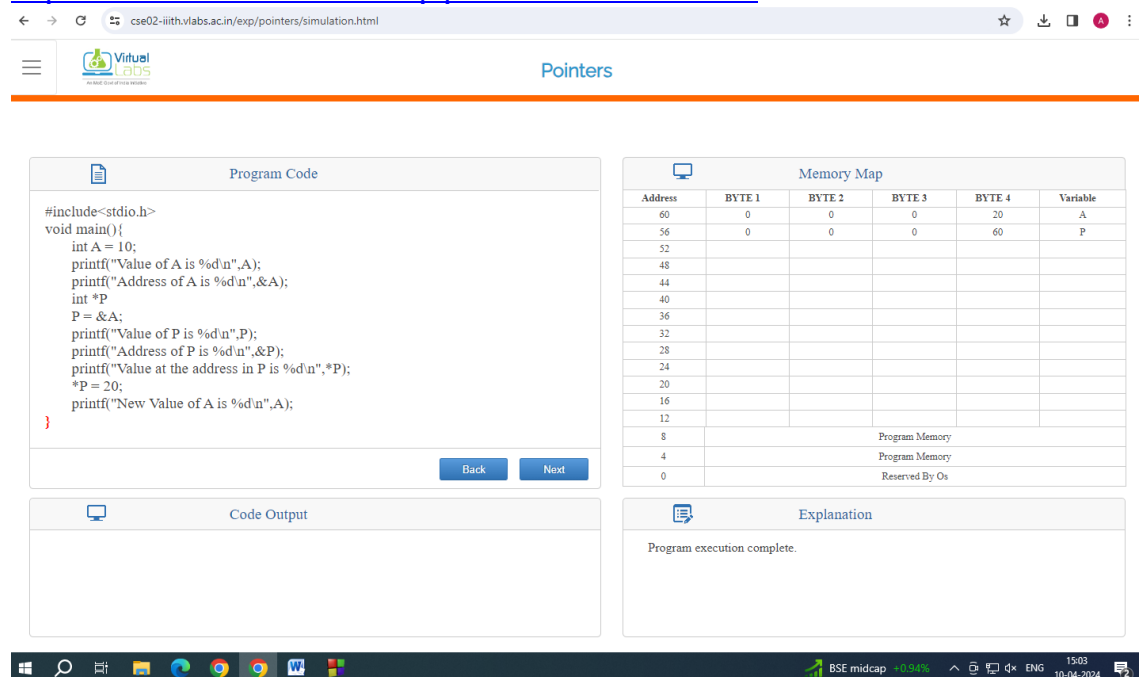
```

Aaryan Sharma
16010123012
Number of characters: 17
Number of lines: 2
Process returned 0 (0x0)    execution time : 0.085 s
Press any key to continue.

```

- Virtual Lab for pointers

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



The screenshot displays the Virtual Lab for pointers simulation interface. The interface is divided into four main sections: Program Code, Memory Map, Code Output, and Explanation.

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);
}

```

Memory Map:


Address	BYTE 1	BYTE 2	BYTE 3	BYTE 4	Variable
60	0	0	0	20	A
56	0	0	0	60	P
52					
48					
44					
40					
36					
32					
28					
24					
20					
16					
12					
8	Program Memory				
4	Program Memory				
0	Reserved By Os				


Code Output:

Explanation:

Program execution complete.

← → ↻ cse02-iiith.vlabs.ac.in/exp/pointers/simulation.html ☆ ⬇️ ⬅️ 🔴


 **Pointers**

 **Program Code**


```
#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;
}
```

Back Next

 **Memory Map**

Address	BYTE 1	BYTE 2	BYTE 3	BYTE 4	Variable
60	0	0	0	9	A
56	0	0	0	5	B
52					
48					
44					
40					
36					
32					
28					
24					
20					
16					
12					
8					Program Memory
4					Program Memory
0					Reserved By Os


 **Code Output**

 **Explanation**

Program Execution Complete

Windows Taskbar: BSE midcap +0.94% 15:05 10-04-2024

← → ↻ cse02-iiith.vlabs.ac.in/exp/pointers/pretest.html ☆ ⬇️ ⬅️ 🔴

 **HOME PARTNERS CONTACT**

Computer Science and Engineering > Computer programming > Experiments

Aim

Theory

Objective

Pretest

Procedure

Simulation

Posttest

References

Feedback

Pointers

1. What would be the equivalent pointer expression for referring the array element a[i][j][k][l]

☐ a: (((a+i)-j)+k)-l

☒ b: 'l'('l'(a+i)-j)+k)-l

☐ c: (((a+i)-j)+k)-l

☐ d: ((a+i)-j+k)-l

2. NULL pointer points to the 0th memory address:

☒ a: True

☐ b: False

Submit Quiz

2 out of 2

Community Links

Sakshat Portal

Outreach Portal





FAQ: Virtual Labs

Contact Us

Phone: General Information: 011-26582050

Email: support@vlabs.ac.in

Follow Us

AGPL 3.0 & Creative Commons (CC BY-NC-SA 4.0)

Windows Taskbar: BSE midcap +0.94% 15:04 10-04-2024

PIC Sem II/January-May 2024