

INDEX

Name: Mahesh Kumar Udas

Roll No.: 21

Faculty: BSc.CSIT

Semester: First

Subject: C Programming

Year : First

| S.N. | Title | Date of Experiment | Date of Submission | Sign |
|------|-----------------------------------------------------------------------------------|--------------------|--------------------|------|
| | Lab 5 | | | |
| 1. | To define, declare, initialize structure and access member variable of structure. | 2080/12/16 | 2081/01/03 | |
| 2. | To pass structure to the function. | 2080/12/16 | 2081/01/03 | |
| 3. | To pass structure pointer to the function. | 2080/12/16 | 2081/01/03 | |
| 4. | To pass array of structure to function. | 2080/12/16 | 2081/01/03 | |
| 5. | To open and close a data file. | 2080/12/16 | 2081/01/03 | |
| 6. | To use different formatted and unformatted I/O functions in a data file. | 2080/12/16 | 2081/01/03 | |
| 7. | To demonstrate record I/O in data file. | 2080/12/16 | 2081/01/03 | |
| 8. | To demonstrate random access in a data file. | 2080/12/16 | 2081/01/03 | |

LAB 5

Experiment No. 1

TITLE:

To define, declare, initialize structure and access member variable of structure.

OBJECTIVE:

- To learn about structure in C programming.
- To learn how to declare, define, initialize and access member variables of structure.

THEORY:

Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure. We can create a structure by using the **struct** keyword and declare each of its members inside curly braces:

Syntax

```
struct structure_name {  
    data_type variable_1;  
    data_type variable_2;  
    .....  
    data_type variable_n  
};
```

To access the structure, we must create a variable of it. Use the **struct** keyword inside the main() method, followed by the name of the structure and then the name of the structure variable and can access members of a structure by using dot symbol(.):

```
int main() {  
    struct structure_name struct_var;  
    struct_var.member_var = value;  
    return 0;  
}
```

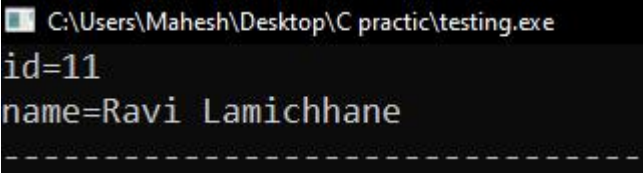
PROGRAM:

```
#include<stdio.h>  
  
struct student{ // structure definition...  
    // structure members...  
    int id;  
    char name[20];  
};
```

```
void main(){
    // structure declaration and initialization...
    struct student s1 = {11,"Ravi Lamichhane"};

    // accessing structure members...
    printf("id=%d\nname=%s",s1.id,s1.name);
}
```

Output:



```
C:\Users\Mahesh\Desktop\C practic\testing.exe
id=11
name=Ravi Lamichhane
-----
```

RESULTS AND DISCUSSION:

The experiment was successful to demonstrate structure in C program. Defining, decelerating,initializing and accessing the structure named student C programming language.

CONCLUSION:

This laboratory exercise provided a hands-on experience in use of structure in C program. Students gained practical knowledge of the basic of function in C programming and are now better equipped to undertake more complex programming tasks in the future.

Experiment No. 2

TITLE: To pass structure to the function.

OBJECTIVE:

- To explain and demonstrate passing structure in a function in C programming.

THEORY:

To pass the structure to the function, we have to define the structure globally in the c program. At the main() function, the structure variable is passed to the function calling as a actual parameter. New structure variable is declared as a formal parameter in the function.

Syntax

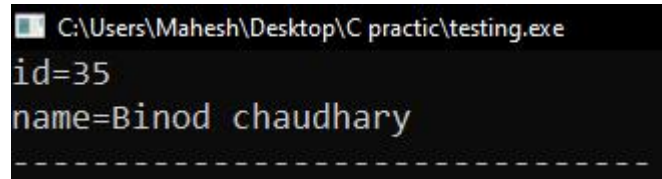
```
struct structure_name {  
    data_type variable_1;  
    data_type variable_2;  
    .....  
    data_type variable_n  
};  
  
data_type function_name(struct structure_name str_var){  
    //statements...  
}  
  
int main() {  
    struct structure_name struct_var = {values...};  
    function_name(struct_name); // passing structure...  
    return 0;  
}
```

PROGRAM:

```
#include<stdio.h>  
  
struct student{ // structure definition...  
    // structure members...  
    int id;  
    char name[20];  
};  
  
// function declaration and definition.  
void display(struct student std){  
    // accessing structure members...  
    printf("id=%d\nname=%s",std.id,std.name);  
}
```

```
void main(){  
    // structure declaration and initialization...  
    struct student s1 = {35, "Binod chaudhary"};  
  
    display(s1); // passing structure to the function..  
}
```

Output:



```
C:\Users\Mahesh\Desktop\C practic\testing.exe  
id=35  
name=Binod chaudhary  
-----
```

RESULTS AND DISCUSSION:

The experiment was successful to demonstrate structure passing to the function in C program. Entire structure was passed to the function to display their member.

CONCLUSION:

This laboratory exercise provided practical experience of passing structure to the function in C programming. Now students can perform different categories of function and make the useful programs in C programming.

Experiment No. 3

TITLE: To pass structure pointer to the function.

OBJECTIVE:

- To learn how to pass structure pointer to the function.

THEORY:

To pass the structure pointer to the function, we have to first define the structure globally in the c program. At the main() function, the structure address is passed to the function calling as a actual parameter. New structure pointer variable is declared as a formal parameter in the function. The structure pointer points the whole structure to do the operation in structure.

Syntax

```
struct str_name {  
    //member_variables...  
}  
function_type function_name(struct str_name *str_ptr){  
    //accessing structure members using pointer..  
}  
main(){  
    struct str_name str_var;  
    function_name(&str_var);  
}
```

PROGRAM:

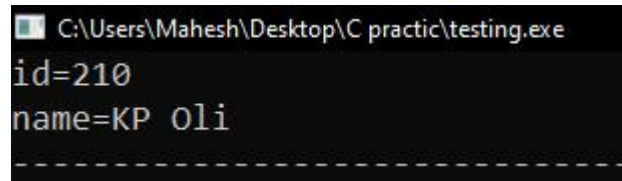
```
#include<stdio.h>
```

```
struct student{ // structure definition...  
    // structure members...  
    int id;  
    char name[20];  
};
```

```
void display(struct student *std){ //function deceleration and definition...  
    // accessing structure members...  
    printf("id=%d\nname=%s",std->id,std->name);  
}
```

```
void main(){  
    // structure decleration and initialization...  
    struct student s1 = {210,"KP Oli"};  
  
    display(&s1); // passing address structure to the function..  
}
```

Output:



```
C:\Users\Mahesh\Desktop\C practic\testing.exe
id=210
name=KP Oli
-----
```

RESULTS AND DISCUSSION:

The experiment was successful to demonstrate structure pointer passing to the function in C program. Address of structure was passed to the function to point the member of the structure to display their member.

CONCLUSION:

This laboratory exercise provided practical experience of passing structure pointer to the function in C programming. Now students can perform different categories of function and make the useful programs in C programming.

Experiment No. 4

TITLE: To pass array of structure to function.

OBJECTIVE:

- To learn how to pass array of structure to function.

THEORY:

An array of structures in C is a data structure that allows us to store multiple records of different data types in a contiguous memory location where each element of the array is a structure. In this article, we will learn how to pass an array of structures from one function to another in C.

We can pass an array of structures to a function in a similar way as we pass an array of any other data type i.e. by passing the array as the pointer to its first element.

Syntax

```
returnType functionName(struct structName arrayName[]);
```

PROGRAM:

```
#include<stdio.h>

struct student{
    int id;
    char name[20];
};

void display(struct student std[]){
    int i;

    printf("\nThe records of students.\n\nID\t\tName");
    for(i=0;i<5;i++){
        printf("\n%d\t\t%s",std[i].id,std[i].name);
    }

void main(){

    struct student std[5];
    int i;

    printf("Enter record of 5 students.\n");

    for(i=0;i<5;i++){
        printf("\nEnter ID:"); scanf("%d",&std[i].id);
```



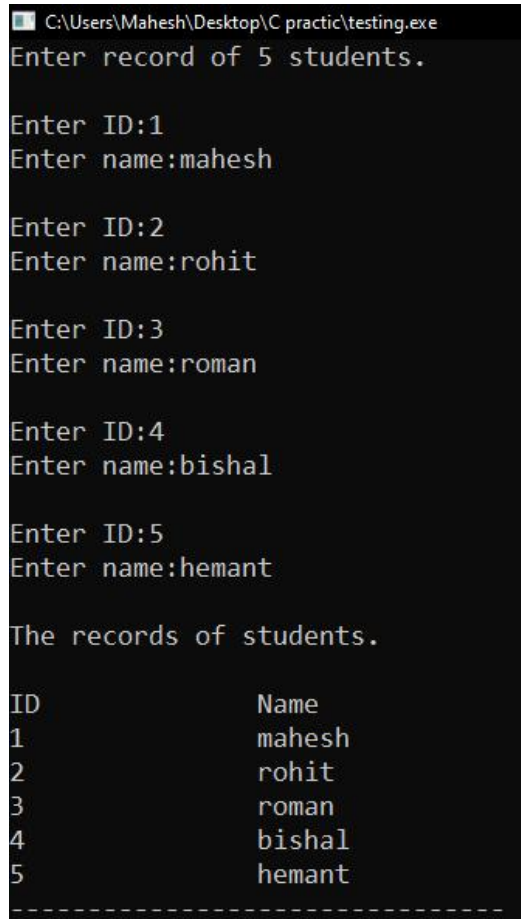
```

        printf("Enter name:"); scanf("%s",&std[i].name);
    }

    display(std);
}

```

Output:



```

C:\Users\Mahesh\Desktop\C practic\testing.exe
Enter record of 5 students.

Enter ID:1
Enter name:maresh

Enter ID:2
Enter name:rohit

Enter ID:3
Enter name:roman

Enter ID:4
Enter name:bishal

Enter ID:5
Enter name:hemanth

The records of students.

ID          Name
1           maresh
2           rohit
3           roman
4           bishal
5           hemanth
-----

```

RESULTS AND DISCUSSION:

The experiment was successful to demonstrate passing array of structure to the function in C program. Address of first element of structure was passed to the function to access the member of the structure to display their member.

CONCLUSION:

This laboratory exercise provided practical experience of passing array of structure to the function in C programming. Now students can perform different categories of function with structure and make the useful programs in C programming.

Experiment No. 5

TITLE: To open and close a data file.

OBJECTIVE:

- To learn how to open and close the data file.

THEORY:

File handling in C is the process in which we create, open, read, write, and close operations on a file. C language provides different functions such as `fopen()`, `fprintf()`, `fclose()` etc. to perform input, output, and many different C file operations in our program.

In this laboratory, we were discuss about opening and closing a file using `fopen()` and `fclose()` function.

Syntax

```
FILE *file_pointer;  
file_pointer = fopen("file_path", "file_mode");  
  
//for closing a file..  
fclose(file_pointer);
```

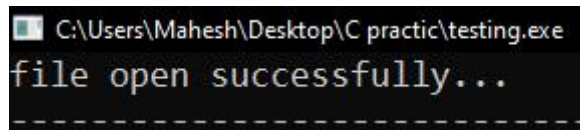
Modes to open a file.

- R - to read the file.
- W - to write the file.
- A - to append the file.
- R+ - to read and write the file.
- W+ - to write and read the file.
- A+ - to append and read the file.

PROGRAM:

```
#include<stdio.h>  
  
void main(){  
    FILE *file;  
  
    file = fopen("text.txt", "r"); //opening the file...  
  
    if(file == NULL)  
        printf("Unable to open the file. !!!!");  
    else  
        printf("file open successfully...");  
  
    fclose(file); //closing the file...  
  
    }
```

Output:

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\Mahesh\Desktop\C practic\testing.exe". The command prompt shows the text "file open successfully..." in a monospaced font. Below the text is a dashed line.

```
C:\Users\Mahesh\Desktop\C practic\testing.exe  
file open successfully...  
-----
```

RESULTS AND DISCUSSION:

The experiment was successful to demonstrate to open and close the file in C program. The text.txt file was opened in read mode. If there is no and file name text.txt exist, then the file could not able to open and error message will appear..

CONCLUSION:

This lab successfully demonstrated to open and close the file in C. We learned different types of mode to open the file in the C program.

Experiment No. 6

TITLE:

To use different formatted and unformatted I/O functions in a data file.

OBJECTIVE:

- To demonstrate the use of formatted and unformatted input/output (I/O) functions in C programming language.
- To understand the differences between formatted and unformatted I/O operations.

THEORY:

Formatted I/O Functions

Formatted I/O functions provide precise control over the input and output format within a program. They achieve this through format specifiers, which define how data is displayed or interpreted. Common formatted I/O functions include:

- `printf`: Prints formatted data to the console.
- `scanf`: Reads formatted data from the console.
- `fprintf`: Prints formatted data to a file.
- `fscanf`: Reads formatted data from a file.

Unformatted I/O Functions

Unformatted I/O functions handle data in its raw form, without any formatting control. These functions offer lower-level interaction and are typically used for specific scenarios. Some examples include:

- `getchar`: Reads a single character from the console (unbuffered).
- `putchar`: Writes a single character to the console (unbuffered).
- `fread`: Reads a block of data from a file into a buffer.
- `fwrite`: Writes a block of data from a buffer to a file

PROGRAM:

To write the file

```
#include <stdio.h>

int main() {
    FILE *fp;
    char name[50]; int id; float marks;

    // Creating a file
    fp = fopen("student.txt", "w");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
}
```

```

// Get student information
printf("Enter student name: "); gets(name);
printf("Enter student ID: "); scanf("%d", &id);
printf("Enter student marks: "); scanf("%f", &marks);

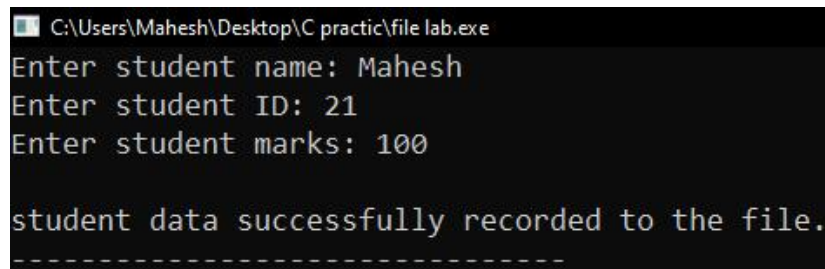
// Write formatted data to file
fprintf(fp, "Name: %s, ID: %d, Marks: %.2f\n", name, id, marks);
printf("\nstudent data successfully recorded to the file.");

// Close the file for writing
fclose(fp);

return 0;
}

```

Output:



```

C:\Users\Mahesh\Desktop\C practic\file lab.exe
Enter student name: Mahesh
Enter student ID: 21
Enter student marks: 100

student data successfully recorded to the file.
-----

```

Reading unformatted data from file.

```

#include <stdio.h>

int main() {
    FILE *fp;
    char name[50], string[100];
    int id; float marks;

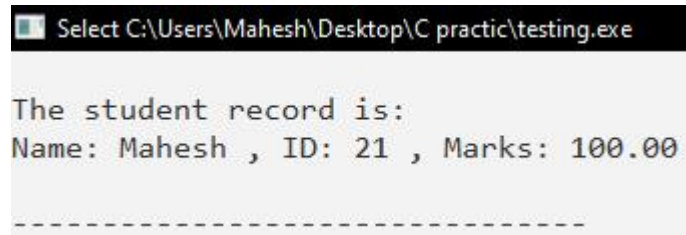
    // Open the file for reading
    fp = fopen("student.txt", "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Read unformatted data
    fgets(string, 100, fp);
    printf("\nThe student record is:\n");
    printf("%s", string);

    fclose(fp);
    return 0;
}

```

Output:



```
Select C:\Users\Mahesh\Desktop\C practic\testing.exe

The student record is:
Name: Mahesh , ID: 21 , Marks: 100.00
-----
```

Reading formatted data from file.

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

```
int main() {
    FILE *fp;
    char name[50], string[100];
    int id;
    float marks;

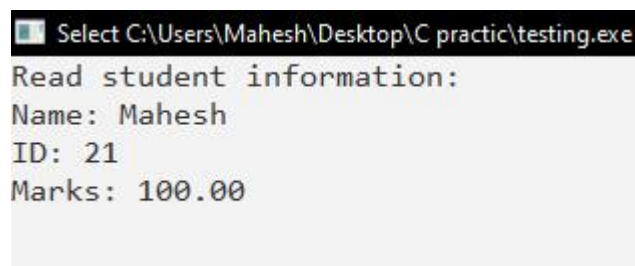
    // Open the file for reading
    fp = fopen("student.txt", "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Read formatted data from file
    fscanf(fp, "Name: %s , ID: %d , Marks: %f ", &name, &id, &marks);
    printf("Read student information:\n");
    printf("Name: %s\n", name);
    printf("ID: %d\n", id);
    printf("Marks: %.2f\n", marks);

    // Close the file for reading
    fclose(fp);

    return 0;
}
```

Output:



```
Select C:\Users\Mahesh\Desktop\C practic\testing.exe

Read student information:
Name: Mahesh
ID: 21
Marks: 100.00
-----
```

RESULTS AND DISCUSSION:

- Formatted input/output functions allowed us to read and write data in a human readable format.
- Unformatted input/output functions allowed us to read and write raw data directly from/to the file.

CONCLUSION:

Through this experiment, we have successfully demonstrated the use of both formatted and unformatted I/O functions in C programming language. These functions provide flexibility in handling different types of data and file operations, catering to various requirements of the program.

Experiment No. 7

TITLE: To demonstrate record I/O in data file.

OBJECTIVE:

- To record and read the data from file.

THEORY:

A file is used to store huge data. C provides multiple file management functions like file creation, opening and reading files, Writing to the file, and closing a file. The file is used to store relevant data and file handling in C is used to manipulate the data.

In this program, we were discuss the and demonstrate the record I/O operations in data file in C programming. We required fwrite() and fread() functions to the this operation.

Syntax

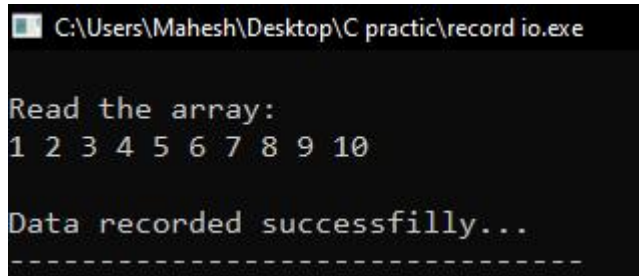
```
fread(&ptr, size_of_array/structure, number_of_array/structure, fptr);  
fwrite(&ptr, size_of_array/structure, number_of_array/structure, fptr);
```

PROGRAM:

- To write the data in a file:

```
#include<stdio.h>  
int main(){  
    FILE *file;  
    int nums[10],i;  
  
    file = fopen("number.txt","wb");  
    if(file == NULL){  
        printf("Error occured during file opening...!!!");  
        return 1;  
    }  
    printf("\nRead the array:\n");  
    for(i=0;i<10;i++)  
        scanf("%d",&nums[i]);  
  
    if(fwrite(&nums,sizeof(nums),1,file))  
        printf("\nData recorded successfilly...");  
    else  
        printf("\nError to record the data !!!");  
  
    fclose(file);  
    return 0;  
}
```


Output:



```
C:\Users\Mahesh\Desktop\C practic\record io.exe

Read the array:
1 2 3 4 5 6 7 8 9 10

Data recorded successfilly...
-----
```

- **To read the data from a file:**

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

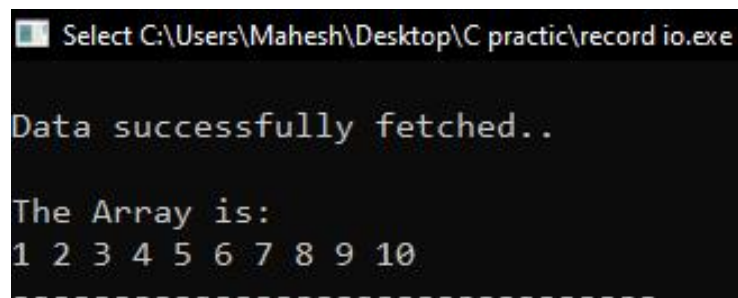
```
int main(){
    FILE *file;
    int nums[10],i;

    file = fopen("number.txt","rb");
    if(file == NULL){
        printf("Error occured during file opening...!!!");
        return 1;
    }

    if(!fread(&nums,sizeof(nums),1,file))
        printf("\nError to fetch the data !!!");
    else{
        printf("\nData successfully fetched..\n");
        printf("\nThe Array is:\n");
        for(i=0;i<10;i++)
            printf("%d ",nums[i]);
    }

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

Output:



```
Select C:\Users\Mahesh\Desktop\C practic\record io.exe

Data successfully fetched..

The Array is:
1 2 3 4 5 6 7 8 9 10
-----
```

RESULTS AND DISCUSSION:

In this laboratory, we were demonstrate the record I/O in the data file. Here the array data are recorded into the number.txt file and fetching the array from the file and display to the program.

CONCLUSION:

This lab successfully to demonstrate the record I/O in the data file. This laboratory exercise provided practical experience of file handling in C programming. Now students can perform different types of operation with file and make the useful programs in C programming.

Experiment No. 8

TITLE: To demonstrate random access in a data file.

OBJECTIVE:

- To record and read the data from file.
- To access and display matched data.

THEORY:

random file access in C, enabling direct data reading or writing without processing all preceding data. Distinct from sequential access, it offers enhanced flexibility for data manipulation. Random access, ideal for large files, involves functions like `ftell()`, `fseek()`, and `rewind()`. This method, akin to choosing a song on a CD, requires more coding but offers superior efficiency and flexibility in file handling.

In this lab, we were discussing about to add and read the data to/from a file and searching the specific record of particular employee and display it. To move the cursor at the beginning of the file pointer, we use `rewind()` function.

PROGRAM:

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

struct employee{
    int id;
    char name[20];
    float salary;
};

int main(){
    FILE *file;
    struct employee emp;
    char ch,sname[20];
    int df = 0; // data found..

    file = fopen("employee.txt","a+b");
    if(file == NULL){
        printf("\n!!! Error !!! file can't open...");
        return 1;
    }

    printf("Enter the record of employee...\n");
    do
    {
        printf("ID: "); scanf("%d",&emp.id);
```

```

printf("Name: "); scanf("%s",emp.name);
printf("Salary: "); scanf("%f",&emp.salary);

if(!fwrite(&emp,sizeof(emp),1,file)){
    printf("\n!!! Error while write the data !!!\n");
    return 1;
}

printf("\nDo you want to add more [y/n]:");
scanf("%s",&ch);
}while(ch == 'Y' || ch == 'y');

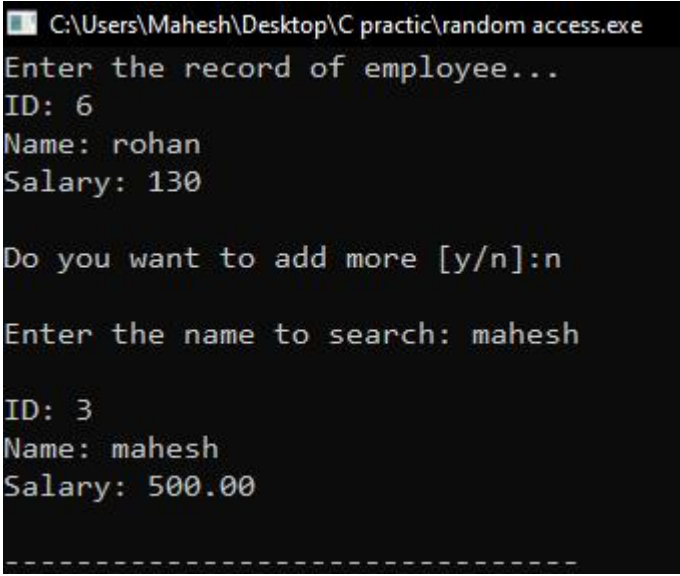
printf("\nEnter the name to search: ");
scanf("%s",sname);

rewind(file);
while(fread(&emp,sizeof(emp),1,file)){
    if(strcmp(emp.name,sname) == 0){
        df = 1;
        printf("\nID: %d",emp.id);
        printf("\nName: %s",emp.name);
        printf("\nSalary: %.2f\n",emp.salary);
    }
}
if(df == 0){
    printf("\nData not found...");
}

fclose(file);
return 0;
}

```

Output



```

C:\Users\Mahesh\Desktop\C practic\random access.exe
Enter the record of employee...
ID: 6
Name: rohan
Salary: 130

Do you want to add more [y/n]:n

Enter the name to search: mahesh

ID: 3
Name: mahesh
Salary: 500.00
-----

```

RESULTS AND DISCUSSION:

Here we opens the file called employee.txt for read appending in binary mode. The structure having members id, name and salary of the employee. The program basically write the multiple employee data and can also read employee data one by one and search for the specific data entered by the user to search the name of employee.

CONCLUSION:

This lab successfully to demonstrate the random access in the data file. This laboratory exercise provided practical experience of file handling in C programming. Now students can perform different types of operation with file and make the useful programs in C programming.