

## **Experiment:16-Develop a C program for implementing random access file for processing the employee details.**

Aim:

The aim of this program is to demonstrate the concept of Random Access File handling in C. Specifically, the program will:

- Store employee details (such as employee ID, name, and salary) in a binary file.
- Allow random access to employee records using their ID (which can be used as an index).
- Perform operations such as adding, updating, and displaying employee details.

Procedure:

1. Create a Structure: Define a structure Employee to store employee details such as ID, name, and salary.
2. Open a Binary File: Open a file in binary mode (rb+ for read/write).
3. Random Access: Use the fseek() function to randomly access employee records based on the employee ID or index.
4. Add Employee: Add employee details at a specific index using fseek() and fwrite().
5. Update Employee: Modify existing employee details by seeking to the employee's position and writing new data.
6. Display Employee Details: Display the employee information stored in the file.
7. Close the File: Properly close the file after all operations.

Steps:

1. Define Employee Structure: Define the structure to hold employee data (ID, name, and salary).
2. File Operations: Open a file for reading and writing binary data (fopen() with "rb+" mode).
3. Add Employee Data: Seek the desired position in the file and write new employee details using fseek() and fwrite().
4. Update Employee Data: Seek the position and write the updated details using fseek() and fwrite().
5. Display Employee Data: Read the file and display employee details using fread() and fseek().
6. Close the File: Ensure that the file is properly closed after performing operations.

C Program to Implement Random Access File for Processing Employee Details:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Define the Employee structure

struct Employee {

    int id;

    char name[50];

    float salary;

};


// Function to display employee details

void displayEmployee(struct Employee emp) {

    printf("Employee ID: %d\n", emp.id);

    printf("Employee Name: %s\n", emp.name);

    printf("Employee Salary: %.2f\n", emp.salary);

}


// Function to add an employee to the file

void addEmployee(FILE *file, int position, struct Employee emp) {

    fseek(file, position * sizeof(struct Employee), SEEK_SET);

    fwrite(&emp, sizeof(struct Employee), 1, file);

}


// Function to update an employee's details in the file

void updateEmployee(FILE *file, int position, struct Employee emp) {

    fseek(file, position * sizeof(struct Employee), SEEK_SET);
```

```

        fwrite(&emp, sizeof(struct Employee), 1, file);
    }

// Function to read employee details from the file
void readEmployee(FILE *file, int position) {
    struct Employee emp;

    fseek(file, position * sizeof(struct Employee), SEEK_SET);

    fread(&emp, sizeof(struct Employee), 1, file);

    displayEmployee(emp);
}

int main() {
    FILE *file;

    struct Employee emp;

    int choice, position;

    // Open the file for reading and writing (binary mode)
    file = fopen("employee_data.dat", "rb+");

    if (file == NULL) {
        // If the file doesn't exist, create it
        file = fopen("employee_data.dat", "wb+");

        if (file == NULL) {
            printf("Unable to open file!\n");

            return 1;
        }
    }
}

```

```
while (1) {

    printf("\nMenu:\n");

    printf("1. Add Employee\n");

    printf("2. Update Employee\n");

    printf("3. Read Employee\n");

    printf("4. Exit\n");

    printf("Enter your choice: ");

    scanf("%d", &choice);


    switch (choice) {

        case 1: // Add Employee

            printf("Enter Employee ID: ");

            scanf("%d", &emp.id);

            printf("Enter Employee Name: ");

            getchar(); // To consume newline character left by previous scanf

            fgets(emp.name, 50, stdin);

            emp.name[strcspn(emp.name, "\n")] = '\0'; // Remove the trailing newline character

            printf("Enter Employee Salary: ");

            scanf("%f", &emp.salary);


            printf("Enter the position to add the employee: ");

            scanf("%d", &position);


            addEmployee(file, position, emp);

            printf("Employee added successfully!\n");

            break;
```

case 2: // Update Employee

```
printf("Enter the position of the employee to update: ");  
  
scanf("%d", &position);  
  
printf("Enter updated Employee ID: ");  
  
scanf("%d", &emp.id);  
  
printf("Enter updated Employee Name: ");  
  
getchar(); // To consume newline character left by previous scanf  
  
fgets(emp.name, 50, stdin);  
  
emp.name[strcspn(emp.name, "\n")] = '\0'; // Remove the trailing newline character  
  
printf("Enter updated Employee Salary: ");  
  
scanf("%f", &emp.salary);  
  
updateEmployee(file, position, emp);  
  
printf("Employee updated successfully!\n");  
  
break;
```

case 3: // Read Employee

```
printf("Enter the position of the employee to read: ");  
  
scanf("%d", &position);  
  
readEmployee(file, position);  
  
break;
```

case 4: // Exit

```
fclose(file); // Close the file  
  
printf("Exiting program.\n");
```

```
return 0;
```

```
default:
```

```
printf("Invalid choice! Please try again.\n");
```

```
}
```

```
}
```

```
return 0;
```

```
}
```

Output:

Output

Clear

Menu:

1. Add Employee
2. Update Employee
3. Read Employee
4. Exit

Enter your choice: 1

Enter Employee ID: 101

Enter Employee Name: 192372048

Enter Employee Salary: 50000

Enter the position to add the employee: 0

Employee added successfully!

Menu:

1. Add Employee
2. Update Employee
3. Read Employee
4. Exit

Enter your choice: 3

Enter the position of the employee to read: 0

Employee ID: 101

Employee Name: 192372048

Employee Salary: 50000.00

Menu:

1. Add Employee
2. Update Employee

