

16. Develop a C program for implementing random access file for processing the employee details.

AIM:

To develop a C program that implements random access files for processing employee details.

ALGORITHM:

1. Define a structure Employee with fields such as ID, name, and salary.
2. Create a random access file where employee details will be stored.
3. Provide functionality to add, modify, delete, and display employee records.
4. Use fseek() for random access to specific records.
5. Use ftell() to determine the position in the file.
6. Implement a menu-driven program to interact with the user.

PROCEDURE:

1. Define the Employee structure with fields like ID, Name, and Salary.
2. Create a file to store employee records in binary format.
3. Implement functions for:
 - Adding a new employee to the file.
 - Modifying an existing employee's details.
 - Deleting an employee record.
 - Displaying all employee details.
4. Use fseek() to navigate to specific records by byte offset.
5. Use fwrite() and fread() to store and retrieve records from the file.
6. Implement user options to interact with the program.

CODE:

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

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

```
#define MAX_NAME_LEN 100
```

```
#define FILE_NAME "employee.dat"
```

```
typedef struct {
```

```
    int id;
```

```
    char name[MAX_NAME_LEN];
```

```
    float salary;
```

```
} Employee;
```

```
void add_employee(FILE *fp) {
```

```
    Employee emp;
```

```
    printf("Enter employee ID: ");
```

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

```
    getchar();
```

```
    printf("Enter employee name: ");
```

```
    fgets(emp.name, MAX_NAME_LEN, stdin);
```

```
    emp.name[strcspn(emp.name, "\n")] = '\0';
```

```
    printf("Enter employee salary: ");
```

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

```
    fseek(fp, 0, SEEK_END);
```

```
    fwrite(&emp, sizeof(Employee), 1, fp);
```

```
}
```

```
void modify_employee(FILE *fp) {
```

```
    int id;
```

```

printf("Enter employee ID to modify: ");
scanf("%d", &id);
Employee emp;
int found = 0;
while (fread(&emp, sizeof(Employee), 1, fp)) {
    if (emp.id == id) {
        found = 1;
        printf("Enter new employee name: ");
        getchar();
        fgets(emp.name, MAX_NAME_LEN, stdin);
        emp.name[strcspn(emp.name, "\n")] = '\0';
        printf("Enter new employee salary: ");
        scanf("%f", &emp.salary);
        fseek(fp, -sizeof(Employee), SEEK_CUR);
        fwrite(&emp, sizeof(Employee), 1, fp);
        break;
    }
}
if (!found) {
    printf("Employee not found.\n");
}
}

void delete_employee(FILE *fp) {
    FILE *temp_fp = fopen("temp.dat", "wb");
    int id;
    printf("Enter employee ID to delete: ");
    scanf("%d", &id);
    Employee emp;

```

```

int found = 0;

while (fread(&emp, sizeof(Employee), 1, fp)) {
    if (emp.id != id) {
        fwrite(&emp, sizeof(Employee), 1, temp_fp);
    } else {
        found = 1;
    }
}

fclose(fp);

remove(FILE_NAME);

rename("temp.dat", FILE_NAME);

if (found) {
    printf("Employee deleted successfully.\n");
} else {
    printf("Employee not found.\n");
}
}

void display_all_employees(FILE *fp) {
    Employee emp;

    fseek(fp, 0, SEEK_SET);

    while (fread(&emp, sizeof(Employee), 1, fp)) {
        printf("ID: %d, Name: %s, Salary: %.2f\n", emp.id, emp.name, emp.salary);
    }
}

int main() {
    FILE *fp;

    fp = fopen(FILE_NAME, "rb+");

```

```
if (fp == NULL) {  
    fp = fopen(FILE_NAME, "wb+");  
    if (fp == NULL) {  
        printf("Unable to open file.\n");  
        return 1;  
    }  
}  
  
int choice;  
  
while (1) {  
    printf("\nMenu:\n");  
    printf("1. Add employee\n");  
    printf("2. Modify employee\n");  
    printf("3. Delete employee\n");  
    printf("4. Display all employees\n");  
    printf("5. Exit\n");  
    printf("Enter your choice: ");  
    scanf("%d", &choice);  
    switch (choice) {  
        case 1:  
            add_employee(fp);  
            break;  
        case 2:  
            modify_employee(fp);  
            break;  
        case 3:  
            delete_employee(fp);  
            break;  
        case 4:
```

```

        display_all_employees(fp);

        break;
    case 5:
        fclose(fp);

        return 0;

    default:
        printf("Invalid choice. Please try again.\n");

    }
}

return 0;
}

```

OUTPUT:

The screenshot displays the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'Create New Project', 'My Projects', 'Classroom' (marked as 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and a 'Logout' button. The main area on the right has a black background with white text for the program's output. The output shows a menu with five options: 1. Add employee, 2. Modify employee, 3. Delete employee, 4. Display all employees, and 5. Exit. The user enters choice 1, then provides employee ID 323, name TEJA, and salary 100000. The menu is shown again, and the user enters choice 5. The program then finishes with the message: '...Program finished with exit code 0 Press ENTER to exit console.'