B. Siva Shirish-192324016

34. Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous records. Design a C program to simulate the file allocation strategy.

AIM

To design a C program that simulates a **sequential file allocation strategy**, where therecords of the file are stored one after another both physically and logically, and each record can only be accessed by reading all the previous records.

ALGORITHM

1. Start
2. Define a structure FileRecord to represent a record in the file.
3. Create an array to hold the file records and initialize them.
4. Define functions for file operations such as adding a new record, displaying allrecords, and accessing a specific record (sequentially).
5. Add a new record to the file at the end (sequential allocation).
6. To simulate the access strategy, iterate through all previous records beforeaccessing the desired record.
7. Display the records as they are stored sequentially.
8. Stop

PROCEDURE

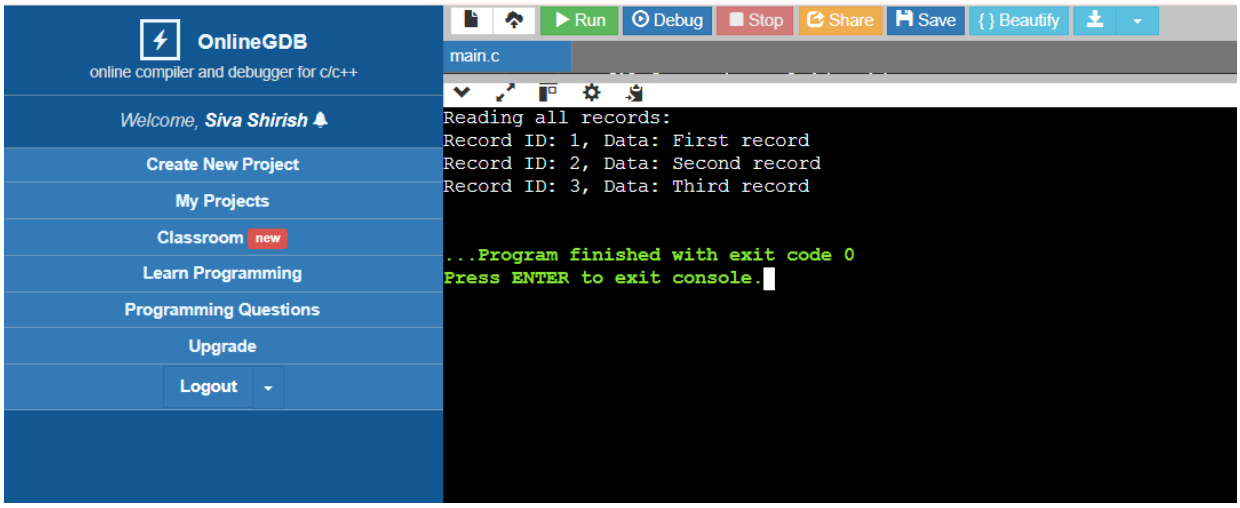
1. Include necessary libraries (stdio.h for input/output and stdlib.h for memory allocation).
2. Define a FileRecord structure to represent a file record.
3. Create a function addRecord() to simulate the addition of new records to the file.
4. Create a function displayRecords() to display the current file records sequentially.

5. Create a function accessRecord() to simulate sequential access by readingprevious records.
6. Initialize the file and perform operations like adding records and displaying oraccessing them sequentially.
7. End

CODE:

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

#define MAX_RECORDS 100

typedef struct {
    int id;
    char data[256];
} Record;

Record file[MAX_RECORDS];
int recordCount = 0;

void addRecord(int id, const char *data) {
    if (recordCount < MAX_RECORDS) {
        file[recordCount].id = id;
        snprintf(file[recordCount].data, sizeof(file[recordCount].data), "%s", data);
        recordCount++;
    } else {
        printf("File is full. Cannot add more records.\n");
    }
}

void readRecords() {
    for (int i = 0; i < recordCount; i++) {
        printf("Record ID: %d, Data: %s\n", file[i].id, file[i].data);
    }
}

int main() {
    addRecord(1, "First record");
    addRecord(2, "Second record");
    addRecord(3, "Third record");

    printf("Reading all records:\n");
    readRecords();

    return 0;
}
```

OUTPUT:


OnlineGDB
online compiler and debugger for c/c++
Welcome, Siva Shirish
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
main.c
Reading all records:
Record ID: 1, Data: First record
Record ID: 2, Data: Second record
Record ID: 3, Data: Third record
...Program finished with exit code 0
Press ENTER to exit console.