B. Siva Shirish-192324016

35. Consider a file system that brings all the file pointers together into an index block. Theith entry in the index block points to the ith block of the file. Design a C program to simulate the file allocation strategy.

AIM

To design a C program that simulates a **File Allocation Strategy** using an **Index Block**, where all the file pointers are brought together into an index block, and each entry in the index block points to the respective block of the file.

ALGORITHM

1. Start
2. Define a structure FileBlock to represent a block in the file.
3. Create an array indexBlock[] to represent the index block that holds pointers to file blocks.
4. Create a function to add a new record to the file and update the index block accordingly.
5. Create a function to display the current file blocks and index block.
6. Create a function to access a specific file block using the index block.
7. Stop

PROCEDURE

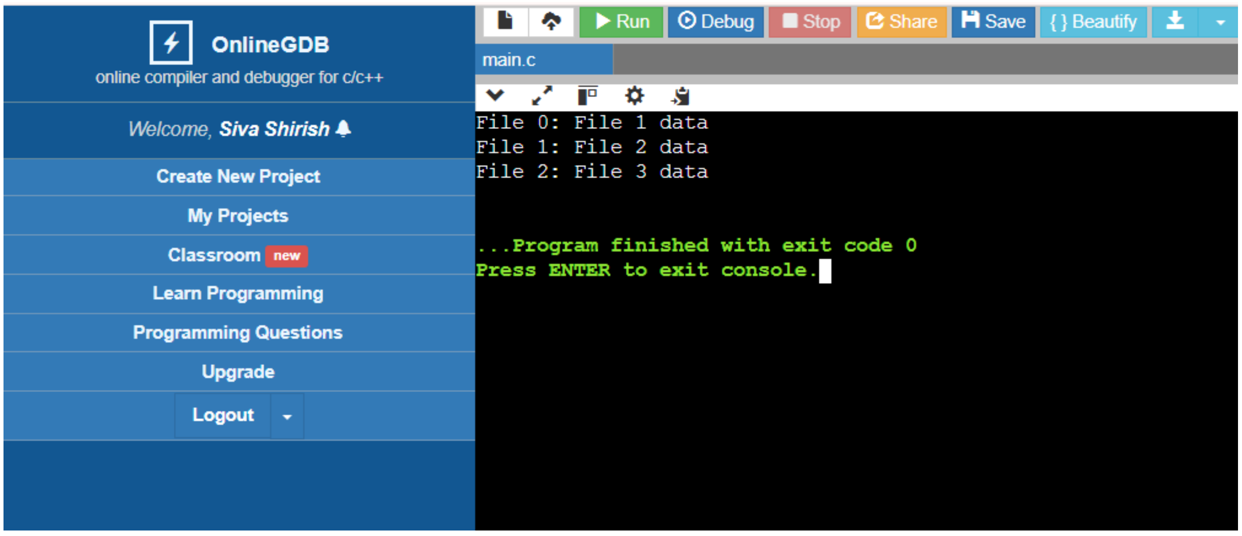
1. Include necessary libraries (stdio.h for input/output, stdlib.h for dynamic memory management).
2. Define a structure FileBlock to represent each file block (with data).
3. Define an array indexBlock[] to simulate the index block storing pointers to file blocks.
4. Implement functions to add new file blocks (addFileBlock()), display file blocks (displayFile()), and access specific blocks (accessFileBlock()).
5. Initialize the file system and perform operations such as adding file blocks, displaying file contents, and accessing blocks using the index.
6. End

CODE:

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

#define MAX_FILES 10
#define BLOCK_SIZE 1024

typedef struct {
    int block_number;
    char data[BLOCK_SIZE];
} Block;

typedef struct {
    Block blocks[MAX_FILES];
    int index[MAX_FILES];
    int file_count;
} FileSystem;
```

```
    void initialize(FileSystem *fs) {
        fs->file_count = 0;
    }

    void allocate_file(FileSystem *fs, const char *data) {
        if (fs->file_count < MAX_FILES) {
            fs->blocks[fs->file_count].block_number = fs->file_count;
            snprintf(fs->blocks[fs->file_count].data, BLOCK_SIZE, "%s", data);
            fs->index[fs->file_count] = fs->file_count;
            fs->file_count++;
        } else {
            printf("File system is full.\n");
        }
    }

    void display_files(FileSystem *fs) {
        for (int i = 0; i < fs->file_count; i++) {
            printf("File %d: %s\n", fs->index[i], fs->blocks[i].data);
        }
    }

    int main() {
        FileSystem fs;
        initialize(&fs);

        allocate_file(&fs, "File 1 data");
        allocate_file(&fs, "File 2 data");
        allocate_file(&fs, "File 3 data");

        display_files(&fs);

        return 0;
}
```

OUTPUT:



```
File 0: File 1 data
File 1: File 2 data
File 2: File 3 data


...Program finished with exit code 0
Press ENTER to exit console.
```