Programming and Data Structures - I

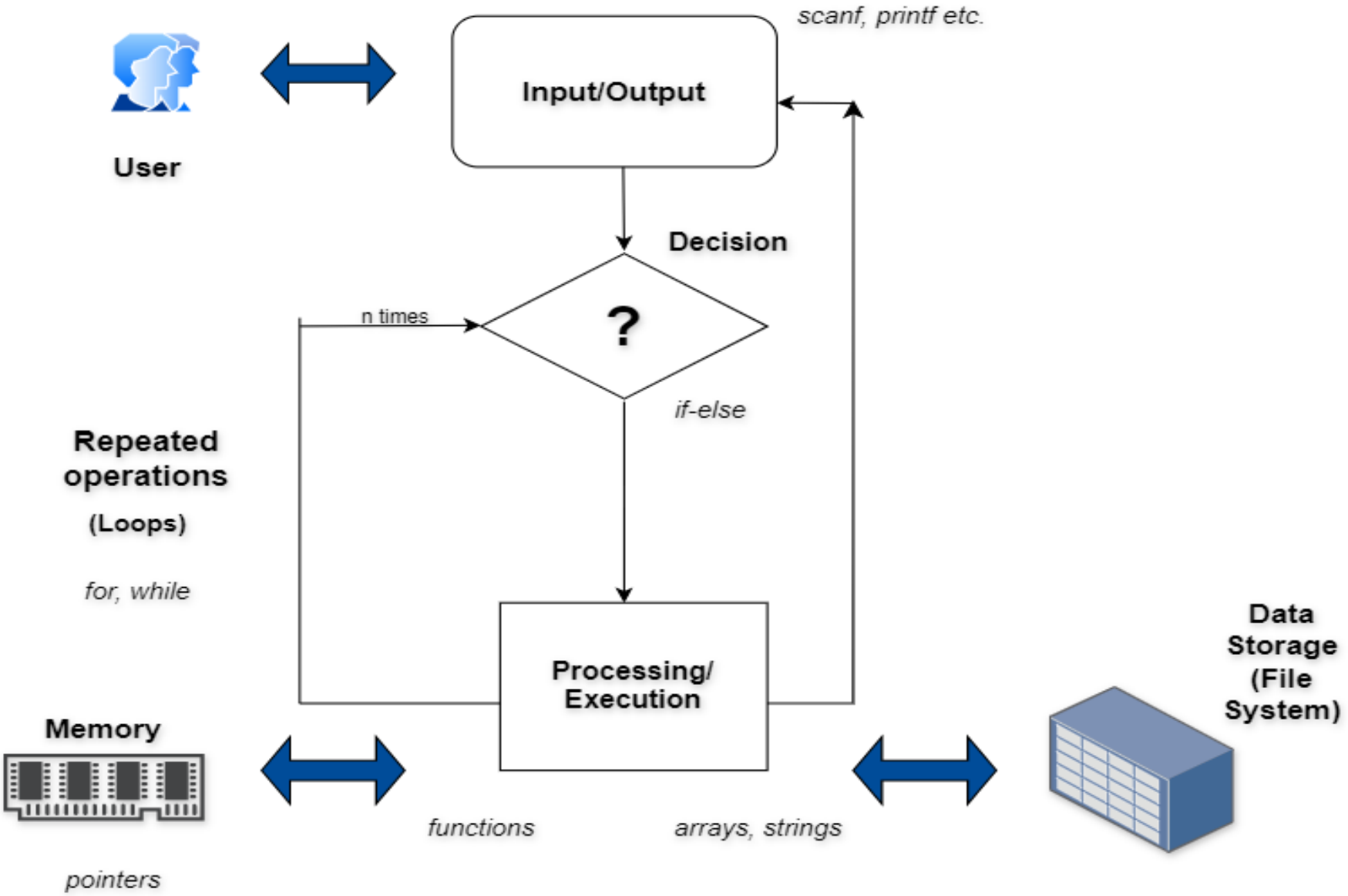
Lecture 14

Kripabandhu Ghosh

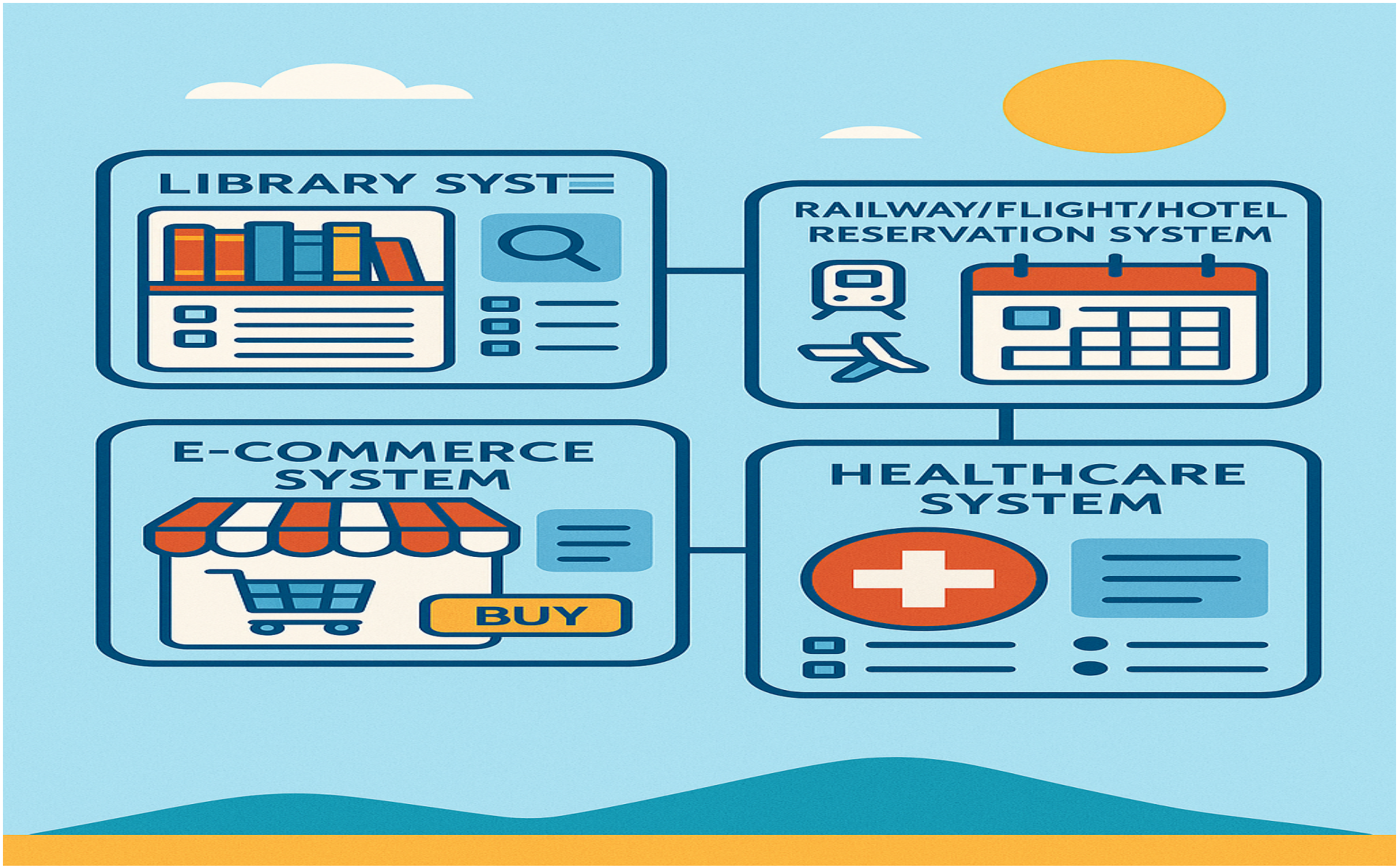
CDS, IISER Kolkata

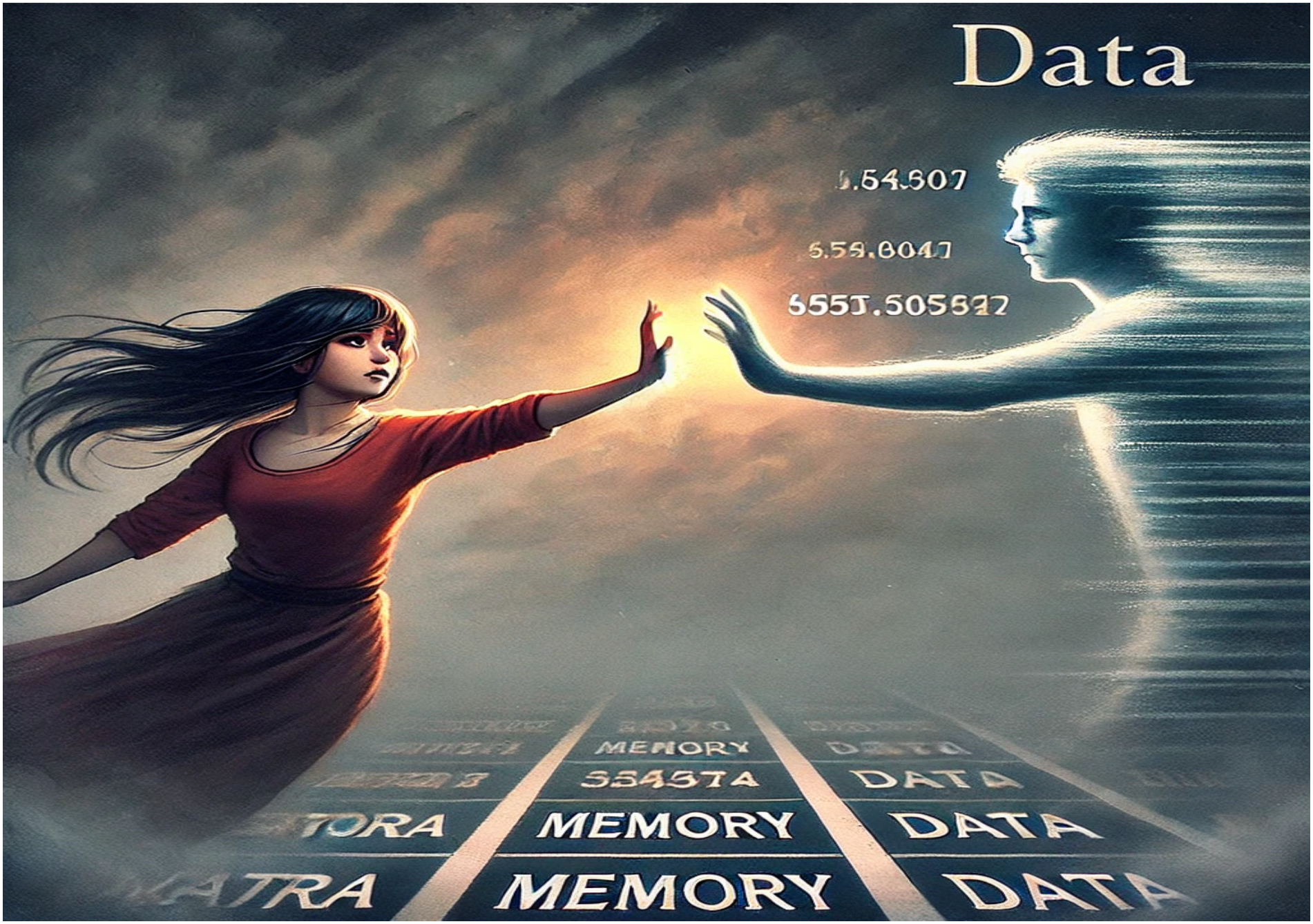
FILE MANAGEMENT

The Big Picture (Again!)



Real-life Systems




Data
MEMORY
DATA
MEMORY
DATA
MEMORY
DATA

Why Files?

- **Storage Preservation:** Entire data is lost when the program terminates $\Rightarrow$ storing in a file will preserve data even if the program terminates
- **Update Preservation:** Changes run in runtime not preserved across runs $\Rightarrow$ files allow changes to be retained
- **Storage Capacity:** Large amount of data can be better handled in files, if needed can be partly read
- **Storage Portability:** Transfer of data from one computer to another

File Management

Aspects

- **Filename**: A string with (optional) extension (e.g. books.txt, program.c)
- **Purpose**: Read, write and append

File stored in secondary memory (hard disk, external storage)

Defining and Opening a File

Format

```
FILE *fptr;
fptr = fopen("filename", "mode");
```

- **FILE**: A structure defined in I/O library
- **fptr**: pointer to FILE type (points to the first byte of the file content)
- **mode**: read, write and append

Defining and Opening a File: Modes

Type	Meaning
r	Open the file for reading only.
w	Open the file for writing only.
a	Open the file for appending (adding) data only.
r+	An existing file is opened for reading and writing.
w+	Same as **w** except both for reading and writing.
a+	Same as **a** except both for reading and writing.

Closing a File

Format

```
fclose(fptr);
```

Reading and Writing: fread

Format

fread(void *ptr, size_t size, size_t nmemb, FILE *fptr);

fptr: Pointer to the file from which **nmemb** elements of **size** bytes are to be read and stored at **ptr**

Example

fread(&book, sizeof(BOOK), 1, fptr);

Reading and Writing: fwrite

Format

fwrite(const void *ptr, size_t size, size_t nmemb, FILE *fptr);

fptr: Pointer to the file to which **nmemb** elements of **size** bytes are to be written from **ptr**

Example

fwrite(&book, sizeof(BOOK), 1, fptr);

Reading and Writing

Program

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

typedef struct book{
   char title[50];
   char id[10];
   int no;
}BOOK;

void fileWrite(char *filename, BOOK* books, int no_of_records)
{
   FILE *fptr;
   int i;
   printf("\nWriting in file %s:\n\n", filename);
   fptr = fopen(filename, "w");
   if(fptr == NULL)
   {
      printf("Error opening file!\n");
      exit(1);
   }
   for(i = 0 ; i < no_of_records ; i++)
   {
      fwrite(&books[i], sizeof(BOOK), 1, fptr);
   }
   fclose(fptr);
}
```

Reading and Writing

Program (contd.)

```
void fileRead(char *filename)
{
   FILE *fptr;
   BOOK book;
   printf("\nReading from file %s:\n\n", filename);
   fptr = fopen(filename, "r");
   if(fptr == NULL)
   {
      printf("Error opening file!\n");
      exit(1);
   }
   while(fread(&book, sizeof(BOOK), 1, fptr))
   {
      printf ("Title = %s id = %s stock = %d\n", book.title, book.id, book.no);
      printf("Current position: %ld \n", ftell(fptr));
   }
   fclose(fptr);
}
```

Reading and Writing

Program (contd.)

```
void fileUpdate(char *filename)
{
  FILE *fptr;
  BOOK book;
  printf("\nUpdating in file %s:\n\n", filename);
  fptr = fopen(filename, "r+");
  if(fptr == NULL)
  {
    printf("Error opening file!\n");
    exit(1);
  }
  while(fread(&book, sizeof(BOOK), 1, fptr))
  {
    printf ("Title = %s id = %s stock = %d\n", book.title, book.id, book.no);
    printf("Current position: %ld ", ftell(fptr));
    if(strcmp(book.id, "B123") == 0)//Check for a particular book id
    {
      book.no = book.no - 2;//Updating the stock
      fseek(fptr, -sizeof(BOOK), 1);
      fwrite(&book, sizeof(BOOK), 1, fptr);
      break;
    }
  }
  rewind(fptr);
  printf("\n\nAfter update:\n");
  while(fread(&book, sizeof(BOOK), 1, fptr))
  {
    printf ("Title = %s id = %s stock = %d\n", book.title, book.id, book.no);
    printf("Current position: %ld \n\n", ftell(fptr));
  }
  fclose(fptr);
}
```

Reading and Writing

Program (contd.)

```
void fileUpdateDirect(char *filename)
{
  FILE *fptr;
  BOOK book;
  long int n, block_no;
  printf("\nUpdating via direct access in file %s:\n\n", filename);
  fptr = fopen(filename, "r+");
  if(fptr == NULL)
  {
    printf("Error opening file!\n");
    exit(1);
  }
  block_no = 2;//Number of record to be updated
  n = (block_no - 1)*sizeof(BOOK);//The starting address of the appropriate record
  printf("\nn = %ld\n", n);
  fseek(fptr, n, 0);
  printf("Current position: %ld ", ftell(fptr));
  fread(&book, sizeof(BOOK), 1, fptr);
  book.no = book.no - 2;
  fseek(fptr, -sizeof(BOOK), 1);
  fwrite(&book, sizeof(BOOK), 1, fptr);
  rewind(fptr);
  printf("\n\nAfter update:\n");
   while(fread(&book, sizeof(BOOK), 1, fptr))
  {
    printf ("Title = %s id = %s stock = %d\n", book.title, book.id, book.no);
    printf("Current position: %ld \n", ftell(fptr));
  }
  fclose(fptr);
```

Reading and Writing

Program (contd.)

```
int main(void)
{
   BOOK books[2];
   char filename[] = "books1.txt";

   strcpy(books[0].title, "Let us C");
   strcpy(books[0].id, "B123");
   books[0].no = 5;

   strcpy(books[1].title, "Introduction to Algorithms");
   strcpy(books[1].id, "B567");
   books[1].no = 10;

   fileWrite(filename, books, 2);
   fileRead(filename);
   fileUpdateDirect(filename);
   return 0;
}
```

Reading and Writing

Output

Writing in file books1.txt:

Reading from file books1.txt:
Title = Let us C id = B123 stock = 5
Current position: 64
Title = Introduction to Algorithms id = B567 stock = 10
Current position: 128

Updating via direct access in file books1.txt:
n = 64
Current position: 64

After update:
Title = Let us C id = B123 stock = 5
Current position: 64
Title = Introduction to Algorithms id = B567 stock = 8
Current position: 128

File I/O functions

Name	Operation
fread()	Reading blockwise from a file
fwrite()	Writing blockwise to a file
getc()	Reading characters from a file
putc()	Writing characters to a file
fscanf()	Reading formatted data from a file
fprintf()	Writing formatted data to a file
fgets()	Reading line-wise from a file
fputs()	Writing line-wise to a file

File I/O functions: Random access

fseek(fptr, offset, position)

- **offset**: Number of positions (int long: bytes) to be moved from **position**
- **position**:
  - **0**: beginning of file
  - **1**: current position
  - **2**: end of file

Name	Operation
fseek(fptr, 0L, 0)	Go to the beginning
fseek(fptr, 0L, 1)	Stay at the current position
fseek(fptr, 0L, 2)	Go to the end
fseek(fptr, m, 0)	Go to the (m + 1)th byte
fseek(fptr, m, 1)	Go forward by m bytes
fseek(fptr, -m, 1)	Go backward by m bytes
fseek(fptr, -m, 1)	Go backward by m bytes from the end

File I/O functions: Random access

Name	Operation
ftell(fptr)	Returns the number of bytes written (or read)
rewind(fptr)	Goes to the beginning of the file

Reading and Writing: Line-wise

Program

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

void main()
{
  FILE *fptr;
  char c, name[3][50] = { "Durga", "Durgatinashini", "Mahishashurmardini" }, buff[100];
  int i;
  fptr = fopen("Data2.txt", "w");
  for(i = 0; i < 3 ; i++)
  {
    fputs(name[i], fptr);
    fputs("\n", fptr);
  }
  fclose(fptr);
  fptr = fopen("Data2.txt", "r");
  while (fgets(buff, 100, fptr) != NULL)
  {
    printf("%s", buff);
  }
  fclose(fptr);
}
```

Reading and Writing: Line-wise

Program

Durga
Durgatinashini
Mahishashurmardini

Reading and Writing: Characters

Program

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

void main()
{
   FILE *fptr;
   char c, name[] = "Mahishashurmardini";
   int i = 0;
   fptr = fopen("Data.txt", "w");
   c = name[i++];
   while(c != '\0')
   {
      putc(c, fptr);
      c = name[i++];
   }
   fclose(fptr);
   fptr = fopen("Data.txt", "r");
   c = getc(fptr);
   while (c != EOF)
   {
      printf("%c", c);
      c = getc(fptr);
   }
   fclose(fptr);
}
```

Reading and Writing: Characters

Program

Mahishashurmardini


Data
Data
Data
DATA


THANK
YOU