Data File Handling

DELTA | Pg No. 1.
Date / /

DATA:- Data is a collection of facts and figures.

Information:- It is the data can be Processed so that it becomes meaningful and understandable.
=> Data Arranged in ordered and useful form.

>>FILE>>

A File Consists of number of records and each record consists of a number of items known as fields.
The Entire Information of System Consisting of given records is referred to as a file.

Files Arrangement:-

(i) Ascending Order / Descending order
(File Arranged According to Ascending or descending order of a Key field)

(ii) Alphabetical Order
(If a field is of string or Character type then this Arrangement is used.

(iii) Chronological Order
(The Records are arranged in the order of their Occurance.) Arranged according to date and events.

File Organization Techniques

The Former System is Knowns as Manual Filling system and later is called as EDP (Electronic Data Processing) filling system.

New File Organization Techniques

(i) Sequential File organization
(ii) Direct access file organization
(iii) Indexed sequential file organisation

(FILES IN "C")

- Text :: Contains ASCII codes only
- Binary :: can contain non-ASCII Characters.

Stream:- A Stream is a data flow from a source to a sink.
The source and sinks can be any of the input/output devices or files.

"C" Supports a very large number of I/O functions, capable of performing a wide variety of tasks.

1.) The Stream I/O system
2.) The Consol I/O system
3.) The low-level I/O system

FUNCTION		PURPOSE
chmode	⟶	Change the mode of a file.
Close	⟶	Close the file
eof	⟶	test for end of file
file length	⟶	determine the length of file
Iseek	⟶	move or read the position of a file pointer.
Open	⟶	Open a file.
Read	⟶	Read data from a file
Setmode	⟶	set mode of a file.
write	⟶	write data to a file.

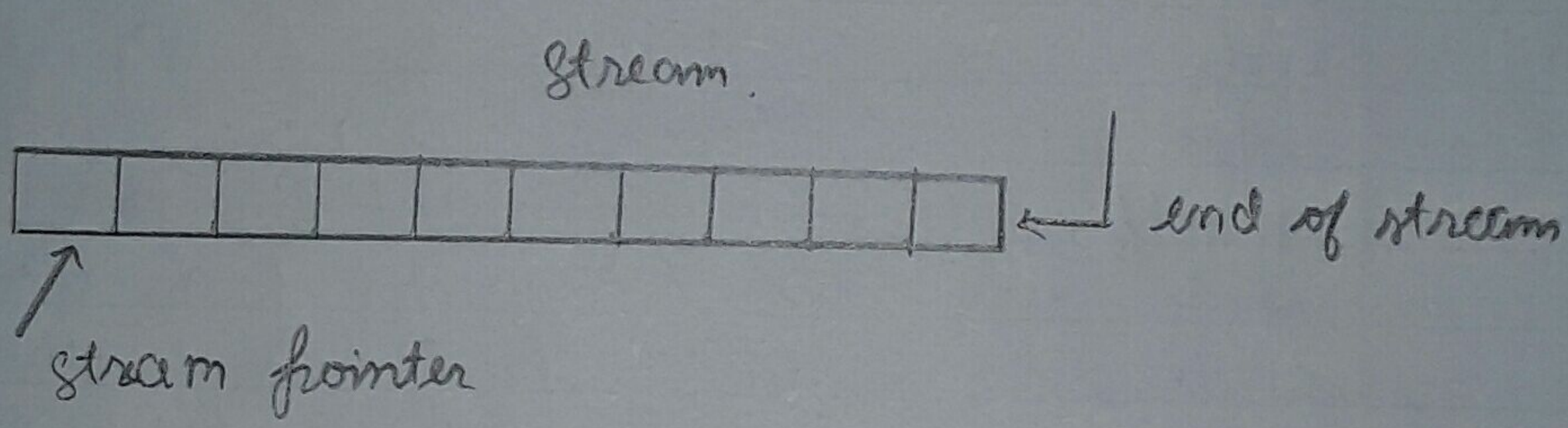
Low-level I/O functions.

STREAM I/O FUNCTIONS

Function	Purpose.
fclose	Close a stream
feof	Test for end of file
fflush	Flush a stream
fgetc	Read a Character from a stream
fgetchar	Read a Character from a Stream
fgets	Read a string from a stream
fopen	Open a stream
fprintf	send formatted data to a stream
fputc	send a Character to a stream
fputs	send a string to a stream
fread	Read a block of datal from a stream.
fscanf	Read formatted data from a stream
fseek	Reposition a stream pointer.
fwrite	write a block of data to a stream
fputs	send a string to a stream.

CONSOL I/O FUNCTION.

cgets	Read a string from the Consol
clrscr	Clear text window.
cprintf	write formatted data to Consol
getch	Read a Character from the Consol
getche	Read a character from consol and echo it
gotoxy	move the cursor to a specified locatin
putch	write a Character to the Consol



WORKING OF A STREAM

Working with files using Stream I/O

A Stream is accessed by a stream pointer. The Stream pointer points to the beginning of a stream.

Opening a file

STEP-1. Declare a pointer (say ptr) of type FILE => FILE * ptr;
STEP-2. Open the file using a stream I/O function called fopen().
=> ptr = fopen ("Myfiles", "r");

Closing a file

A file can be Closed by a function called fclose() which takes one argument of type FILE.
Thus, The file ("myfile") can be closed by the following statement.

fclose(prt);

```
#include<stdio.h>
main()
{ FILE *fp;
   fp = fopen("data.txt", "r");
   if (fp == NULL) {
     Printf ("File does not exist, please check!\n");
        }
            fclose (fp);
   }
```

MODES	PURPOSE
"r"	Open a file for reading. If file does not exist NULL is returned.
"w"	Open a file for writing. If the file does not exist a new file is created. If the file exists, the new contents overwrite the previous contents.
"r+"	Open the file for both reading and writing. If the file does'not exist. NULL is returned.
"w+"	Open the files for both reading and writing. If the file doesnot exist, the previous contents are overwritten by the new one.
"a"	Open a file for appending. If the file exists, the new data is written at the end of the file else the new file is created.
"a+"	Open the file for reading and appending. If the file doesn't exist, a new file is created.

Unformatted File I/O Operations

An opened file can be read or written by following set of unformatted I/O functions.

(i) fget()	(iv) get()	(vii) fgetchar()
(ii) getw()	(v) fputc()	(viii) putc()
(iii) fputchar()	(vi) fputs()	(ix) putw().

- fputs()
  fputs (< string >, < file pointer >)
- fgets()
  fgets (< string >, n, < file-pointer >)

Similarly,

fgetc() and fputc() functions can be used to read from or write a character to a file.

fgetc() takes only one argument the file pointer.

mple

ch = fgetc(ptr) :- Reads a character in ch from a file pointed by the pointer called ptr.
fputc (ch, ptr) :- Writes a character in ch from a pointed by the pointer called ptr.

DELTA Pg No. 6
Date / /

REQUIRED PROGRAM

- (fputs())

```
#include <stdio.h>
main()
{ FILE *ptr;
  ptr = fopen("message.dat","w");
  if (!ptr)
{ printf("\n The file cannot be opened");
  exit(1);
} fputs("When an apple fell, Newton was distursb\n",ptr);
fputs("but when he found that all apples fell,\n",ptr);
fputs("It was gravitation and he was satisfied",ptr);
fclose(ptr); }
```

- (fgets())

```
#include <stdio.h>
main()
{ char text[80];
  FILE *ptr;
  ptr = fopen("message.dat","r");
  if (!ptr)
{ printf("\n The file Cannot be opened");
  exit(1);
} while(!feof(ptr)
{ fgets(text,80,ptr);
  printf("\n %s", text);
}   fclose(ptr);
}
```

DELTA | Pg No. 7
Date / /

Formatted file I/O Operations

'C' supports fscanf() and fprintf() functions to read to or write file.

fscanf (<file pointer>, <format string>, <argument list>);
fprintf (<file pointer>, <format string>, <argument list>);

<File pointer> : Is the pointer to the file in which I/O operations are to be done.
<Format string> : Is the list of format specifiers i.e. %d, %f, %s etc.
<Argument list> : Is the list of arguments.

Reading or writing blocks of data in files :

An array or a structure can be written a file by using fwrite() function and can be read by fread() fun^c.

fread (<address of object>, <size of object>, <number of items>, <file pointer>);
fwrite(< ">, < ">, < ">, < ">);

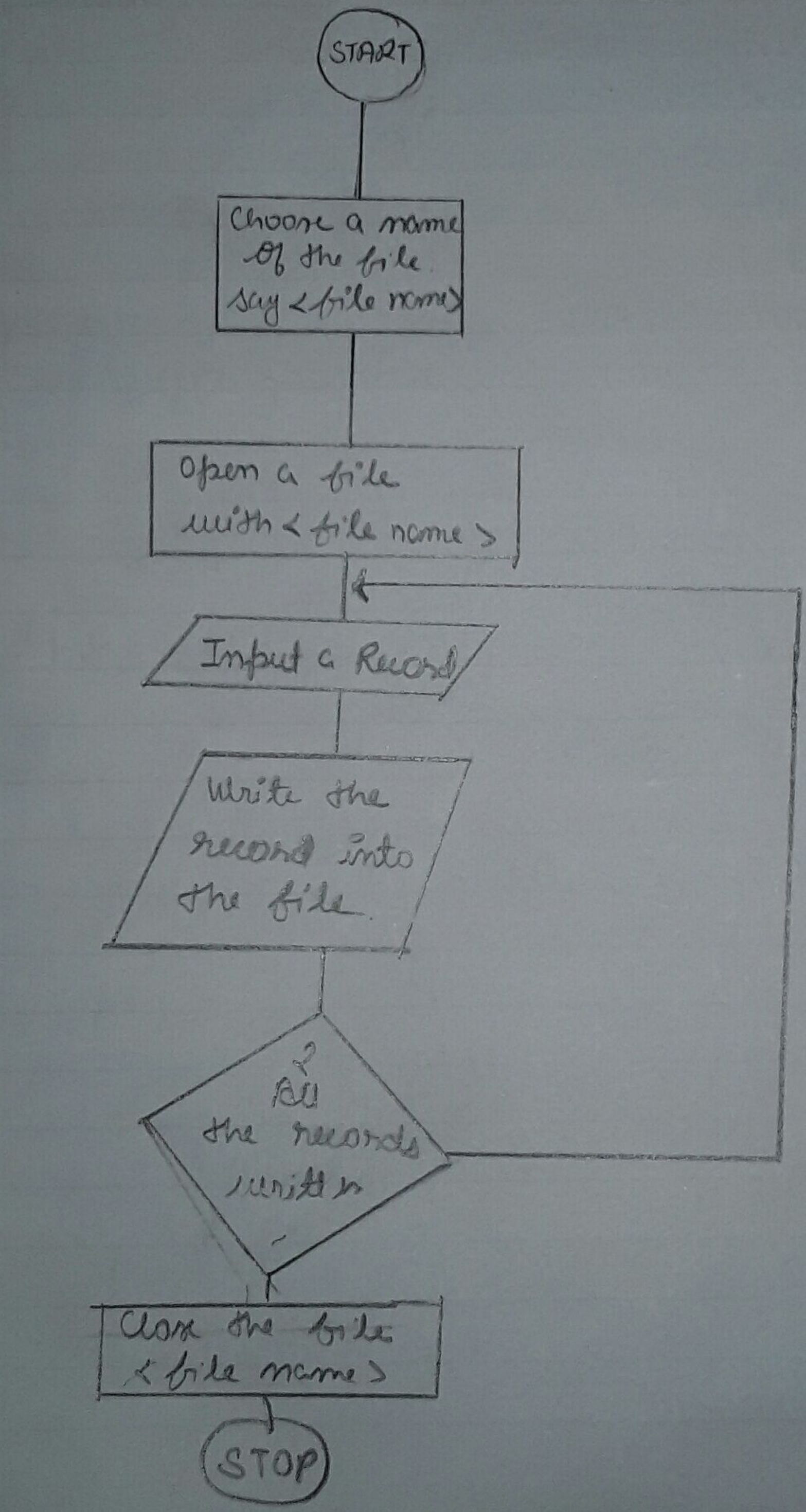
<Address of object> : Is the address or pointer to the object from which the data is to be read/written on the file.
<Size of object> : The size of the object is computed by the fun^c size of () and included in the Argument list.
<Number of items> : The Number of data items to be read or writ
<File pointer> : Is the pointer to the file from which data is to read or written.

DELTA | Pg No. 8.
Date / /

- A Program that makes a list of numbers from user through standard input device. The number -9999 marks the end of the list. Using fprintf() function.

Soln:
```
#include <stdio.h>
main()
{ FILE *ptr;
  int val, i;
  char outfile[20];
  printf("\n Enter the name of the file:");
  scanf("%s", outfile);
  ptr = fopen(outfile, "w");
  do{ printf("\n Enter a value");
      scanf("%d", &val);
  if (val != -9999) fprintf(ptr, "%d", val);
}
  while (Val != -9999);
  fclose(ptr);
}.
```

A FLOW CHART OF READING OF FILE



DELTA | Pg No. 9
Date / /

SEQUENTIAL FILE ORGANIZATION

- This file organization is the most simple way to store and retrieve records of a file.

(1) The Sequential file can be created on a storage device

(i) A name for the file is Chosen. The System is requested to Open a file on the storage device with Chosen file-name

(ii) The Records for the file are input one by one into the Computer which in turn stores them in the order of their arrival on to the file.

(iii) After all the Records are transferred into the file, the system is requested to close the file.

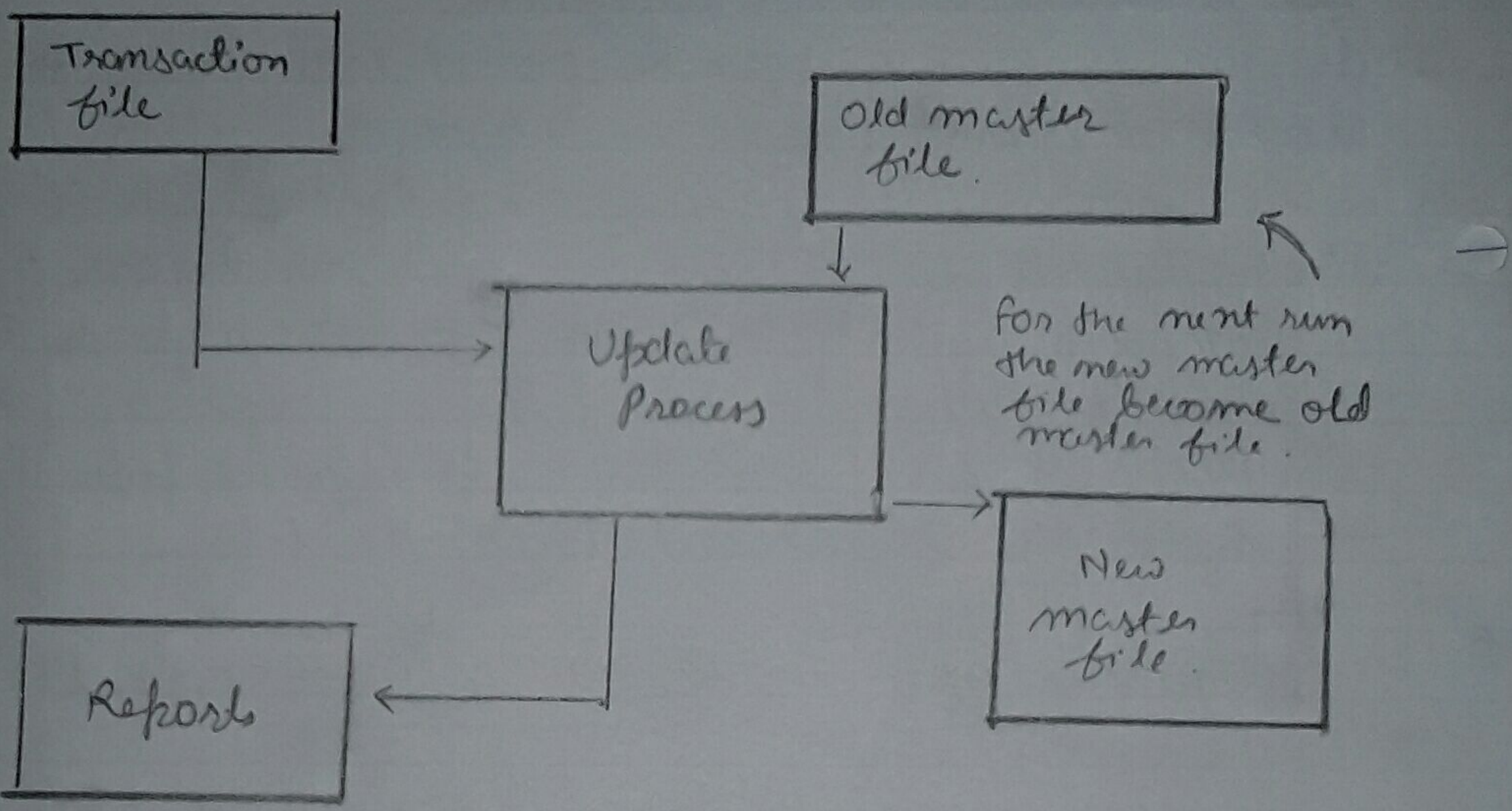
READING AND SEARCHING A SEQUENTIAL FILE

To Read a Sequential file, the system always starts at the beginning of the file and the records are read one by one till the required record is Reached or EOF

The Reading of files involves the following operations.

1.) Open the file.
2.) Read a record
3.) Check if the record is the desired one.
4.) Check for the end of file (EOF).
5.) Close the file.

UPDATING A SEQUENTIAL FILE



m

APPENDING A SEQUENTIAL FILE

The Append operation is done with a view of adding more records to an existing file.

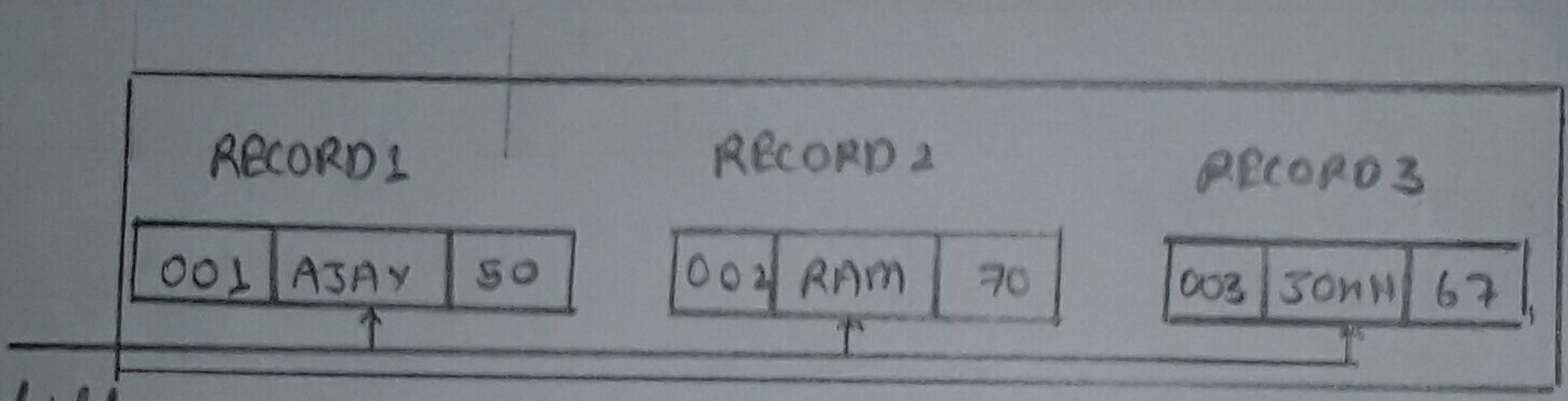
=> Operations

1.) Open the file for append operation.
2.) Read the file till EOF is encountered.
3.) Read the record to be added.
4.) Write the record of file.
5.) Close the file.

UPDATING A SEQUENTIAL FILE

- The Original file is known as a Master file or Old master file.
- And, The temporary file which contains the changes or transactions is known as Transaction file.

Both files are sorted in the same order on the same key. Thus, Master file and Transaction file become Co-sequential. A new file called as New Master file.



ARRANGEMENT OF RECORDS IN FILE

DELTA | Pg No. 11
Date / /

DIRECT FILE ORGANIZATION

The Direct files allow records to be read or written on to a file without proceeding sequentially from the beginning.
The Record is then stored at the generated address. The Mathematical function is called as a Hash function.

fseek (<file pointer>, <offset>, <start-position>)

<file pointer> :- Is the pointer to the file.
<start position> :- Is the start position of the file. It can have either of the following values

VALUE	MEANING
0.	The starting position is the begining of file
1.	Use the current file pointer as starting pos
2.	The starting position is end of a file.

<Offset> :- Is the offset i.e how much the pointer should advance from the starting positio

A Relative file organization wherein the records are accessed directly in relation with the start point specified within the file.

Offset = Record Number + Size of a Record.

GRADED PROBLEMS

The macroprocessor replaces all occurrences of ma name in a document by its corresponding defi

```
# define    CE      Computer Engineering.
# define    PE      Personal Engineering.
  MEND
```

<Text of the Document>

The Program Implements the macro processor

FUNCTION	DESCRIPTION
1.) get-token()	: To Read a word from the input text doc
2.) Build-Mtable()	: To Build a table of macro names and definitions using array of structure
3.) Write out()	: Write the output to the output text file

⇨ The get_token() and write out() functions of previous program would be used in this program to read a token and to write the tokens to a given file

1.) <int> ator(string):
This function Converts an integer to an Equivalent. This function takes a string of characters and Converts it into an equivalent Integer.

2.) void itoa(int, string, <radix>):
This function converts an integer to an equivalent string. <radix> is the radix of the integer such as 2, 10 depending upon the number being binary or decimal.