Assignment - 5

1 Define pointer. How can you declare it. Write a C program to read and print an array of elements using pointers

A Pointers :

- Pointer is a variable that stores or holds the address of the another variable of same data type. A pointer is a derived data type in C
- Pointer initialization is the process of assigning address of a variable to pointer variable. Pointer variable contains address of a variable of same data type
- By the help of (*) (in direction operator) we can print the values of pointer variable
- & (amphacent) is called as reference operator which gives address of a variable

Declaration of pointer :-

```
data-type * pointer variable-name ;
  int * P ;
```

WACP to read and print an array of elements using pointers

```
#include <stdio.h>
int main() {
int n,i ;
int * ptr ;
printf (" Enter the number of elements : ");
scanf (" %d ",&n);
int arr[n];
ptr = arr ;
printf ("Enter %d elements : \n", n);
for (i = 0; i < n ; i++ ) {
scanf ("%d", (ptr+i));
printf (" The elements in the array are \n ");
for (i=0; i < n ; i++)
{
printf ("%d", *(ptr+i));
}
```

```
printf("\n");
return 0;
}
```

OUTPUT

```
Enter the number of elements : 5
Enter 5 elements :
10
20
30
40
50
The elements in the array are :
10  20  30  40  50
```

2 Write a C program to count number of character, lines and words of a file

A

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp;
char path[100];
char ch;
int characters, words, lines;
printf("Enter source file path : ");
scanf("%s", path);
fp = fopen(path, "r");
if(fp == NULL)
{
printf("\n Unable to open file.\n");
printf("Plase check if file exists and you have read privilge.\n");
exit(EXIT_FAILURE)
}
```

```
characters = words = lines = 0;
while ((ch = fgetc (file)) != EOF)
{
 characters ++;
if (ch == '\n' || ch == '\0')
 lines ++;
if (ch == ' ' || ch == '\t' || ch == '\n' || ch == '\0')
words ++;
}
if (characters > 0)
{
 words ++;
 lines ++;
}
printf ("\n");
printf (" Total characters = %d\n", characters);
printf (" Total words = %d\n", words);
printf (" Total lines = %d\n", lines);
 fclose (file);
 return 0;
}
```

Creating a text file :

File.txt

This is ISL college at bandlaguda

OUTPUT

```
Enter source file path : C:\Users\user3\One Drive\Documents\file.txt
 Total characters = 33
 Total words = 6
 Total lines = 1
```

3 Write a short note on

i) Sequential / Random

ii) Command line argument

A i) Sequential Access

Sequential Access is not the optimum method for reading the data in the middle of the file if the file size is too large. A sequential file is a kind of file organization in which we can keep the data continuously and every record is stored after the other and can be accessed the data sequentially

Random Access

A Random Access file in C is kind of file that enables us to write or read any data in disk file without having to read or write each section of data before it. In this file we may instantly find the data, modify it, or even remove it.

ii) Command line argument.

- Command line arguments are the values given after the name of the program in the command line • shell of operating system.
- Command line arguments are handled by the main function of a C program
- To pass the command line arguments we define main function with two arguments

i) No of command line arguments

ii) list of command line arguments

Syntax: int main (int argc, char* argv [])

```
{
// body of the function)
}
```

4 Explain the following file handling

i) fseek ( )

ii) ftell ( )

iii) rewind ( )

iv) feof ( )

A i) fseek ( )

The fseek ( ) function is used to set the file pointer to the specified offset. It is used to write data into file at desired location

Syntax : fseek (FILE *stream, long int offset, int whence)

ii) ftell ( )

The ftell ( ) function returns the current file position of the specified stream. We can use ftell ( ) function to get the total size of a file after moving file pointer at the end of file. We can use SEEK_END constant to move the file pointer at the end of file

syntax : n = ftell (fp)

n would give the relative offset (in bytes)

iii) rewind ( )

This function places the file pointer to the beginning of the file, irrespective of where it is present right now. It takes file pointer as an argument

syntax : rewind (fp);

iv) feof

feof ( ) function finds end of file

Explain the concept of array of pointer with example program ?

An array of pointers is a collection of memory addresses (pointers) where each pointer in the array points to either a single variable or the first element of another array

- Each to element in the array is a pointer
- It is particularly useful when you need to store addresses of multiple variable or strings
- The array itself resides in contigous memory, but the data it points to can be scattered across memory

ex:-

```
#include <stdio.h>
int main () {
const char * names [ ] = { "Alice", "Bob", "Charlie", "Diana" };
int size = sizeof(names) / sizeof(names[0]);
for ( int i =0 ; i < size ; i++ )
{
  printf("Names [%d]: %s\n", i , names [i]);
}
return 0;
}
```

OUTPUT

Name [0]: Alice

Name [1]: Bob

Name [2]: Charlie

Name [3]: Diana

6 Write in detail about 'file modes'

A File modes specify how a file should be accessed whether for reading, writing, appending, or a combination of these operations. They also determine whether the file is treated as a text or binary file. When opening a file using the fopen() function you provide a mode string that defines the intended file operation

Common File Modes

1. Read Mode ("r" / "rb");

- Used to open an existing file for reading
- The file pointer is positioned at the beginning of the file
- If the file doesn't exist fopen() returns NULL

ex: FILE * file = fopen("output.txt", "r");

```
if (file == NULL) {
    perror("Error opening file");
    return 1;
} /* Read operations */
fclose(file);
```

2. Write Mode ("w" / "wb"):

- Used to create a new file for writing or to overwrite an existing file
- If the file exists, its contents are cleared
- If the file doesn't exist, a new file is created

Example: FILE* file = fopen("output.txt", "w");

```
if (file == NULL) {
    perror("Error opening file");
    return 1;
}
fclose(file);
```

3. Append Mode ("a" / "ab"):

- Used to open a file for appending data at the end of file. Data is written at the the end of the file. If the file doesn't exist, a new file is created
- The file pointer is positioned at the end of the file

example: FILE *file = fopen("log.txt", "a");

```
if (file == NULL) {
perror ("error opening file");
return 1;
}
fclose (file)
```

4. Read and Write mode ("r+" / "rb+"):

- Opens an existing txt file for reading and writing. If the file doesn't exists, fopen() returns NULL
- The file pointer at the beginning of the file

example

```
FLLE *file = fopen("data.txt", "r");
if (file == NULL) {
perror ("error opening file");
return 1;
}
fclose (file);
```

5. Write and Read Mode ("w+" / "wb+"):

- Used to create a new file for both reading and writing, or to overwrite an existing file if the file ~~doesn't~~ exists, its contents are cleared.
- If the file doesn't exist, a new file is created

example

```
FILE *file = fopen ("output.txt", "w+");
if (file == NULL) {
perror ("error opening file");
return 1;
}
fclose (file);
```

c. Append and Read mode ("a+" / "ab+"):

- Used to open a file for for both reading and appending.
- If the file doesn't exist, it is created
- The file pointer is positioned at the end of the file

Q What is self referential structure?

A Self Referential Structure

A structure consists of at least a pointer member var pointing to the same structure is used to create data structures like linked lists, stacks. etc.

Syntax:
```
struct tag_name
{
 Type member 1;
 Type member 2;
  :     :
  :     :
 type  member N;
struct tag_name * name;
}
```

example:
```
struct emp
{
 int code;
struct emp* name
}
```

- Following is example of this kind of structure.
- self referential structure allows the structure to refer itself by creating a linked data structure
- EMP in the given program is self referential structure by because it contains a pointer (*) to our name objid

8 Differentiate between text and binary file.

A

Text File	Binary File
• A file where data is stored in human readable format typically as ASCII or Unicode text	• A file where data is stored in a raw, binary
• Data is stored as a sequence	• Data is stored as raw byte values without translation
• Commonly use extensions like .txt, .csv, etc	• Commonly uses extensions like .bin, .exe, .dat etc
• Less efficient because characters and formating (like newline) are converted to human-readable form	• More efficient because data is stored in compact binary form
• Size of text files are usually larger due to the overhead of character encoding and line breaks	• Binary files are smaller because they store raw data without extra encoding
• Platform-independent due to standard text encoding	• May not be portable across platform due to differences in byte ordering or data alignment