KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

Module V

Pointers and Files-Basics of Pointer: declaring pointers, accessing data though pointers, NULL pointer,array access using pointers, pass by reference effect. File Operations: open, close, read, write, append Sequential access and random access to files: In built file handling functions (rewind() ,fseek(), ftell(),feof(), fread(), fwrite()), simple programs covering pointers and files.

POINTERS

The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture. However, in 32-bit architecture the size of a pointer is 2 byte.

Consider the following example to define a pointer which stores the address of an integer variable.

int n = 10;

*int *p = &n; // Variable p of type pointer is pointing to the address of the variable n of type integer.*

Declaration of Pointer Variable

The pointer in C language can be declared using * (asterisk symbol). It is also known as **indirection operator used to dereference a pointer**.

```
datatype *ptrvar;
        where ptrvar is any identifier
int  *a;        //pointer to integer
char *c;        //pointer to char
```

Initialization of Pointer Variable

The process of assigning the address of a variable to a pointer variable is known as initialization. All uninitialized pointers have some garbage value so it should be initialized.

Example

```
int a;
int *p;         //declaration
p=&a;           //initialization
```

Accessing Data through Pointers

A variable can be accessed through its pointer using unary operator '*'. Also known as indirection operator or dereferencing operator.

Consider the following example:

```
#include<stdio.h>
void main()
{
        int n = 10;
        int *p;
        p = &n;
        printf("Address of n=%u\n",&n);
        printf("Value of n=%d\n",n);
        printf("Address of p=%u\n",&p);
        printf("Value of p=%u\n",p);
        printf("Value of *p=%d\n",*p); //            *p = value of n;

}
```

OUTPUT

```
Address of n=6487628
Value of n=10
Address of p=6487616
Value of p=6487628
Value of *p=10
```


Each variable has two properties: address and value. Address of a variable is the memory location allocated for that variable. Value of a variable is the value stored in the memory location allocated to them. The address of a variable can be derived using **address of (&) operator**. In the example above, the address of n is 6487628 and value is 10.

Variable Name	Address	Value
n	6487628	10

When p is assigned with &n, p stores the address of n. Thus p points to n. In order to retrieve the value of n, we can make use of *p.

Pointer Name	Address of p	Value of p	Asterisk Value(*p)
p	6487616	6487628 Address of variable n	10 Value of n

NULL Pointer

A pointer that is not assigned any value but NULL is known as the NULL pointer. If you don't have any address to be specified in the pointer at the time of declaration, you can assign NULL value. It will provide a better approach.

*int *p=NULL;*

OPERATIONS ON POINTERS

- Address of a variable can be assigned to a pointer variable.
- One pointer variable can be assigned to another pointer variable provided both points to the items of same datatype.
- A NULL value can be assigned to a pointer variable.
- An integer quantity can be added to or subtracted from a pointer variable. The result will be a pointer.
- When we increment a pointer, its value is increased by the length of its data type.
- One pointer variable can be subtracted from another pointer variable provided both points to the elements of the same array. The result will be an integer value.
- Two pointer variables can be compared provided both points to the items of same datatype.
- A pointer variable can be compared with NULL values.

Write a program to find the sum of two numbers using pointers

```
#include<stdio.h>
void main()
{
        int  x,y,sum,*xp,*yp;
        printf("Enter value of x:");
        scanf("%d",&x);
        printf("Enter value of y:");
        scanf("%d",&y);
        xp=&x;
        yp=&y;
        sum=*xp+*yp;
        printf(" Sum is %d  \n",sum);
}
```

OUTPUT

```
Enter value of x:10
```

```
Enter value of y:20
Sum is 30
```

Write a program to swap two variables using pointers

```
include<stdio.h>
void main()
{
        int x=10,y=20, temp,*a,*b;
        printf("\nBefore Swapping");
        printf("\nValue of x is %d",x);        //output 10
        printf("\nValue of y is %d ",y);       //output 20
        a=&x;
        b=&y;
        temp= *a;
        *a= *b;
        *b=temp;
        printf("\nAfter swapping");
        printf("\nValue of x is %d \n",x);    //output 20
        printf("\nValue of y is %d \n",y);    //output 10
}
```


OUTPUT

```
Before Swapping
Value of x is 10
Value of y is 20
After swapping
Value of x is 20
Value of y is 10
```

POINTERS AND ARRAY

- In C programming, name of the array always points to address of the first element of an array.

Array Access using Pointers

1) Write a program to read and display an array using pointer.

```
#include<stdio.h>
#include<malloc.h>
void main()
{
        int p[30],i,n;
```

```
        printf("Enter the size of array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                scanf("%d",p+i);
        }
        printf("Array Content\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",*(p+i));
        }
}
```

OUTPUT

```
Enter the size of array:5
10 15 20 25 30
Array Content
10      15      20      25      30
```

2) Write a program to sort the content of an array using pointers

```
#include<stdio.h>
#include<malloc.h>
void main()
{
        int i,n,temp,j;
        int *p = malloc(30 * sizeof(int)); // equivalent to int p[30];
        printf("Enter the size of array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                scanf("%d",p+i);
        }
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-i-1;j++)
                {
                        if(*(p+j) > *(p+j+1))
                        {
                                temp = *(p+j);
                                *(p+j) = *(p+j+1);
                                *(p+j+1) = temp;
                        }
                }
        }
        printf("Array Content\n");
```

```
        for(i=0;i<n;i++)
        {
                printf("%d\t",*(p+i));
        }
}
```

OUTPUT

```
Enter the size of array:5
16   7   14  2   5
Array Content
2     5     7     14     16
```

POINTER TO POINTER

A pointer points to an address of another pointer. Such pointers are known as pointer to pointer or double pointers.

Example

```
#include<stdio.h>
void main()
        {
            int a=10;
            int *p,**q;
            p=&a;
            q=&p;
            printf("\nValue of a  is %d \n",a);
            printf("\nValue of *p is %d  \n",*p);
            printf("\nValue of  **q is %d \n ",**q);
        }
```


OUTPUT

```
Value of a is 10
Value of *p is 10
Value of **q  is 10
```

PASS BY REFERENCE

In C programming, it is also possible to pass addresses as arguments to functions. To accept these addresses in the function definition, we can use pointers. In this method, the address of the actual parameters is passed to formal parameters. So any change in formal parameters will be reflected in the actual parameters.

Consider the program to swap two numbers using pass by reference method,

```
#include<stdio.h>
int swap(int *a,int *b)
{
        int temp=*a;
        *a= *b;
        *b=temp;
}

void main()
{
        int x,y;
        printf("\nEnter the numbers:");
        scanf("%d%d",&x,&y);
        printf("\nBefore swapping : x=%d\ty=%d\n",x,y);
        swap(&x,&y);
        printf("\nAfter swapping : x=%d\ty=%d",x,y);
}
```

OUTPUT

Enter the numbers:10 20

Before swapping: x=10 y=20

After swapping: x=20 y=10

Differences between pass by value and pass by reference

Pass by Value	Pass by Reference
In *call by value*, a copy of actual arguments is passed to formal arguments of the called function and any change made to the formal arguments in the called function have no effect on the values of actual arguments in the calling function.	In *call by reference*, the location (address) of actual arguments is passed to formal arguments of the called function. This means by accessing the addresses of actual arguments we can alter them within from the called function.
In call by value, actual arguments will remain safe, they cannot be modified accidentally.	In *call by reference*, alteration to actual arguments is possible within from called function; therefore the code must handle arguments carefully else you get unexpected results.

FILES

A file is a collection of related data that a computer treats as a single unit. Computers store files to secondary storage so that the contents of files remain intact when a computer shuts down. When a computer reads a file, it copies the file from the storage device to memory; when it writes to a file, it transfers data from memory to the storage device. When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates. It is easy to move the data from one computer to another without any changes. C uses a structure called FILE (defined in stdio.h) to store the attributes of a file. When working with files, you need to declare a pointer of type FILE. This declaration is needed for communication between the file and the program.

*FILE *fp; // *fp – file pointer variable*

Types of Files

There are two types of files

- Text files
- Binary files

1. Text files

Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad. When you open those files, you'll see all the contents within the file as plain text. It is easy to edit or delete the contents. They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

2. Binary files

Binary files are mostly the .bin files in the computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's). They can hold a higher amount of data, are not readable easily, and provides better security than text files.

FILE OPERATIONS

1) Opening a file

A file must be "opened" before it can be used. Opening a file is performed using the fopen() function defined in the stdio.h header file.

The syntax for opening a file in standard I/O is:

*FILE *fp*

fp = fopen("filename","mode");

- *fp is declared as a pointer to the data type FILE.(C has a special "data type" for handling files which is defined in the standard library 'stdio.h '.) It is called the file pointer and has the syntax FILE*.*
- *filename is a string - specifies the name of the file.*
- *fopen returns a pointer to the file which is used in all subsequent file operations.*

File Opening Mode

Sl. No	Mode	Description
1	**r**	Opens an existing text file for reading purpose.
2	**w**	Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
3	**a**	Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
4	**r+**	Opens a text file for both reading and writing.
5	**w+**	Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
6	**a+**	Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

File Location

We can provide the relative address of the file location or absolute address of the file. Consider your working directory is C:\CP\Test\ . Now you want to open a file hello.c in read mode. Two ways to provide the file location are as given below:

fp = fopen("hello.c","r");

OR

fp = fopen("C:\\CP\\Test\\hello.c","r")

2. Closing a file

The file (both text and binary) should be closed after reading/writing. Closing a file is performed using the fclose() function.

fclose(fp);

Here, fp is a file pointer associated with the file to be closed.

- fclose() function closes the file and returns **zero** on success, or **EOF** if there is an error in closing the file.
- This **EOF** is a constant defined in the header file **stdio.h**.

3. Reading and writing to a file

Sl. No	Function Name	Description	Syntax
1	fgetc	To read a character from a file	ch = fgetc(fp)
2	fputc	To write a character to a file	fputc(ch,fp)
3	fscanf	To read numbers, strings from a file	fscanf(fp,"%d",&n)
4	fprintf	To write numbers, strings to a file	fprintf(fp,"%d",n)
5	fread	To read binary content from a file. It is used to read structure content.	Refer Note
6	fwrite	To write as binary content to a file.	Refer Note

Note:

1) **The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.**
2) **To write into a binary file, you need to use the fwrite() function. The functions take four arguments:**
    - Address of data to be written in the disk
    - Size of data to be written in the disk
    - Number of such type of data
    - Pointer to the file where you want to write.

    fwrite(addressData, sizeData, numbersData, pointerToFile);

3) **Function fread() also take 4 arguments similar to the fwrite() function as above.**

    fread(addressData, sizeData, numbersData, pointerToFile);

feof()

The C library function **int feof(FILE *stream)** tests the end-of-file indicator for the given stream. This function returns a non-zero value when End-of-File indicator associated with the stream is set, else zero is returned.

RANDOM ACCESS TO A FILE

1) **rewind()**

The rewind() function sets the file pointer at the beginning of the stream. It is useful if you have to use stream many times.

Syntax: rewind(file pointer)

2) **fseek()**

If you have many records inside a file and need to access a record at a specific position, you need to loop through all the records before it , to get the record. This will waste a lot of memory and operation time. An easier way to get to the required data can be achieved using fseek().

*fseek(FILE * stream, long int offset, int pos);*

The first parameter stream is the pointer to the file. The second parameter is the position of the record to be found, and the third parameter specifies the location where the offset starts.

Different positions in fseek()

Position	**Meaning**
SEEK_SET	Starts the offset from the beginning of the file.
SEEK_END	Starts the offset from the end of the file.
SEEK_CUR	Starts the offset from the current location of the cursor in the file.

3) ftell()

ftell() in C is used to find out the position of file pointer in the file with respect to starting of the file.

Syntax of ftell() is:

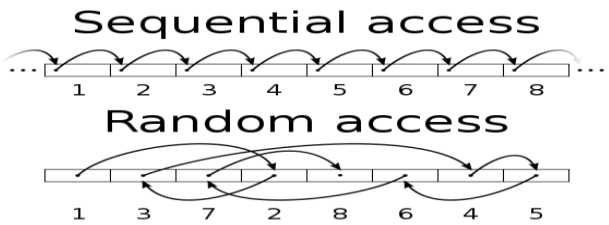
*ftell(FILE *pointer)*

Difference between sequential and random access

Sequential file access is the method employed in tape drives where the files are accessed in a sequential manner. So if you have to get a file in the end of the tape you have to start from the beginning till it reaches the beginning of the file.

Random access files are similar to the one in Hard Disks and Optical drives, wherever the files is placed it will go to that particular place and retrieve it.

Accessing data sequentially is much faster than accessing it randomly because of the way in which the disk hardware works.



1) Write a program to display the content of a file.

```
#include<stdio.h>
void main()
{
        FILE *fp;
        char ch;
        fp = fopen("test.txt","r");
        while(feof(fp) == 0)
        {
                ch=fgetc(fp);
                printf("%c",ch);
        }
       fclose(fp);
}
```

Content of test.txt
Hello, Welcome to C Programming Lectures.

OUTPUT
Hello, Welcome to C Programming Lectures.

2) Write a program to count number of vowels in a given file.

```
#include<stdio.h>
void main()
{
    FILE *fp;
    char ch;
    int countV=0;
    fp = fopen("test.txt","r");
    while(feof(fp) == 0)
    {
            ch=fgetc(fp);
            if(ch == 'a' || ch == 'A' || ch=='e' ||ch=='E' || ch == 'I' || ch == 'i' ||ch == 'O' ||
ch=='o' || ch == 'U' ||ch == 'u')
            {
                    countV++;
            }
    }
    printf("Count of Vowels=%d",countV);
       fclose(fp);
}
```

Content of test.txt
Hello, Welcome to C Programming Lectures.

OUTPUT
Count of Vowels=12

3) Write a program to copy the content of one file to another.

```
#include<stdio.h>
void main()
{
    FILE *f1,*f2;
    char ch;
    f1 = fopen("test.txt","r");
    f2 = fopen("copy.txt","w");
    while(feof(f1) == 0)
    {
        ch=fgetc(f1);
        fputc(ch,f2);
    }
    printf("Successfully Copied");
    fclose(f1);
    fclose(f2);
}
```

Content of test.txt
Hello, Welcome to C Programming Lectures.

OUTPUT
Successfully Copied

Content of copy.txt
Hello, Welcome to C Programming Lectures.

4) Write a program to merge the content of two files.

```
#include<stdio.h>
void main()
{
    FILE *f1,*f2,*f3;
    char ch;
    f1 = fopen("file1.txt","r");
    f2 = fopen("file2.txt","r");
    f3 = fopen("merge.txt","w");
    while(feof(f1) == 0)
    {
        ch=fgetc(f1);
        fputc(ch,f3);
    }
    while(feof(f2) == 0)
    {
```

```
            ch=fgetc(f2);
            fputc(ch,f3);
        }
        printf("Successfully Merged");
}
```

Content of file1.txt
Hello, Welcome to C Programming Lectures.

Content of file2.txt
C is very easy to learn.

OUTPUT
Successfully Merged

Content of merge.txt
Hello, Welcome to C Programming Lectures. C is very easy to learn.

5) Write a program to read numbers from a file and display the largest number.

```
#include<stdio.h>
void main()
{
        FILE *f1;
        int large,num;
        f1 = fopen("number.txt","r");
        fscanf(f1,"%d",&large); // setting first element as largest element
        while(feof(f1) == 0)
        {
            fscanf(f1,"%d",&num);
            if(large<num)
            {
                large= num;
            }
        }
        fclose(f1);
        printf("Largest element = %d",large);
}
```

Content of number.txt
15 21 7 29 36 78 67 56 10

OUTPUT
Largest element = 78

6) Consider you are a content writer in Wikipedia. You are the person who write the known facts about APJ Abdul Kalam. After his death, you need to change all is to was. Write a program to replace all is' to was' to a new file.

```
#include<stdio.h>
#include<string.h>
void main()
{
	FILE *f1,*f2;
	char str[30];
	f1 = fopen("apj.txt","r");
	f2 = fopen("new.txt","w");
	fscanf(f1,"%s",str);
	while(feof(f1) == 0)
	{
		if(strcmp(str,"is")==0)
			fprintf(f2,"was");
		else
			fprintf(f2,"%s ",str);
		fscanf(f1,"%s",str);
	}
	fclose(f1);
	fclose(f2);
	printf("Replaced String Successfully\n");
}
```


7) Write a program to reverse each content of file to another.

```
#include<stdio.h>
#include<string.h>
void main()
{
	FILE *f1,*f2;
	char str[30],rev[30];
	int i,j;
	f1 = fopen("test.txt","r");
	f2 = fopen("new.txt","w");
	while(feof(f1) == 0)
	{
		fscanf(f1,"%s",str);
		j=0;
		for(i=strlen(str)-1;i>=0;i--)
		{
			rev[j]=str[i];
			j++;
		}
```

```
                rev[j]='\0';
                fprintf(f2,"%s ",rev);
        }
        fclose(f1);
        fclose(f2);
}
```

Content of test.txt
Welcome to C programming

Content of new.txt after execution
emocleW ot C gnimmargorp

8) Write a program to count number of words and lines in a file.

```
#include<stdio.h>
#include<string.h>
void main()
{
        FILE *f1,*f2;
        int countW=0,countL=0;
        char ch;
        f1 = fopen("test.txt","r");
        while (feof(f1) == 0 )
        {
                ch = fgetc(f1);
                if(ch == ' ' || ch == '\t' || ch == '\n')
                        countW++;
                if(ch == '\n')
                        countL++;
        }
        printf("Count of words = %d\n",countW);
        printf("Count of Lines = %d",countL);
        fclose(f1);
}
```

Content of test.txt
Welcome to C programming.
C is very easy to learn

OUTPUT
Count of words = 4
Count of Lines = 2

9) Write a program to append some data to already existing file.

```
#include<stdio.h>
void main()
{
        FILE *f1;
        char str[30];
        f1 = fopen("test.txt","a");
        printf("Enter the string:");
        gets(str);
        fprintf(f1,"%s",str);
        fclose(f1);
}
```

10) Write a program to display content of a file two times without closing the file.

```
#include<stdio.h>
void main()
{
        FILE *fp;
        char ch;
        fp = fopen("test.txt","r");
        while(feof(fp)==0)
        {
                ch=fgetc(fp);
                printf("%c",ch);
        }
        rewind(fp);
        while(feof(fp)==0)
        {
                ch=fgetc(fp);
                printf("%c",ch);
        }
        fclose(fp);
}
```


OUTPUT

Hello, Welcome to C programming. Hello, Welcome to C programming.

11) Write a program to read the details of "n" Employees with following fields – name, empid and salary and write the details into a file. Then read details from file and display the name of employee who has highest salary.

```
#include<stdio.h>
struct Employee
{
        char name[30];
        int empid;
        double salary;
```

```
};
void main()
{
        FILE *fp;
        struct Employee e[10],res,temp;
        int i,n;
        fp = fopen("employee.dat","w");
        printf("Enter the limit:");
        scanf("%d",&n);
        printf("Enter the details of Employee\n");
        for(i=0;i<n;i++)
        {
                printf("Name:");
                scanf("%s",e[i].name);
                printf("EmpId:");
                scanf("%d",&e[i].empid);
                printf("Salary:");
                scanf("%lf",&e[i].salary);
                fwrite(&e[i],sizeof(e[i]),1,fp);
        }
        fclose(fp);
        fp = fopen("employee.dat","r");
        res.salary=-1.0;
        while(feof(fp) == 0)
        {
                fread(&temp,sizeof(temp),1,fp);
                if(res.salary < temp.salary)
                {
                        res = temp;
                }
        }
        printf("Name of Employee with highest Salary:%s",res.name);
}
```

OUTPUT

Enter the limit:2

Enter the details of Employee

Name:John

Salary:10000

Name:Kiran

EmpId:102

Salary:20000

Name of Employee with highest Salary:Kiran

PREVIOUS YEAR UNIVERSITY QUESTIONS

1. Write a function in C which takes the address of a single dimensional array (containing a finite sequence of numbers) and the number of numbers stored in the array as arguments and stores the numbers in the same array in reverse order. Use pointers to access the elements of the array. ***[KTU, MODEL 2020]***
2. With an example, explain the different modes of opening a file. ***[KTU, MODEL 2020]***
3. With a suitable example, explain the concept of pass by reference. ***[KTU, MODEL 2020]***
4. With a suitable example, explain how pointers can help in changing the content of a single dimensionally array passed as an argument to a function in C. ***[KTU, MODEL 2020]***
5. Differentiate between sequential files and random access files? ***[KTU, MODEL 2020]***
6. Using the prototypes explain the functionality provided by the following functions.
   - rewind()
   - fseek()
   - ftell()
   - fread()
   - fwrite() ***[KTU, MODEL 2020]***
7. What are the file I/O functions in C? Explain about the task performed by each function. ***[KTU, DECEMBER 2019]***
8. Explain with suitable examples, how & and * operators are used with pointer variable. ***[KTU, DECEMBER 2019]***
9. Write a C program to copy the content from one file to another file. ***[KTU, DECEMBER 2019]***
10. When fopen() is not able to open a file, what it returns? Explain in brief. ***[KTU, DECEMBER 2019]***
11. Compare the use of fread( ) and fwrite( ) functions with fscanf( ) and fprintf( ) functions. ***[KTU, DECEMBER 2019]***
12. How do pointers differ from variables in C? ***[KTU, DECEMBER 2019]***
13. Write a C program to swap the content of two variables using pointers. ***[KTU, DECEMBER 2019]***

14. How will you declare a pointer to a function? Write down the syntax and explain. ***[KTU, DECEMBER 2019]***
15. What is the significance of EOF? ***[KTU, DECEMBER 2019]***
16. With a suitable example explain how does a pointer point to another pointer? ***[KTU, MAY 2019]***
17. Discuss about unformatted data files and write on any two library functions associated with this. ***[KTU, MAY 2019]***
18. What is the output of the following program? Justify your answer.

```
#include<stdio.h>
void main()
{
    char *p = "wxyz";
    printf("%c", *p++);
    printf("%c", *p);
}
```
[KTU, MAY 2019]

19. List the advantages of using pointers in C. ***[KTU, MAY 2019]***
20. Using pointers, write a function that receives a character string and a character as argument and deletes all occurrences of this character in the string. The function should return the corrected string with no holes. Also write the main function to invoke the above function. ***[KTU, MAY 2019]***
21. Write a C program to write a set of numbers to a file and separate the odd and even numbers to two separate files. ***[KTU, MAY 2019]***
22. Explain the use of indirection operator with the help of an example? ***[KTU, MAY 2017], [KTU, JULY 2017]***
23. Explain the difference between *ptr++ and (*ptr)++ if ptr is pointing to the first element of an integer array. ***[KTU, MAY 2017]***
24. With an example, show how you can access the members of a structure variable using a pointer to the variable. ***[KTU, MAY 2017]***
25. What is NULL pointer? ***[KTU, MAY 2017]***
26. How do you declare a pointer variable? What is the significance of the data type included in the declaration? ***[KTU, JULY 2017]***
27. What are array of pointers? How do you declare an array of pointers? ***[KTU, JULY 2017]***
28. For the declaration int p=1, q=1, r[25]={1}, s[5][25]={{1}}; , check the validity of the given pointer usages and if valid provide the value of the statement. Support your answer with proper explanation.
    i) *p ii) &(p+q) iii) *(&p) iv) *2017 v) *s[0] ***[KTU, APRIL 2018]***
29. Explain any three file opening modes? ***[KTU, MAY 2017], [KTU, JULY 2018]***

30. How can you perform read and write operations of an unformatted data file? ***[KTU, MAY 2017]***
31. Assume that there are two files first.txt and second.txt. Write a C program to merge the contents of two file into a new file third.txt ***[KTU, MAY 2017]***
32. What do you mean by opening of a file? How is this accomplished? ***[KTU, JULY 2017], [KTU, DECEMBER 2019]***
33. Write a program to count the number of vowels, consonants, digits and special characters in a text file. ***[KTU, JULY 2017]***
34. Write a C program to copy the content of a given text file to a new file after replacing every lowercase letters with corresponding uppercase letters. ***[KTU, APRIL 2018]***
35. With suitable example explain any four different File I/O operations in C? ***[KTU, APRIL 2018]***
36. Discuss the concept of binary file in C. ***[KTU, DECEMBER 2018]***
37. What is the purpose of fopen() and fclose() functions in C ***[KTU, DECEMBER 2018]***
38. When a program is terminated, all the files used by it are automatically closed. Why is it then necessary to close a file during the execution of the program? ***[KTU, DECEMBER 2018]***
39. What is the purpose of getw() and putw() functions ***[KTU , DECEMBER 2018]***
40. Write a C program to create a file and store information about a person, in terms of his name, age and salary ***[KTU, DECEMBER 2018]***
41. Write any two file handling functions used to write data into text files***. [KTU, DECEMBER 2018]***
42. Write a C program to create a text file and display its contents. ***[KTU, JULY 2018]***
43. Briefly explain unformatted file in C. Give the format of fwrite functions in C. ***[KTU, JULY 2018]***
44. Write a C program to read set of numbers from input file 'value.dat' and store the sorted numbers in an output file 'sort.res' ***[KTU, JULY 2018]***
45. Explain any Six File opening modes available in C. ***[KTU, APRIL 2018]***
46. List any three advantages of pointers. What is pointer to a one-dimensional array? ***[KTU, MAY 2019]***