Module 5

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.

int n = 10;

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

Declaring a pointer

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

*int *a;//pointer to int*

*char *c;//pointer to char*

Accessing Data through Pointers

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 variables 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	Value	Asterisk Value *
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;*

Array Access using Pointers

Write a program to display the content of an array using pointer.

```
#include<stdio.h>

void main()
{
	int A[]={10,20,30,40,50};
	int i;

	int *p = A; //Variable p of type pointer is pointing to the address of an integer array A.

	printf("Array Content\n");
	for(i=0;i<5;i++)
	{
		printf("%d\t",*(p+i));
	}
}
```

Output

```
Array Content
10	20	30	40	50
```

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

```
#include<stdio.h>
#include<malloc.h>
void main()
{
	int i,n;
	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);
	}
	printf("Array Content\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",*(p+i));
	}
}
```

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

Array Content

10 15 20 25 30

3) 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

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 to 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("Enter the numbers:");
        scanf("%d%d",&x,&y);
        printf("Before swapping : x=%d\ty=%d\n",x,y);
        swap(&x,&y);
        printf("After 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

Difference 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.

File

A file represents a sequence of bytes, regardless of it being a text file or a binary file. 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. 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:

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");

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.

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, string 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 fprint() 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:

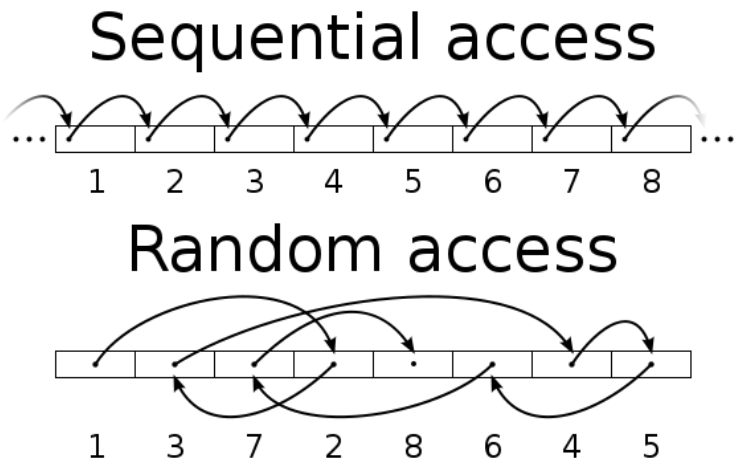
*ftell(FILE *pointer)*

Difference between sequential and random access.

Sequential file access is the method employed in tape drives where the files are access 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.



Examples Programs

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 numbers 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 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 A");
		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 copy the content of a file to another in reverse order.

```
#include<stdio.h>
void main()
{
        FILE *f1,*f2;
        int count,begin,end;
        char ch,i;
        f1 = fopen("test.txt","r");
        f2 = fopen("new.txt","w");

        // Code to find count of characters in a file.
        begin = ftell(f1);
        fseek(f1, -1, SEEK_END);
        end = ftell(f1);
        count = end - begin;  // Count of characters.
        printf("Count of characters=%d",count);

        // Copy the content of file in reverse order
        i=-1;
        while (count != -1)
        {
                ch = fgetc(f1);
                fputc(ch, f2);
                i--;
                fseek(f1, i, SEEK_END);     // shifts the pointer to the previous character
                count--;

        }
        fclose(f1);
        fclose(f2);
}
```

9) 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 == ' ')
			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

10) 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);
}
```

11) Write a program to display content of 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.

12) 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:Sangeeth

EmpId:101

Salary:10000

Name:Poornima

EmpId:102

Salary:20000

Name of Employee with highest Salary:Poornima