Teaching Notes

Programming In 'C' Language

1st Year 1st Sem Computer Science

S. Bhavani
Lecturer
Department of Computer Science
Govt. Degree College
URAVAKONDA

GOVERNMENT DEGREE COLLEGE, URAVAKONDA

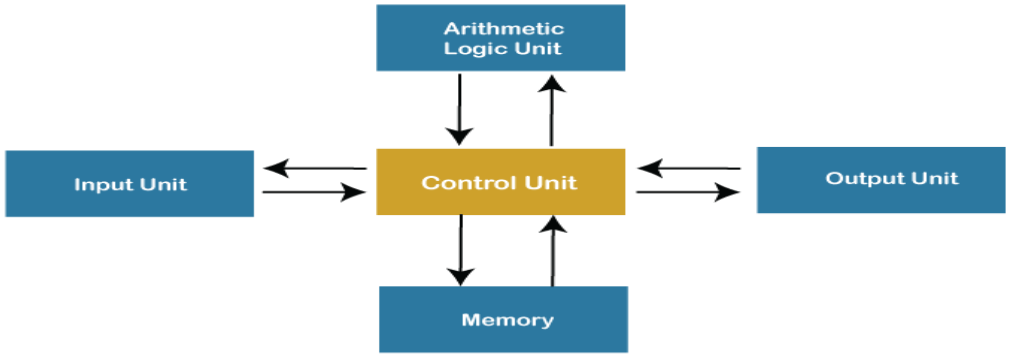
TEACHING NOTES

Month: November	**Year: 2021-22.**
Name of the Department / Subject:	Computer Science
Name of the Lecturer:	S. Bhavani
Course / Class:	I Year I Sem (MPCs & MSCs)
Paper:	I
Name of the Topic:	General Fundamentals of computers, Principles of Programming Languages.
Hours required:	14
Learning Objectives:	Students will learn Computer basics, Programming language basics.
Previous Knowledge to be reminded:	Introduction to Computers.

TOPIC SYNOPSIS: UNIT I

Introduction to Computers:

A computer is an electronic device, operating under the control of instructions stored in its own memory that can accept data (input), process the data, produce information (output), and store the information.



Characteristics of Computers:

Speed: Computers are a high-speed electronic machine. They can carry around 3-4 million instruction per second.

Accuracy: Computers are also known for their accurate performance. They can complete the given jobs at almost 100% accuracy.

Storage Capacity: Computers can easily store a massive size of data.

Reliability: Computers are reliable and consistent; they can process the same tasks any number of times without throwing any error.

Versatility: The variety of tasks that a computer can perform are almost infinite. That means computers can perform different tasks back-to-back without making errors;

Classification of Computers:

Supercomputer: Supercomputers are the fastest and the most expensive type of computer. They are large and require more space for installation. These types of computers are mainly designed to perform massive data-based and complex tasks.

Mainframe Computer: Mainframe computers are comparatively smaller in size as compared to supercomputers. These computers can handle heavy tasks, including complex calculations and can store vast amounts of data. They are best suited for big organizations such as banking, telecom, and educational sectors.

Microcomputer: Microcomputers are cheap in price and support multi-user platform. They are best suited for internet café, schools, universities, offices, etc. A microcomputer is also referred to as the 'Personal Computer (PC)' in general life. Laptop and desktop are examples of microcomputers.

Minicomputer: Minicomputers are also referred to as Mini frame computers. They are suitable for billing, accounting, education, and business purposes. Tablet PC, Notebooks, and cell phones are examples of minicomputers.

Workstation: Workstation is a powerful, single-user computer. A workstation is a personal computer with a faster microprocessor, a massive amount of RAM, higher-quality monitors, high graphic memory, etc. This is best suited for performing any specific type of task professionally.

Advantages of Using Computers

- Computers can perform given tasks at incredible speed.
- Computers can perform the same task multiple times with the same accuracy.
- Computers allow doing several tasks simultaneously as they are best suited for multitasking.
- Computers keep the stored data secure and inaccessible from unauthorized users.
- Computers can automatically perform routine tasks with automation, making humans available for more intelligent tasks.

Disadvantages of Using Computers:

1. Computers cannot work on their own. They need instructions from humans to complete tasks.
2. Computers need a power supply to work. Without a power supply, they are just useless.
3. Working on a computer continuously for a long period can cause several health issues.
4. Wastage of computers and their parts leave a negative impact on the environment.
5. Computers are taking human jobs in many sectors. They are replacing human work and thus increasing unemployment.

Generations of Computers:

There are five generations of the computer, which can be classified as below:

First Generation (1946 - 1959): During the first generation, computers were based on electronic valves (Vacuum Tubes). Some popular computers of first-generation are ENIAC, EDVAC, UNIVAC, etc.

Second Generation (1959 - 1965): During the second generation, computers were based on Transistors. Some popular computers of second-generation are IBM 1400, IBM 1620, IBM 7000 series, etc.

Third Generation (1965 - 1971): During the third generation, computers were based on Integrated Circuits (ICs). Some popular computers of the third generation are IBM 360, IBM 370, PDP, etc.

Fourth Generation (1971 - 1980): During the fourth generation, computers were based on very large scale integrated (VLSI) circuits. Some popular computers of fourth-generation are STAR 1000, CRAY-1, CRAY-X-MP, DEC 10, etc.

Fifth Generation (1980 - Present): The fifth generation is still ongoing. The computers are based on multiple technologies, such as ultra large scale integration (ULSI), artificial intelligence (AI), and parallel processing hardware. The fifth generation of computers includes Desktop, Laptop, NoteBook etc.,

Algorithm

An algorithm is a step-by-step procedure of solving the given problem statement. All algorithms must satisfy the following characteristics.

Input: Zero or more quantities are externally supplied.

Output: At least one quantity is produced.

Definiteness: Each instruction is in clear format.

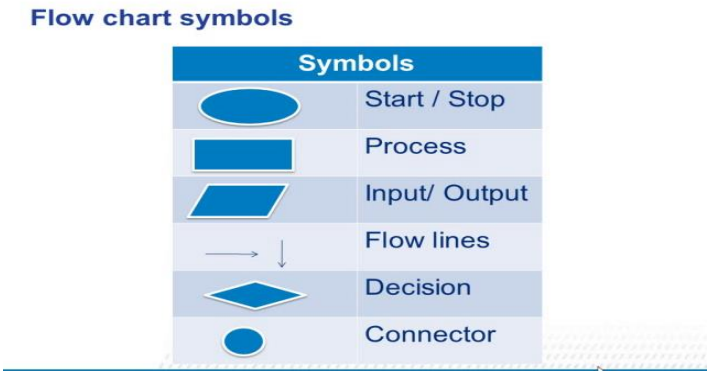
Finiteness: The algorithm must terminate after a finite sequence of steps.

Effectiveness: Every instruction must be in basic format.

Flowchart

Pictorial or Graphical representation of an algorithm is called a flowchart. Flowcharts are designed by using some specific symbols.

The most import symbols used for designing flowcharts are:



PROGRAMMING LANGUAGES:

A programming language is a language specifically designed to express computations that can be performed by the computer.

The concept of generations of programming languages is closely connected to the advances in technology that brought about computer generations.

- Machine language (1GL)

- Assembly language (2GL)
- High - level language (3GL)
- Very high - level Language (4 GL)
- Fifth generation Languages (5GL)

Structured programming is a top-down approach in which the overall program structure is broken down into separate modules.

It allows the code to be efficiently loaded into the memory and to be reused in other programs.

Structure oriented program is subset of procedural oriented programming approach.

In this approach reusability of subroutines are between the programs.

C is a structured oriented (or) procedure-oriented programming language.

Examples / Illustrations	
Additional inputs	Sample code
Teaching Aids used	Black Board & Chalk, Through PPTs OOPs Concepts
References cited	*http://www.tutorialspoint.com &* *https://www.cprogramming.com/*
Student Activity planned after the teaching	Questions and answers
Activity planned outside the Class room, if any	Assignments
Any other activity	

Signature of the Lecturer

GOVERNMENT DEGREE COLLEGE, URAVAKONDA

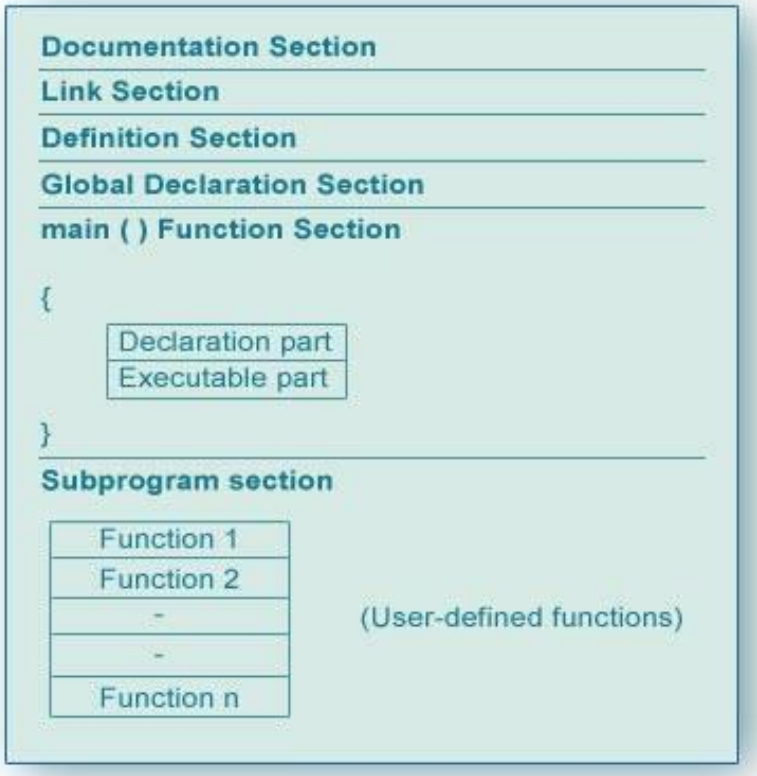
TEACHING NOTES

Month : December	**Year: 2021-22.**
Name of the Department / Subject:	Computer Science
Name of the Lecturer:	S, Bhavani
Course / Class:	I Year I Sem(MPCs & MSCs)
Paper:	I
Name of he Topic:	Introduction to C
Hours required:	14
Learning Objectives:	Student to learn basics of C language.
Previous Knowledge to be reminded:	Structured Programming basics.

TOPIC SYNOPSIS : UNIT II

Introduction to C:

C is a programming language which born at "AT & T's Bell Laboratories" of USA in 1972. It was written by Dennis Ritchie. This language was created for a specific purpose: to design the UNIX operating system. From the beginning, C was intended to be useful to allow busy programmers to get things done.



Structure of a C program:

```
/*Example program for illustrating input and output functions*/
#include<stdio.h>
void main()
{
        int a;
        printf("Enter a number : ");
        scanf("%d",&a);
        printf("Given number a=%d",a);
}
```

Steps to execute C program

1. Save the program with filename.c — Example: first.c
2. Compile the program to rectify the errors. — Press **Alt + f 9**
3. Execution (Run) the program — Press **Ctrl + f 9**
4. To see output after execution (run) — Press **Alt+ f5**

Using Comments, Keyword:

C tokens are the basic buildings blocks in C language which are constructed together to write a C program.

Each and every smallest individual units in a C program are known as C tokens. C tokens are of four types. They are,

Keywords Identifiers Constants Variables

Identifiers, Basic Data Types in C:

The nature of data is called Data type. A variable can store data to perform a specific operation. Data types are used to define a variable before to use in a program.

"C" supports **three** classes of data types

1. Primary data types.
2. Derived data types.
3. User defined data types.

Variable:C variable is an identifier, which is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable. The value of the variable may get change in the program.

Example: a, same, item235

Constant: Constant is an identifier, which is a named location to hold fixed values. These values do not change during the execution of a program.

Constants are mainly classified into two types.

Input function: scanf( ) is used to read characters, strings, as well as numeric variables from the standard input device i.e. keyboard.

Syntax : scanf (" format strings", &arguments list);

Some format strings are:

Data type	**Format String**
int	%d
float	%f
char	%c
double	%lf
long int	%ld

Output function: printf ( ) is used to display values, results and message on the screen. This can be used to output any combination of numerical values, single characters and strings.

Syntax : printf ("format strings, messages, white space characters", arguments) ;

Escape sequence characters are:

Character	**Meaning**
\a	Bell alert
\b	Back space
\n	New line
\t	One horizontal tab space
\v	One vertical tab space
\0	Null value

Operators: The symbols which are used to perform logical and mathematical operations in a C program are called C operators.

C language offers many types of operators. They are

1. Arithmetic operators
2. Assignment operators
3. Relational operators
4. Logical operators
5. Increment/decrement operators
6. Conditional operators (ternary operators)
7. Bit wise operators
8. Special operators

Control statements in C- programming language can be classified in to 2 types. They are

1. Conditional Branching Statements
2. Conditional Looping Statements (or) Iterative statements

1. Conditional Branching Statements:

Conditional Branching statements are of two types

I. Two way branching Statements (or) Decision making Statements.

II. Multiple branching Statements.

Two way branching Statements are

i. Simple if ii. if…. else iii. nested if iv. else…. if….ladder.

II. Multiple branching Statements (switch statement): In general if…else statements provides a two way branching of control. To branch two options out of many options or choices, a multiple branching statement is useful called switch statement.

Syntax :

```
switch (exp)
{
    case value1:    Statements;
                    break;
    case value2:    Statements;
                    break;
                    ----------
                    ----------
    case value-n:   Statements;
                    break;
    default:        default statements;
}
```

2. Conditional Looping statements

In control statements, Conditional looping statements allows a set of instructions to be repeatedly executed until condition is satisfied.

There are three types of looping statements.

i. while loop

ii. do…. while loop

iii. for loop

i. while loop

Syntax :

```
Initialization;
while (condition)
{
        body of the loop;
        increment/decrement;
}
```

ii. do…. while

Syntax:

```
        initialization;
do
{
        body of loop ;
        increment/decrement;
 }while(condition);
```

iii. for loop

This statement is different from other two loops. It has 3 expressions.

They are

- Initialization expression
- The condition expression.
- Increment or Decrement expression, which prepares for the next repeat.

Syntax:

```
for (initialization; condition; inc/dec)
{
        body of loop;
}
```

Functions:

A large C program is divided into basic building blocks called C function. C function contains set of instructions enclosed by "{ }" which performs specific operation in a C program.

Uses of functions

1. C functions are used to avoid rewriting same code over and over in a program.
2. There is no limit in calling C functions to make use of same functionality wherever required.
3. Programs with functions are compact.
4. We can call functions any number of times in a program.
5. From any place in a program we can call a function.
6. The length of the source program can be reduced.
7. It facilitates top-down modular programming.

8. It increases program readability and helps documentation.

Creating a Function: To create a function in a C program you must follow the following steps.

1. C function declaration 2.Function call and 3.Function definition

1. Function declaration or prototype: This informs compiler about the function name, function parameters and return value's data type.

Syntax : return_type function_name (argument list);

Example : void fun( );

2. Function call :A function can be called by simply using the function name in a statement.

Syntax : function_name (arguments list);

Example : fun( );

3. Function definition: Function definition contains all the statements to be executed.

Syntax : return_type function_name (arguments list)

```
{
    Function body;
}
```

Example :

```
void fun( )
{
    printf("User defined function");
}
```

Classification of functions: All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function. Based on these functions are classified into four types. They are

1. Function without arguments (parameters) and without return value
2. Function with arguments (parameters) and without return value
3. Function with arguments (parameters) and with return value
4. Function without arguments (parameters) and with return value

Examples / Illustrations	
Additional inputs	Sample code
Teaching Aids used	Black Board & Chalk, Through PPTs
References cited	*http://www.tutorialspoint.com & https://www.cprogramming.com/*
Student Activity planned after the teaching	Questions and answers
Activity planned outside the Class room, if any	Assignments
Any other activity	

Signature of the Lecturer

GOVERNMENT DEGREE COLLEGE, URAVAKONDA

TEACHING NOTES

Month: January	**Year: 2021-22.**
Name of the Department / Subject:	Computer Science
Name of the Lecturer:	S. Bhavani
Course / Class:	I Year I Sem(MPCs & MSCs)
Paper:	I
Name of the Topic:	Arrays
Hours required:	14
Learning Objectives:	Student to learn creating arrays in C.
Previous Knowledge to be reminded:	Basics of C Language.

UNIT III
ARRAYS

Arrays: An Array is a collection of values of the same data type. We can store group of data of same data type in an array. Always, Contiguous (adjacent) memory locations are used to store array elements in memory. It is a best practice to initialize an array to zero or null while declaring, if we don't assign any values to array.

Types of arrays: There are 2 types of C arrays. They are

1. One dimensional array 2. Multi dimensional array (Two dimensional array)

1. One dimensional array: A one dimensional array contains only one subscript(size of array) . The subscript is also called index. It starts from zero (0).

Syntax : data-type array_name[array_size];

Here Array size must be a constant value.

Example: int a[10];

In the above example variable 'a' can stores 10 integer values.

Array initialization

Syntax: **data_type** array_name [array_size]= {value1, value2, value3,….};

Example: int a[5]={1,2,3,4,5};

char str[10]={'A','T','P'}

Accessing array

Syntax: arr_name[index];

Example: int a [3]; a[0]=1; a[1]=2; a[2]=3;

Storing values in Arrays

Index :	0	1	2
a	**10**	**20**	**30**
Address:	2000	2002	2004

UNIT v: Pointers Pointer is a variable that stores or points the address of another variable. Pointer is used to allocate memory dynamically i.e., at run time. The variable might be any of the data type such as int, float, char, double, short etc.

Declaring a pointer :

Syntax : data_type *var_name;
Example : int *p;

char *p;

Here, * is used to denote that "p" is pointer variable and not a normal variable.

Program

```
/* Example program for pointer */
#include <stdio.h>
void main()
{       int *ptr, q;
       q = 50;
       ptr = &q;
       printf("%d", *ptr);
}
```

Test Data & Output
50

Types of pointers: There are two different types of pointers.

1) **NULL pointer** **2.Generic pointers**

NULL pointer:A NULL pointer has a fixed reserved value that is not zero or space, which indicates that no object is referred. NULL pointers are used in C as compile-time constant.

Example: void *pointer = NULL;

Generic pointers: When a variable is declared as being a pointer to type **void** it is known as a generic pointer. A void pointer can be really useful if the programmer is not sure about the data type of data inputted by the end user. In such a case the programmer can use a void pointer to point to the location of the unknown data type.

Example:- void *ptr; int a; ptr=&a;

Examples / Illustrations	
Additional inputs	Sample code
Teaching Aids used	Black Board & Chalk, Through PPTs OOPs Concepts
References cited	*http://www.tutorialspoint.com & https://www.cprogramming.com/*
Student Activity planned after the teaching	Questions and answers
Activity planned outside the Class room, if any	Assignments
Any other activity	

Signature of the Lecturer

GOVERNMENT DEGREE COLLEGE , URAVAKONDA

TEACHING NOTES

Month : February	**Year: 2021-22.**
Name of the Department / Subject:	Computer Science
Name of the Lecturer :	S. Bhavani
Course / Class :	I Year I Sem(MPCs & MSCs)
Paper :	I
Name of the Topic :	Functions, Structures, Unions, Enumerated DataTypes, Files
Hours required :	14
Learning Objectives :	Student to learn Functions, Structures, Unions.
Previous Knowledge to be reminded :	Basics of C Language.

Functions:

A large C program is divided into basic building blocks called C function. C function contains set of instructions enclosed by "{ }" which performs specific operation in a C program.

Uses of functions

1. C functions are used to avoid rewriting same code over and over in a program.
2. There is no limit in calling C functions to make use of same functionality wherever required.
3. Programs with functions are compact.
4. We can call functions any number of times in a program.
5. From any place in a program we can call a function.
6. The length of the source program can be reduced.
7. It facilitates top-down modular programming.
8. It increases program readability and helps documentation.

Creating a Function: To create a function in a C program you must follow the following steps.

2. C function declaration 2.Function call and 3.Function definition

1. Function declaration or prototype: This informs compiler about the function name, function parameters and return value's data type.

Syntax : return_type function_name (argument list);

Example : void fun( );

2. Function call :A function can be called by simply using the function name in a statement.

Syntax : function_name (arguments list);

Example : fun( );

3. Function definition: Function definition contains all the statements to be executed.

Syntax : return_type function_name (arguments list)

```
{
        Function body;
}
```

Example : void fun( )

```
{
        printf("User defined function");
}
```

Classification of functions: All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function. Based on these functions are classified into four types. They are

5. Function without arguments (parameters) and without return value
6. Function with arguments (parameters) and without return value
7. Function with arguments (parameters) and with return value

Structures: A Structure is a collection of different data types which are grouped together and each element in a structure is called structure member. If you want to access structure members in a program, structure variable should be declared. You can declare any number of structure variables for same structure and memory will be allocated for each separately. **For using structures you have to follow three steps. They are**

1. Defining a structure 2. Declaring structure variables
3. Accessing the members of a structure

1. Defining a structure: Before going to use a structure you have to define a structure.

Syntax:
```
struct tag_name
{       data type var_name1;
        data type var_name2;
        data type var_name3;
};
```

Here tag_name is an identifier.

Example :
```
struct employee
{       int  eno;
        char ename[20];
        float salary;
};
```

2. Declaring structure variables: After defining a structure you can declare variables to the structure. You can declare variables in two ways.

Syntax 1:
```
struct tag_name
{       data type var_name1;
        data type var_name2;
        data type var_name3;
}Structure_variable;
```

Example :
```
struct employee
{       int  eno;
        char ename[20];
        float salary;
}emp;
```

Syntax 2: `struct tag_name Structure_variable;`

Example : `struct employee emp;`

3. Accessing the members of a structure

After declaring variables to the structure you can access members of the structure through

variables by using dot(**.**) operator.

Syntax : Structure_variable.member;

Example: emp.eno; emp.ename; emp.salary;

Passing structure to function:A structure can be passed to any function from main function or from any sub function.

Unions: Unions are a concept borrowed from structures and therefore follow the same syntax as structures and unions arise in terms of storage.

For using unions you have to follow three steps. They are,

1. Defining a **union** **2.** Declaring **union** variables

3.Accessing the members of a **union**

1. Defining a union: Before going to use a union you have to define a union.

```
Syntax:   union tag_name
          {     data type var_name1;
                data type var_name2;
                data type var_name3;
          };
```

Here tag_name is an identifier.

```
Example : union employee
          {     int  eno;
                char ename[20];
                float salary;
          };
```

2. Declaring union variables: After defining a union you can declare variables to the union. You can declare variables in two ways.

```
Syntax 1: union tag_name
          {     data type var_name1;
                data type var_name2;
                data type var_name3;
          } union _variable;
```

```
Example : union employee
          {     int  eno;
                char ename[20];
                float salary;
          }emp;
```

Syntax 2: union tag_name union _variable;

Example : union employee emp;

3. Accessing the members of a union

After declaring variables to the union you can access members of the union through variables by using dot(**.**) operator.

Syntax : union _variable.member;

Example: emp.eno;

emp.ename;

emp.salary;

Enumerated Data Types

Enumerated Types allows you to create your own symbolic names for a list of related ideas. The key word for an enumerated type is **enum**. It is an user defined data type.

Syntax:

enum identifier {value1, value2,.... Value n};

Here "identifier" is name of user defined data type. Value1,Value2,Value3….. etc creates one set of enum values. Using "identifier" we are creating your own variables.

Example : **enum** day {Sunday,Monday,Tuesday,Wednesday};

enum day sday;

In the above example "day" is User defined data type . It has 4 values as given in the **pair of braces**. "sday" variable is declared of type "day" which can be initialized with 4 values. The default numeric value assigned to the first enum value is 0. To change the default value assign the value whatever you want using assignment operator.

Example : **enum** day {Sunday=1,Monday,Tuesday,Wednesday};

Unit – V File Input and Output operations

After completion of executing a program the entire data used as input and output is lost. If you want to keep large amount of data, it is time consuming to enter large amount of data. But, if you create a file, this information can be stored permanently on secondary storage devices.

C Language supports large numbers of functions to handle file I/O operations.

File Operations

1. Creating a new file 2.Opening a file 3. Writing information into a file
4. Reading information from a file 5.Closing a file

1. Creating a new file: To create a file, you need to declare a pointer of type FILE. This declaration is needed for communication between file and program.

Syntax: FILE *file_pointer; Here FILE is a Data type and file_pointer is a pointer variable.

Example: FILE *fp;

2. Opening a file: The **fopen( )** function is used to create a new file or to open an existing file, this function call will initialize an object of the data type **FILE**, which contains all the information necessary to control the stream.

Syntax: file_pointer=fopen("filename", "mode");

Here **filename** is a string literal, which you will use to name your file and **mode** is a file accessing mode. The accessing modes of the file are as follows:

Mode	Description
r	This mode opens an existing text file for reading purpose.
w	This mode opens a text file for writing, if it does not exist then a new file is created. Here your program will start writing data from the beginning of the file.
a	This mode opens a text file for writing in appending mode, if it does not exists then a new file is created. If exists program will start appending data into the existing file content.
r+	This mode opens a text file for reading and writing both.
w+	This mode opens a text file for reading and writing both. It first removes the file to zero length if it exists otherwise create the file if it does not exist.
a+	This mode opens a text file for reading and writing both. It creates the file if it does not exist and reading will start from the beginning. If exists it appends data into the existing file content.

Example : fp=fopen("One.txt","r");

3. Writing information into a file

You can write data into the file using different functions. They are putw, fprintf , fputs etc.

putw : This function is used to write integer type data into the file.

Syntax: putw(int_var, file_pointer); Example: putw(a,fp);

Fprintf: This function is used to write different types of data into the file.

Syntax: fprintf(file_pointer, "control string", var1,var2…);

Example: fprintf(fp,"%d%s",a,b);

Here fp is a file pointer that associated with a file opened for writing. Variable a is of type integer and b is of type character.

Fputs: This function is used to write string values into the file.

Syntax: fputs(" text", file_pointer); Example: fputs("Welcome", fp);

4. Reading information from a file: After writing data into the file you can read data from the file. To read data from the file C supports different functions. They are getw, fscanf, and fgets.

getw : This function is used to read integer type data from the file.

Syntax: getw(file_pointer); Example: getw(fp);

Fscanf: This function is used to read different types of data from the file.

Syntax: fscanf(file_pointer, "control string", &var1,&var2…);

Example: fscanf(fp,"%d%s",&a,&b);

Here fp is a file pointer that associated with a file opened for writing. Variable a is of type integer and b is of type character.

Fgets: This function is used to read string values from the file.

Syntax: fgets(char *var,int size, file_pointer);

Here var is a variable of type character. Size is length of the string reading from the file.

Example: fgets(var,255,fp);

5. Closing a file: After completion of the file reading or writing it must be close. To close the file C provides a function called fclose().

Syntax: fclose(file_pointer); Example: fclose(fp);

Detecting the End-of-file:

Ans) In c language EOF represent End Of file. It is used in file concept. Generally file is collection information. Information may be in any data type like built in data type, derived data type and user defined data types.

There are two ways represent the end-of file.

1. Reading a single character 2. Reading the big information from a file.

1. Reading a single character: In c language the user can be enter the data through the keyboard. Here we can read the character form the keyboard. It will be stored in a memory. Last character must and should be end with the enter key symbol. In this situation we use the terminology EOF.

By default EOF will value is -1.The following simple program explain as follows

2. Reading the information from a file: The other way is use the following predefined function called feof().

feof(): It is used to stream of data as an argument for the checking and after completion of execution it return s 0;

Syntax: int feof(FILE *fp);

Here feof is function which returns the non zero(1) value for true and zero for false. true means our sending argument is successfully received.

Ex: File *fp; Feof(fp);

Error Handling during File Operations

File: file is nothing but stream of data in programming language. It contains many number operations like

- Creating a file
- Remove a file
- Insert information into the file,
- Modify the information in file
- Delete the information from the file
- Closing the file.

In this situation a file may contain many number of errors at the time the above operations like.

When trying to read a file beyond indicator

When trying to read a file that does not exist
When trying to use a file that has not been opened.
When trying to use a file in appropriate mode i.e writing data to a file that has been opened for reading
When trying to a file that is write-protected.
In the above situations two types functions are supported.
clearerr(): The clearerr() function is used to clear the EOF and error indicators for the stream. The clearerr() function clears the error for the stream pointed by stream.
Syntax: void clearer(FILE *stream);
perror(): The perror() function stands for print error. In case of an error programmer can determine the type of error that has occurred using the p error.
Void perror(char *msg);
Perror takes one argument msg which points to optional user-defined message. This message is printed first followed by colon, and the implementation-defined message that describes the most recent error.
Generally some more functions also available like
Fseek() Fsetpos() Rewind() In the above situations also errors are may be raised.
Command Line arguments: Command line arguments are arguments giving input from command prompt at the time of executing a c program. It will take the values and store in argument variables.
Sysntax: void main(int argc, char * argv[])
Here argc integer variable represent the count the no of argument.
Argv is an array variable which is possible to store many number of values in the form of strings.

Steps to execute C program

1. Save the program with filename.c Example: add.c
2. Compile the program to rectify the errors. Press **Alt + f 9**
3. Execution (Run) the program Press **Ctrl + f 9**
4. Goto Filemenu ➔DOS Shell / OS Shell
5. C:\TC>add 10 20

Test data & Output :
The sum is : 30

Examples / Illustrations	
Additional inputs	Sample code
Teaching Aids used	Black Board & Chalk, Through PPTs OOPs Concepts
References cited	*http://www.tutorialspoint.com & https://www.cprogramming.com/*
Student Activity planned after the teaching	Questions and answers
Activity planned outside the Class room, if any	Assignments
Any other activity	

Signature of the Lecturer