JMS

INDEX

Name: Sudhanshu Ranjan Adarsh Std.: _____ Sec.: _____

Roll No.: 23-----80 Sub.: Prog. in C

DCE

S. No.	Date	Title	Page No	Teacher's Sign / Remarks
1.	2024	Algorithm and Program Development	1-9	
2.	2024	Program Structure	10-23	
3.	2024	Control structures	24-30	
4.	2024	Pointers	31	
5	2024	Functions	32-40	
6.	2024	Array and string	41-49	
7.	2024	structure and Union	50-55	
8	2024	file Handling	56-69	

Set of instructions - Program

JMS

Date:
Page No. 1

Unit-I: Algorithm and Program Development

The sequence of steps required to solve a problem, is called program. It is a list of instructions where each statement tells the computer to do a specific task.

⊙ Steps in Development of Program

The various steps in developing a program as follows:-

i> Development of Program:- The first step in developing a program for particular problem is to study the problem. The next task is to prepare a detailed list of step to obtain the required output.

ii> Compiling the Program:- By using C compiler the program is translate into machine language and this process of translation called compilation.

iii> Linking the Program:- After the compilation stage, the machine code version of C program is taken but it is not an executable file. To produce an executable file, these object code are to linked together with library function.

iv> Testing the Program:- Testing is done to make the program correct and performs task for all possible input. In this phase, program is executed with all input data for which result are known.

v> Document the Program:- The term 'documentation' refers to recording the important information of program. It enables everyone connected with program to understand the purpose of program.

vi> Deploying and maintaining the program:- The final phase in program development is deployment and maintenance. In this, program is installed at user's site. In program maintainence, the programming team fixes program error.

Q.1. Ans

Q.2. Ans

Algorithm

An algorithm is step-by-step instruction set for solving a problem.

Solving
Correctness
Definiteness
Effectiveness

Advantages of Algorithm:

Following are the advantages of algorithm:-

- i> Efficiency:- It can streamline processes, resulting faster and optmize solution.
- ii> Problem Solving:- It provide systematic approach to tackle complex problem.
- iii> Easy to Understand:- With step by step representation of program makes it easier to understand.
- iv> Easy to Debug:- Each step has its logical sequence makes it easier to debug.
- v> Precision:- The steps are precisely stated.

Q.1. Write an algorithm to find sum of two number.

Ans

step 1- Start
step 2- Input the given number A & B
step 3- sum = A + B
step 4- Print the sum and store
step 5- End

Q.2. Write an algorithm to find even number between 0 to 100.

Ans

step 1- start
step 2- I = 0
step 3- Write I
step 4- I = I + 2
step 5- If (I <= 98), then go to step 3
step 6- End

Flowcharts ⊙

A graphical method of representing the logical flow of the program, is called flowchart. It is a pictorial representation of algorithm. It uses different shapes to denote different types of instructions.

While generating a solution for a problem, an algorithm is created first and then it is represented through flow chart. From the flowchart, actual program is created by using programming language.

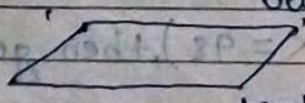
Flowchart Symbols

A flowchart uses different symbols to represent different functions in the program as follows :-

i> Terminal :- It is used to represent begining ending and pause in program's logical flow. The first and last is generally terminal symbol.

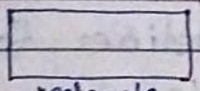
Oval

ii> Input/Output :- It is used when input is taken from the user and output is displayed to user.



parallelogram

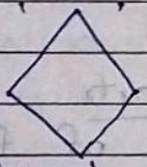
iii> Processing':- The processing symbol is used in a flowchart to represent arithmatic and data movement ~~structure~~ instruction.



rectangle

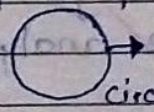
iv> Flow Lines':- It is used to indicate the flow of execution of a program. Flowlines indicate the exact sequence in which ~~are~~ the instructions are to be executed.

v> Decision :- A decision symbol is used in a flowchart, at a point where a number of options are available to user and user has to select one way depending upon input supplied by user. The condition is written in decision box.



Diamond

vi> Connector':- It is used to extend very long in flowchart on next page. When the flowchart becomes very long, we use connector symbol to extend the flowchart to next page.



circle

vii> Delay :- It indicates a delay in the process.

Guidelines for drawing Flowcharts

Following are the guidelines for drawing flowcharts:-

- i> To draw a proper flowchart, all the requirements should be listed out in a logical order.
- ii> The flowcharts should be clear, neat and easy to flow.
- iii> The usual direction of flow in chart is from left to right and top to bottom.
- iv> Ensure that flowchart has logical start and finish.

Q.1 Ans

Advantages of flowcharts

The advantages of flowcharts are as follows:-

- i> Communication:- It is easier for a programmer to explain the logic of program to other through flowchart
- ii> Effective Analysis:- Through flowchart, problem can be analysed in effective way.
- iii> Proper Documentation:- Flowchart serves a good program documentation.
- iv> Efficient Coding:- It acts as a guide during program development.
- v> Proper Debugging:- It helps in debugging process.

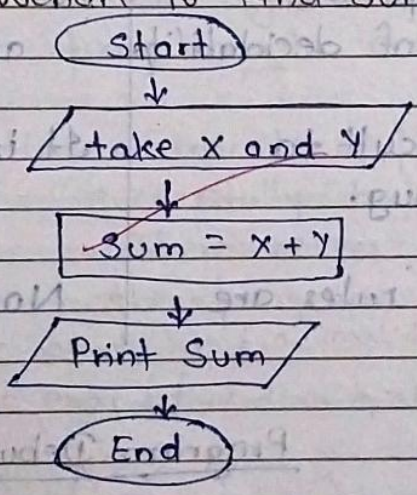
Q.2 Ans

Disadvantages of Flowcharts

- i> Complex Logic :- The program logic is quite complicated so, flowchart becomes complex and clumsy.
- ii> Alternation :- If alternations are required, flowchart may re-draw completly
- iii> Reproduction :- As flowchart symbols can't be typed, reproduction of program becomes a problem.
- iv> Time Consuming :- It takes more times to create and update the algorithm

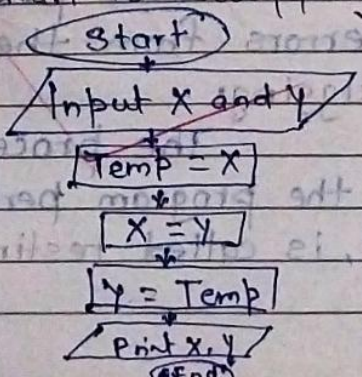
Q.1 Draw a flowchart to find sum of two given numbers.

Ans:



Q.2. Draw a flowchart for swapping the contents of two variables

Ans:



Difference between Algorithm and Flowcharts

	Algorithm	Flowchart
i>	It is a step-by-step procedure to solve a problem.	It is a pictorial representation of logic of problem.
ii>	Text-based readability	Graphical based readability
iii>	It is comparatively difficult to create.	It is very easy to create and understand.
iii>	It doesn't include geometrical pattern.	It uses various kinds of geometrical diagram.
iv>	Used to represent concept of decidability.	It is used in designing and documenting.
v>	It is difficult to debug.	It is easy to debug.
vi>	Predefined rules are used.	No rules are used.

Program Debugging

The process of removing the errors from the program, is called debugging.

The process of making sure that the program performs the intended task, is called testing (or running).

Types of Errors

Different types of errors can occur while creating and executing the program as follows:-

(not compile)
```
int main(){
int i=10, j=5, sum;
printf("i+j=%d", sum)
return 0;
}
```
semicolon

i> Syntax Errors:- It reffers to the errors of syntax or rules in a programming language like - incorrect punctuation, undefined terms etc.

```
int main(){
int i=10, j=5;
printf("sum=%d", i+j);
return 0;
}
```

ii> Logical Errors:- It refers to an error in planning the program's logic. Such logical errors cause the program not to crash during execution.

```
int main(){
int i=10, j=0;
printf("%d", i/j);
return 0;
}
```

iii> Run-time Errors:- A program may compile successfully but not work properly. These refers to an error when the program is running.
Ex- divisible by 0, invalid memory access or file handling error

iv) Semantic error (float divide by int)

(not compile)

Aug 09 2024

For C Prg-
- Save - F2
- Compile - Alt + F9
- Col Run - Ctrl + F9

JMS

Date:
Page No.: 10

Unit-I2 Program Structure..

C language is computer-based procedural programming language. It was developed in 1972 by 'Ken Thompson' and 'Denis M Ritchie' at Bell Laboratory in USA. Its old name was BCPL and B.

Basic Combined Prog. Lang. Martin Richard in 1966

Some other features of C:-
i> It is a middle-level language but it has feature of both low as well as high level language.
ii> Unix OS is written in C language
iii> The extension of C language is .C
iv> It is a case sensetive language (mean 'A' and 'a' represents different symbol ~~language~~)

The increasing popularity of C is due to its desirable quantity:-
i> It is portable, means program written for one computer can run on other system with no modification.
ii> It is easy to use and learn. ~~and~~
iii> It is very compact language.
iv> It is compiler-based programming langugage.
v> Graphic-based programming is possible in C.

Some rules regarding the C - Program

Following are the some rules which may followed when writting a program in C as follows:-

i> 'Header files' are included in the program before 'main' function.

ii> Only one 'main()' function can be used in one C-Program.

iii> Semicolon(;) can't be wed before 'main()' function.

iv> Each statement ends with semicolon(;) but it is not allowed with while, for loop, statements etc.

v> Two types of comments is used in documentation section,

i> // single line Comment

ii> /* Double line Comment */

vi> The header files always begins with hash(#) symbol.

i> #include <stdio.h>

ii> #include <conio.h>

iii> #include <math.h> etc.

Structure of a C- Program

The basic structure of a C-Program is divided into six parts which makes it easy to read, modify and understand in a particular format.

Six sections of C-Program are mentioned:-

i) Documentation
ii) Preprocessor
iii) Definition
iv) Global declaration
v) Main() function
vi) Sub Program

i) Documentation:- It consists of description, name and creation date and time of program. It specified at the start of program in the form of comment.

Like- //-------

/*-------

-------*/

ii) Preprocessor:- All the header files of the program will be declared in this section. Header files help us to access other's improved code.

Like:- #include<stdio.h>

#include<math.h>

iii> Definition:- Preprocessor are the programs that process our source code before the process of compilation. Preprocessor directives starts with # (hash) symbol.
The '#define' preprocessor used to create a constant in the program.

iv> Global declaration:- It contains variables, function declaration etc.
like - int sum = 18;

v> Main() function:- Operation like declaration and execution are performed inside curly brace ({}) of main program.
void main() tells the compiler that program will not return any value. The int main() tells the compiler to return an integer value.
Ex- void main()
int main()

vi> Sub Program:- User-defined function is called in this section. The control of program is shifted to called function whenever they are called from main() function.
Ex- int sum (int x, int y)
{
return x+y;
}

Structure of C program

```
// add two numbers                          ← Comment (Single line)
#include <stdio.h>                          ← Header file
                                            ← Preprocessor
int main()                                  ← Main function
{                                           ← Start of Pr. Exe.
    int a, b;                               ← Datatype; To define two integer value
    printf("Enter number a:");              ← Print as it is
    scanf("%d", &a);                        ← Function is used to take value for user; Address of operator
    printf("Enter number b:");
    scanf("%d", &b);                        ← Format Specifier of 'int'
    printf("The sum is:%d");                ← To print sum
    return 0;                               ← Return '0' to OS (Successful Program)
}                                           ← End of program Execution
```

```
#include <stdio.h>                    ← Link Section
# define X 20                         ← Definition
 int sum (int y);                     ← Global Declaration
  int main() {
    int y = 55;
    printf("sum: %d", sum(y));        } Main() function
    return 0;
  }
    int sum(int y)
  {                                   } Subprogram
    return y + X;
  }
```

Program: /*..... Program 2.1 ... */

```
#include <stdio.h>
main()
{
printf ("\n \t Hello Chap");
}
```

Explanation: In the above C program, it consist of six lines. The first line is a comment and is non-executable. Second line is a preprocessor directives which enables to add header file stdio.h by compiler. Third line is executable and the function main starts here. The pair of brace at fourth and sixth line contain output function printf of standard input/output header file(stdio.h) with agrument value \n \t Hello chap. \n is used to control new line. \t transfers it to tabposition & Hello Chap is printed. It completes the program printig Hello Chap on screen.

A process in which breaking a source code file into smaller component (token) is called Tokenization

JMS

Tokens in C

A token in C can be defined as the smallest individual element that is meaningful to compiler. It is a collection of strings.

Some of types of tokens in C are:-

i> Identifiers
ii> Keywords
iii> Constants
iv> Operators
v> Punctuators

i> Identifiers:- It is used to name variables and functions. It must began with an alphabetic character. They are formed by using alphabet, digits and underscore (_) character.

ii> Keywords:- They are reserved words that having pre-defined meaning in a language. It can't use for personal use. C language supports 32 keywords like int, void, else etc.

iii> Constants:- It refers to those variable whose value do not change during the program execution.

Types:
i> Integer Constant:- It refers to sequence of digits.
ii> Character Constant:- It is represented within single quote.
iii> Floating Constant:- It consists of integer part, a decimal point or a fraction part.
iv> String Constant:- It is a sequence of alpha-numeric character. It is in double quote.

iv) Operators:- It operates on some data to give the result.

v) Punctuator:- They are used to separate tokens in a program.

Datatypes in C ⊙

It specify how the data is enter in the program and what type of data enter in the program. It is used to declare a variable.

Three types of datatypes in C:-
- i) Premitive datatype
- ii) Derieved datatype
- iii) User-defind datatype.

i) Premitive Datatype:- These are the most basic datatype that are used for representing simple value like integer, float etc.

Types:-

i) Integer:- It is used to store an integer value. It require 2-bytes of memory space.

Format Specifier- %d

Syntax:

```
int a, b;
```

ii> Character :- It allows its variable to store only a single character. It require 1-byte of memory space.
Format Specifier - %c

double - 8 byte

Syntax:
char a, b;

iii> Float :- It is used to represent decimal-point value. It require 4-byte of memory space.
Format Specifier - %f

Syntax:
float a, b;

iv> Void :- It is used to specify that no value is present.

ii> Derived Datatype :- The datatype that are derived from premitive datatype, is called derived datatype.

Types:
i> Array :- It is a collection of same type of data element. It is used to represent a group of related data items.
A[0], A[1], ... A[n-1]

ii> Function :- It is a self contain block of statements that perform some task.

iii> Pointers:- It is a variable that contains the address of another variable in memory. If first variable, contains the address of second then first point to second.

iii> User defined datatype:- It is defined by the user.

Types: i> Class: It is a group of object with same attribute and common behaviour.

ii> Structure: It is a method of grouping data of different datatype.

Operators in C ⊙

It is the symbol that help us to perform some mathematical and logical computation on operands.

Ex- 1 + 2 = 3 ← operator

There are mainly three types of Operators in C:-

i> Unary Operator
ii> Binary Operator
iii> Ternary Operator

i> Unary Operator:- It works on single operands.
ii> Binary Operator:- It works on double operands.
iii> Ternary Operator:- It works on three operands.

Include Program in all operator
'pow' → power

JMS

C language provides a wide range of operators :-

i> Arithematic Operator :- It is used to perform arithematical or mathematical operation on operands.
Ex- Addition (+), Subtraction (-), Multiply (*) etc

ii> Relational Operator :- It is used for comparison of two operands.
Ex- Less than (<), greater than (>),
Lessthan or equal to (<=),
Greater than or equal to (>=),
Equal to (==), Not equal to (!=)

iii> Logical Operator :- The result of operation of a logical operator is either true (1) or false (0).
Ex- AND (&&) - It returns true, if both operands are true.
ii> OR (||) :- It returns true, if both or any of operand is true.
iii> NOT (!) :- It gives result true, if operand is false and false, if operand is true.

iv> Bitwise Operator :- It is used to perform bit-level operation on operands.
like Bitwise AND (&) Bitwise OR (|)
Bitwise XOR (^) Bitwise left shift (<<)
Bitwise right shift (>>)

v> Assignment Operator:- It is used to assign some value to a variable.

Ex- x = 40 (Value '40' is assign to a variable 'x')

Like- Assignment (=), Addition Assignment (+=)
Subt. Assign. (-=), Multi. Assignment (*=)
Division Assign. (/=) Modulus Assign. (%=)

These are shorthand op.
Ex a=a+b
∴ a+=b

vi> Increment/Decrement Operator:- These are called unirary operator because they operates on single variable only.

Increment (++), Decrement (--)

Types:

i> Prefix Increment:- In prefix increment, first the value is increment, then operation is performed.

Ex- z = ++a

ii> Postfix Increment:- In this, first operation is performed, then value is increment.

Ex- z = a++

vii> Conditional Operator:- It is only a ternary operator.

Syntax: Operand1 ? operand2 : Operand3;
(Condition) ? (Do if TRUE) : (Do if false);

viii> Comma Operator:- It is a binary operator that evaluate its first operand and discards result, then it evaluates second operand and return this value.

Expression in C

It is a combination of operators, constants and variables.

Ex-

result = a + b * c

Variable to store the expression value.

Operand

operator

There are mainly three types of expression:-

i> Constant Exp.
ii> Integral Exp.
iii> Float Exp.

i> Constant Expression:- It consist only constant values. A constant value is one that doesn't change

ii> Integral Expression:- Those expression which produce integer results.

iii> Float Expression:- Those expression which produce results in floating-point values after execution. like - 1.0, 10.000

Datatype Casting

It is a process of converting one data type to another data type by programmer using casting operator.

Ex- int a = 10;
float b = 10.000;

There are two major types to perform casting :-

i> Implicit
ii> Explicit

i> Implicit type casting :- It performs automatic by compiler with program interference.

ii> Explicit type casting :- It allows to create the variable and to change its type by using type of operator.

Header file

It contains a set of predefined library function. The '.h' is the extension of header file and we request to use a header file by including it with C preprocessing directive '#include'.

The '# include' preprocessor directs the compiler that header file needs to be processed before compilation and include all datatypes and function definition

There are two types of header files in C :-

i> Standard Header file
ii> Non-standard Header file

i> Standard Header File :- It contains libraries defined in ISO standard of C programing language. They are stored in default directory of compiler and there are 31 standard header file in latest C language are- <assert.h>, <math.h>, <float.h>, <stdio.h> etc.

ii> Non-standard Header file :- These are difined by programmer itself for purposes like containing custom library functions. There are lots of non-standard libraries for C language are <conio.h>, <gtk/gtk.h>

09/09/2024

Unit-I Control Structures in C

It is used to alter the flow of control in a program from normal sequencer execution.

A compound statement is a sequence of statement that is treated as a single statement

Syntax:

```
{
    int temp = x;
    x = y;
    y = temp;
}
```

There are three types of control structure:- i> Sequential Statement
ii> Conditional State.
iii> Looping State.

Conditional Statement

When the programmers are required to execute a particular set of statement depending upon a particular test condition, then it is used.

Various types of statement:-
i> if st., ii> if...else st.,
iii> nested if st; iv> switch st.
v> goto st., vi> Continue vii> break st

Forms of if
- if, if-else
- if-else-if-else
- nested if

i) if statement :- It is wed when applied condition is true, otherwise skip.

Syntax:

```
if (condition)
{// it executes a block of
 code if condition is true.
}
```

Example:-

```
# include <stdio.h>
int main()
{
int x, y
clrscr();
printf("Enter any two number \n");
scanf("%d, %d", &x, &y);
printf("\n The given two numbers are:
 x = %d, y = %d \n", x, y);
if (x>y)
printf("\n x = %d is largest", x);
}
```

Explain code

ii) if..... else statement :- It executes a block of code if the specified condition is true and false.

Syntax:

```
if (condition)
{
//The true-block statement
}
else{
//false- block statement
}
```

Example:

```
#include <stdio.h>
int main()
{
int number, rem;
clrscr();
printf("Enter an integer number:");
scanf("%d", &number)
printf("\n Number is %d\n", number);
rem = number % 2;
if(rem == 0);
printf("Number is even %d\n", number);
else
printf("Number is odd %d\n", number);
}
```

Explain code

iii) nested if.... else statement :- A conditional statement can be used within another conditional statement.

Syntax:

```
if (condition1){
// code to execute if condition1 is true
    if (condition2){
// code to execute if both con1 & con2 are true
    }else{
// code to execute if con1 is true & con2 is false
    }
}else{
// code to execute if condition1 is false
}
```

Explain code

iv) Switch Statement:- The switch statement provides an alternative to else if construct. It causes a particular group of statement to chose. The selection is based on current value of expression is included within switch.

Syntax:

```
switch (expression)
{
case value 1: statement-1;
              break;
case value 2: statement-2;
              break;
:
case value-n: statement-n;
              break;
default: default-statemen;
}
```

Explain code

v) Goto statement:- The goto statement is used to alter the normal ~~space~~ sequence of program execution by transfering control to other part of program.

Syntax:

```
goto label;
```

here, label is an identifier used to label the target to which controll will be transferred.

vi) Break Statement:- It is used to terminate control from loop statement. It can be used for loop & switch statement.

Synt: break;

Array & its types of switch → 2 1st sem

JMS

Date:
Page No.: 28

Looping Statement ⊙

It allows a set of statements or instructions to be performed repeatedly untill a condition is fullfilled.

C provides three types of loop:-

i) for loop
ii) while loop
iii) do.... while loop

i) For Loop:- It causes a block of statement to execute for fixed number of times.

Syntax:

```
for (initialize; test; update)
{
  //
  // body for loop
  //
}
```

Example:

```
#include <stdio.h>
int main()
{
  int i;
  clrscr();
  for (i=0; i<=5, i++)
  {
    printf("Hello");
  }
  return 0;
}
```

Explain code

ii> While Loop:- It is used when a set of statements are to be executed repetedly. A set of statement written in while will be executed untill condition is true. It checks condition at begning of program.

Entry Controlled loop

Syntax:

```
initialization_expression;   ← Declaration
while (test_expression)
{
  // body of while loop
}
```

Variable declare before condition

Example:

```
#include <stdio.h>
int main()
{
  int a;
  clrscr();
  a=1
  while (a<=10)
 {
  printf("\n%d", a); }
 return 0;
}
```

Explain code

iii> Do.... while loop:- In this, condition is checked after execution of loop. The loop body will execute at least once. It checks condition as end of program.

exit controlled loop

Syntax:

```
initialization_expression;
do
{
  // body of do-while loop
} while (test_expression);
```

Example:

```
# include <stdio.h>
 main ()
{
  int a;
      a = 1;
 do
{
   printf (" %d", a);
} while (a <= 10);
return 0;
}
```

Explain code

Q.1. Write a program to print area of circle.

Prog.

```
# include <stdio.h>
 main ()
{
 float pi, r, area;
 clrscr ();
  pi = 3.14
   r = 5
 area = pi * r * r;
  printf (" the area of circle = %f", area);
  return 0;
}
```

It is declared using refrence operator and append to refrence's type - reference

JMS

Page No. 01

Unit-II 4. Pointers

Pointer is used to return multiple values

Pointer variable not multiplied by constant

A variable that holds the address of data item, like variable or array, is called pointer. It is denoted by (*) before variable name.

'*' ← value at address ope
'&' address of operator

Advantage:
i) function cannot return more than one value.
ii) In the case of array, we can decide the size of array at run time by allocating the necessary space.

variable से पहले (*) होगा

Disadvantages:
i) If sufficient memory is not available the program may crash.
ii) If the programer is not careful and conitent with pointer, then program crash.

Refering → address of point
Derefing → value of point

Initialization of pointers:
We can assign the address of variable using address operator (&).

11/9/2024

Defrencing: To access or modify the value at address store in pointer, we use defrence operator (value access करने के लिए, जो pointer point करता है)

We of pointer
- Dynamic memory allocation
- Array & string manipulation
- Function call back
- Build complex data structure

Null Pointer: A pointer can be assign null value to indicates that it doesn't point to any value location.

Syntax!

```
int age = 15;
int *ptr = &age;
int _age = *ptr;
```

value at pointer ← *ptr

address of operator ← &age

Format Specifier = %p ← (Pointer address)

(This return hexadecimal value)

%u ← return unsigned int

```
int main() {
  int x = 5;
  int *ptr = &x;
  printf("Value: %d", x);          (5)
  printf("Address of x: %d", &x);  (0x7ffee)
  printf("value using ptr: %d", *ptr);  (5)
  printf("Address in ptr: %d", ptr);    (0x7fa)
```

'&ptr' = pointer of address

→ Ptr++, Ptr-- depends on size of datatypes in memory

→ Only addition(+) or subtr(-) can be performed on pointer on integer and on other datatypes
Incr++, --decr

1) Value of pointer
2) Address of pointer
3) Incr/Decr value
4) Incr/Decr address

→ int *ptr

(*) asterisk tells compiler that variable 'ptr' is a pointer.
and 'int' datatype tells compiler, int datatype pointer is point to.

→ pointer to pointer (store address of pointer)
(int **ptr;)

Feat.
- They save memory space
- Fast data processing
- structure & function handled easily
- More Efficient
- It support run-time memory allo.

Unit-II 5. Function

A complex problem may be decomposed into smaller parts, is called function. It is a self contain block of statement that perform a specific task.

Adv:- i) Easy to write
ii) Easy to read and debug
iii) Easy to maintain and modify
iv) It can be called any number of times.

Function Declaration

A function definition has a name, parenthesis with more parameter and body.

The general format of function defini:-

Header → datatype functionname (datatype arg1, ...)
{
body of function
;
return (expression);
}

Ex:-
```
int heading(void)
{
  // statements
  return 0;
}
```

Return Values

A return statement is used to terminate a function to its caller. It is used to return/terminate function without returning a value.

Return only 1 value

Syntax:

```
return;
return (expression);
```

here, expression may be a constant, variable.

Types of function

There are two types of function in C :-

i> Built-in function
ii> User-defined function

i> Built-in function :- These are predefined inbuilt function

pow(num, 2) ∴ num²

ii> Us

```
//Program for user-define function
#include <stdio.h>
int add(int a, int b,)
{
  return a+b;
}
int main(){
 int x=5, y=10;
  int sum = add(x,y);
  printf("The sum is: %d \n", sum);
 return 0;
}
```

...me ..., it #include ...), strcpy(...

...programm...

Types of User Defined Ft.

Parameter and Parameter Passing

A function is divided into four way depending whether parameter are passed or not and value return or not.

i) Function with no parameter and no value
ii) Function with parameter and no value
iii) function with no parameter & return value
iv) Function with parameter & return value

i) **Function with no parameter and no value:-** (Void function())

A function which invoke without passing any argument and does not return value, is called function.....

ii) **Function with parameter and no value:-**

```
void sum(int a, int b);
```

A function which passes some argument but does not return value, is called.

iii) **Function with no parameter but return value:-**

```
int sum();
```

A function which invoke without any argument and return value, is called.

iv) **Function with parameter and return value:-**

```
int sum(int a, int b);
```

A function which pass some formal argument to function and compute some value.

Parameter - Placeholder

Actual and Formal Argument (Input)

Function Call Arg.
value of Argu.

The values that are declared within a function when the function is called, is called argument

There are two types of argument:-
i> Actual Arg.
ii> Formal Arg.

i> Actual Argument:- The value that is passed to function when it is called, is called actual argument. It must be constant, single variable.

Syntax:
```
{
int x,y;
void add(int x, int y);
// function declaration
:add(x,y);
// x and y are actual arg.
}
```

ii> Formal Argument:- The value, declared in function header, used to recieve value of actual argument when function calls, is called formal arg.

Syntax:
```
{ int x,y;
  void add(int x, int y);
  : add(x,y);
} void add(int a, int b) // formal arg.
{//body of function
}
```

Passing Argument to Function

Data is passed to a function via argument. There are two type of argument :-

i) Call by value
ii) Call by refrence

lue :- In this method, name of actual argument is wed to function call. The value of actual argument is passed.

Example:

```
#include <stdio.h>
int mul (int, int, int)
void main()
{
  int a, b, c, Prod;
printf ("Enter three numbe\n");
 scanf ("%d %d %d, &a, &b, &c);
 Prod = mul (a, b, c);
printf (" Product is : %d", Prod);
 return 0;
}
int mul (int X, int Y, int Z)
{
 int P;
P = X * Y * Z
return (P);
}
```

When control is transfered to called function, the value of actual parameter are substituted to formal argument and body of function is executed.

ii> Call by refrence :- In this method, address of actual argument is used in function call. The address of actual argument is passed.

Example: (both actual & formal parameter share same memory location) (if change in formal then, actual also be change)

```
# include <stdio.h>
void main()
{
    int x,y;
    printf ("x = ");
    scanf (" %d", &x);
    printf ("y = ");
    scanf ("%d", &y);
    void swap (int *x, int *y);
    swap (&x, &y);
    printf ("value after swap ");
    printf ("x = %d", x);
    printf ("y = %d", y);
    return();
}
void swap (int *x, int *y)
{
    int temp;
    *x = *y;
    *y = temp
}
```

both variable declare

initialization for swap value

swap statement

swaping templelate

When control is transfered to called function, the address of actual parameter are substituted to formal argument & body of function is executed.

Q. What is difference between call by value & call by refrence.

Call by Value	Call by ref.
i> When argument passed by value, any modification in formal argument is not reflected in calling function.	When argument passed by refrence any modification in formal argument is reflected backin calling function.
ii> Actual value is not change	Actual value is change
iii> Formal argument get separate memory	No separate memory is alloted to formal argument.
iii> Not suitable for passing argument of large size.	Suitable for passing argument of large size
iv> Separate copy consume a lot of memory space.	No separate copy is made.
v> It includes argument constant, variable, expression.	It include argument only a variable.

Variable in C

A variable is a memory location that helps to store some data and retrieved it when required.

Always start with alphabet (A-z), underscore (_)

There are two types of variable:-

i> Local Variable

ii> Global Variable

i> Local Variable:- A variable that is declared inside a function, is called local variable. Its scope is limited to function in which it is declared.

Ex:-

```
#include <stdio.h>
void function()
{
    int x = 10; //local vari.
    printf("%d", x);
}
    int main()
{
    function();
}
```

ii> Global Variable:- A variable that is declared outside a function, is called global variable. We can access it anywhere.

Ex:-

```
#include <stdio.h>
int x = 20; // global vari.
void function1()
{ printf("function1: %d \n", x); }
void function2()
{ printf("function 2: %d \n", x); }
int main()
{
    function1();
    function2();
    return 0;
}
```

Recursion in C

A function which calls itself directly or indirectly again and again, is called recursive function and such kind of function calls, is called recursive calls.

Syntax: It is used for constructing link list, tree etc.

```
type function_name (arg)
{
 // function statement
 // base condition
 // recursion case (recursive call)
}
```

Q. Calculate the sum of first N natural number using recursion.

```
#include <stdio.h>
int nSum(int n)
{ // base condion to terminate recursion
   when N=0
 if (n == 0)
 { return 0;
 } // recursive case
  int res = n + nSum(n-1);
  return res;
}
  int main()
{ int n = 5;
  // calling function
  int sum = nSum(n);
printf("Sum of first %d Natural No: %d",
    n, sum);
  return 0;
}
```

Adv:
- Easy to understand & implement
- Reduce code size
- Use only problem easily
- Stack management

Disadv:
- High storage
- Performance overhead
- Difficult to debug
- Limited size

Adv:
1) Simp
2) Eff
3) Simp
4) best

Du
1) Find
2) Ineff
3) Work
4) Limit

lower bound → 0
upper bound → -6

Unit-III 6. Array and String

An array is a collection of identical data which are stored in memory under a common heading or variable name. It can be used to store collection of premitive data type like int, charetc.

Ex:-

A[0], A[1], A[2] ... A[n-1]
(0 based indexing)

There are two types of array:-
i> One dimensional
ii> Multi dimensional

i> One dimensional Array:- Those array that have only one dimension, is called one dimensional array. It is also called 1-D array.

Syntax:

```
array_name [size];
```

Example:

```
#include <stdio.h>
int main()
{ // 1d array declaration
int arr [5];
// 1d array initialization using for loop
for (int i=0; i<5; i++)
{ printf ("%d", arr[i]);
}
return 0;
}
```

Adv
- simple representation
- easy access
- efficient memory utilization
- simple implementation
- best for knew data type

Dis
- Fixed size
- Inefficient insertion/Deletion
- wasted memory
- Limited

iii) Multidimensional Array :- Those array that have more than one dimension, is called multi-dimensional array. Most popular array mutt of thae are 2D array and 3D array.

Syntax:

```
array_name[size1][size2];
```

Use:
- Represent tabular data
- Better Organization
- Direct access
- Simple processing

Disadvantage:
- Memory overhead
- Complex syntax
- Fixed size
- Difficult to manage directly

Two dimensional Array

An array that has exactly two dimension, is called 2-D array. It is in rows and columns form organized in two-dimensional plane.

Example:

```
#include <stdio.h>
void main()
{
    int i, j;
    int x[3][3] = {
        {6, 9, 12},
        {45, 67, 18},
        {11, 16, 19}
    };
printf("The array are:");
for(i=0; i<3; i++)
{
    for(j=0; j<3; j++)
printf("%d", x[i][j]);
return 0;
}}
```

Array Declaration → `int i, j;`

Elements of array declaration → `int x[3][3] = { {6,9,12}, {45,67,18}, {11,16,19} };`

Row & Column declare → `for(i=0; i<3; i++)` and `for(j=0; j<3; j++)`

Array Declaration

We can declare an array by specifying its name, type of element, and size of dimension. When we declare an array, the compiler allocates memory of size to array.

Syntax:

```
data_type array_name [size];
Ex- int name[5];
```

Example:

```
#include <stdio.h>
int main()
{
    // declaring array
    int arr-int[5];
    return 0;
}
```

Array Initialization

Initialization is the process to assign some initial value to variable. There can be multiple way to initialize array are initialization with declaration, initialization after declaration etc.

Syntax:

```
data-type array-name [size]
={value 1, value 2, .... value N};
```

length of string can't exceed 49

JMS

Passing an Array to a Function

An array is always passed as pointer to a function. To pass an array to a function, the array name must specify without any bracket.

While passing, the called function and compiler don't know the size of array.

Syntax:

```
void fl(int x[], int n)
{
:
}
```

Strings in C

A string is a sequence of contingous character terminated with null character. The string is stored as an array of character. The string constant are enclose in double quotes.

Example:

```
#include <stdio.h>
void main()
{ char name[50];
printf("Enter name:");
scanf("%s", name);
printf("Your name is %s", name);
return 0;
}
```

dv:
ary representation
redefined library function
ethod of string
Dynamic Use

ound:
imited string size
se with its use
mbly manipulation

Declaring a String

It is declared by one-dimensional method.

Syntax:

```
char string_name [size];
```

Array of String

It can be represented by 2-D array where, first size specify number of string and second size specify number of character in each string.

Syntax:

```
char variable_name [r] [m]
= {list of string};
```

(r: no. of string)

(m: no. of character)

Example:

```
#include <stdio.h>
int main()
{
  char arr [3][10] = { "Ram ","Shyam",
  Patna"};
  printf (" string array element : \n");
  for (int i = 0; i <3 ; i++)
  {
    printf ("%s \n", arr [i]);
  }
  return 0;
}
```

Library Function

There are predefine inbuilt function which are design to perform some specific task. The number and type of function dependent on compiler. Before using this, it is necessary to includer header file.

String Function

The C string function are built-in function that can be used for various operation and manipulation on string like copy, comparison, length etc.
The <string.h> header file is used for these string function.

Some commonly used string function are :-

i> strlen()
ii> strcpy()
iii> strcat()
iv> strcmp()

i> strlen() :- This function calculates the length of string. It doesn't count null character '\0'.

syntax:

```
int strlen(const char *str);
```

Example:

```
#include <string.h>
int main()
{
  //
  char str[] = "Rose";
  //
  size_t length = strlen(str);
  //
  printf("string: %s \n", str);
  printf("length: %zu\n", length);
  return 0;
}
```

Explain code

ii> strcpy(): - It is a standard library function which is used to copy one string to other.

String:

```
char* strcpy(char* dest, const char* src);
```

Example:

```
#include <string.h>
int main()
{
  //
  char source[] = "Ram";
  char dest[20];
  //
  strcpy(dest, source);
  printf("source: %s \n", source);
  printf("destination: %s\n", dest);
  return 0;
}
```

Explain code

<math.h> is used for mathematical function sqrt() is used,

iii> strcat():- It is wed for string concatenation and combination.

syntax:

```
char * strcat (char * dest, const char *src);
```

Example:

```
#include <string.h>
int main()
{
char dest [50] = "....";
char src [50] = "...";
printf ("dest: %s \n", dest);
//
strcat (dest, src);
printf ("dest : %s", dest);
return 0;
}
```

Explain code

iv> strcmp():- This function compare two strings, character by character.

syntax:

```
int strcmp(const char * str1, const char *str2);
```

```
int main()
{
char str1 [] = "One";
char str2 [] = "Two";
int result = strcmp(str1, str2);
printf ("Compare of str1, str2);
return 0;
}
```

Explain code

Pointer to an Array

Pointer is a variable that that can appear on left side of assignment operator.

Syntax:

```
data_type (*var_name)[size];
```

Ex:-

```
int main()
{
 int arr[5] = {1,2,3,4,5};
 int *ptr = arr;
 printf("%p", ptr);
return 0;
}
```

In this program, the pointer ptr points to 0th element of array.

Pointer to a String

A pointer to string is a pointer that hold memory address of first character in the array.

Syntax:

```
char *str;
```

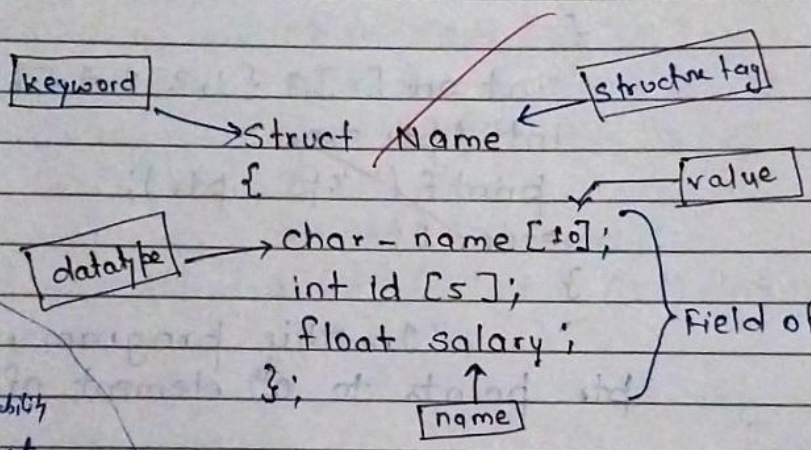
Ex:-

```
#include<string.h>
int main(){
 char *str = "Ram";
 printf("Sring: %s", str);
 return 0;
}
```

Unit-IV 7. Structure and Union

collection of heterogenous data type

A structure is a collection of logically related data item. It is a method of grouping data of different data types. The structure definition is specified by keyword "struct".



Adv:
1) Group related data
2) Improve code readability
3) Easy data management
4) Better data representation
5) Memory Efficiency
6) Support Nested structure

Disadv:
1) No data hiding
2) Lack of function
3) Manual Memory management
4) Inefficient for empty data
5) Limited size

Structure Declaration

In this, we specify its member variable along with datatype. We can use struct keyword to declare.

Syntax:

```
struct name
{
  data-type name
  :
};
```

Access Structure Member

We can access structure member by using (.) dot operator.

Syntax:

```
structure_name.member1;
```

Structure Initialization

A structure can be initialize like other data type.

Example:

```
struct student
{
  int roll no;
  char name[50];
};
```

Structure Initialization

We can initialize the structure by providing the values in the list during declaration.

Syntax:

```
structName varName={value1,---valueN};
```

Example:

```
struct student{
    int id;
    char name[50];
    float percentage;
}; int main()
{ struct student
student1={1,"Ram",85.5};
printf("ID: %d \n" student1.id);
printf("Name: %s \n" student1.name);
printf("percent: %.2f");
```

Same type of variable can assign to other structure

JMS

Structure Assignment

The value of one structure variable can be assign to another variable of same type using structure assignment.

If S1 and S2 are same type of structure variable, then the statement,

S1 = S2

assigns value of structure variable S2 to S1.

Example:

```
struct one
{
    int x;
    float y;
    char z;
};
```

Member of structure

Nested Structure

A nested structure is a structure within structure. One structure can be declared inside another structure in same way as structure member are declared. It is declared as member of another structure.

Syntax:

```
struct name1
{
    member1;
    :
    membern;
    struct name2
    {member1;
    :
    }var1;
```

Example:

```
struct date
{
 int day;
 int month;
};
 struct student
{
 int roll_no;
 char name[50];
};
```

Pointers to a Structure

The pointer which points to the address of memory block that stores a structure, is called structure pointer. Trees, Graph etc are created with the help of structure pointer.

Member is access by arrow operator

Syntax: (name of structure) (pointer to structure)

```
struct name *ptr1, ....
```

Example:

```
#include <stdio.h>
 struct point
{
 int value;
};
 int main()
{
 struct point s;
 struct point* ptr = &s;
 return 0;
}
```

Self Refrential Structures

A structure having refrences to itself, is called self-refrential structure. Useful to implement linklist, tree etc.

The self refrential pointers in structure points to next node of list.

Example:

```
struct list
{
    int data;          // node data
    list *next;        // pointer to next node
};
```

(struct list is self-refrential structure because it has a pointer to next node.)

UNION

The union is a user-defined data type that contain elements of different data types. All members in union are stored in same memory.

The union can be declared by using "union" keyword.

Example:

```
union item
{
    int x;
    float y;
    char a;
} code          // variable of type union
```

Adv:
- i) Memory Efficient
- ii) Flexibility
- iii) ~~Type Dummy~~ versatile
- iv) Simplified data handling

Disad:
- i) Limited Access
- ii) No type Safety
- iii) Difficult to debug
- iv) Portability issue

Q. Difference between structure and Union in C:

i> It is a collection of data of different data type	It is an object, having all member in same location.
ii> Each data object is a member	Variable can be represent the value of member.
iii> Each member is assigned a unique storage	All member share same storage area.
iii> All members may be initialized	Only first member is initialized.
iv> We can access all member at a time.	Only one member is accesed at a time.
v> Memory is require for all variable.	Memory is require for specific variable.
vi> 'struct' keyword is wed to declare	'union' keyword is wed to declare

Array	Structure
i> It is a collection of same data type.	It is a collection of different data type.
ii> It allocates static memory	It allocates dynamic memory
iii> It wes index/subscript for accesing element	It wes (.) operator for accessing member.
iv> It is a pointer to first element	It is a not pointer to first element.
v> It takes less time to access element	It takes more time to access element.

Unit V 8. File Handling

File is a place on the disk where a group of related data is stored permanently.

Defining and Opening a File

A file has to be opened before read and write operation. It provides a link between OS and file function.

When a request is made to OS for opening a file, the request is granted and OS points to structure FILE.

Features
1) Reusability
2) Portability
Efficient Storage

Example:

File *fp = fopen("Abc.doc", "r");

- File: data structure (include header file)
- *fp: File pointer
- fopen: function to open
- "Abc.doc": name of file
- "r": mode of open file

here, 'fp' is declare as pointer which address of file.

Modes of Opening File

The mode show the purpose for which the file is to be opened. Following are modes used in C:-

i> Read mode (r) :- 'r' symbol is used for reading in a file. If the file exists, it sets up a pointer which points first character.

ii> Write mode (w) :- The symbol 'w' is used to write in file. If the file exists, its contents are overwritten.

iii> Append mode (a) :- The symbol 'a' is used to add data in existing file. If the file not exists, a new file is created

iv> Read + (r+) :- The symbol 'r+' is used to read existing content, write new content and modify existing content.

v> Write + (w+) :- The symbol 'w+' is used to write new content, read and modify existing content.

vi> Append + (a+) :- The symbol 'a+' is used to read existing content, add new content but not modify content.

Note: The fclose function is used to close existing open file.
∴ fclose (fp);

Program to open file

```
#include <stdio.h>
  void main()
  {
    FILE *fp;
    Fp = fopen("Abc.d",'r");
   if (fp == Null)
  {
    printf("Not open file");
  exit(1);
  }
    fclose(fp);
    return 0;
  }
```

Reading from a File

There exist a number of function to read a character from a file. The character to which file pointer points to is 'read', then shifted to next.

i> getw():- It is an integer oriented function which is used to read an integer from a file.

Syntax:

```
a = getw (fp);
```

↑ integer type variable — ↑ Function — ↑ integer type variable

ii> fcanf ():- This function is used to read any type of variable - int, char, etc.

syntax:

```
fscanf(fp, "%c", &ch);
```

fp → file pointer
"%c" → control string
&ch → character type variable

iii> fgets():- This function is used to read string from a file.

syntax:

```
fgets(str, n, fptr);
```

str → array of character
n → maximum length
fptr → file pointer

iv> fread():- This function read a block of statement from a file.

syntax:

```
fread(ptr, m, n, fptr);
```

ptr → address
m → size
n → maximum length
fptr → file pointer

Writing to a File

There are several functions to write a characters or numbers in file.

i> fputc():- This function is used to write character in a file at specified pointer.

syntax:

```
fputc(c, fp);
```

fputc → function
c → character type variable
fp → file pointer

0 - begin
1 - correct code
2 - end

JMS

Date:
Page No.: 60

ii> fputw( ):- This function is wed to print a number in a file.

Syntax:

```
fputw(a, fp);
```

fp ← file pointer
fputw ← function
a ← Integer type variable

iii> fprintf( ):- This function is wed to print character, number, string in file.

Syntax:

```
fprinf(fp, "%d", a);
```

fp ← file pointer
"%d" ← control string
a ← Numeric variable.

Direct and Random Access

The function that is used to access directly the data item is "fseek( )".

Syntax:

```
fseek(fptr, offset, mode);
```

fptr ← file pointer
offset ← No. of bites to move
mode ← contain value

i> fseek( ):- This pointer/functions moves the file pointer from one to other position.

ii> ftell( ) :- This function is used to return integer type value.

syntax:

```
n = ftell (fp);
```

n: value of type long int
fp: file pointer

iii> rewind( ) :- It is used to set the file pointer to starting of file.

syntax:

```
rewind (fp);
```

fp: file pointer

iv> a+ (append + read) :- This mode is used for reading and appending the record.

example:

```
fptr = fopen ("rec.dat", "a+");
```

v> Closing a file :- When the program finished with reading and writing file, it must be closed.

syntax:

```
fclose (fptr);
```

fptr: file pointer

CR - Carriage Return
LF - Linefeed

JMS

Date:
Page No.: 62

Structure of File

```
main()
{
  FILE *fptr, *fopen();
  fptr = fopen("file name", "mode");
  -----
  -----
  fclose(fptr);
}
```

Eol in DOS - CR, LF

i> Newline Character (Point):- The end of line is signaled by single character, the newline character "\n". While writing a file, it is translated into CR/LF before writing and while reading, it is translated back into newline character.

ferror - It detects error occur in file on file pointer

unlink - delete files from directry

feof - It detects whether a file is at end of file or not

ii> End of file (EOF):- EOF is an integer value sent to program by OS. Its value is predefined in header file.
no duplicate value is stored. EOF transmit signal when OS detects last character.

read & write character

Character Input and Output

i> Writing to file using fputc():- The function which is used to writing one character at a time is "fputc".

Syntax:

```
fputc(ch, fptr);
```

ch ← character variable
fptr ← file pointer

Example:-

```
#include <stdio.h>
    main()
{
  FILE *fptr
   char name[15], ch;
printf("Enter file name: ");
 scanf("%s", name);
if((fptr=fopen(name,"r")==Null)
{
 printf("file doesn't exist");
 exit(1);
}
 else
{
  fptr1 = fopen("rec.txt", "w");
while(ch = fgetc(fptr)! = EOF)
fputc(ch, fptr);
}
 fclose(fptr1);
  fclose(fptr1);
}
```

ii> Reading from file using fgetc():- This function read a single character from a file and increments the file pointer position.

Syntax:

```
fgetc(fptr);
ch = fgetc(fptr);
```

ch → character variable
fptr → file pointer

Example:

```
#include <stdio.h>
  main()
{
  FILE *fopen(), *fptr
charch;
if(fptr = fopen("rec.dat","r")==Null)
printf ("File not exist")
 else
 {
 while ((ch=fgetc(fptr)!= EOF)
printf ("%c",ch);
 }
  fclose(fptr);
}
```

read & write string

String Input and Output

i> Writing to file using fputs():- This function is used to write string to a file.

read & ch

Syntax:

```
fputs(sptr, fptr);
```

sptr: String pointer
fptr: File pointer

Example:

```
#include <stdio.h>
  main()
 {
  FILE * fptr
  char name [20], arr[50];
  scanf("%s, name);
```

```
if((fptr = fopen(name,"w")) == Null)
{
printf(" File not open");
exit(1);
}
else
{
printf("The string is : ");
gets(arr);
fputs(arr, fptr);
}
fclose(fptr);
}
```

ii> Reading from file using fgets( ) :- This function is used to read string from a file and copies the string to memory.

Syntax:

```
fgets(sptr, max, fptr);
```

sptr ↑ string pointer; max ↑ length of array; fptr ↖ file pointer

maximum length

Example:

```
#include <stdio.h>
main()
{
FILE * fptr
char name[20], arr[50];
int i = 0;
printf("Enter file name");
scanf("%s", name);
if((fptr = fopen(name,"r")) == Null)
{
```

```
printf("File not exist");
exit();
}
else
if(fgets(arr,50,fptr)!=NULL)
while(arr[i]!='\0')
{
putchar(arr[i]);
i++;
}}
```

Formatted Input and Output

Print & write everything fprintf

i> Writing to File using fprintf():- It is same as printf() function but it writes data into file.

Syntax:

```
fprintf(fptr, "control character", variable name);
```

Example:

```
#include <stdio.h>
main()
{
FILE *fopen(), *fptr;
char name[10];
fptr = fopen("rec.dat", "r");
printf("Enter name your");
scanf("%s", name);
fprintf(fptr, "My name is %s", name);
fclose(fptr);
}
```

ii> Reading from file using fscanf():- It is same as scanf() function but it reads data from file.

Example:

```
#include <stdio.h>
  main()
  {
   FILE *fopen, *fptr;
  char name[10];
  intsal;
  p = fopen("rec.dat", "r");
  fscanf(fptr, "%s %d", name, &sal);
  while(!feof(fptr))
  {
  printf("%s %d", name, sal);
  fscanf(fptr "%s %d", name, sal);
  }
  fclose(fptr);
  }
```

read or write block of data

Record (Block) Input & Output

i> Write to file using fwrite():- This function is used for writing an entire block to a file.

Syntax:

```
fwrite(ptr, size, nst, fptr);
```

ptr → pointer
size → size of structure
nst → No. of structure
fptr → file pointer

iii> Reading from file using fread() :- This function is used to read entire block from a file.

Syntax:

```
fread(ptr, size, nst, fptr);
```

Printing a File

Print file can be created by opening a file in 'w' mode and then, writing the contents of file on printer.

Syntax:

```
fptr = fopen("prn", "w");
```

Example:

```
#include <stdio.h>
main()
{
 FILE *fptr1, *fptr2;
 char name[20], ch;
 printf("Enter file name:");
 scanf("%s", name);
if((fptr1 = fopen(name,"r")==Null)
{
 printf(" File does not exist");
}
else
{
 if((fptr2 = fopen("prn","w"))==Null)
{
 printf("Error to printer");
exit(1); }
```

```
else
{
 while ((ch = getc(fptr1)) != EOF)
 putc(ch, fptr2);
}
 fclose(fptr1);
 fclose(fptr2);
}
```

Q. Write a Program For Rewind Function.

Ans

```
#include <stdio.h>
  main()
 {
  FILE *fopen(), *fptr;
 char ch;
  fptr = fopen("text", "r");
 fseek(fptr, 2L, 0);
  rewind(fptr);
 ch = fgetc(fptr);
 while (!feof(fptr))
 {
  printf("%c", ch);
 ch = fgetc(fptr);
 }
  fclose(fptr);
 } return 0;
}
```

Thu 24/10/2024