Programming in C and Data Structures

June/July 2017 Question Paper Solutions

Prepared by Prof.Chandrika C N and Prof.Nagashree C

Module 1

1a. Define pseudo code. Explain with Example. 5 marks

Answer

- Pseudocode is series of steps to solve a given problem written using a mixture of English and C language.
- It is the first step in writing a program.
- It is written using a mixture of English and C language.
- It is a series of steps to solve a given problem
- It acts as a problem solving tool.

Advantage:

- Easy to write and understand
- It is relatively easy to convert English description solution of small programs to C program.

Disadvantage:

- It is very difficult to translate the solution of lengthy and complex problem in English to C.

Example:

Sum and average of three numbers

- Get the numbers[a,b,c]
- Compute addition [Sum= a + b+c]
- Compute average[avg=sum/n]
- Print the results [Sum].

1b. Write a C program to find biggest among three numbers using ternary operator. 5 marks.

Answer:

```
#include<stdio.h>
void main()
{
```

```
int a,b,c,big;
printf("enter the values of a,b,c\n");
scanf("%d%d%d",&a,&b,&c);
big=(a>b)?(a>c?a:c):(b>c?b:c);
printf("big=%d",big);
}
```

1c. Explain the following constants with example. 6 Marks

1.Integer Constant 2.Floating Constant 3.Character Constant

Answer

1. **Integer constant**
   - Definition: An integer is whole number without any decimal point.no extra characters are allowed other than + or _ .
   - Three types of integers
   - **Decimal integers**: are constants with a combination of digits 0 to 9,optional + or -

     Ex: 123 , -345, 0 , 5436 , +79
   - **Octal integers:** are constants with a combination of Digits from 0 to 7 but it has a prefix of 0

     Ex: 027 , 0657 , 0777645
   - **Hexadecimal integers:** Digits from 0 to 9 and characters from a to f, it has to start with 0X or 0x

     Ex: 0X2 0x56 0X5fd 0xbdae

2. **Real constants/Floating point:**
   - **Floating point** constants are base 10 numbers that are represented by fractional parts, such as 10.5.They can be positive or negative.
   - Two forms are:

   i. **Fractional form:**
      - A floating point number represented using fractional form has an integer part followed by dot and a fractional part.
      - Ex: 0.0083
      - 215.5
      - -71.
      - +0.56 etc..

   ii. **Exponential form:**
      - A floating point number represented using **Exponent form has 3 parts**
      - **mantissa e exponent**
      - Mantissa is either real number expressed in decimal notation or integer.

- **Exponent must be integer with optional + or –**
- Ex 9.86 E 3 => $9.86*10^3$
- Ex 9.86 E -3 => $9.86*10^{-3}$

Character constant

- A symbol enclosed within pair of single quotes is called character constant.
- Each character is associated with unique value called ASCII value.
- Ex: '9'
- '$'

- Backslash constants(Escape sequence character)
  - Definition: An escape sequence character begins with backslash and is followed by one character.
  - A backslash along with a character give rise to special print effects.
  - It always start with backslash ,hence they are called as backslash constants.

character	name	Meaning
\a	Bell	Beep sound
\b	Backspace	Cursor moves towards left by one position
\n	Newline	Cursor moves to next line
\t	Horizontal tab	Cursor moves towards right by 8 position
\r	Carriage return	Cursor moves towards beginning of the same line
\"	Double quotes	Double quotes
\'	Single quotes	Single quotes
\?	Question mark	Question mark
\\	Backslash	Backslash
\0	NULL	

2a. List the formatted input/output functions in C language. Explain the basic structure of C program with proper Syntax and example. 6 Marks

Answer:

The formatted input/output functions in C language are **scanf() and printf()**

Structure of C Program

Comments/Documentation Section
Preprocessor Directives
Global declaration section
void main( ) [Program Header] **{** **Local Declaration part** **Execution part** **Statement 1** **----------** **----------** **Statement n** **}**
User defined Functions

- **Comments/Documentation Section**

- Comments are short explaining the purpose of the program or a statement.
- They are non executable statements which are ignored by the compiler.
- The comment begins **with /* and ends with */.**
- The symbols /* and */ are called **comment line delimiters.**
- We can put any message we want in the comments.

Example: /* Program to compute Quadratic Equation */

Note: Comments are ignored by C compiler, i.e. everything within comment delimiters

- **Preprocessor Directives**

- Preprocessor Directives begins with a # symbol
- Provides instructions to the compiler to include some of the files before compilation starts.
- Some examples of header files are

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>

#include<stdlib.h> Etc…

- **Global Declaration Section**
  - There will be some variable which has to be used in anywhere in program and used in more than one function which will be declared as global.
- **The Program Header**
  - Every program must contain a main() function.
  - The execution always starts from main function.
  - Every C program must have only one main function.
- **Body of the program**
  - Series of statements that performs specific task will be enclosed within braces { and }
  - It is present immediately after program header.
    It consists of two parts
    1. **Local Declaration part**: Describes variables of program
       Ex:**int sum = 0;**
       int a , b:
       float b;
    2. **Executable part**: Task done using statements.
       All statements should end with semicolon(;)

Subroutine Section: All user defined functions that are called in main function should be defined.

---

2 b. Define an algorithm. Write an algorithm to find area of circle and triangle. 6 Marks

Answer:

Algorithm

- It is a step by step procedure to solve a problem
- A sequential solution of any problem that is written in natural language.
- Using the algorithm, the programmer then writes the actual program.
- The main use of algorithm is to help us to translate English into C.
- It is an outline or basic structure or logic of the problem.

Algorithm for area of Circle:

Algorithm: Area of circle

Input: radius r

Output: area

Step 1: Start

Step 2: [input radius]

Read r

Step 3: [compute area]

area = 3.142*r*r

Step 4: print area

Step 5: Stop

Algorithm for area of Triangle:

Algorithm: Area triangle

Input: breadth and height

Output: area

Step 1: Start

Step 2: [input breadth and height]

Read b and h

Step 3: [compute area]

area = 0.5*b*h

Step 4: print area

Step 5: Stop

2c.Evaluate the following expressions/code Segment:

i) 22+3<6&&!5||22==7&&22-2>=5

ii) a+2>b||!c&&a==d||a-2<=e where a=11, b=6, c=0, d=7, e=5

Refer Classwork

Module 2

3. a List all the branching statements. Explain any two with proper syntax and example.

6 Marks

Answer:

There are 2 types of branching statements present **1.Conditional Statement**
2.Unconditional Statement.

1.Conditional Statement

Definition: The statements that transfers the control from one part of the program to another part of the program based on a condition is called as **Conditional statements.**

There are **5 Conditional** statements in C

1. **if control construct(simple if)**
2. **if else control construct**
3. **nested if else control construct**
4. **else if ladder or cascaded**
5. **switch control construct**

2.Unconditional Statements

Definition:

The statements that transfer the control from one part of the program to another part without checking for any conditions are called unconditional branch statements.

The 4 Unconditional branch statements are

1. goto
2. break
3. continue
4. return

The if statement (Simple if/ One way selection)

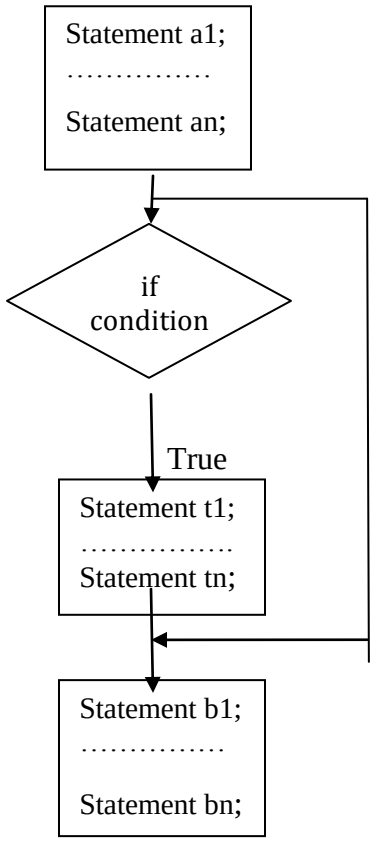
- An if statement is a single selection statement.
- It is used to execute a set of statements if the condition is true, if the condition is false, it skips executing those set of statements. Hence it is called **one way selection**.

Syntax:

```
Statement a1;
      ……..
Statement an;
if(condition)
{
      Statement t1;
       ……..
      Statement tn;
}
      Statement b1;
      ……..
      Statement bn;
```

False

Flowchart:



Explanation

- The keyword **if** must be followed by an **expression** and expression must be enclosed within parentheses.
- First statement a1 to an is executed sequentially(one after another).
- The if statement is executed next. The expression inside the parentheses is evaluated. i.e Condition is checked, which results in either **TRUE** or **FALSE** .
- If condition is **TRUE** the statements t1 to tn are executed and then statements b1 to bn are executed.
- If condition is **FALSE** the statements t1 to tn are skipped and then statements b1 to bn are executed.

Advantage of if-statement

- Output of if-statement is true or false.
- if-statement is used as one way decision/selection statement.

Disadvantage

- If one action has to be performed when the condition is true and another action has to be performed when the condition is false then if-statement is not recommended
- This disadvantage is overcome using two- way decision/selection statement called " if-else statement"

Example:

```
#include<stdio.h>
void main()
{
    int n;
    printf(" Enter the number\n")
    scanf("%d",&n);
    if(n%2==0)
    {
        printf("Even no");
    }


}
```

The if – else statement (two way selection).

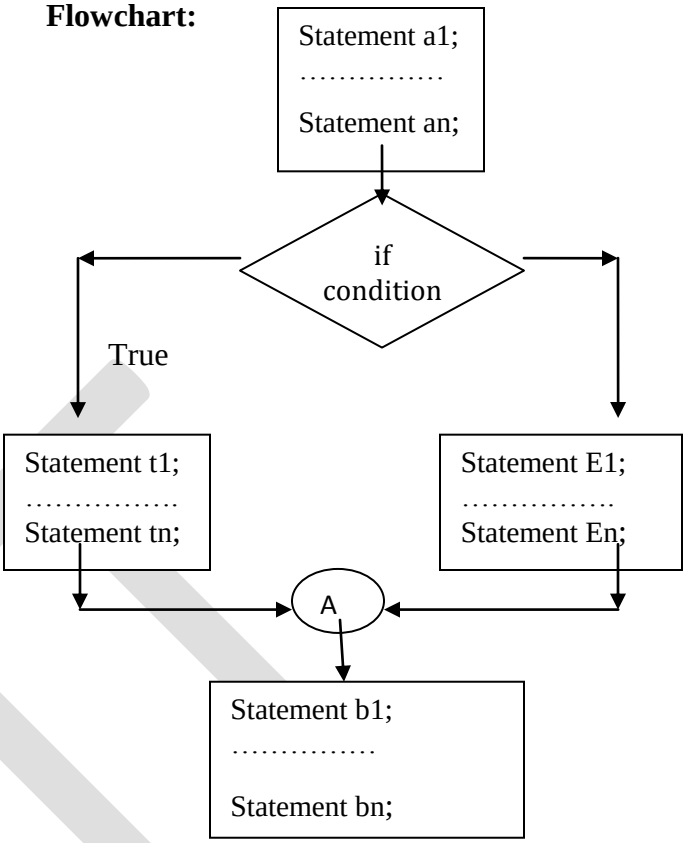
It is used to execute a set of statements if the condition is true, and another set of statements if the condition is false. Hence it is called two way selection.

Syntax:

```
Statement a1;
      ……..
Statement an;
if(condition)
{
      Statement t1;
       ……..
      Statement tn;
}
else
{    Statement E1;
       ……..
      Statement En;
}
      Statement b1;
      ……..
      Statement bn;
```

False

Flowchart:



Explanation

- The key wed by an expression and expression must be enclosed within parentheses.
- First statement a1 to an is executed sequentially(one after another).
- The if statement is executed next. The expression inside the parentheses is evaluated. i.e Condition is checked, which results in either **TRUE** or **FALSE** .
- If condition is **TRUE** the statements t1 to tn are executed and else block is skipped then statements b1 to bn are executed.
- If condition is **FALSE** the statements if block is skipped and then statements E1 to En are executed and then statements b1 to bn are executed.

Example:

An Example which illustrates if-else statement: To print given no is an even no or odd no.

```
#include<stdio.h>
void main()
{
      int n;
      printf(" Enter the number\n")
       scanf("%d",&n);
```

```
    if(n%2==0)
    {
        printf("Even no");
    }
     else
    {
         printf("odd no");
      }

}
```

3b. Explain switch case statement with syntax and example. 5 Marks

Answer:

Switch statements are used in following scenarios

- When a decision has to be made between many alternatives
- When the selection condition reduces to an integer value.

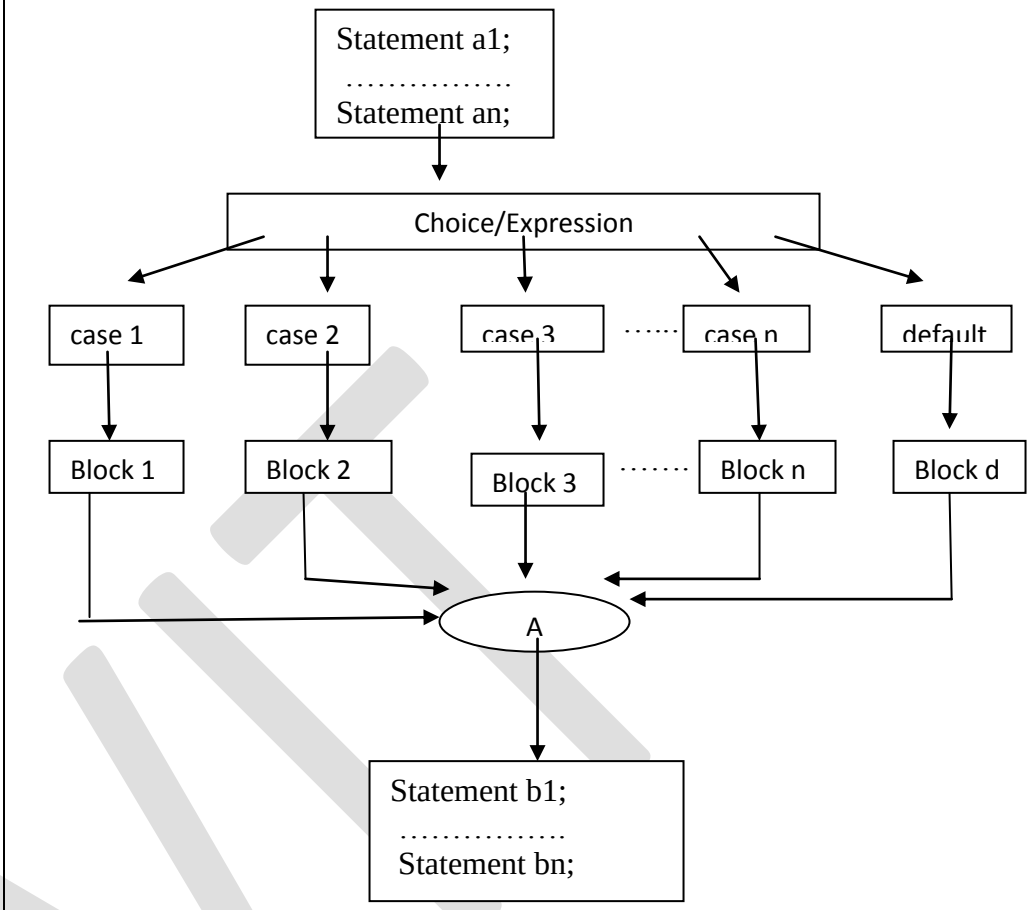
Definition: The switch statement is a control statement used to make a selection between many alternatives. i.e multiple selection mechanism is implemented using *switch*.

- ***switch*** is alternative for two way selection multiple ***if-else-if***.

Statement a1;
…….
Statement an;

```
switch(choice)
{
case value1:   Block1;
                 break;
case value2: Block 2;
                 break;
case value3:  Block3;
                 break;
default:  Block d;
}
```

Statement b1;
…….
Statement bn;



Explanation

- The switch statement is a multi-way decision that tests whether an expression matches one of a number of constant integer values, and branches accordingly.
- Each case is labeled by one or more integer-valued constants or constant expressions.
- If a case matches the expression value, execution starts at that case.
- All case expressions must be different.
- The case labeled default is executed if none of the other cases are satisfied.
- A default is optional; if it isn't there and if none of the cases match, no action takes place. Cases and the default clause can occur in any order.

- The break statement causes an immediate exit from the switch.
- The expression of a switch statement must result in an *integral type*, meaning an integer (byte, short, int, long) or a char.
- The expression cannot be a Boolean value or a floating point value (float or double)
- No two case labels can have the same constant value.

An Example which illustrates switch control statement:

Program to simulate a simple calculator that performs arithmetic operations only on integers. Error message should be reported if any attempt is made to divide by 0.

```
#include<stdio.h>
void main()
{
   int a,b,res;
   char choice;
   printf(" Enter your choice\n");
   scanf("%c",&choicec);
   printf(" Enter two operands\n");
   scanf("%d%d",&a,&b,);
   switch(choice)
   {
      case '+':   res = a+ b;
                  printf("%d", res);
                   break;
      case '-':   res = a- b;
                  printf("%d", res);
                  break;
      case '*':   res = a-*b;
                  printf("%d", res);
                  break;
       case '/':   if(b==0)
                 {
                    printf("error :Divided by   0");
                  }
                  else
                 {
                    res = a/ b;
                    printf("%d", res);
                   }
                   break;
        default: printf( "invalid op");
```

```
        }
    }
```

Advantages:

1. Improves readability of a program.
2. More Structured way of writing program.

Disadvantages:

1. Used only if the expressions used for checking results in integer value.
2. Cannot be used when a decision is based on range of values.

3c. Write a C program to find whether given year is leap year or not. 5 Marks

Answer

```
#include<stdio.h>
void main()
{
        int year;
        printf("enter a year:\n");
        scanf("%d",&year);
        if((year%4==0 && year%100!=0)||(year%400==0))
                printf("%d is a leap year",year);
        else
                printf("%d is not a leap year\n",year);
        }
```

4.a Write the syntax of all looping control statements. Explain how break and continue statements are used in C program with example. 6 Marks

Looping Control statements in C are:
1.while loop
2.do-while loop
3.for loop

while loop(syntax)	**do-while loop(syntax)**	**for loop(syntax)**
Statement a1; Statement a2; **initialization;** ***while*(Condition check)** **{** Statement b1; Statement b2;	Statement a 1; Statement a2; **initialization;** ***do*** **{** Statement b1; Statement b2;	**Statement A1;** **Statement A2;** ***for* (exp1 ;exp2; exp3)** **{** **Statement B1;** **Statement B2;** }

updation; } Statement c1; Statements c2;	**updation**; } ***while*(Condition check);** Statement c 1; Statements c 2;	**Statement C1;** **Statements C2;**

The break statement:

- The break statement is an unconditional jump statement.
- The break statement can be used as the last statement in each case's statement list.
- A break statement causes control to transfer to the end of the switch statement.
- If a break statement is not used, the flow of control will continue into the next case.

"Uses of break statement"

The **break** statement in C programming language has the following two usages:

- When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.
- It can be used to terminate a case in the switch statement.
- If you are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

switch (option)
{
case 'A':
aCount++;
break;
case 'B':
bCount++;
break;
case 'C':
cCount++;
break;
}
	for(……)
	{
	…….
	if(…….)
	{
	……
	break;
	……..
	……..
	}
	…….
	……..
	}

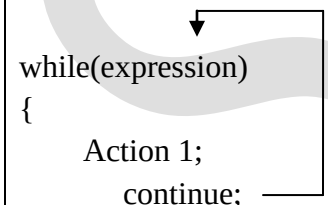
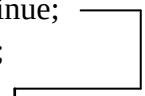
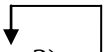
Example: Refer switch statement example program	Example:
	#include<stdio.h>
	void main()
	{
	int i,n=5;

	for(i=0;i<=n;i++)
	{
	if(i==3)
	break;
	printf("%d\n",i);
	}
	}
	
	Output: 1 2

The continue Statement

- The continue statement can be placed only in the body of a for loop, a while loop, or a do...while loop.
- The continue statement is used only in the loops to terminate the current iteration and continue with remaining iterations.
- In the **while** statement and **do-while** statement, whenever a **continue statement** is executed, the rest of the statements within the body of the loop are skipped and the **conditional** expression in the loop is executed.
- But, when **continue** is executed in the **for loop** ,control is transferred to **expression 3**.
- The pictorial representation is shown below:

Current iteration is skipped	Current iteration is skipped	Control is transferred exp3


while(expression)
{
Action 1;
continue;
Action n;
}
	do
	{
	Action 1;
	continue;
	Action n;
	 
	} while(expression);
		
		for(exp1,exp2,exp3)
		{
		Action 1;
		continue;
		Action n;
		}

Example:

```
#include<stdio.h>                    Output: 1 2 4 5
void main()
{
   int i,n=5;
   for(i=1;i<=5;i++)
   {
```

```
            if(i==3)
            continue;
            printf("%d\n",i);
        }
    }
```

4b. Write a C program to find the square root of a given number without using library function. 5 Marks

Answer:

```
#include<stdio.h>
void main()
{
    float n,x,root;
    int i;
    printf("ENTER ANY NUMBER\n");
    scanf("%f",&n);
    x=1;
    for(i=0;i<20;i++)
    {
        root=(x*x+n)/(2*x);
        x=root;
    }
    printf("square root of given number is %f",root);
}
```

4c. List the differences between while and do-while loop 5 Marks

Answer:

Difference between while loop and do while loop

while loops ***pretest loop***	***do while loops*** ***post test loop***
Syntax Statement a1; Statement a2; **initialization;** ***while*(Condition check)**	**Syntax** Statement a 1; Statement a2; **initialization;** ***do***

{ Statement b1; Statement b2; **updation**; } Statement c1; Statements c2;	**{** Statement b1; Statement b2; **updation**; } ***while*(Condition check);** Statement c 1; Statements c 2;
It is a **top testing/entry controlled loop** since the condition is checked in the beginning itself.	It is a bottom testing /exit controlled loop since the condition is checked in the bottom of the loop.
It is a **pre test loop**, so if the expression is false in the beginning itself, the statements within the body of the loop will not be executed.	It is a **post test loop,**and the body of the loop will be executed atleast once.
While loop is a Entry controlled loop ,because the condition is checked first and if that condition is true than the block of statement in the loop body will be executed	***Do while*** loop is a Exit controlled loop the body of loop will be executed first and at the end the test condition is checked, if condition is satisfied then body of loop will be executed again.

Module 3

5 a. Define the array. How one and two dimensional arrays are declared and initialized? Explain. 7 Marks

Answers:

Arrays: Array is a sequential collection of similar data items.

Pictorial representation of an array of 5 integers

10	20	30	40	50

A[0] A[1] A[2] A[3] A[4]

- An array is a collection of similar data items.
- All the elements of the array share a common name .
- Each element in the array can be accessed by the subscript(or index) and array name.
- The arrays are classified as:

1. **Single dimensional array**
2. **Multidimensional array.**

Single Dimensional Array.

- A single dimensional array is a linear list of related data items of same data type.
- In memory, all the data items are stored in contiguous memory locations.

Declaration of one-dimensional array(Single dimensional array)

Syntax:

```
datatype  array_name[size];
```

- **datatype** can be int,float,char,double.
- **array_name** is the name of the array and it should be an valid identifier.
- **Size** is the total number of elements in array.

For example:

int a[5];

The above statement allocates 5*2=10 Bytes of memory for the array **a.**

a[0] a[1] a[2] a[3] a[4]

float b[5];

The above statement allocatests 5*4=20 Bytes of memory for the array **b.**

- Each element in the array is identified using integer number called as i**ndex.**
- If n is the size of array, the array index starts from **0** and ends at **n-1.**

Storing Values in Arrays

- Declaration of arrays only allocates memory space for array. But array elements are not initialized and hence values has to be stored.
- Therefore to store the values in array, there are 3 methods
  1. Initialization
  2. Assigning Values
  3. Input values from keyboard through **scanf()**

3.1.2 Initialization of one-dimensional array

- **Assigning the required values to an array elements before processing is called initialization.**

data type array_name[expression]={v1,v2,v3…,vn};

Where

- ✓ datatype can be char,int,float,double
- ✓ array name is the valid identifier
- ✓ size is the number of elements in array
- ✓ v1,v2,v3……..vn are values to be assigned.

- Arrays can be initialized at declaration time.
  Example:
  int a[5]={2,4,34,3,4};

2	4	34	3	4
a[0]	a[1]	a[2]	a[3]	a[4]

- The various ways of initializing arrays are as follows:
  1. **Initializing all elements of array(Complete array initialization)**
  2. **Partial array initialization**
  3. **Initialization without size**
  4. **String initialization**

1. Initializing all elements of array:

- Arrays can be initialized at the time of declaration when their initial values are known in advance.
- In this type of array initialization, initialize all the elements of specified memory size.
- Example:

  int a[5]={10,20,30,40,50};

10	**20**	**30**	**40**	**50**

2. Partial array initialization

- If the number of values to be initialized is less than the size of array then it is called as partial array initialization**.**
- In such a case elements are initialized in the order from 0^{th} element.
- The remaining elements will be initialized to **zero automatically by the compiler.**
- Example:
  int a[5]={10,20};

10	20	0	0	0

3. **Initialization without size**

- In the declaration the array size will be set to the total number of initial values specified.
- The compiler will set the size based on the number of initial values.
- Example:

  int a[ ]={10,20,30,40,50};
- **In the above example the size of an array is set to 5**

4. **String Initialization**

- Sequence of characters enclosed within double quotes is called as string.
- The string always ends with NULL character**(\0)**

char s[5]=”SVIT”;

We can observe that string length is 4,but size is 5 because to store NULL character we need one more location.

So pictorial representation of an array **s** is as follows:

S	V	I	T	\0
S[0]	S[1]	S[2]	S[3]	S[4]

Two Dimensional arrays:

- In two dimensional arrays, elements will be arranged in rows and columns.
- To identify two dimensional arrays we will use two indices(say i and j) where I index indicates row number and j index indicates column number.

Declaration of two dimensional array:

data_type array_name[exp1][exp2];

Or

data_type array_name[row_size][column_size];

- **data_type** can be int,float,char,double.
- **array_name** is the name of the array.
- **exp1 and exp2** indicates number of rows and columns

For example:

```
int a[2][3];
```

- ✓ The above statements allocates memory for 3*4=12 elements i.e 12*2=24 bytes.

Initialization of two dimensional array

Assigning or providing the required values to a variable before processing is called initialization.

```
Data_type array_name[exp1][exp2]={
                                  {a1,a2,….an},
                                  {b1,b2, .bn},
                                  …….
                                  ……..
                                  {z1,z2,…zn}
                                 };
```

- Data type can be int,float etc.
- exp1 and exp2 are enclosed within square brackets .
- both exp1 and exp2 can be integer constants or constant integer expressions(number of rows and number of columns).
- a1 to an are the values assigned to 1st row ,
- b1 to bn are the values assigned to 2nd row and so on.

Example:

```
int a[3][3]={
             {10,20,30},
             {40,50,60},
             {70,80,90}
            };
```

10	**20**	**30**
40	**50**	**60**
70	**80**	**90**

Partial Array Initialization

- If the number of values to be initialized is less than the size of array, then the elements are initialized from left to right one after the other.
- The remaining locations initialized to zero automatically.
- Example:

```
int a[3][3]={
             {10,20},
             {40,50},
             {70,80}
```

};

10 ➢	**20**	**0**
40	**50**	**0**
70	**80**	**0**

5b. . Write a C program to evaluate polynomial f(x) = a4*x*4 + a3*x*3 + a2*x*2 + a1*x* + a0, for a given constant *x* and its coefficients **4 Marks**

Answer:

```
#include<stdio.h>
void main()
{
        int sum;
        int x,i,n,a[20];
        printf("enter the degree of polynomial\n");
        scanf("%d",&n);
        printf("enter the co-efficients of the polynomial\n");
        for(i=0;i<=n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("enter the values of x\n");
        scanf("%d",&x);
        sum=a[n];
        for(i=n-1;i>0;i--)
                sum=(sum+a[i])*x;
        sum=sum+a[0];
        printf("\n the value of polynomial is %d\n",sum);

}
```

5c.Explain string Input/output functions with example. 5 Marks

Formatted Input Function: scanf()

- The formatted input function is scanf().
- It reads a string from the keyboard
- The format specifier is **%s**
- **The string is terminated by NULL character(\0)**
  Syntax: scanf("%s",str);
- Example:
  char str[5]="SVIT";
  scanf("%s",str);

NOTE: scanf() cannot read spaces and any special symbols i.e conversion code cannot read spaces, it will terminated as soon as space appear.

Formatted Output Function: printf()

- The formatted output function is printff().
- It prints/displays a string which is stored on memory locations on monitor
  Syntax: printf("%s",str);
- Example:
  char str[5]="SVIT";
  printf("%s",str);

Write a program to read and print an string using scanf() and printf()

```
#include<stdio.h>
void main()
{
char str[20];
printf("enter the string\n");
scanf("%s",str);
printf("The entered string is \n");
printf("%s",str);
}
```

UnFormatted Input Function: scanf()

- The Unformatted input function is gets().
- It reads a sequence of characters(line) from the keyboard with spaces in between and store them in memory locations.
  Syntax: gets(str);

UnFormatted output Function: puts()

- The Unformatted output function is puts().
- This function displays all the character(line) stored in variable **str** on the monitor till it encounters **\0(Null Character)**
  Syntax: gets(str);

➢ Example:
char str[20];
printf("enter the string\n");
gets(str);
	➢ Example:
	char str[20]="HELLO";
	printf("the string is \n");
	puts(str);

Write a program to read and print an string using gets() and puts()

```
#include<stdio.h>
#include<string.h>
void main()
{
char str[20];
printf("enter the string\n");

gets(str);
printf("The entered string is \n");
puts(str);
}
```

OutPut:

```
Enter the string
HELLO HOW R U
The entered string is
HELLO HOW R U
```

H	E	L	L	O		H	O	W		R		U	\0					

6a. Explain how strings are declared and initialized with syntax and example.6 Marks
Answer:

String Declaration:

- Like all other variable a string variable a string variable also has to be declared before it is used.

Syntax: char string_name[size];

Example: char s[10];

The above declaration statement allocates 10 bytes of memory to string s as follows:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

String Initialization:

Initialization is a process of assigning values to a string, before doing manipulation.

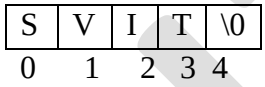
Strings are initialized in 4 ways:

1. Initializing character by character
2. Partial Array Initialization
3. Initialization without size
4. Initialization of Array with string

1.Initializing character by character

- ✓ Consider following declaration and initialization

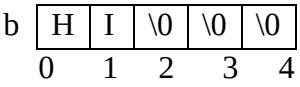
  char b[5]={ 'S','V','I','T'};
- ✓ The complier allocates 5 memory locations and these locations are initialized with the character in the order specified.



2. Partial Array Initialization

- ✓ If the number of characters to be initialized is less than the size of array then the remaining locations will be initialized to NULL as follows:

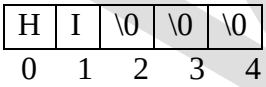
Char b[5]={ 'H','I'};



3.Initialization without Size

- ✓ If a string is declared without size then compiler will set the array size to the total number of initialized values.

char b[]={ 'S','V','I','T'};

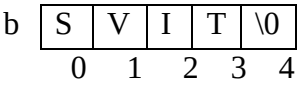


Size of b is 4

4.Initialization of array with string

char b[]="SVIT";

In the above initialization the string length is 4 bytes but size is 5 bytes.



6b. Write a C program to find addition of two matrices. 4 Marks

Answer:

```
#include<stdio.h>
void main()
{
        int m,n,i,j,a[3][3],b[3][3] ,c[3][3];
        printf("enter number of rows and columns\n");
        scanf("%d %d",&m,&n);
        printf("enter array  a elements\n");
        for(i=0;i<m;i++)
```

```
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",&a[i][j]);
                }
        }

        printf("enter array  b elements\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",&b[i][j]);
                }
        }
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        c[i][j]=a[i][j]+b[i][j];
                }
        }
        printf("resultant matrix c is \n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d\t",c[i][j]);
                }
                printf("\n");
        }
}
```

6c. Explain function definition, function call and function declaration with example. Elements of User Defined Function. 6 Marks

Answer:

The Three Elements of User Defined function structure consists of :

1. **Function Definition**

2. **Function Declaration**
3. **Function call**

1. Function Definition:

- A program Module written to achieve a specific task is called as function definition.
- Each function definition consists of two parts:
  - **i. Function header**
  - **ii. Function body**

General syntax of function definition

Function Definition Syntax	**Function Definition Example**
datatypefunctionname(parameters) **{** **declaration part;** **executable part;** **return statement;** **}**	**void add()** **{** **int sum,a,b;** **printf("enter a and b\n");** **scanf("%d%d",&a,&b);** **sum=a+b;** **printf("sum is %d",sum);** **}**

i. Function header

Syntax

datatype functionname(parameters)

- It consists of **three** parts
  - **a) Datatype:**
    - ✓ **The data type can be int,float,char,double,void.**
    - ✓ This is the data type of the value that the function is expected to return to calling function.
  - **b) <u>functionname</u>**:
    - ✓ The name of the function.
    - ✓ It should be a valid identifier.
  - **c) <u>parameters</u>**
    - ✓ The parameters are list of variables enclosed within parenthesis.
    - ✓ The list of variables should be separated by comma.

Ex: **void add(int a, int b)**

- In the above example the return type of the function is **void**
- the name of the function is **add** and
- The parameters are **'a' and 'b'** of type integer.

ii. Function body

- The function body consists of the set of instructions enclosed between **{ and } .**
- The function body consists of following three elements:
  a) **declaration part:** variables used in function body.
  b) **executable part:** set of Statements or instructions to do specific activity.
  c) **return :** It is a keyword,it is used to **return control back to calling function.**

If a function is not returning value then statement is:

return;

If a function is returning value then statement is:

return value;

2. Function Declaration

- The process of declaring the function before they are used is called as function declaration or function prototype.
  - function declaration Consists of the data type of function, name of the function and parameter list ending with semicolon.

Function Declaration Syntax
datatypefunctionname(type p1,type p2,……type pn); **Example** **int add(int a, int b);** **void add(int a, int b);**

3. Function Call:

- The method of calling a function to achieve a specific task is called as function call.
- A function call is defined as **function name followed by semicolon.**
- A function call is nothing but invoking a function at the required place in the program to achieve a specific task.

Ex:

```
void main()
{
        add( ); // function call without parameter
}
```

MODULE 4

7.a Define structure. Explain how structure members are accessed using dot(.) operator with example. 5 Marks

Answer:

Definition: A *structure* is defined as a collection of data of same/different data types. All data items thus grouped are logically related and can be accessed using variables. Thus, structure can also be defined as a group of variables of same or different data types. The variables that are used to store the data are called *members of the structure* or *fields of the structure*. In C, the structure is identified by the keyword **struct**

Syntax:	Example:

struct tag_name
{
data type member1;
data type member1;
data type member1;
---------- 	
};
	struct student
	{
	char name[10];
	int usn;;
	float marks;
	};

- struct is a keyword which informs the compiler that a structure is being defined.
- tagname name of a structure.
- members of structure are member1,member2
  datatype can be int,float,char,double.

Accessing structures

- The members of a structure can be accessed by specifying the variable followed by dot operator followed by the name of the member.
- For example,
  consider the structure definition and initialization along with memory representation as shown below:

```
struct student
{
        char   name[20];
        int    usn;
        float  marks;
```

} s1;
struct student s1 = {"aditi",002,40};

By specifying
Variblename . membername
Example
S1.name
S1.usn
S1.marks

---

7 b. Show how structure variables are passed as a parameter to a function with example. **5 marks**

Answer:

1. **By passing individual members of a structure to a function**
   - A function can be called by passing individual members of a structure as actual parameters.
   - Consider the below program which demonstrates the multiplication of fraction:

Program	User defined function to multiply
#include<stdio.h> **typedef struct** **{** **int n;** **int d;** **}FRACTION;** **void main()** **{** **FRACTION a,b,c;** **printf("enter fraction 1");** **scanf("%d%d", &a.n, &a.d);** **printf("enter fraction 2");** **scanf("%d%d", &b.n, &b.d);** **/* function call with numerator of a and b*/** **c.n=multiply(a.n , b.n);** **/*same as above function call with denominator of a and b*/** **c.d=multiply(b.d, b.d);** **printf("fraction c is %d / %d", c.n, c.d);** }	***Continued….*** **int multiply(int x,int y)** **{** **return x*y;** **}**

Continued to next user defined function….	

7 c. Write a C program to maintain a record of n student details using an array of structures with four fields (Roll number, Name, Marks, and Grade). Assume appropriate data type for each field. Print the marks of the student, given the student name as input.

6 Marks

Answer:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
Typedef struct
{       int rollno,marks;
        char name[20];
        char grade[2];
}STUDENT;
void main()
{               int i,n;
                STUDENT s[10];
                char key[20];

                printf("enter the number\n");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                {
                        printf("enter student details\n”);
                        printf("enter the roll number:\n");
                        scanf("%d",&s[i].rollno);
                        printf("enter the student name:\n");
                        scanf("%s",&s[i].name);
                        printf("enter the marks:\n");
                        scanf("%d",&s[i].marks);
                        printf("enter the grade:\n");
                        scanf("%s",&s[i].grade);
```

```
        }
    printf("enter the key student name:\n");
    scanf("%s",key);
    for(i=0;i<n;i++)
    {
        if(strcmp(s[i].name,key)==0)
        {   printf("\n marks of the student is %d \n",s[i].marks);
            exit(0);
        }
    }
    printf("student name NOT FOUND\n");
}
```

8 a. Define file. Explain the different modes of file with suitable example. 8 Marks

Answer:

File: A file is defined as a collection of data stored on the secondary device such as hard disk. **FILE** is type defined structure and it is defined in a header file "stdio.h". FILE is a derived data type.

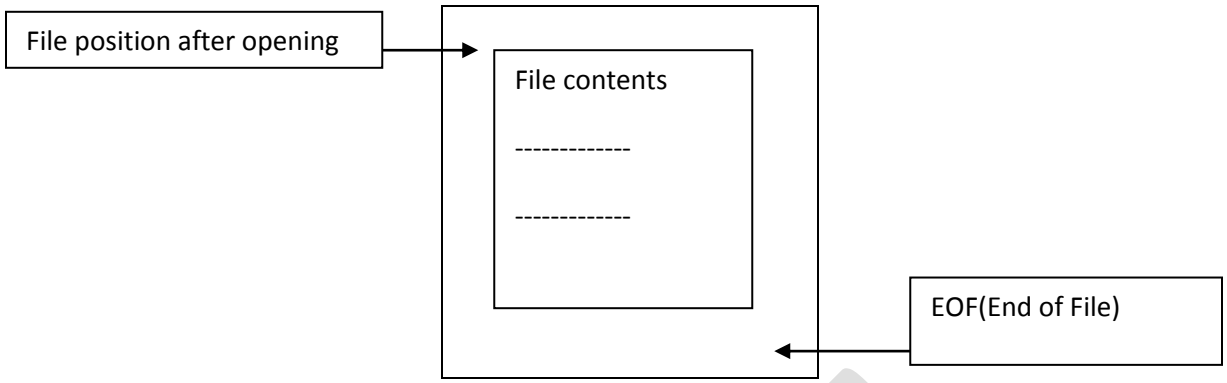
Modes of File

The various mode in which a file can be opened/created are:

Mode	Meaning
"r"	opens a text file for reading. The file must exist.
"w"	creates an text file for writing.
"a"	Append to a text file.

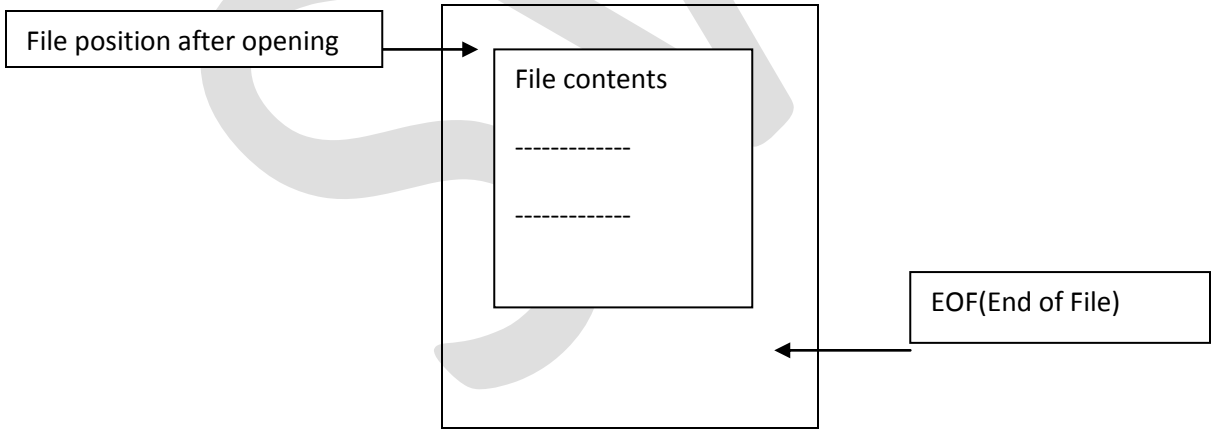
1.Read Mode("r")

- This mode is used for opening an existing file to perform read operation.
- The various features of this mode are
  - Used only for text file
  - If the file does not exist, an error is returned
  - The contents of the file are not lost
  - The file pointer points to beginning of the file.



2.Write Mode("w")

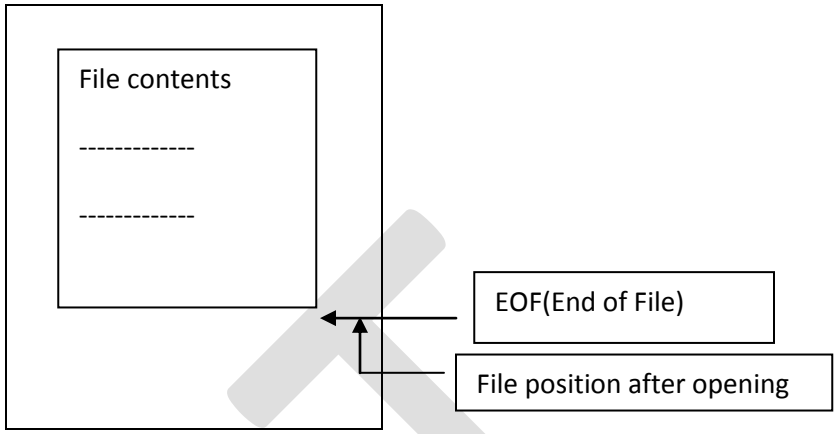
- This mode is used to create a file for writing.
- The various features of this mode are
  - If file does not exist, a new file is created.
  - If a file already exists with the same name, the contents of the file are erased and the file is considered as a new empty file.
  - The file pointer points to the beginning of the file.



3.Append Mode("a")

- This mode is used to insert the data at the end of the existing file.
- The various features of this mode are
- Used only for text file.
- If the file already exist, the file pointer points to end of the file.

- If the file is does not exist. A new file is created.
- Existing data cannot be modified.



Examples

Read Mode	Write Mode

#include<stdio.h>
 	
FILE *fp // File Pointer
fp=fopen("civil.txt","r"); //opening file civil in read mode
 	
if (fp == NULL) //file does not exist
{
printf("Error in opening the file\n");//error message
exit(0); //terminate the program
}
 	
fclose(fp); // close the file civil.txt
	#include<stdio.h>
	
	FILE *fp // File Pointer
	fp=fopen("ecb.txt","w"); //opening file ecb in write mode
	
	if (fp == NULL) //file does not exist
	{
	printf("Error in opening the file\n");//error message
	exit(0); //terminate the program
	}
	
	fclose(fp); // close the file ecb.txt

Append Mode

#include<stdio.h>
 	
FILE *fp // File Pointer
fp=fopen("ecb.txt","a"); //opening file ecb in write mode

```
if (fp == NULL)          //file does not exist
{
printf("Error in opening the file\n");//error message
exit(0);                 //terminate the program
}

fclose(fp); // close the file ecb.txt
```

8 b. Explain the following functions with example.
i)fopen() ii)fprintf() iii)fscanf() iv)fgets() 8 Marks
Answer:
i)fopen()

The file should be opened before reading a file or before writing into a file.

Syntax:

```
FILE *fp;
……..
……..
fp = fopen(filename, mode)
```

Where,

fp is a feile pointer

fopen() function to open file

mode is **"r","w","a"**

- **fopen()** function will return the starting address of opened file and it is stored in file pointer.
- If file is not opened then fopen function returns NULL

```
if (fp == NULL)
{
        printf("Error in opening the file\n");
        exit(0);

}
```

Example:

```
#include<stdio.h>
```

```
FILE *fp                        // File Pointer

fp=fopen("civil.txt","r");   //opening file civil in read mode

if (fp == NULL)          //file does not exist
{
printf("Error in opening the file\n");//error message
exit(0);                 //terminate the program

}

fclose(fp); // close the file civil.txt
```

ii) **fscanf():**

The function **fscanf** is used to get data from the file and store it in memory.

Syntax:

fscanf(fp, "format string", address list);

where,

"fp" is a file pointer.It points to a file from where data is read.

"format String": The data is read from file and is stored in variable s specified in the list ,will take the values from the specified pointer fp by using the specification provided in format sting.

"**address list":**address list of variables

Note:fsanf() returns number of items successfully read by fscanf function.

Example:

FILE *fp
fp=fopen("name.txt","r");
fscanf("fp,"%s",name);
	FILE *fp
	fp=fopen("marks.txt","r");
	fscanf("fp,"%d%d%d",&m1,&m2,&m3);

iii) **fprintf():**

The function **fprintf** is used to write data into the file.

Syntax:

fprintf(fp, "format string", variable list);

where,

"fp" is a file pointer.It points to a file where data to be print.

"format String": group of format specifiers.

"**address list":**list of variables to be written into file

Note:fprinf() returns number of items successfully written by fprintf function.

Example:

FILE *fp
fp=fopen("name.txt","w");
fscanf("fp,"%s",name);
	FILE *fp
	fp=fopen("marks.txt","w");
	fscanf("fp,"%d%d%d",m1,m2,m3);

iv) **fgets()**

fgets() is used to read a string from file and store in memory.

Syntax:

ptr=fgets(str,n,fp);

where

fp ->file pointer which points to the file to be read

str ->string variable where read string will be stored

n ->number of characters to be read from file

ptr->If the operation is successful, it returns a pointer to the string read in.

Otherwise it returns NULL.

The returned value is copied into ptr.

Example:

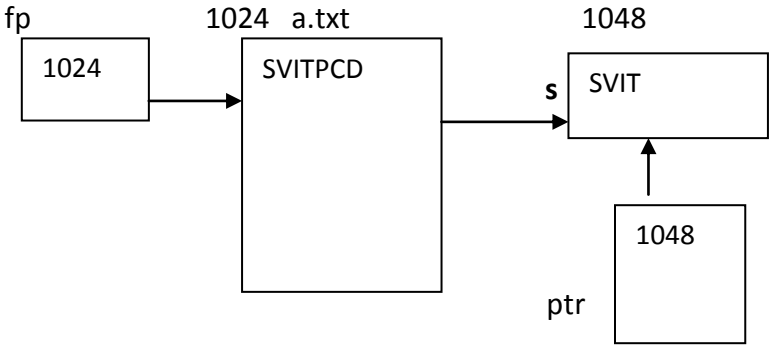
```
FILE *fp;

char s[10];

char *ptr;

fp=fopen("a.txt","r");


if(fp==NULL)
```



```
{
printf("file cnnnot be opened);
exit(0);
}
ptr=fgets(s,4,fp);
fclose(fp);
```

MODULE 5

9a) What is pointer? Explain how pointer variable is declared and initialized. 5 Marks

Answer:

Definition:

- ***A pointer is a variable which contains the address of another variable.*** A pointer is a derived data type. This is the one which stores the address of data in memory.

Declaring of a pointer variable

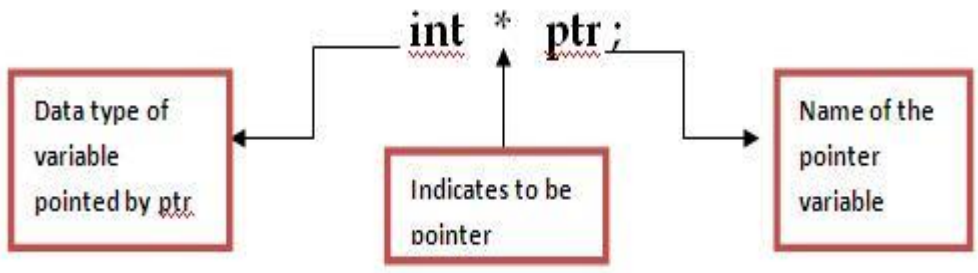
General syntax :

```
data_type *pointer_name;
```

where,

- The asterisk (*) indicates that the variable pointer_name is a pointer variable.
- Pointer_name is a identifier.
- pointer_name needs a memory location.
- pointer_name points to a variable of type data_type which may be int, float, double etc..

Example:



where

- ptr is not an integer variable but ptr can hold the address of the integer variable
- i.e. it is a **pointer to an integer variable'** ,but the declaration of pointer variable does not make them point to any location .
- We can also declare the pointer variables of any other data types .

Example:

double * dptr;

char * ch;

float * fptr;

Initialization of pointers

- Initializing a pointer variable is a important thing, it is done as follows:

Step 1: Declare a data variable

Step 2:Declare a Pointer variable

Step 3:Assign address of data variable to pointer variable using & operator and assignment operator.

Example:

```
int  x;
int  *p
p = &x;
```

---

9 b. Explain any two preprocessor directives in C with Example. 6 Marks

Answer:

Types of preprocessor

1. **Symbolic names**
2. **Macros**
3. **File inclusion**
4. **Conditional compilation**

Symbolic names/symbolic constants

- These are the names which are used to define/give the names for a constant values.

- The ***# define*** directive is used to define the names for a constant value.
- Since it is used to define constant, it is also termed as **defined constant.**

 # define name value

Where

- **# define is a directive**
- **name is symbolic name**

Example # define MAX 30

 # define PI 3.14

File inclusion

- It is a include preprocessor directives
- #include specifies to insert the content of the specified files to the program.
- This directive includes a file in to the code.
- It has two possible forms

#include <file>

Or

#include"file"

Example:

```
#include<stdio.h>
void main()
{
        printf("ECE,EEE,CIV\n");
}
```

---

9 c. Write a C program to swap two numbers using pointer Concept. 5 Marks

Answer

```
#include<stdio.h>
void swap(int m,int n);
void main()
{
```

```
        int a,b;
        printf("enter values for a and b:");
        scanf("%d %d",&a,&b);
        printf("the values before swapping are a=%d b=%d \n",a,b);
        swap(&a,&b);
        printf("the values after swapping are a=%d b=%d \n",a,b);
 }
 void swap(int *m, int *n)
{
        int temp;
        temp=*m;
        *m=*n;
        *n=temp;
 }
```

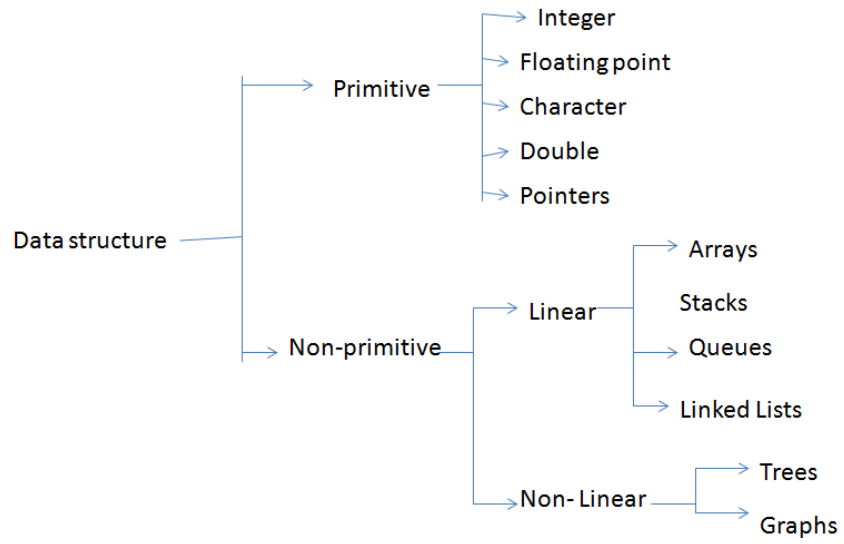
---

10 a. What are primitive and Non primitive data types? Explain. 5 Marks

Ans:

Types of data structure:

A data structure can be broadly classified as shown below:



(i) Primitive data structure

The data structures, typically those data structure that are directly operated upon by machine level instructions i.e. the fundamental data types such as int, float, double in case of 'c' are known as primitive data structures.

Data Types and Sizes:

There are only a few basic/fundamental data types available in C:

char- a single byte, capable of holding one character in the local character set.

int - an integer, typically reflecting the natural size of integers on the host machine.

float- single-precision floating point.

double- double-precision floating point.

void- does not return any value.

(ii) Non-primitive data structure

The data structures, which are not primitive, are called non-primitive data structures.

The non-primitive data types cannot be manipulated by machine instructions.

- **Linear Data Structures:-**

A list, which shows the relationship of adjacency between elements, is said to be linear data structure. The most, simplest linear data structure is a 1-D array, but because of its deficiency, list is frequently used for different kinds of data.

Stack, Queues and linked list are linear data structures.

- Stack: A stack is a linear data structure in which an element may be inserted or deleted only at one end called the top end of the stack .A stack follows the principle of last-in-first-out (LIFO) system.
- Queues: Queue is a linear data structure in which insertion can take place at only one end called **rear end** and deletion can take place at other end called **top end**. The front and rear are two terms used to represent the two ends of the list when it is implemented as queue. Queue is also called First In First Out (FIFO) system since the first element in queue will be the first element out of the queue.
- Linked List: A linked list is a data structure which is collection of zero or more nodes with each node consisting of two field's data and link. Data field consists the information of be processed. Link field contains the address of the next node.

- **Non-linear data structure:-**

A list, which doesn't show the relationship of adjacency between elements, is said to be non-linear data structure.

Trees and graph are non linear data structures.

---

10 b. list the applications of stack and queue data structure. 5 Marks

Answer

- **Application of Stacks**
  - There are two applications of stacks.

- Recursion: A recursion function is a function which calls itself. The problems like towers of Hanoi, tree manipulation problems etc can be solved using recursion.
- Arithmetic/Evaluation of Expression: The conversion of an expression in the form of either postfix or prefix can be easily evaluated.
- Conversions of expressions: Evaluation of infix expressions will be very difficult and hence it needs to be converted to prefix or postfix expression which needs the use of stack.

Applications of Queues:

- Number of print jobs waiting in a queue, when we use network printer. The print jobs are stored in the order in which they arrive. Here the job which is at the front of the queue, gets the services of the network printer.
- Call center phone system will use a queue to hold people in line until a service representative is free.
- Buffers on MP3 players and portable CD player, iPod playlist are all implemented using the concept of a queue.
- Movie ticket counter system, it maintains a queue to take tickets based on first come first serve means who is standing first in a queue, that person will get the ticket first than second person, and so on.

---

10 c. Write a C program to find sum and mean of all elements in an array using pointer.
6 Marks

Answer:

```
#include<stdio.h>
#include<math.h>
void main()
{
   int n,i;
   float a[20],sum,mean, var, sd;
   printf("\n enter size of array");
   scanf("%d",&n);
   printf("\n Enter array elements");
   for(i=0;i<n;i++)
  scanf("%f",(a+i));
   sum=0;
   for(i=0;i<n;i++)
         sum=sum+ *(a+i);
         printf("\n sum=%f",sum);
   mean=sum/n;
```

```
        sum=0;
  for(i=0;i<n;i++)
        sum=sum+(*(a+i)-mean)*(*(a+i)-mean);
  var=sum/n;
  sd=sqrt(var);
  printf("\n Mean=%f \n Variance=%f \n StdDev=%f",mean,var,sd);
 }
```