PROGRAMMING IN C AND DATA STRUCTURES

Dec 2015/ Jan 2016

Module 1

1a. What is Variable? Explain the rules for constructing variables in C language. Give Examples for Valid and invalid variables.

Answer:

- Variables are the names given to computer memory location.
- These refer to names of variables, functions and arrays.
- These are user defined names and do not have fixed meaning.

Rules for identifiers/variables

- First character must be an alphabet or an underscore(_)
- First character is followed by any number of letters or digits.
- Only first 31 characters are significant
- Keywords cannot be used
- Must not contain blank space/white space
- Must contain only alphabets, digits and one special symbol underscore

Examples: **Vaild:** svit, svit06, _svitu, svit_06, svit_06_be, _ _svit

Invalid: 06svit, int, svit 06, svit@06

1b. Evaluate the following expressions:

i) 100%20<=20-5+100%10-20==5>=1!=20

ii) a+=b*=c-=5 where a=3, b=5 and c=8

Answer:

Refer class notes.

1c. Write a C program to find the area and perimeter of rectangle.

Answer:

```
#include<stdio.h>
void main()
{
        float length,breadth,area,perimeter;
        printf("\nEnter the length of the rectangle:");
```

```
        scanf("%f",&length);
          printf("\nEnter the breadth of the rectangle:");
        scanf("%f",&breadth);
          area = length * breadth
        perimeter = 2*(length + breadth);
          printf("\nArea of the rectangle is:%f",area);
        printf("\n\nPerimeter of the rectangle is:%f",perimeter);
}
```

2a. Write a C Program which takes input as p, t, r. Compute the simple interest and displays the result.

Answer:

```
#include<stdio.h>
void main()
{
        float p,t,r,si;
        clrscr();
        printf("\nEnter values of p, t and r:");
        scanf("%f%f%f",&p,&t,&r);
        si=p*t*r/100;
        printf("\nSimple interest is:%f",si);
}
```

2b. Convert the following mathematical expression into C expressions:

Answer:

i) $\frac{x}{b+c} + \frac{y}{b-c}$

➔ **(x/(b+c))+(y/(b-c))**

ii) $a + \frac{b(ad+c)}{b-a} - \frac{c}{d}$

→ **a+(b*(a*d+c))/(b-a)- (c/d)**

2c. What is the value of "x" in the following code segments ? Justify your answers.

Answer:

i) int a,b;
float x;
a=4;
b=5;
x=b/a;
	ii) int a,b;
	float x;
	a=4;
	b=5;
	x=(float)b/a;

The Value of x is 1.000000. The reason is both a and b are int variables and the division will give integer value itself and there is no type conversions involved	The Value of x is 1.250000. The reason is both a and b are int variables and the division will give integer value itself and there is explicit type conversions involved converting it into floating value.

Module 2

3a. Explain the syntax of do-while statement. Write a C program to find the factorial of a number using do-while, where the number n is entered by user.

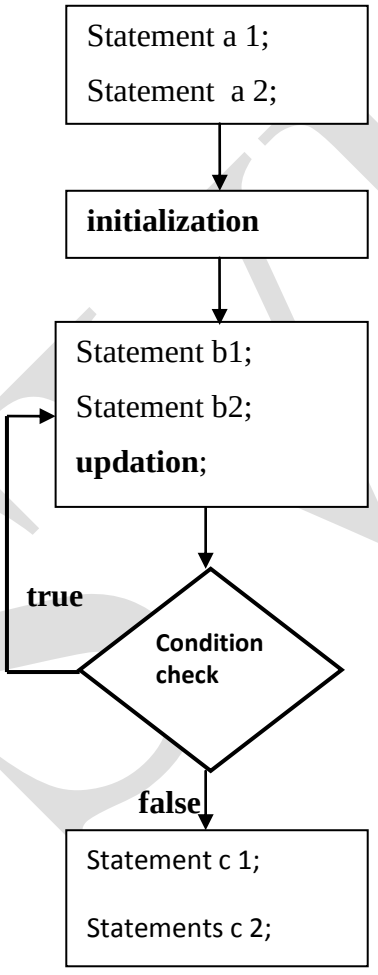
Answer:

The *do while* loop

- **Definition:** A do while loop is a looping statement which are used to execute a set of statements repeatedly.
- Since a set of statements have to be executed until a condition is true, it is also called ***condition controlled loop***
- The body of the loop is executed first and then the expression is evaluated to true or false. hence it is called as ***post test loop/bottom testing loop***
- As the expression is evaluated at the end ,the do while loop is also called as ***exit controlled loop.***
- ***Syntax:***
  do and while are keywords.(also called reserved words).
- **Working:**
  - Statements a1 to an are executed in sequence.
  - First Statements inside the body of the loop is executed one after the other.

- Then the expression/condition is evaluated ,if it is ***true,*** body of the do while loop is executed.
- Thus the body of loop is repeatedly executed as long as the condition is ***true***.
- Once the condition/expression is evaluated to ***false*** ,the control comes out of loop and statements c1 to cn are executed.
- The moment expression is evaluated to ***false***, the control comes out of the ***do while*** loop.
- **The body of the loop is executed at least once before the condition is checked ,** hence it is also called as ***post test loop*** .

Syntax	Flowchart	Example

Statement a 1;
Statement a2;
initialization;
do
{
Statement b1;
Statement b2;
updation;
} ***while***(**Condition check);**
Statement c 1;
Statements c 2;
		#include<stdio.h>
		void main( )
		{
		int i;
		i=1;
		do
		{
		printf("INDIA\n");
		i++;
		} ***while(i<=5);***
		}
		
		Output:
		INDIA
		INDIA
		INDIA
		INDIA
		INDIA

Program:

#include<stdio.h>

```
void main()
{
  int n, fact;
  clrscr();
  printf("\nEnter values of n:");
  scanf("%d",&n);
  fact=1;
  do{
       fact=fact*n;
       n--;
     }while(n<0);
  printf("\n fact is :%d",fact);
}
```

3b. What is two way selection statement? Explain if, if-else and cascaded if else with examples.

Answer:

A selection statement where a block of code is executed when the condition is true and another block of code if the condition is false is termed as two way selection statement.

if statement:
Synatx:
Statement 1;
Statement 2;
if(condition)
{
Statement 3;
Statement 4;
}
Statement 5;
	Example :
	#include<stdio.h>
	void main()
	{
	int n;
	printf(" Enter the number\n")
	scanf("%d",&n);
	if(n%2==0)
	{
	printf("Even no");
	}
	}

Explanation

- The keyword if must be followed by an expression and expression must be enclosed within parentheses.
- First statement1 is executed followed by statement2.
- Condition is checked
  - if false control directly jumps to statement5 ignoring statement3 amd 4.

- if true control goes to statement3 , statement4 and automatically goes to statement5.

if else statement:
Synatx:
Statement 1;
Statement 2;
if(condition)
{
Statement 3;
Statement 4;
}
else
{
Statement 5;
Statement 6;
}
Statement 7;
	Example :
	#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");
	}
	}

Explanation

- The keyword if and else must be followed by an expression and expression must be enclosed within parentheses.
- First statement1 is executed followed by statement2.
- Condition is checked
  - if true control goes to statement3 , statement4 and automatically goes to statement7.
  - else control goes to statement5 , statement6 and automatically goes to statement7.

Cascaded if-else statement:
Synatx:
Statement 1;
Statement 2;
if(condition 1)
{
Statement 3;
}
else if(condition 2)
{
Statement 4;
}
else if(condition 3)
{
Statement 5;
}
else
{
Statement 6;
}
	Example :
	#include<stdio.h>
	void main()
	{
	inta,b,c;
	clrscr();
	printf(" Enter the 3 numbers\n")
	scanf("%d%d%d",&a,&b,&c);
	if(a>b && a>c)
	{
	printf("a largest");
	}
	else if(b>a && b>c)
	{
	printf("b largest");
	}
	else
	{
	printf("c largest");
	}

Statement 7;	}

Explanation

- The keyword if and else must be followed by an expression and expression must be enclosed within parentheses.
- First statement1 is executed followed by statement2.
- Condition 1 is checked
  - if true control goes to Statement3 and automatically goes to Statement7.
- false Condition 2 is checked
  - if true control goes to Statement4 and automatically goes to Statement7.
- false Condition 3 is checked
  - if true control goes to Statement5 and automatically goes to Statement7
- false Statement6 and automatically goes to S4.

4a. Write a C program that takes from the user an arithmetic operator(‘+’, ‘-’, ‘*’ or ‘/’) and two operands. Perform the corresponding arithmetic operation on the operands using switch statement.

Answer:

```
#include<stdio.h>
void main()
{
    int a,b,res;
    char op;
        printf(“ Enter your choice\n”);
        scanf(“%c”,&op);
        printf(“ Enter two operands\n”);
        scanf(“%d%d”,&a,&b,);
        switch(op)
    {
                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");
    }
}
```

4b. What is an array? How to declare and Initialize the two dimensional arrays?

Answer:

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.

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**

Module 3

5a. What is a function? Write a C program to find the cube of a number using function.

Answer:

A function is a self contained block of code that performs a particular task.

Or

A function as series of instructions or group of statements with one specific purpose.

Program:

```
#include<stdio.h>
int cube(int a);
void main()
{
    int n,res;
    printf("enter values for n :");
    scanf("%d",&n);
```

```
        res=cube(n); / * Function Call */
        printf("cube is:%d",res);
}
int cube(int a)                         /* Function Header */
{
        return (a*a*a);
}
```

5b. Write a C program to check a number is a prime or not using recursion.

Answer

```
#include<stdio.h>
int isPrime(int,int);
void main()
{
   int num,prime;
   printf("Enter a positive number: ");
   scanf("%d",&num);
   prime = isPrime(num,num/2);
   if(prime==1)
      printf("%d is a prime number",num);
     else
    printf("%d is not a prime number",num);
}
int isPrime(int num,int i)
{
   if(i==1)
   {
     return 1;
   }
   else
   {
         if(num%i==0)
```

```
            return 0;
        else
            isPrime(num,i-1);
    }
}
```

5c. Write a Program to replace each constant in a string with the next one except letter 'z', 'Z' and 'a', 'A'. thus the sting "Programming in c is fun" should be modified as "Qsphsannojoh jo D jt gvo".

Answer:

```
#include<stdio.h>
void main()
{
    int c,i;
    char str[100],ch;
    printf("enter the string :");
    fflush(stdin);
    gets(str);
    for(i=0;str[i]!='\0';i++)
    {
        ch=str[i];
        if(ch=='a'|| ch=='A'|| ch=='z'|| ch=='Z'|| ch==' ')
          continue;
        else
        {
          c=ch;
          c++;
          str[i]=c;
        }
    }
    printf("\n the final string after modification ");
    puts(str);
```

```
}
```

6a. Write a C program to sort the elements by passing array as function argument.

Answer:

```
#include<stdio.h>
void sort(int a[],int n);
void main()
{
   int n, a[50],i;
   printf("Enter the size of array ");
   scanf("%d",&n);
   printf("Enter the array elements ");
   for(i=0;i<n;i++)
     scanf("%d",&a[i]);
    printf("\n Elements before sorting ");
         for(i=0;i<n;i++)
     printf("\n%d",a[i]);
    sort(a,n);
    printf("\n Elements aftersorting ");
    for(i=0;i<n;i++)
     printf("\n%d",a[i]);
}

void sort(int a[],int n)
{
   int i,j;
   for(i=0;i<n;i++)
   {
     for(j=0;j<n-i-1;j++)
     {
         if(a[j]>a[j+1])
         {
```

```
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                    }
                }
        }
}
```

6b. Write a C program to concatenate two strings without using built - in function strcat( ).

Answer:

```
#include<stdio.h>
void main()
{
        char str1[100], str2[100];
        int i,j;
        printf("\n enter the first string\n");
        fflush(stdin);
        gets(str1);
        printf("\n enter the second string\n");
        fflush(stdin);
        gets(str2);
        i=0;
        while(str1[i]!='\0')
                i++;
        j=0;
        while(str2[j]!='\0')
        {
                str1[i]=str2[j];
                i++;
                j++;
        }
    str1[i]='\0'
```

```
        printf("\n the first string is :");
        puts(str1);
        printf("\n the second string is:");
        puts(str2);
}
```

MODULE 4

7a.what is structure? Explain the c syntax of structure declaration with example. [5M]

Answer:

Structures

- **Definition:** A *structure* is defined as a collection of variables of same data type or dissimilar datatype grouped together under a single name.

Syntax	Example
struct tagname **{** **datatype member1;** **datatype member2;** **datatype member3;** **}**	***struct*** *student* **{** **char name[10];** **int usn**; **float marks;** **};**

where

struct is a keyword which informs the compiler that a structure is being defined

tagname: name of the structure

member1,member2: members of structure:

type1,type 2 : int,float,char,double

Structure Declaration

As variables are declared ,structure are also declared before they are used:

Three ways of declaring structure are:

- Tagged structure
- Structure without tag

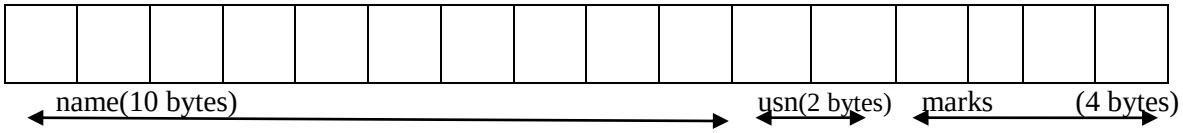
- Type defined structures

1.Tagged structure	
syntax **struct tag_name** **{** **data type member1;** **data type member2;** **----------** **----------** **};**	**Example** **struct student** **{** **char name[20];** **int usn;** **float marks;** **};**
Declaration of structure variables	
struct tagname v1,v2,v3…vn;	**struct student s1,s2,s3;**

2.structure without tagname	
syntax **struct** **{** **data type member1;** **data type member2;** **;** **----------** **----------** **}v1,v2,v3;**	**Example** **struct** **{** **char name[20];** **int usn;** **float marks;** **}s1,s2;**

3.Type defined structure	
syntax **typedef struct** **{** **data type member1;** **data type member2;** **----------** **----------** **}TYPE_ID;**	**Example** **typedef struct** **{** **char name[20];** **int usn;** **float marks;** **}STUDENT;**
Declaring structure variables	
TYPE_ID v1,v2,v3…vn;	**STUDENT s1,s2,s3;**

Memory Allocation for structure variable s1:



memory allocated for a structure variable s1=memory allotted for name+usn+marks

10+2+4

16 bytes.

7b. write a c program to pass structure variable as function argument. [7M]

Answer:

```
#include<stdio.h>
#include<conio.h>
typedef struct
{
        int  n;
        int d;
}FRACTION;
int multiply(int x,int y)
{
        return x*y;
}
Void main()
{
        FRACTION a,b,c;
        Printf("enter  fraction1 in the form x/y");
        Scanf("%d%d",&a.n,&a.d);
        Printf("enter fraction2 in the form x/y");
        Scanf("%d%d",&b.n,&b.d);
        c.n=multiply(a.n,b.n);
        c.d=multiply(b.n,b.d);
        printf("fraction c is %d/%d",c.n,c.d);
```

}

7c.Explain fopen and fclose function [4M]

Answer:

- **Definition: A** file is defined as collection of data stored on the secondary device such as hard disk.

fopen()

- The file should be opened before reading a file or before writing into a file.
- The syntax to open a file is:

  FILE *fp;

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

- fp is a file pointer.
- fopen is a function to open a file.
- Mode can be r,w, or 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 file\n");
        exit(0);
}
```

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.

fclose()

- Closing a file: fclose function()
- When we no longer need a file ,we should close the file .this is the last option to be performed on a file.
- A file can be closed using fclose() function.
- If a file is closed successfully,0 is returned, otherwise EOF is returned.

Syntax:

```
fclose(fp);
```

8a.write a c program to store and print name,USN,subject and IA marks of student using structure. [8M]

Answer:

```
#include <stdio.h>
struct student
{
   char name[50];
   int usn;
   char sub1[10];
   float m1,m2,m3;
} s[10];

void main()
{
   int i,n;
   printf("\n enter num of students:");
   scanf("%d",&n);
   printf("Enter information of students:\n");
   for(i=0; i<n; i++)
   {
      printf("\n enter usn");
      scanf("%d",&s[i].usn);
           printf("Enter name: ");
```

```
    scanf("%s",s[i].name);
        printf("Enter subject name: ");
    scanf("%s",s[i].sub1);
    printf("Enter marks of three internals: ");
    scanf("%f%f%f",&s[i].m1,&s[i].m2,&s[i].m3);
    printf("\n");
}
printf("Displaying Information:\n\n");
printf("\nName\tusn\tsub\tm1\tm2\tm3\n");
for(i=0; i<n; i++)
{
        printf("%s\t%d\t%s\t%f\t%f\t%f\n",s[i].name,s[i].usn,
        s[i].sub1,s[i].m1,s[i].m2,s[i].m3);
    printf("\n");
}
```

8b. Explain fgets ,fgetc and fputs and fputc function with syntax. [8M]

Answer:

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

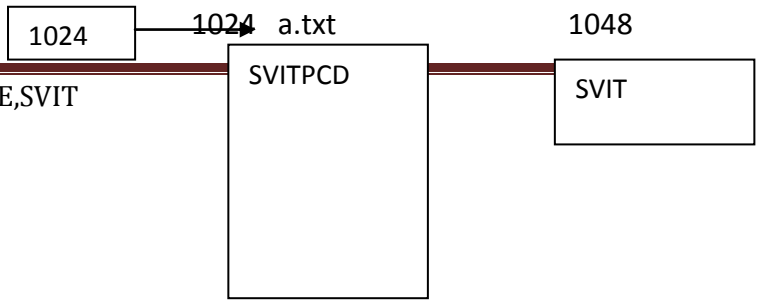
ptr->If the operation is successful, it returns a pointer to the string read in.

Otherwise it returns NULL.

The returned value is copied into ptr.

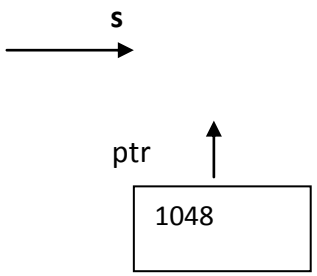
Example:

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);
```



fputs()

fputs() is used to write a string into file.

Syntax:

fputs(str,fp);

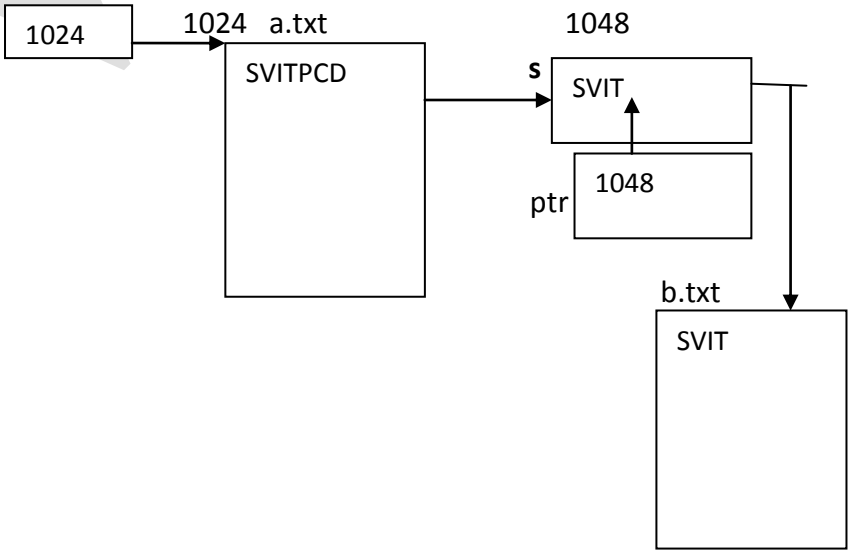
where

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

str ->string variable where read string will be stored

Example:

```
FILE *fp,*fp1;
char s[10];
char *ptr;
fp=fopen("a.txt","r");
fp1=fopen("b.txt","w");
if(fp==NULL)
{
printf("file cnnnot be opened);
exit(0);
}
ptr=fgets(s,4,fp);
fputs(s,fp1);
fclose(fp);
fclose(fp1);
```



3.fgetc()

fgetc() function is used to read a character from file and store it in memory.

Syntax:

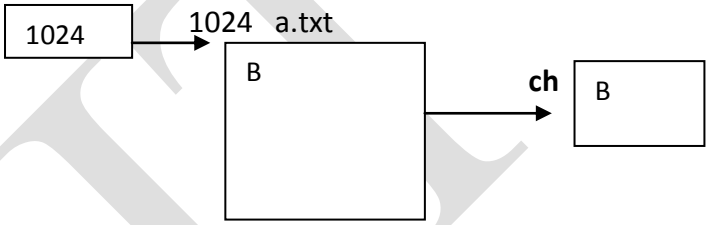
ch=fgetc(fp);

Example 1:

```
FILE *fp;
fp=fopen("sec.txt","r");
ch=fgetc(fp);
```

Example 2:

```
FILE *fp;
char ch;
fp=fopen("a.txt","r");
ch=fgets(fp);
fclose(fp);
```



3.fputc()

fgetc() function is used to write a character into a file.

Syntax:

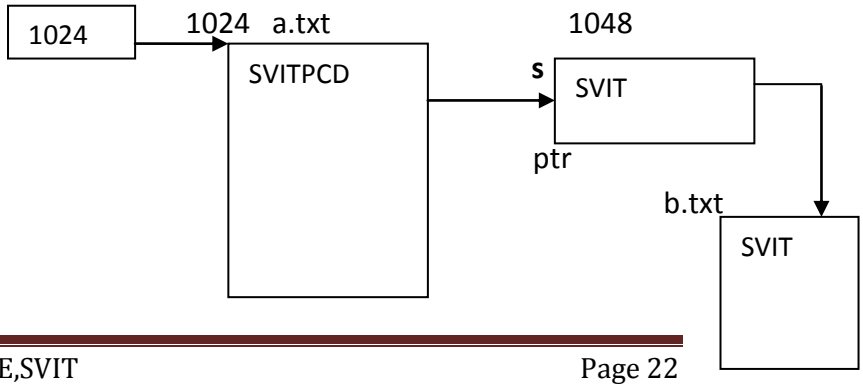
fputc(ch,fp);

Example 1:

```
FILE *fp;
fp=fopen("sec.txt","w");
fputc(ch,fp);
```

Example 2:

```
FILE *fp,*fp1;
char ch;
fp=fopen("a.txt","r");
fp1=fopen("b.txt","w");
ch=fges(fp);
fputc(ch,fp1);
fclose(fp);
```



```
fclose(fp1);
```

MODULE 5

9a. what is a pointer ?write a c program to find sum and mean of all elements in an array using pointers. [8M]

Answer:

- **A pointer is a variable which contains the address of another variable.**

Advantages of pointer

- Enables us to access a variable that is defined outside the function.
- Can be used to pass information back and forth between a function and its reference point.
- More efficient in handling data tables.
- Reduces the length and complexity of a program.
- Sometimes also increases the execution speed.

```
/* c program to find sum and mean of all elements in an array using pointers.*/
#include<stdio.h>
void main()
{
        int i,n,a[20],sum=0;
        printf("enter the n value\n");
        scanf("%d",&n);
        printf("enter the elements of the array\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",(a+i));
        }
        for(i=0;i<n;i++)
        {
```

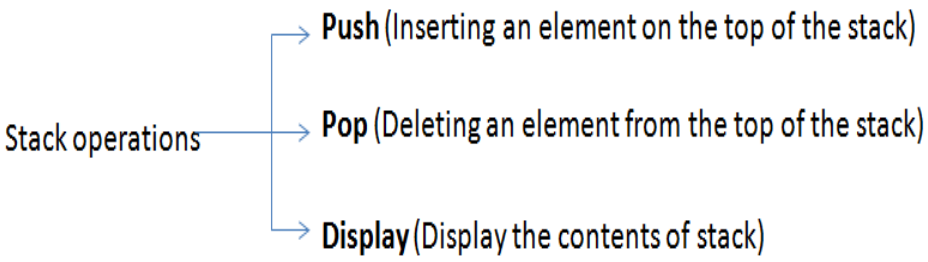
```
            sum=sum+*(a+i);
        }
        mean=sum/n;
        printf("the sum of all elements are %d",sum);
        printf("the mean of all elements are %d",mean);
}
```

9b. What is stack? Explain its operation with examples. [8M]

Answer:

Stacks

- 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.
- The various operations that can be performed on stacks are shown below:



- **Push: inserting the element to the stack**

Here we need to check the stack overflow condition which means whether the stack is full.

To insert an element two activities has to be done

1. Increment the top by 1
2. Insert the element to the stack at the position top.

- **Pop: deleting an element from the stack**

Here we need to check the stack underflow condition which means whether the stack is empty or not.

To delete the element we need to perform following operations.

1. Access the top element from the stack.
2. Decrement top by 1

- **Display: printing the elements of the stack**

Here we need to check the stack underflow condition which means whether the stack is empty or not.

If empty there will be no elements to display.

Display the elements from the 0^{th} position of stack till the stack top.

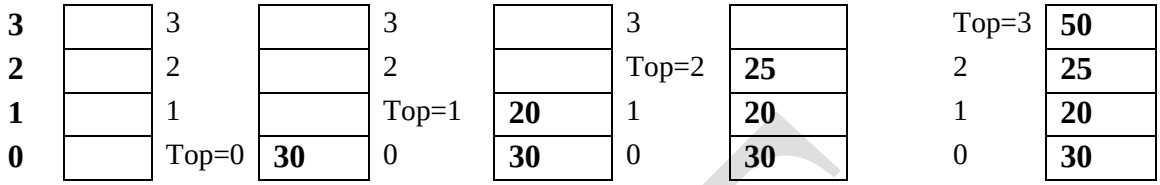


Figure stack push operation

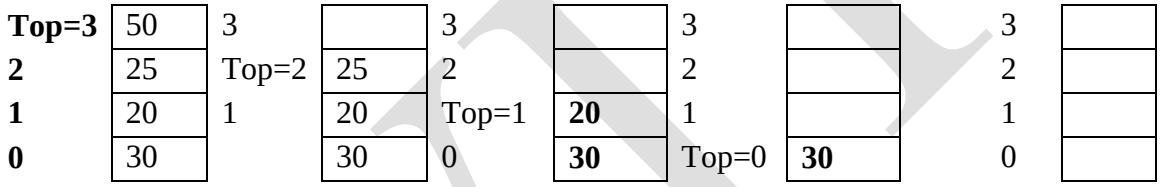


Figure stack pop operation

- **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.

10a.write a C programs to swap two numbers using call by address. [6M]

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;
 }
```

10b.Explain any five preprocessor directives in C. [5M]

Answer:

- The various types of preprocessor directives are:

1. Symbolic Names
2. Macros
3. File Inclusion
4. Conditional Compilation

symbolic constant:

These are the names which are used to define the names for a constant values. The "#define" directive is used to specify the constant hence it is also termed as defined constant.

Example:

```
#define max 30
```

where,

#define specifies the directive

max specifies the name of the constant

30 indicates the constant value assigned to max

consider the following program

```
#include<stdio.h>
#define pi 3.142
void main()
```

```
{
        printf("value is %f",pi);
}
```

Defining macros:

Macro is the name given to the group of statements. When a macro is included in the program , the program replaces the set of instructions defined. The #define directive is used to define macro.

Syntax:

#define <macro> (set of instructions)

for example

1. maximum of two variables can be written as

   #define max(a,b) ((a) > (b) ? (a):(b))

2. converting degrees into radians

   #define deg_to_rad(x) (x*m_pi/180.0)

Include preprocessor directive

#include specifies to insert the content of the specified files to the program.

- ➔ This directive includes a file into code.
- ➔ It has two possible forms:

  #include <file>

Conditional compilation

The #if, #elif, #else, #endif, #undef, #ifdef, #ifndef are some of the conditional compilation directives.

#if → it tests a compile time condition

#endif → specifies the end of #if directive

#else → specifies the alternative when #if directive tests fails.

```
#define DEBUG 1
#ifdef DEBUG
      printf(" HELLO");
#endif
```

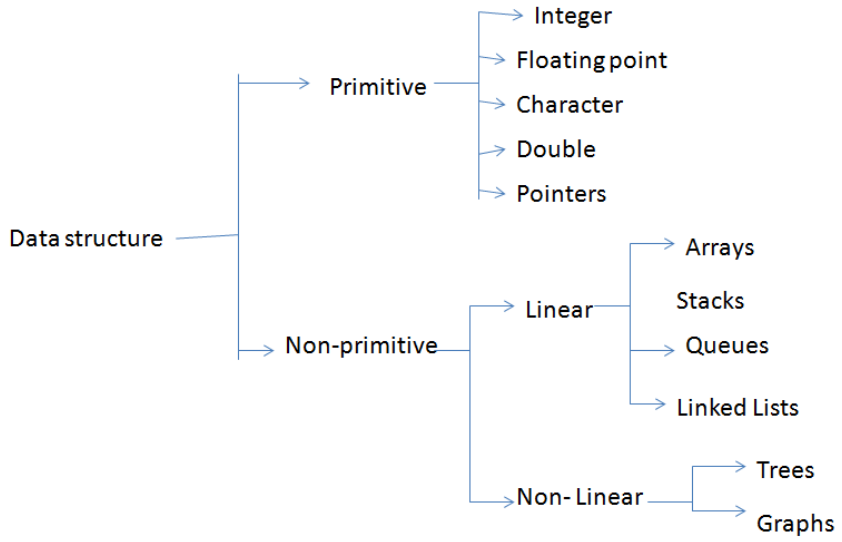
- ➔ Since the DEBUG is defined and set to 1 it includes the printf statement.

10c. What are primitive and non primitive data types? Explain with examples. [5M]

Answer:

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.