

Module-I

BCSE102L_STRUCTURED- AND-OBJECT-ORIENTED- PROGRAMMING

**Prepared by
Dr.M.Suguna
D1 Slot and C1-Slot**

STRUCTURED PROGRAMMING

Data:

A collection of facts, concepts, figures, observations, occurrences or instructions in a formalized manner.

Information:

The meaning that is currently assigned to data by means of the conventions applied to those data(i.e. processed data)

Record:

Collection of related fields.

Data type:

Set of elements that share common set of properties used to solve a program.

2. PROBLEM SOLVING

Problem solving is a process of transforming the description of a problem into the solution of that problem by using our knowledge of the problem domain and by relying on our ability to select and use appropriate problem-solving Strategies, Techniques and Tools.

Problem solving (with in the context of developing programs) refers to analyzing a problem with the intention of deriving a solution for the problem.

1.1 Steps Involved in Computer Programming

What is Program?

A program is a set of instructions written by a programmer. The program contains detailed instructions and complete procedures for performing the relevant tasks.

What is Programming?

Programming involves creativity, analytical thinking, knowledge in the language in which coding or the programming is done.

Steps Involved in Programming

1. Identify the problem
2. Analyzing the Problem
3. Algorithm design
4. Pseudocode conversion
5. Flowchart
6. Coding
7. Debugging
8. Testing
9. Final output
10. Documentation

1. Identify the problem

Every program aims at solving the problem for which it has been developed.

For example: An online banking program solves the problem of transferring money, tracking the transactions, etc., filing your taxes. A word processor solves the problem of document creation. Even a game solves the problem of keeping people entertained. The only way to achieve the goal is to aim in finding the problem involved in achieving it. A program is considered as useful only when it solves the problem. This is the first step to approach any solution.

1. Analyzing the Problem

Once the problem is being identified and target solution is being outlined. Analysis on the problem has to be carried out. Define whether the problem is a continuation of the existing solution or it is a new problem that has been identified. If the problem arises from an existing solution then its quiet easy for us to start our designing from that point. If its a new problem then we need to look in for the base of the problem and then move further for coding.

2. Algorithm Design

In this stage the stepwise procedure involved in arriving at the problem solution is identified and documented in simple English language. It is called as a strategy. These contents are said to be understandable by any person who knows simple English.

Example1: Adding two subject marks for one student

Algorithm:

Step 1. Read the two subject marks that have to be added

Step 2. Add the two subject marks and save it inside a variable.

Step 3. Write the total .

Example2: Adding ten subject marks for one student.

Algorithm:

Step 1. Initialize a sum variable as zero and subject variable as ten.

Step 2. While the number of subjects not equal to zero repeatedly do

Step 2.a. Read the corresponding subject mark that has to be added.

Step 2.b Compute the current sum by adding along with the previous sum value.

Step 3. Write the sum.

Example3: Adding ten subject marks for fifty students.

Algorithm:

Step 1. Initialize a sum variable as zero , subject variable as ten and student variable as fifty.

Step 2. While number of students is not equal to zero repeatedly do

Step 2.a. While the number of subjects not equal to zero repeatedly do

Step 2.a.1. Read the corresponding subject mark that has to be added.

Step 2.a.2. Compute the current sum by adding along with the previous sum value.

Step 2.b. Write the sum for each student.

As the stepwise procedure has been outlined they are easily converted into pseudocode. This helps in the coding the solution in the appropriate programming language.

Example1: Adding two subject marks for one student

Pseudocode:

```
Var mark1, mark2, sum;  
begin  
read(m1);  
read(m2);  
sum=m1+m2;  
write(sum);  
end
```

Example2: Adding ten subject marks for one student.

Pseudocode:

```
Var mark, sum;  
begin  
sum=0;  
subject=10;  
While(subject !=0)  
begin  
read(mark);  
sum = sum +mark;  
subject --;  
end  
write(sum)  
end
```











Example 3 : Adding ten subject marks for fifty students.

Pseudocode:

```
Var mark, sum, student;  
begin  
sum=0;  
student=50;  
subject=10;  
while(student !=0)  
begin  
While(subject !=0)  
begin  
read(mark);  
sum = sum +mark;  
subject --;  
end  
write(sum)  
student --;  
end
```

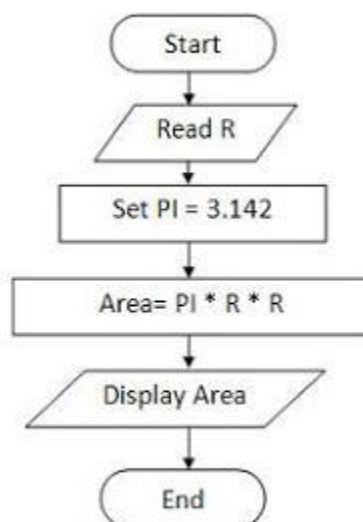
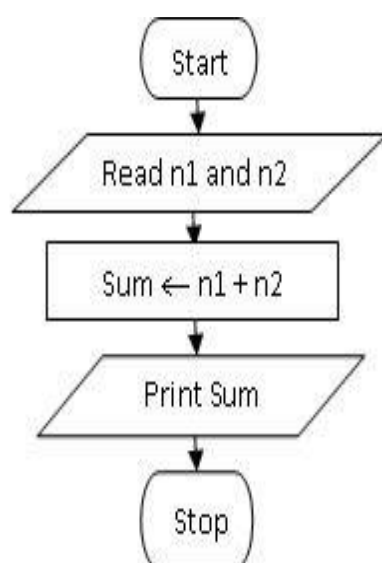
5. FLOW CHART

The flowchart is an diagrammatic representation of the work flow. This consists of various shapes for representing the various activities at each step which are connected with an arrow.

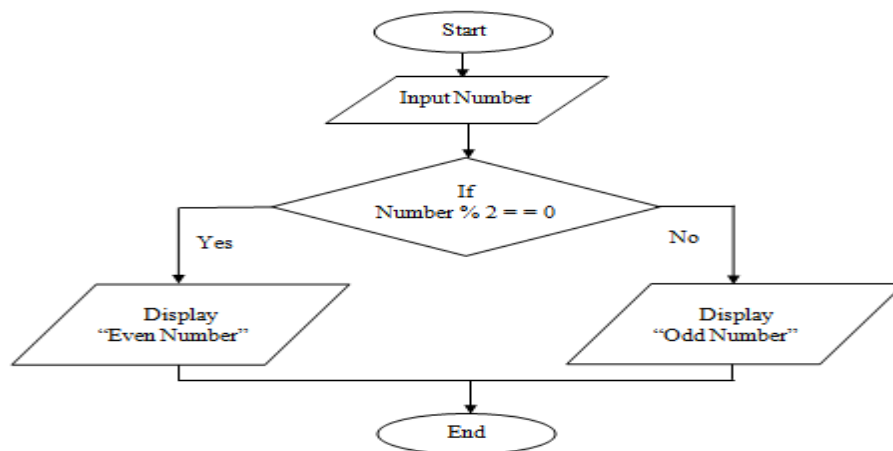
Name	Symbol	Description
Process		Process or action step
Flow line		Direction of process flow
Start/ terminator		Start or end point of process flow
Decision		Represents a decision making point
Connector		Inspection point
Inventory		Raw material storage
Inventory		Finished goods storage
Preparation		Initial setup and other preparation steps before start of process flow
Alternate process		Shows a flow which is an alternative to normal flow
Flow line(dashed)		Alternate flow direction of information flow

Flow Chart Examples:

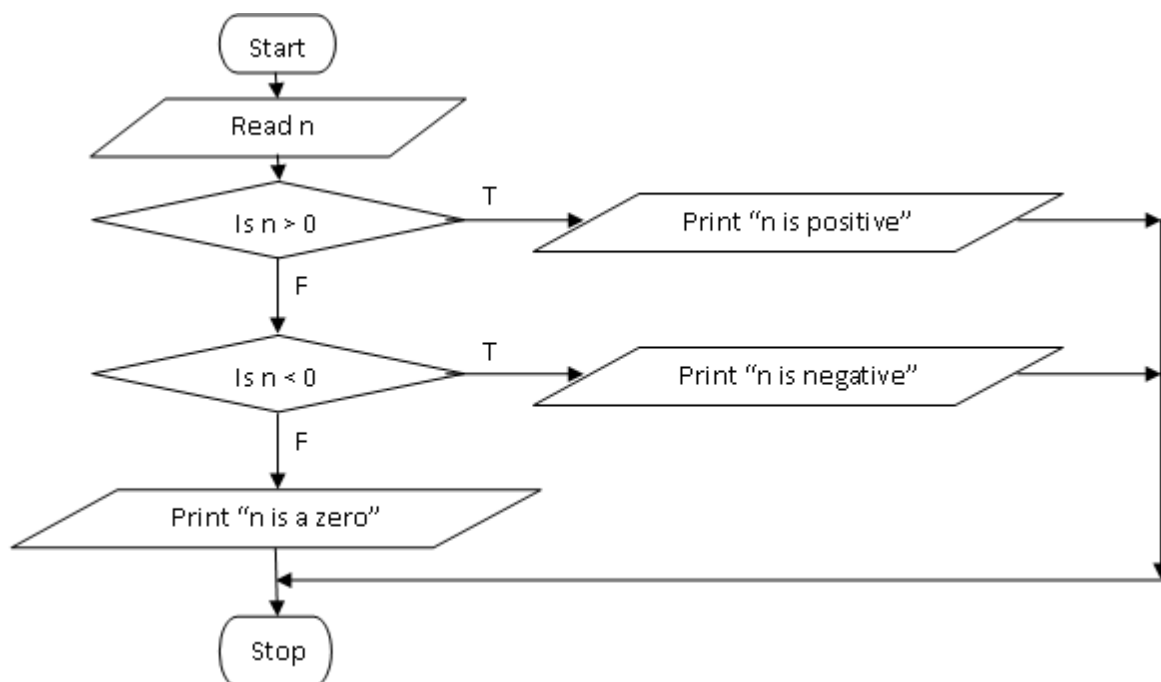
Sequence to find the sum of two number Area of triangle



Flowchart to find the number is odd or Even



Flow Chart to find Positive or Negative



6. CODING:

After the target and the steps has been identified. The coding phase can start. The various factors that should be considered while coding are

- Identify the users of the computer program
- Determine the basic system configuration for executing the computer program
- List the programming skill
- Select the programming language suitable from the list.

Identify the users of the computer program

7. Debugging

After the coding is of executable form. It is run several times manually inorder to remove the errors that has been occurred while typing the code. The program execution done manually is called dry run. This process is continued until the code become error less.

8. Testing- The testing of the codes are done after the completion of the entire coding. There are three types of data that has to be considered while testing the codes they are usual, unusual and invalid data. These type of data's are given to check the behavior of the code.

9. Final Output

This is the last stage in which the programmer has to show the full efficiency of the program. The exact output expected by the end user should be provided.

10. Documentation

This is similar to a manual that is been generated for any electronic product. This type of manual is also necessary for the software that is being coded too. This helps the end user and the programmer to know about the product and helps in long term maintenance.

3-ALGORITHM

Definition:

An Algorithm is a finite set of instructions that, if followed, accomplishes a particular task. In addition, all algorithms should satisfy the following criteria.

1. INPUT → Zero or more quantities are externally supplied.
2. OUTPUT → At least one quantity is produced.
3. DEFINITENESS → Each instruction is clear and unambiguous.
4. FINITENESS → If we trace out the instructions of an algorithm, then for all cases, the algorithm terminates after a finite number of steps.
5. EFFECTIVENESS → Every instruction must very basic so that it can be carried out, in principle, by a person using only pencil & paper.

Issues or study of Algorithm:

How to device or design an algorithm → creating and algorithm.

- How to express an algorithm → definiteness.
- How to analysis an algorithm → time and space complexity.
- How to validate an algorithm → fitness.
- Testing the algorithm → checking for error.

Algorithm Specification:

Algorithm can be described in three ways.

1. Natural language like English:

When this way is choose care should be taken, we should ensure that each & every statement is definite.

2. **Graphic representation called flowchart:** This method will work well when the algorithm is small & simple.

3. Pseudo-code Method:

In this method, we should typically describe algorithms as program, which resembles language like Pascal & algorithm.

Pseudo-Code Conventions:

1. Comments begin with // and continue until the end of line.
2. Blocks are indicated with matching braces { and }.
3. An identifier begins with a letter. The data types of variables are not explicitly declared.
4. Compound data types can be formed with records. Here is an example,

```
Node. Record
{  data type – 1
data-1;
```

```

      .
      .
data type – n data – n;
      node * link;   }

```

Here link is a pointer to the record type node. Individual data items of a record can be accessed with → and period.

5. Assignment of values to variables is done using the assignment statement.
 <Variable>:= <expression>;

6. There are two Boolean values TRUE and FALSE.

→ Logical Operators AND, OR, NOT
 → Relational Operators <, <=, >, >=, =, !=

7. The following looping statements are employed.

For, while and repeat-until While Loop:

```

While < condition > do
{
    <statement-1>
    .
    <statement-n>
}

```

For Loop:

For variable: = value-1 to value-2 step step do

```

{
<statement-1>
.
.
<statement-n>
} repeat-
until:

```

```

repeat
<statement-1>

```

```

.
.
.
<statement-n>

```

```

until<condition>

```

8. A conditional statement has the following forms.

→ If <condition> then <statement>
 → If <condition> then <statement-1>
 Else <statement-1>

Case statement:

```

case
{
    : <condition-1> : <statement-1>
    .
    .
    .
    : <condition-n> : <statement-n>
    : else : <statement-n+1>
}

```


9. Input and output are done using the instructions read & write.
10. There is only one type of procedure: Algorithm, the heading takes the form.

Performance Analysis:

1. Space Complexity:
The space complexity of an algorithm is the amount of memory it needs to run to completion.
2. Time Complexity:
The time complexity of an algorithm is the amount of computer time it needs to run to completion.

C Programs with Solution

C CONCEPTS

1.0 OVERVIEW OF C PROGRAMMING

C language is one of the most popular computer languages today because it is a structured, high level, machine independent language. It allows software developers to develop programs without worrying about the hardware platforms where they will be implemented. C is called a high level, compiler language. The aim of any high level computer language is to provide an easy and natural way of giving a programme of instructions to a computer.

C is one of a large number of high level languages which can be used for general purpose programming, *i.e.*, anything from writing small programs for personal amusement to writing complex applications. It is unusual in several ways. Before C, high level languages were criticized by machine code programmers because they shielded the user from the working details of the computer. The C language has been equipped with features that allow programs to be organized in an easy and logical way. This is vitally important for writing lengthy programs because complex problems are only manageable with a clear organization and program structure.

C allows meaningful variable names and meaningful function names to be used in programs

without any loss of efficiency and it gives a complete freedom of style, it has a set of very flexible loop constructions and neat ways of making decisions. These provide an excellent basis for controlling the flow of programs. Another feature of C is the way it can express ideas concisely. The richness of a language shapes what it can talk about. C gives us the apparatus to build neat and compact programs. C tries to make the best of a computer by linking as closely as possible to the local environment.

The increasing popularity of C is probably due to its many desirable qualities. It is a robust language whose rich set of built-in functions and operators can be used to write any complex program. The C compiler combines the capabilities of an assembly language with the features of a high-level language and therefore it is well suited for writing both system software and business packages. Programs written in C are efficient and fast. This is due to its variety of data types and powerful operators. C is highly portable. This means that C programs written for one computer can be run on another with little or no modification. Another feature of C is its ability to extend itself.

1.1 INTRODUCTION

C is a remarkable language. Designed originally by **Dennis Ritchie**, working at **AT&T Bell** Laboratories in New Jersey, it has increased in use until now it may well be one of the most widely-written computer languages in the world. C is a structured language. It allows variety of programs in small modules. It is easy for debugging, testing, and maintenance if a language is a structured one.

1.2 STRUCTURE OF A C PROGRAM

Include header file section

Global declaration section

main()

{

Declaration part

Executable part

}

User-defined functions

{

Statements

}

Include Header File Section

C program depends upon some header files for function definition that are used in program. Each header file by default is extended with .h. The header file should be included using #include directive as given here.

Global Declaration

This section declares some variables that are used in more than one function. These variables are known as global variables. This section must be declared outside of all the functions.

Function Main

Every program written in C language must contain main () function. The function main() is a starting point of every C program. The execution of the program always begins with the function main ().

Declaration Part

The declaration part declares the entire variables that are used in executable part. The initialisations of variables are also done in this section. Initialisation means providing initial value to the variables.

Executable Part

This part contains the statements following the declaration of the variables. This part contains a set of statements or a single statement. These statements are enclosed between the braces.

User Defined Function

The functions defined by the user are called user-defined functions. These functions are generally defined after the main () function.

1.3 STEPS FOR EXECUTING THE PROGRAM

1. Creation of program

Programs should be written in C editor. The file name does not necessarily include extension C. The default extension is C.

2. Compilation of a program

The source program statements should be translated into object programs which is suitable for execution by the computer. The translation is done after correcting each statement. If there is no error, compilation proceeds and translated program are stored in another file with the same file name with extension “.obj”.

3. Execution of the program

After the compilation the executable object code will be loaded in the computer's main memory and the program is executed.

1.4 C CHARACTER SET

Letters	Digits	White Spaces
Capital A to Z	All decimal digits 0 to 9	Blank space
Small a to z		Horizontal tab
		Vertical tab
		New line
		Form feed

Special Characters

,	Comma	&	Ampersand
.	dot	^	Caret
;	Semicolon	*	Asterisk
:	Colon	-	Minus
'	Apostrophe	+	Plus

4 C PROGRAMS WITH SOLUTIONS

"	Quotation mark	<	Less than
!	Exclamation mark	>	Greater than
	Vertical bar	()	Parenthesis left/right
/	Slash	[]	Bracket left/right
\	Back slash	{ }	Braces left/right
~	Tilde	%	Percent
_	Underscore	#	Number sign or Hash
\$	Dollar	=	Equal to
?	Question mark	@	At the rate

1.5 DELIMITERS

Delimiters	Use
: Colon	Useful for label
; Semicolon	Terminates the statement
() Parenthesis	Used in expression and function
[] Square Bracket	Used for array declaration
{ } Curly Brace	Scope of the statement

# hash	Preprocessor directive
, Comma	Variable separator

1.6 C KEYWORDS

Auto	Double	Int	Struct
Break	Else	Long	Switch
Case	Enum	Register	Typedef
Char	Extern	Return	Union
Const	Float	Short	Unsigned
Continue	For	Signed	Void
Default	Goto	Sizeof	Volatile
Do	If	Static	while

1.7 IDENTIFIERS

Identifiers are names of variables, functions, and arrays. They are user-defined names, consisting sequence of letters and digits, with the letter as the first character.

1.8 CONSTANTS

Values do not change during the execution of the program

Types:

1. Numerical constants:

— Integer constants

*These are the sequence of numbers from 0 to 9 without decimal points or fractional part or any other symbols. It requires minimum two bytes and maximum four bytes.
Eg: 10, 20, + 30, - 14*

— Real constants

*It is also known as floating point constants.
Eg: 2.5, 5.342*

2. Character constants:

— Single character constants

A character constant is a single character. Characters are also represented with a single digit or a single special symbol or white space enclosed within a pair of single quote marks

Eg: 'a', '8', " ".

— String constants

String constants are sequence of characters enclosed within double quote marks.

Eg: "Hello", "india", "444"

1.9 VARIABLES

It is a data name used for storing a data value. Its value may be changed during the program execution. The value of variables keeps on changing during the execution of a program.

1.10 DATA TYPES

Short or int	2	$-32,768$ to $32,767$	%i or %d
Unsigned int	2	0 to 65535	%u
Float	4	$3.4e-38$ to $+3.4e+38$	%f or %g
Long	4	-2147483648 to 2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
Double	8	$1.7e-308$ to $1.7e+308$	%lf
Long double	10	$3.4e-4932$ to $1.1e+4932$	%lf

Data type	Size (Bytes)	Range	Format Specifiers
Char	1	-128 to 127	%c
Unsigned char	1	0 to 255	%c

1.11 OPERATORS

It indicates an operation to be performed on data that yields value.

Types

Type of Operator	Symbolic representation
Arithmetic operators	+, -, *, /, %
Relational operators	>, <, ==, >=, <=, !=
Logical operators	&&, , !=
Increment and decrement operator	++ and --
Assignment operator	=
Bitwise operator	&, , ^, >>, <<, ~
Comma operator	,
Conditional operator	?:

1.12 INPUT AND OUTPUT

Reading data from input devices and displaying the results on the screen are the two main tasks of any program.

Formatted Functions

— The formatted input/output functions read and write all types of values

Input

Scanf()

Output

Printf()

Unformatted Functions

— The unformatted input/output functions only work with the character data type

Input

getch()

getche()

getchar()

gets()

Output

putch()

putchar()

put()

1.13 DECISION STATEMENTS

It checks the given condition and then executes its sub-block. The decision statement decides the statement to be executed after the success or failure of a given condition.

Types:

1. If statement
 2. If-else statement
 3. Nested if-else statement
 4. Break statement
 5. Continue statement
 6. Goto statement
 7. Switch() statement
 8. Nested switch ()case
 9. Switch() case and Nested if
-

Statement	Syntax
If statement	if(condition) Statement;
If-else statement	If (condition) { Statement 1; Statement 2; } else { Statement 3; Statement 4; }
Nested if-else statement	If (condition) {

8 C PROGRAMS WITH SOLUTIONS

	<pre>Statement 1; Statement 2; } Else if (condition) { Statement 3; Statement 4; } Else { Statement 5; Statement 6; }</pre>
Break statement	<pre>Break;</pre>
Continue statement	<pre>Continue;</pre>
Goto statement	<pre>goto label;</pre>
Switch() statement	<pre>Switch (variable or expression) { Case constant A: Statement; Break; Case constant B: Statement;</pre>

	<pre>Break; Default: Statement; }</pre>
--	--

1.14 LOOP CONTROL STATEMENTS

Loop is a block of statements which are repeatedly executed for certain number of times.

Types

1. For loop
2. Nested for loops
3. While loop
4. do while loop
5. do-while statement with while loop

Statement	Syntax
For loop	<pre>For(initialize counter; test condition; re-evaluation parameter) { Statement; Statement; }</pre>
Nested for loop	<pre>for(initialize counter; test condition; re-evaluation parameter) { Statement; Statement; for(initialize counter; test condition; re-evaluation parameter) Statement; Statement; } }</pre>
While loop	<pre>While (test condition) { Body of the loop }</pre>
Do while loop	<pre>do { Statement; }</pre>

	While(condition);
Do-while with while loop	Do while(condition) { Statement; } While (condition);

2 INTRODUCTION—C PROGRAMS

1] Program to find sum of two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,s;
clrscr();
printf("Enter two no: ");
scanf("%d%d",&a,&b);
s=a+b;
printf("sum=%d",s);
getch();
}
```

Output:

```
Enter two no: 5
6
sum=11
```

2] Program to find area and circumference of circle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r;
float pi=3.14,area,ci;
clrscr();
printf("enter radius of circle: ");
```

```

scanf("%d",&r);
area=pi*r*r;
printf("area of circle=%f ",area);
ci=2*pi*r;
printf("circumference=%f ",ci);
getch();
}

```

Output:

```

enter radius of a circle: 5
area of circle=78.000
circumference=31.4

```

3] Program to find the simple interest.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int p,r,t,si;
clrscr();
printf("enter principle, Rate of interest & time to find simple interest: ");
scanf("%d%d%d",&p,&r,&t);
si=(p*r*t)/100;
printf("simple intrest= %d",si);
getch();
}

```

Output:

```

enter principle, rate of interest & time to find simple interest: 500
5
2
simple interest=50

```

4] Program to convert temperature from degree centigrade to Fahrenheit.

```

#include<stdio.h>
#include<conio.h>
void main()

```

```

{

float c,f;
clrscr();
printf("enter temp in centigrade: ");
scanf("%f",&c);
f=(1.8*c)+32;
printf("temp in Fahrenheit=%f",f);
getch();
}

```

Output:

```

enter temp in centigrade: 32
temp in Fahrenheit=89.59998

```

5] Program to calculate sum of 5 subjects and find percentage.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int s1,s2,s3,s4,s5,sum,total=500;
float per;
clrscr();
printf("enter marks of 5 subjects: ");
scanf("%d%d%d%d%d",&s1,&s2,&s3,&s4,&s5);
sum=s1+s2+s3+s4+s5;
printf("sum=%d",sum);
per=(sum*100)/total;
printf("percentage=%f",per);
getch();
}

```

Output:

```

enter marks of 5 subjects: 60
65
50
60
60
sum=300
percentage=60.000

```

6] Program to show swap of two no's without using third variable.

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

void main()
{
    int a,b;
    clrscr();
    printf("enter value for a & b: ");
    scanf("%d%d",&a,&b);
    a=a+b;
    b=a-b;
    a=a-b;
    printf("after swapping the value of a & b: %d %d",a,b);
    getch();
}
```

Output:

```
enter value for a & b: 4 5
after swapping the value of a & b: 5 4
```

7] Program to reverse a given number.

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

void main()
{
    int n,a,r=0;
    clrscr();
    printf("enter any no to get its reverse: ");
    scanf("%d",&n);
    while(n>=1)
    {
        a=n%10;
        r=r*10+a;
        n=n/10;
    }
    printf("reverse=%d",r);
}
```

```
getch();  
}
```

Output:

```
enter any no to get its reverse: 456  
reverse=654
```

8] Program to find gross salary.

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int gs,bs,da,ta;  
clrscr();  
printf("enter basic salary: ");  
scanf("%d",&bs);  
da=(10*bs)/100;  
ta=(12*bs)/100;  
gs=bs+da+ta;  
printf("gross salary=%d",gs);  
getch();  
}
```

Output:

```
enter basic salary: 100  
gross salary=122
```

9] Program to print a table of any number.

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int gs,bs,da,ta;  
clrscr();  
printf("enter basic salary: ");  
scanf("%d",&bs);  
da=(10*bs)/100;  
ta=(12*bs)/100;
```

```

gs=bs+da+ta;
printf("gross salary=%d",gs);
getch();
}

```

Output:

enter a no to know table: 2

2*1=2

2*2=4

2*3=6

2*4=8

2*5=10

2*6=12

2*7=14

2*8=16

2*9=18

2*10=20

10] Program to find greatest in 3 numbers.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("enter value of a, b & c: ");
scanf("%d%d%d",&a,&b,&c);
if((a>b)&&(a>c))
printf("a is greatest");
if((b>c)&&(b>a))
printf("b is greatest");
if((c>a)&&(c>b))
printf("c is greatest");
getch();
}

```

Output: enter value for a, b& c: 5 7 4

b is greatest

11] Program to show the use of conditional operator.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    printf("enter value for a & b: ");

    scanf("%d%d",&a,&b);
    (a>b)?printf("a is greater"):printf("b is greater");
    getch();
}
```

Output:

```
enter value for a & b: 5
7
b is greater
```

12] Program to find that entered year is leap year or not.

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

Output:

```
enter any year: 1947
year is not a leap year
```

13] Program to find whether given no. is even or odd.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
clrscr();
printf("enter any no: ");
scanf("%d",&n);
if(n%2==0)
printf("no is even");
else

printf("no is odd");
getch();
}
```

Output:

```
enter any no: 5
no is odd
```

14] Program to shift inputed data by two bits to the left.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y;
clrscr();
printf("Read the integer from keyboard :- ");
scanf("%d",&x);
x<<=3;
y=x;
printf("\nThe left shifted data is = %d ",y);
getch();
}
```

Output:

```
Read the integer from keyboard :- 2
The left shifted data is = 16
```


15] Program to use switch statement. Display Monday to Sunday.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char ch;
    clrscr();
    printf("enter m for Monday\n t for Tuesday\n w for Wednesday\n h for Thursday\n f for Friday\n s for Saturday\n u for Sunday);
    scanf("%c",&ch);
    switch(ch)
    {
        case 'm':

        case 'M':
            printf("monday");
            break;
        case 't':
        case 'T':
            printf("tuesday");
            break;
        case 'w':
        case 'W':
            printf("wednesday");
            break;
        case 'h':
        case 'H':
            printf("thursday");
            break;
        case 'f':
        case 'F':
            printf("friday");
            break;
        case 's':
        case 'S':
            printf("saturday");
            break;
        case 'u':
```

```

case 'U':
printf("sunday");
break;
default :
printf("wrong input");
break;
}
getch();
}

```

Output:

```

enter m for Monday
t for Tuesday
w for Wednesday
h for Thursday
f for Friday
s for Saturday
u for Sunday: f
Friday

```

16] Program to display arithmetic operator using switch case.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,n,s,m,su,d;
clrscr();
printf("enter two no's : ");
scanf("%d%d",&a,&b);
printf("enter 1 for sum\n2 for multiply\n3for subtraction\n4 for division: ");
scanf("%d",&n);
switch(n)
{
case 1:
s=a+b;
printf("sum=%d",s);
break;
case 2:

```

```

m=a*b;
printf("multiply=%d",m);
break;
case 3:
su=a-b;
printf("subtraction=%d",su);
break;
case 4:
d=a/b;
printf("divission=%d",d);
break;

default:
printf("wrong input");
break;
}
getch();
}

```

Output:

```

enter two no's: 8
4
enter 1 for sum
2 for multiply
3 for subtraction
4 for division: 1
sum=12

```

17] Program to display first 10 natural no. & their sum.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int i,sum=0;
clrscr();
for(i=1;i<=10;i++)
{
printf("%d no is= %d\n",i,I);

```

```
sum=sum+i;
}
printf("sum =%d",sum);
getch();
}
```

Output:

```
1 no is=1
2 no is=2
3 no is=3
4 no is=4
5 no is=5

6 no is=6
7 no is=7
8 no is=8
9 no is=9
10 no is=10
sum=55
```

18] Program to print stars Sequence1:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
printf("*");
printf("\n");
}
getch();
}
```

Output:

```
*
**
```

19] Program to print stars Sequence2.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,k;
    clrscr();
    for(i=1;i<=5;i++)

    {
        for(j=5;j>=i;j--)
            printf(" ");
        for(k=1;k<=i;k++)
            printf("*");
        printf("\n");
    }
    getch();
}
```

Output:

```
  *
 **
***
****
```

20] Program to print stars Sequence3.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,k;
    clrscr();
    for(i=1;i<=3;i++)
    {
```

```

for(j=3;j>=i;j--)
printf(" ");
{
for(k=1;k<=i*2-1;k++)
printf("*");
}
printf("\n");
}
getch();
}

```

Output:

```

      *
    ***
  *****

```

21] Program to print Fibonacci series up to 100.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int a=1,b=1,c=0,i;
clrscr();
printf("%d\t%d\t",a,b);
for(i=0;i<=10;i++)
{
c=a+b;
if(c<100)
{
printf("%d\t",c);
}
a=b;
b=c;
}
getch();
}

```

Output:1 1 2 3 5 8 13 21 34 55 89

22] Program to find factorial of a number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,fact=1;
clrscr();
printf("Enter any no: ");
scanf("%d",&n);

for(i=n;i>=1;i--)
{
fact=fact*i;
}
printf("Factorial=%d",fact);
getch();
}
```

Output:

```
Enter a no: 5
Factorial=120
```

23] Program to find whether given no. is a prime no. or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,r=0;
clrscr();
printf("Enter any no: ");
scanf("%d",&n);
for(i=2;i<=n-1;i++)
{
if(n%i==0)
r=1;
break;
}
if(r==0)
```

```
printf("prime no");
else
printf("Not prime");
getch();
}
```

Output:

Enter any no: 16

Not prime

24] Program to display sum of series $1 + 1/2 + 1/3 + \dots + 1/n$.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,sum=0;
clrscr();
printf("Enter any no: ");
scanf("%d",&n);
printf("1");
for(i=2;i<=n-1;i++)
printf(" 1/%d +",i);
for(i=1;i<=n;i++)
sum=sum+i;
printf(" 1/%d",n);
printf("\nSum=1/%d",sum+1/n);
getch();
}
```

Output:

Enter any no: 7

$1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7$

Sum=1/28

