

# **C Programming**

## **(Basic)**

### **PART-1**

<b>Topic</b>	<b>Page no</b>
compiler	3
Basic structure of c program	4-6
Rules to declare variable names	6
What are datatypes?	7-8
Keywords in c	8
Printf() function	8-9
How to input a variable?	9
Escape sequences	10
operators	14-20
Decision making statements	20-37
loops	37-57
Jump statements	57-67
patterns	67-86
arrays	86-90
<b>How to run a “C” program in “Dev C++”</b>	<b>91-92</b>

## **Compiler:**

A compiler is a computer program which converts the written high level language(program) into machine language(in the form of 0's and 1's )which is understandable by the computer.

The execution starts from the main function, thereby checking out the errors in the program. The compiler doesn't care about what is there in the comments.

Even a small error is displayed for mistakes like forgetting the semicolon at the end of line, forgetting variable declaration etc. By solving the errors we can compile our program without any errors and get the desired output.

# BASIC STRUCTURE OF C PROGRAM

- Comments
- Header files
- Global declaration (if necessary)
- Main function
- Sub functions(if necessary)

## COMMENTS

- Including comments in the program makes the user to understand ‘why this logic is used ?’, ‘what is this program about ?’ such type of queries will be clarified.
- Comments can be included in any part of the program
- There are two types of comments. They are:

### a. SINGLE LINE COMMENTS:

We can add comment to a single line only. Any text can be entered after **ex :** `//` (two forward slashes)  
hello world program

`// declare a variable`

### b. MULTI-LINE COMMENTS:

Comments can be written in multiple lines, but the text must included within `/*` and `*/`

**ex** : /\* you can include any multiple number of lines text here within these forward slash followed by asterisk and asterisk followed by forward slash \*/

/\* hello man !!!!\*/

## HEADER FILES

To access functions of a header file , we need to import them with the help of pre-processor directive called '**#include**'.

Every C program should contain the header file **<stdio.h>** which stands for standard input and output functions to take input with the help of scanf() function and to display output with the help of printf() function.

**Ex:** #include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<conio.h>

#include<time.h>

#include<string.h> etc...

## MAIN FUNCTION

The execution of a C program starts from the main function. Every C program must have main function.

Ex: void main()

{

Block of statements;

}

(or)

```
int main()
{
Block of statements;
return 0;
}
```

### **Example program 1(ep1):**

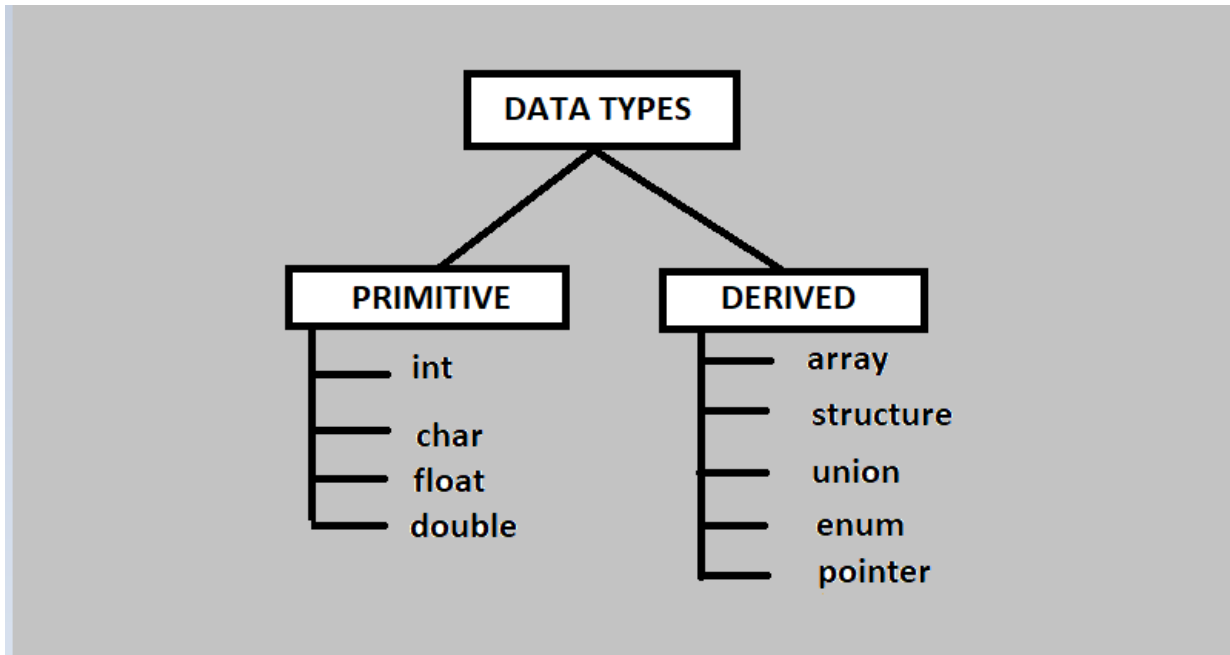
```
/*Program to print my name*/
#include<stdio.h> //header file
void main() //main function
{
printf("CHANDANA PENUMUCHU"); //print statement
}
```

- To go ahead with our programming we need to know ‘what are the rules to declare variable names?’, ‘what are datatypes?’, ‘what are keywords in c ?’ ‘how to print using printf() function?’ , ‘how to input a value to a variable?’and ‘what are escape sequences?’

### **what are the rules to declare variable names?**

- Variable name must begin with letter or underscore.
- Variables are case sensitive.
- They can be constructed with digits, letters.
- No special symbols are allowed other than underscore.
- sum, height, \_value, aA, Aa are some examples for variable name.

## What are datatypes?



### Basic /primitive /fundamental datatypes:

Following are the examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

Ex: `char ch= 'A';` //ch - variable name

- **int:** As the name suggests, an int variable is used to store an integer.

Ex: `int a=27;` // a-variable name

- **float:** It is used to store decimal numbers (numbers with floating point value- till 7 digits after decimal point)

Ex: `float x=9.65766;` // x-variable name

- **double:** It is used to store decimal numbers (numbers with floating point value-till 15 digits after decimal point)

Ex: double a=2.5765655657667; // a-variable name

DATA TYPE	MEMORY (bytes)	RANGE	FORMAT SPECIFIER
char	1	-128 to 127	%c
int	4	-2,147,483,648 to 2,147,483,647	%d
float	4	1.2E-38 to 3.4E+38	%f
double	8	2.3E-308 to 1.7E+308	%lf

### What are keywords in C?

There are 32 keywords in c of which each has a specific purpose .we will get to know some of these keywords ahead .

<b>auto</b>	<b>break</b>	<b>case</b>	<b>char</b>
<b>const</b>	<b>continue</b>	<b>default</b>	<b>do</b>
<b>double</b>	<b>else</b>	<b>enum</b>	<b>extern</b>
<b>float</b>	<b>for</b>	<b>goto</b>	<b>if</b>
<b>int</b>	<b>long</b>	<b>register</b>	<b>return</b>
<b>short</b>	<b>signed</b>	<b>sizeof</b>	<b>static</b>
<b>struct</b>	<b>switch</b>	<b>typedef</b>	<b>union</b>
<b>unsigned</b>	<b>void</b>	<b>volatile</b>	<b>while</b>

### how to print using printf() function?

Everything written inside the double quotes of print statement will be printed as it is.

PRINT STATEMENT	OUTPUT
printf("hello chandu");	hello chandu



int a=27; printf("the value of a is %d",a);	the value of a is 27
char c='p'; printf("the character is %c",c);	the character is p
float val=7.32; printf("decimal num=%f",val);	decimal num=7.32
double d=34.33224224; printf("value=%lf",d);	value=34.33224224
int a=2020,b=2019; char c='s'; printf("bye %d,worst %d,%c",b,a,c);	bye 2019,worst 2020,s
int s1=11,s2=12; printf("sum is %d",s1+s2)	sum is 23

### how to input a value to a variable?

declaration	How to input?
int p;	scanf("%d",&p);
char s;	scanf("%c",&s);
float b;	scanf("%f",&b);
double m;	scanf("%lf",&m);

Whatever the value we give on output screen those will be stored in the memory of corresponding variables.

### what are escape sequences?

Escape Sequence	Name	Description
\a	Bell (alert)	Makes a sound from the computer
\b	Backspace	Takes the cursor back
\t	Horizontal Tab	Takes the cursor to the next tab stop
\n	New line	Takes the cursor to the beginning of the next line
\v	Vertical Tab	Performs a vertical tab
\f	Form feed	
\r	Carriage return	Causes a carriage return
\"	Double Quote	Displays a quotation mark (")
\'	Apostrophe	Displays an apostrophe (')
\?	Question mark	Displays a question mark
\\	Backslash	Displays a backslash (\)
\0	Null	Displays a null character

The mostly used escape sequence is \n.

Ex1: printf("chandana penumuchu");

printf(" harshitha penumuchu ");

Output:

chandana penumuchu harshitha penumuchu

Ex2:printf("chandana penumuchu \n harshitha penumuchu");

Output:

chandana penumuchu

harshitha penumuchu

**Example program 2(ep2):**

*/\*program to add and subtract 2 numbers by initializing values in the program itself\*/*

#include<stdio.h>

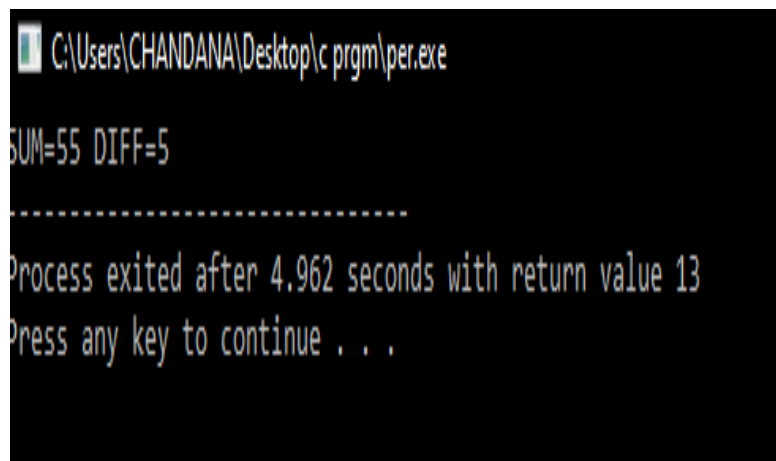
void main()

```

{
int a=25,b=30,c,d; // in memory space with variable names a,b,c,d are created and values of a,b are stored
c=a+b;//value of c will be stored in memory
d=b-a;//value of d will be stored in memory
printf("SUM=%d DIFF=%d",c,d);//printing the values of c and d
}

```

## OUTPUT:

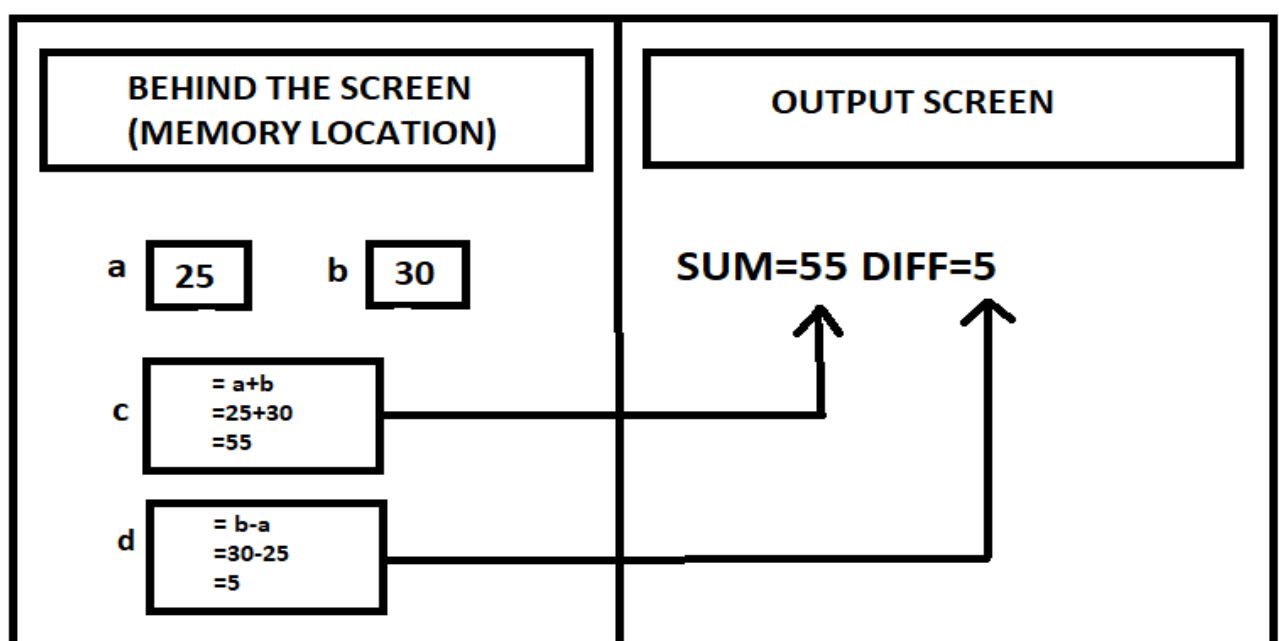


```

C:\Users\CHANDANA\Desktop\c prgm\per.exe
SUM=55 DIFF=5
-----
Process exited after 4.962 seconds with return value 13
Press any key to continue . . .

```

## EXPLANATION:



### Example program 3(ep3):

*/\*program to add and subtract 2 numbers by inputing the values\*/*

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

```
void main()
```

```
{
```

```
int a,b,c,d; // in memory ,space with variable names a,b,c,d are created
```

```
printf("enter a value");//prints on output screen
```

```
scanf("%d",&a);//value of 'a' stores in memory when user gives input
```

```
printf("\nenter b value");//prints on output screen
```

```
scanf("%d",&b); //value of 'b' stores in memory when user gives input
```

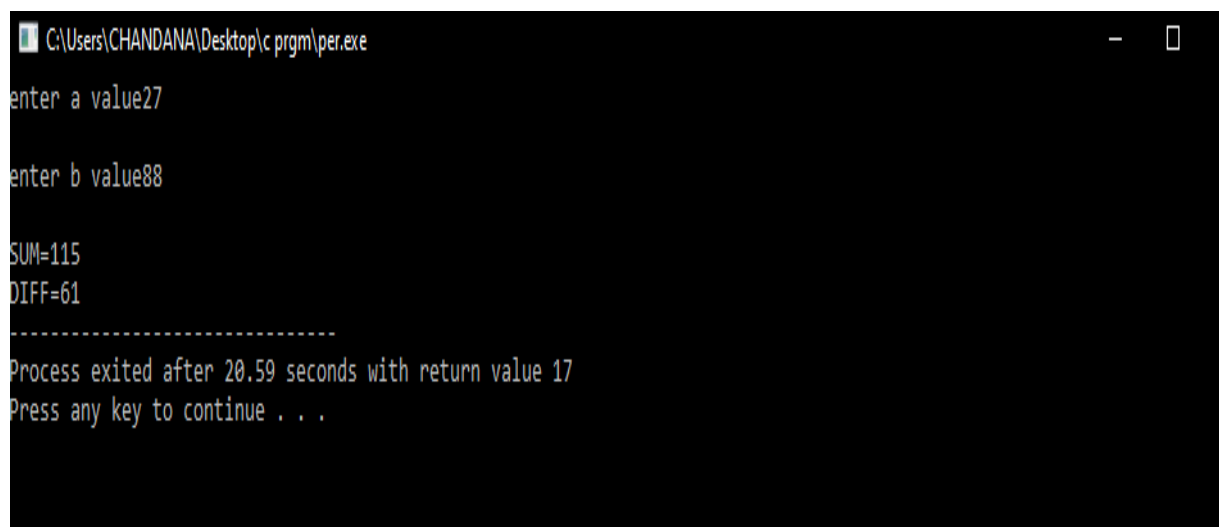
```
c=a+b;//value of c will be calculated and stored in memory
```

```
d=b-a;//value of d will be calculated and stored in memory
```

```
printf("\nSUM=%d \nDIFF=%d",c,d);//printing the values of c and d
```

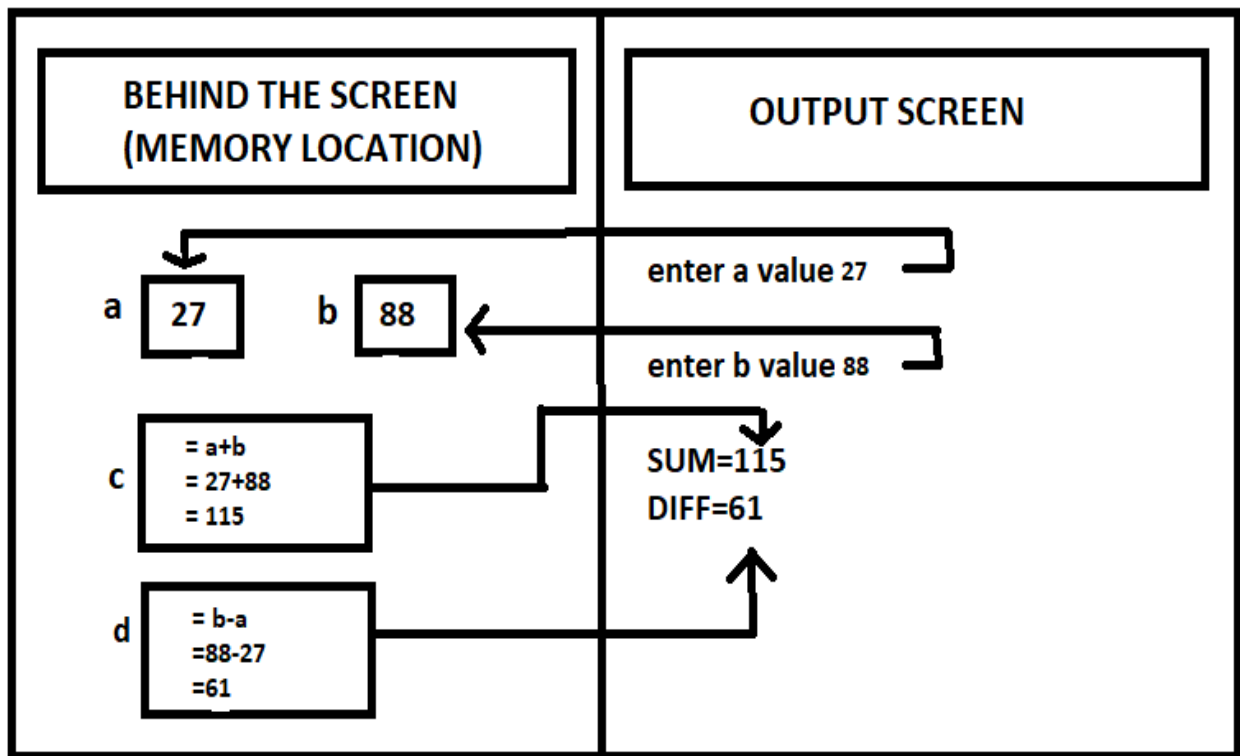
```
}
```

### Output1:



```
C:\Users\CHANDANA\Desktop\c prgm\per.exe
enter a value27
enter b value88
SUM=115
DIFF=61
-----
Process exited after 20.59 seconds with return value 17
Press any key to continue . . .
```

### EXPLANATION:



## Output 2:

\\Users\\CHANDANA\\Desktop\\c prgm\\per.c - [Executing] - Dev-C++ 5.11

```

C:\Users\CHANDANA\Desktop\c prgm\per.exe
enter a value6
enter b value45
SUM=51
DIFF=39
-----
Process exited after 21.06 seconds with return value 16
Press any key to continue . . .

```

When you try it by yourself ,you will learn and find out inetresting ,try to perform the following programs by yourself

**P1:**Program to input two numbers and perform operations like addition ,subtraction and multiplication and print the result as the following sample output

**Sample output:**

```
enter a value27
enter b value12
SUM=39 DIFF=-15
PRODUCT=324
```

**P2:**Program to print square and cube of a given input number

**Sample output:**

```
enter a number7
NUMBER=7
SQUARE=49
CUBE=343
```

By this time you might have got clarity of using printf and scanf statements

Now, lets learn about the operators in c

## **OPERATORS**

An operator is a symbol that operates on a value or a variable.

Ex: a=5,b=7

c=a+b

Here, plus(+) is an operator that performs addition on a,b variables and stores the result in c

**Some operators in c are:**

- 1. ARITHMETIC OPERATORS**
- 2. INCREMENT AND DECREMENT OPERATORS(UNARY)**
- 3. ASSIGNMENT OPERATORS**
- 4. RELATIONAL OPERATORS**
- 5. LOGICAL OPERATORS**
- 6. BITWISE OPERATORS**
- 7. CONDITIONAL OPERATOR**

### **ARITHMETIC OPERATORS**

An arithmetic operator performs mathematical calculations like addition, subtraction, multiplication, division.

<b>OPERATOR</b>	<b>MEANING</b>
+	addition
-	subtraction
*	multiplication
/	division
%	remainder

### **UNARY OPERATORS**

**PRE-INCREMENT OPERATOR(++x):** First the value of the variable will be incremented to plus one of the variable value and then the operation takes place.

**PRE-INCREMENT:** First the value will be incremented and then the operation takes place

**GIVEN** a=3

$$\begin{aligned}x &= (a) + (++a) + (a) \\ &= (3) + (4) + (4) \\ &= 11\end{aligned}$$

**POST-INCREMENT OPERATOR(x++):** First operations takes place and then the value of the variable will be incremented to plus one of the variable value.

**POST-INCREMENT:** First operation takes places and then value will be incremented

**given :** a=3

$$\begin{aligned}x &= (a) + (a++) + (a) \\ &= (3)+(3)+(4) \\ &= 10\end{aligned}$$

**PRE-DECREMENT OPERATOR(--x):** First the value of the variable will be decremented to minus one of the variable value and then the operation takes place.



**PRE-DECREMENT:** First value will be decremented and then the operation takes place

GIVEN a=3

$$\begin{aligned}x &= (a) + (--a) + (a) \\ &= (3) + (2) + (2) \\ &= 7\end{aligned}$$

**POST-DECREMENT OPERATOR(x--):** First operations takes place and then the value of the variable will be decremented to minus one of the variable value.

**POST-DECREMENT:** First operation takes place and then the value will be decremented

GIVEN a=3

$$\begin{aligned}x &= (a) + (a--) + (a) \\ &= (3) + (3) + (2) \\ &= 8\end{aligned}$$

## **ASSIGNMENT OPERATORS**

An assignment operator is used for assigning a value to a variable.

<b>OPERATOR</b>	<b>EQUATION</b>	<b>EQUIVALENT TO</b>
=	a=b	a=b
+=	a+=b	a=a+b

-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

## **RELATIONAL OPERATORS :**

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

## **LOGICAL OPERATORS**

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands is	(A    B) is true.

	non-zero, then the condition becomes true.	
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

## **BITWISE OPERATORS**

Operator	Description	Example(assume A=60 and B=13)
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A   B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e., -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

p	q	p & q	p   q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assume A = 60 and B = 13 in binary format, they will be as follows –

A = 0011 1100

B = 0000 1101

-----

$A \& B = 0000\ 1100$

$A | B = 0011\ 1101$

$A \wedge B = 0011\ 0001$

$\sim A = 1100\ 0011$

## **CONDITIONAL OPERATOR**

Also known as ternary operator.

**Syntax: expression ? statement1: statement2;**

If the expression is true statement1 gets evaluated else statement2 gets evaluated.

Ex:

`(y >= 20) ? printf("good") : printf("bad");`

If  $y = 40$  output will be good

If  $y = 10$  output will be bad

## **DECISION MAKING STATEMENTS**

Decision making statements decides the direction of flow in the program.

1. if statement
2. if-else statement
3. Nested if statements
4. if-else if ladder
5. switch statement
6. Jump statemnts
  - a. break
  - b. Continue
  - c. goto
  - d. return

**if statement:**

## Syntax

```
if(condition) {
```

```
Block of statements;
```

```
}
```

If the condition is true then the block of statements inside the if blocks gets executed.

### **Example program 4(ep4):**

```
/*program to print end of the world if the input year is greater than or  
equal to 2020*/
```

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

```
void main()
```

```
{
```

```
int year;
```

```
printf("\n enter the year");
```

```
scanf("%d",&year);
```

```
if(year>=2020) // if the condition is true then it enters inside the block
```

```
{
```

```
printf("\n end of the world");
```

```
}
```

```
}
```

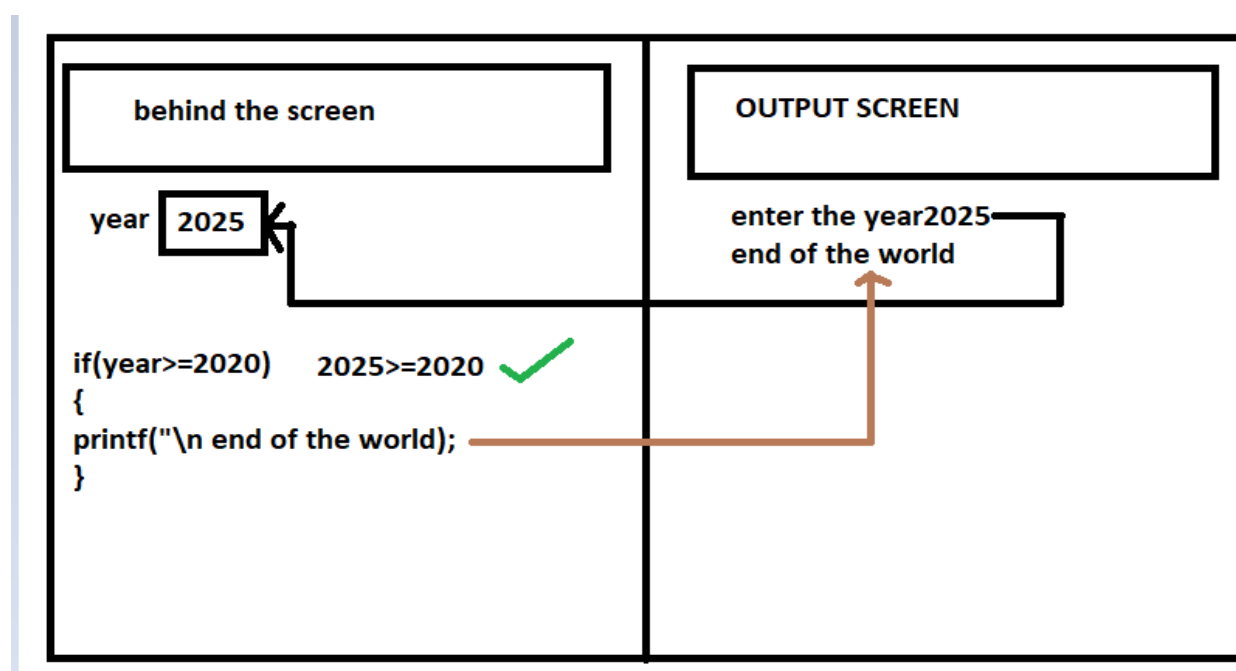
**OUTPUT1:**

```
C:\Users\CHANDANA\Desktop\c prgm\if.exe

enter the year2025

end of the world
-----
Process exited after 80.77 seconds with return value 18
Press any key to continue . . .
```

## EXPLANATION:



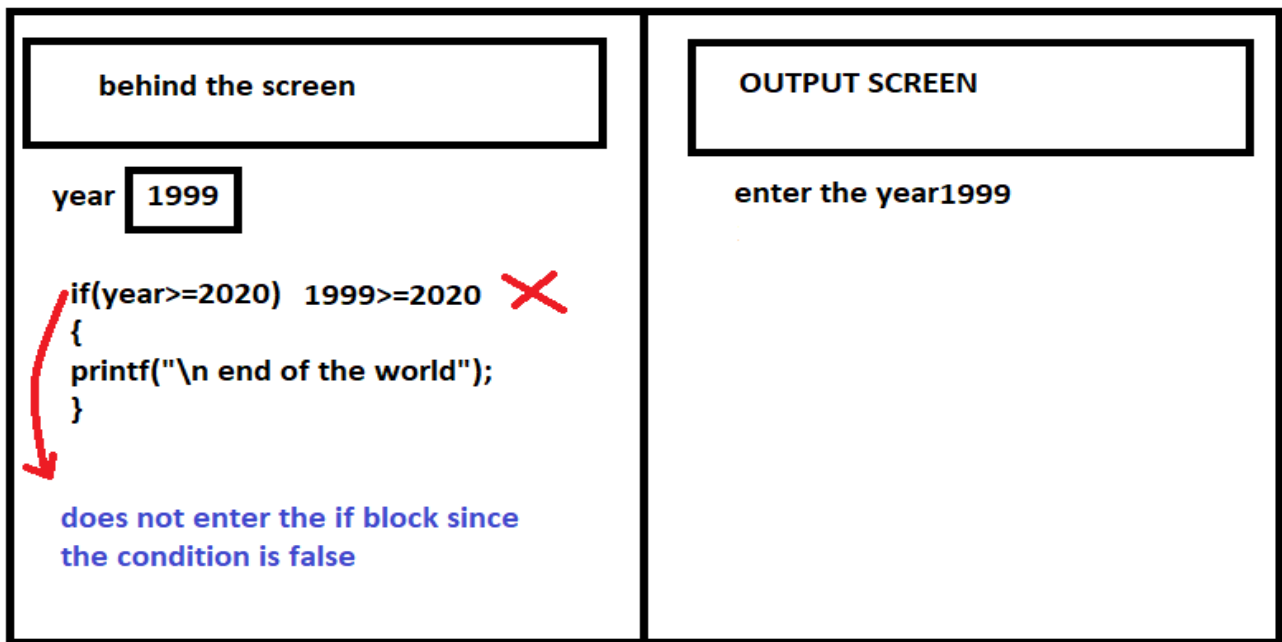
## OUTPUT 2:

```
C:\Users\CHANDANA\Desktop\c prgm\if.exe

enter the year1999

-----
Process exited after 10.02 seconds with return value 1999
Press any key to continue . . .
```

## EXPLANATION:



**P3:**Program to input two numbers and print excellent if sum of two numbers is greater than 500

Sample output:

```
enter two numbers500  
20  
  
excellent  
-----  
Process exited after 22.26 seconds with return value 11  
Press any key to continue . . .
```

## if-else statement

### Syntax

```
if(condition) {  
    Block of statements;  
}  
else  
{
```

Block of statements;

}

If the condition is true then the block of statements inside the if block gets executed otherwise block of statements in the else block will get executed

### **Example program 5(ep5):**

*/\*program to print adult if age greater than or equal to 18 otherwise print child\*/*

**#include<stdio.h>**

**void main()**

**{**

**int age;**

**printf("\n enter your age");**

**scanf("%d",&age);**

**if(age>=18) // if the condition is true then it enters inside the 'if' block**

**{**

**printf("\n adult");**

**}**

**else**

**{**

**printf("\n child");// if the condition is false then it enters inside the 'else' block**

**}**

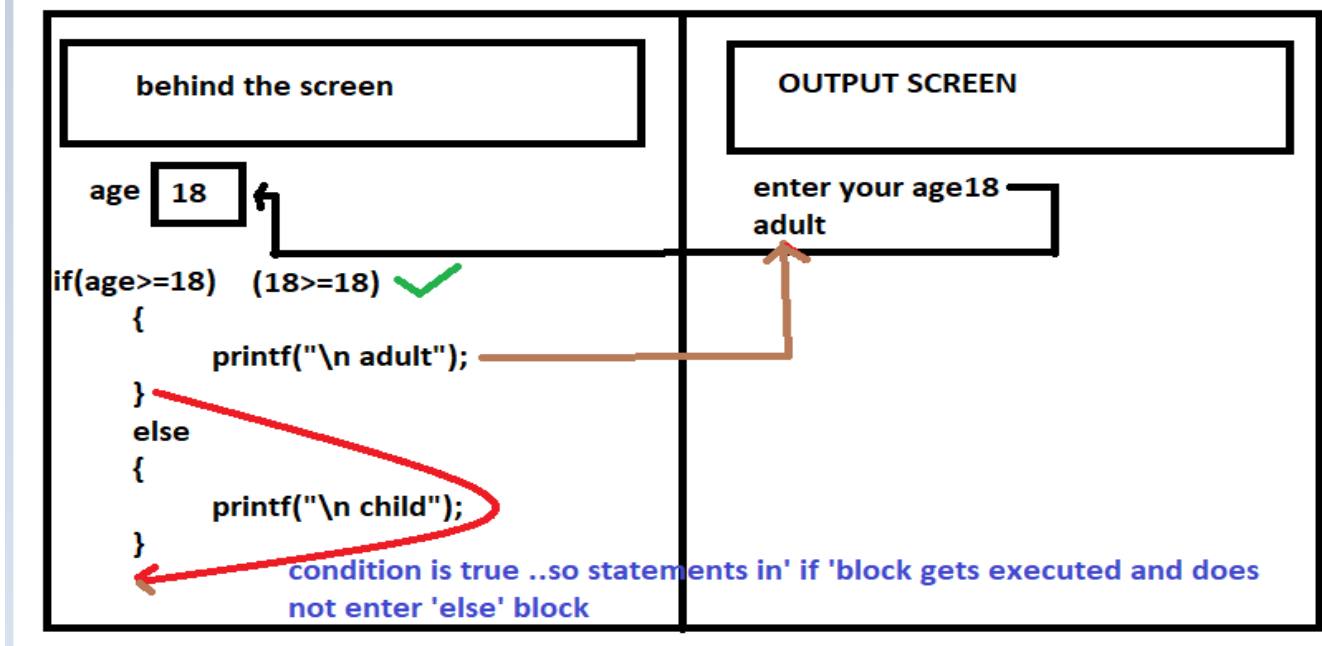
**}**

**OUTPUT1:**



```
enter your age18
adult
-----
Process exited after 11.95 seconds with return value 7
Press any key to continue . . . _
```

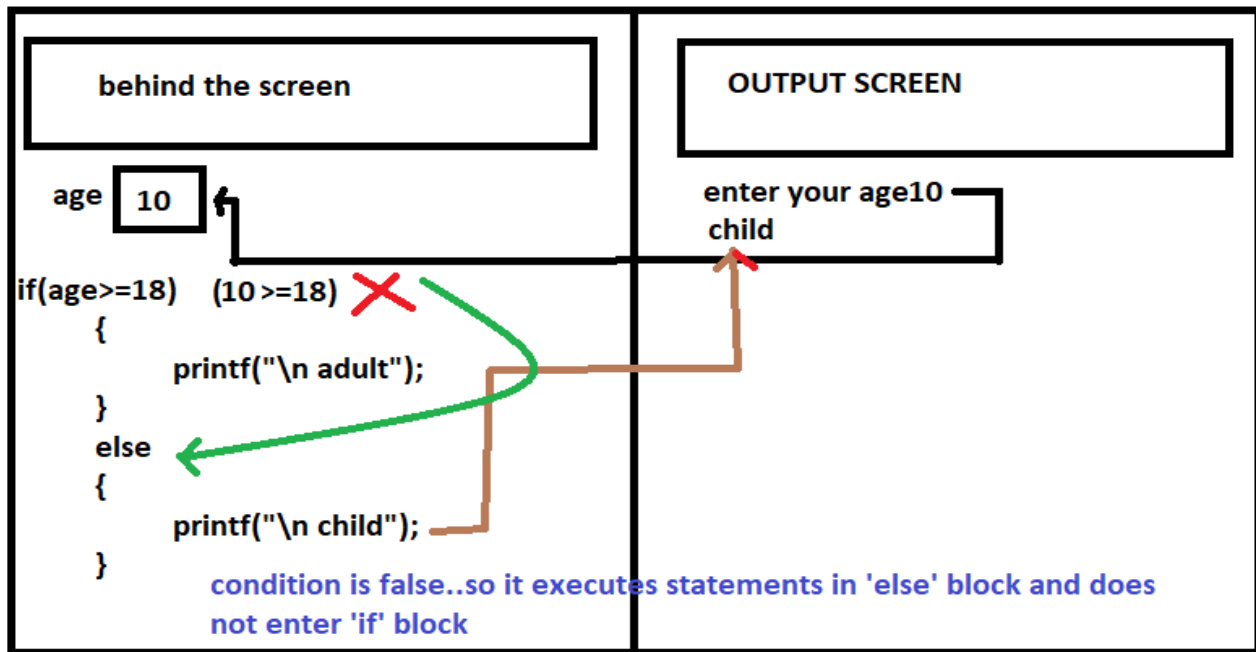
**EXPLANATION:**



**OUTPUT2:**

```
enter your age10
child
-----
Process exited after 11.67 seconds with return value 7
Press any key to continue . . . _
```

**EXPLANATION:**



**P4:** Program to input a number and print even if it is even number else print odd

Sample output:

7

odd

**P5:** Program to print wheather the input character is vowel or consonant

Hint: even numbers are divisible by 2

Sample output:

E

Vowel

## Nested-if statement

### Syntax

```
if(condition1)
```

```
{
```

```
Block of statements1;
```

```
if(condition2)
```

```
{  
Block of statements2;  
}  
}
```

If condition1 is true and condition2 is also true then both block of statement1 and block of statement2 gets executed

If condition1 is true and condition2 is false then only block of statement1 gets executed

If condition1 is false and condition2 is also false then none of the statements gets executed

### **Example program 6(ep6):**

*/\*program to print divisible by number if it is divisible by 5 and print divisible by 7 if it is divisible by 7 also\*/*

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

```
void main()
```

```
{
```

```
int n;
```

```
printf("\n enter number");
```

```
scanf("%d",&n);
```

```
if(n%5==0) // if the condition is true then it enters inside the 'if' block
```

```
{
```

```
printf("\n divisible by 5");
```

```
    if(n%7==0) // if the condition is true then it enters inside the 'if' block
```

```
{
```

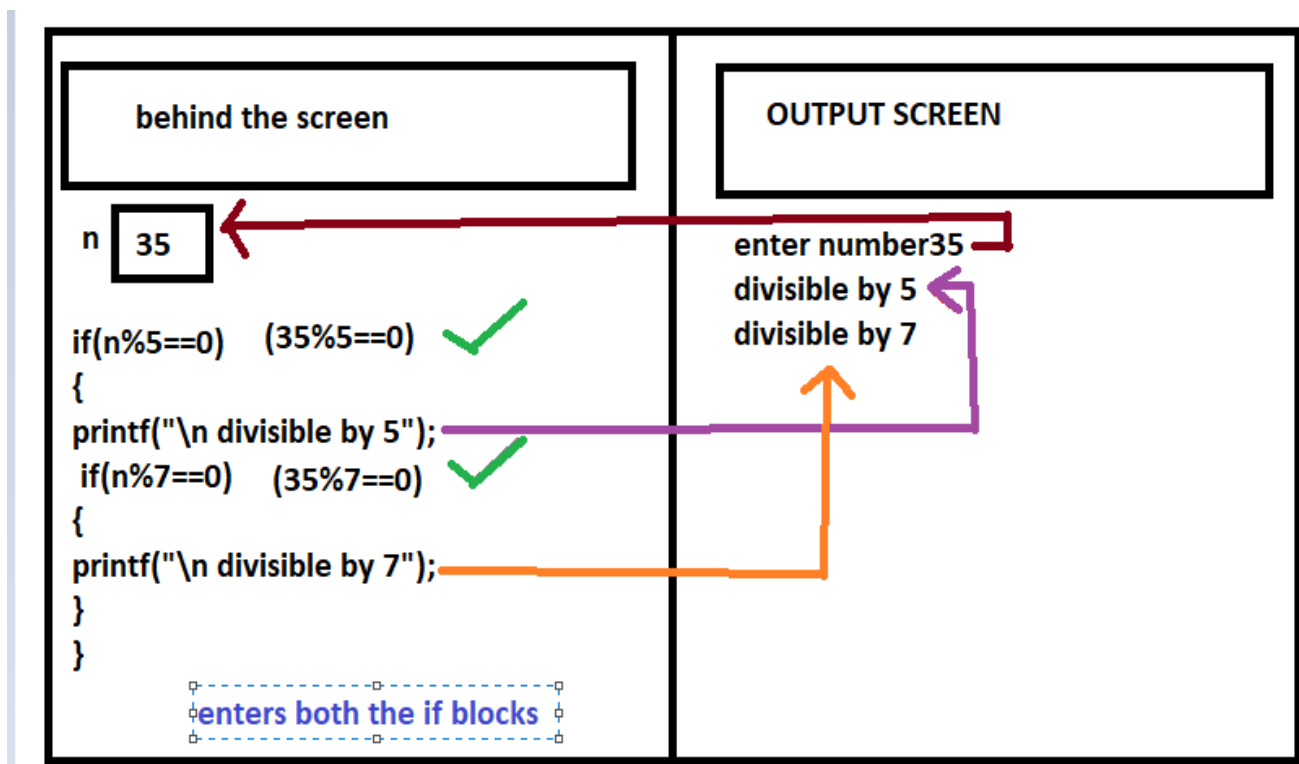
```
printf("\n divisible by 7");
```

```
}  
}  
}
```

## OUTPUT1:

```
enter number35  
  
divisible by 5  
divisible by 7  
-----  
Process exited after 16.23 seconds with return value 16  
Press any key to continue . . .
```

## EXPLANATION:



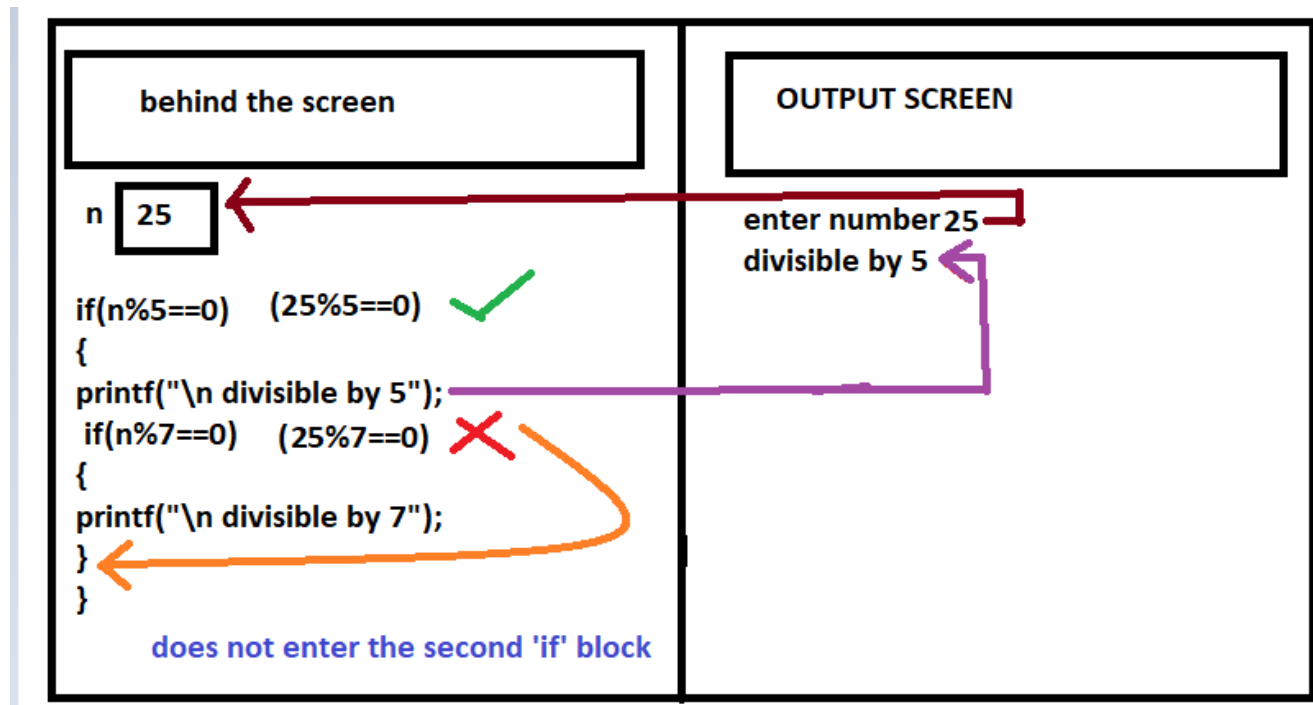
## OUTPUT2:

```
enter number25

divisible by 5

-----
Process exited after 5.97 seconds with return value 4
Press any key to continue . . .
```

## EXPLANATION:



**P6:** Program to print good score if marks greater than or equal to 60 and print well done also if marks greater than or equal to 90

**Sample output:**

98

Good score

Well done

**if-else if ladder**

**Syntax:**

```

if (condition1)
{
    statement;
}
else if(condition2)
{
    statement;
}
.
.
else
{
statement;
}

```

If the condition1 is true it enters the 'if' block otherwise it checks condition2 if it is true then it enters respective 'else if' block and even if the condition2 is false it checks the next condition if it is true then respective block gets executed and if none of the conditions are true then the 'else' block get executed.

**Example program 7(ep7):**

*/\*program to print whether the input character is an alphabet or a number or a special character \*/*

```

#include<stdio.h>

void main()
{
char n;

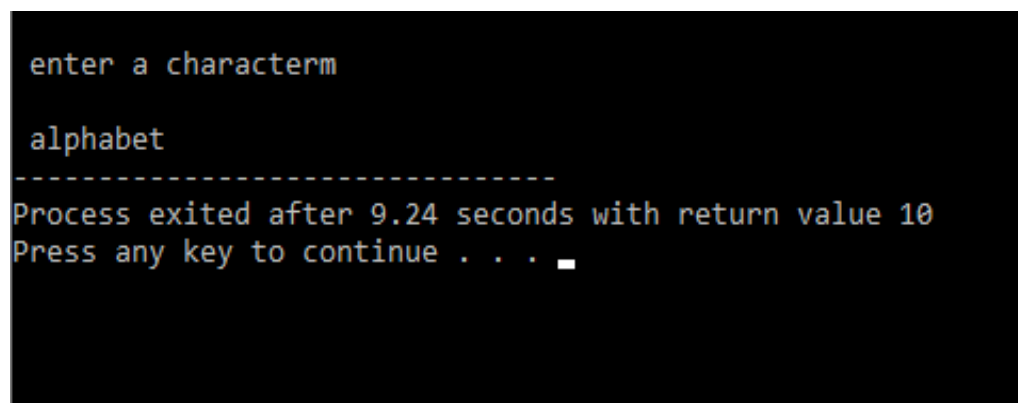
```

```

printf("\n enter a character");
scanf("%c",&n);
    if((n>='A'&&n<='Z')||(n>='a'&&n<='z'))//ascii values of the character gets compared
    {
        printf("\n alphabet");
    }
else if(n>='0'&&n<='9')//it enters this block if input character is a number
{
    printf("\n number");
}
else //other than alphabets and numbers others are special characters
{
    printf("\n special character");
}
}

```

## OUTPUT1:

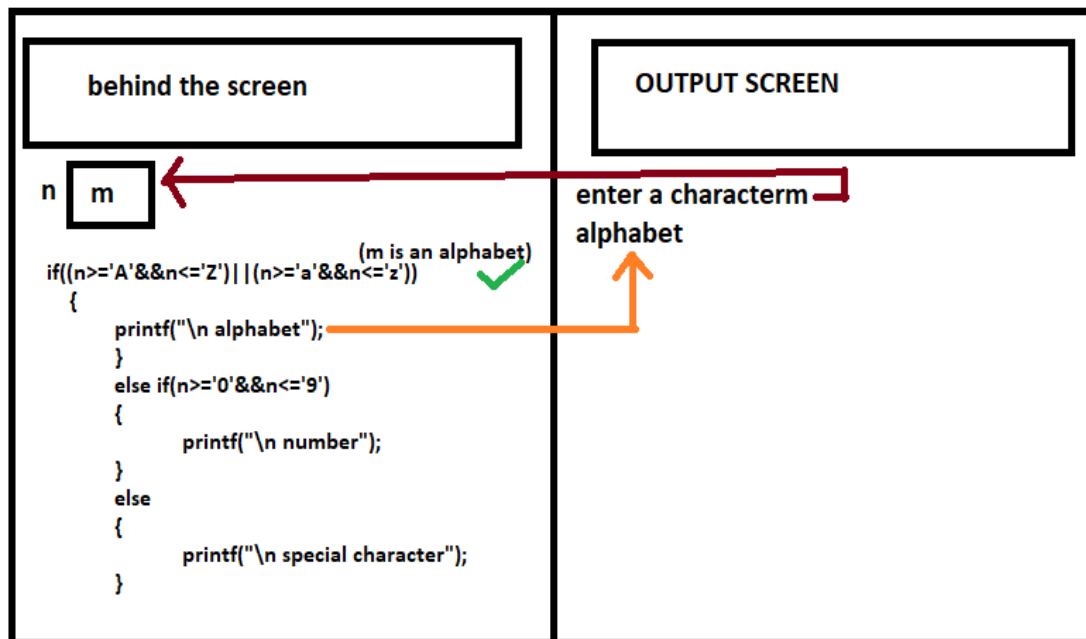


```

enter a characterterm
alphabet
-----
Process exited after 9.24 seconds with return value 10
Press any key to continue . . . 

```

## EXPLANATION:



## OUTPUT2:

```

enter a character7
number
-----
Process exited after 14.59 seconds with return value 8
Press any key to continue . . .

```

## OUTPUT3:

```

enter a character*
special character
-----
Process exited after 12.86 seconds with return value 19
Press any key to continue . . .

```



**P7:**Program to input a character and print add if input character is + ,print sub if input character is -,print product if input character is \* else print wrong

**Sample output:**

\*

Product

### **switch statement**

Switch case statements are a substitute for long if statements that compare a variable to several integral values.

**Syntax:**

switch (n)

{

case 1:

code to be executed if n = 1;

break;

case 2:

code to be executed if n = 2;

break;

.

.

.

default:

code to be executed if n doesn't match any cases ;

}

If n matches with the case value then the corresponding case gets executed else the default statement gets executed

Every case must be ended with break statement( to get out of switch statement).

### **Example program 8(ep8):**

*/\*program to build simple calculator using switch statement \*/*

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

```
void main()
```

```
{
```

```
char c;
```

```
int a,b;
```

```
printf("\n input two numbers\n");
```

```
scanf("%d",&a);
```

```
scanf("%d",&b);
```

```
printf("\n enter symbol \n 1. + for addition \n 2. - for subtraction \n 3.  
* for product \n 4. / for division\n");
```

```
scanf("%c",&c);
```

```
switch(c){
```

```
case '+': // if c is + it enters this block
```

```
printf("\nsum is %d",a+b);
```

```
break;
```

```
case '-': // if c is - it enters this block
```

```
printf("\ndiff is %d",a-b);
```

```
break;
```

```
case '*': // if c is * it enters this block
```

```
printf("\nproduct is %d",a*b);
```

```
break;
```

case '/': // if c is / it enters this block

printf("\ndivision result is %f", (float)a/(float)b);

//since result may be in decimals we converted integers to float

break;

default:

printf("\n wrong choice");

}

}

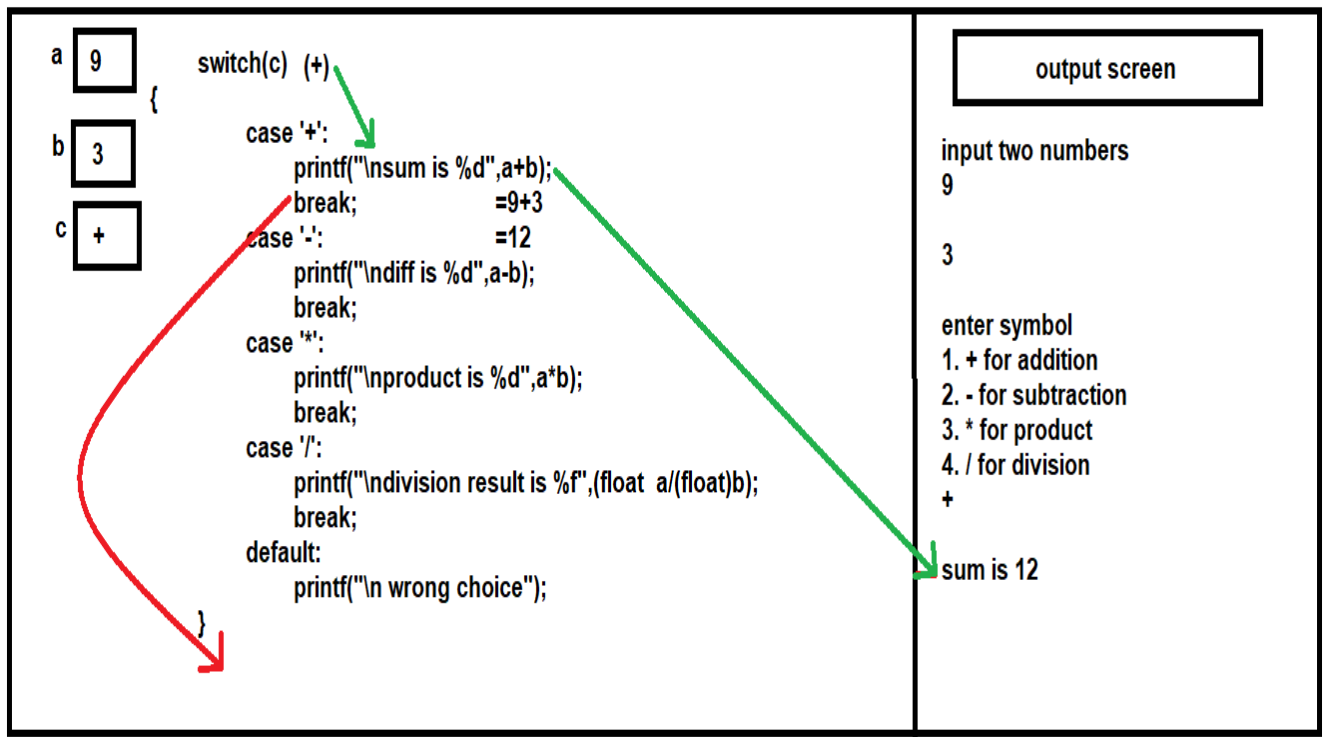
## OUTPUT1:

```
input two numbers
9
3

enter symbol
1. + for addition
2. - for subtraction
3. * for product
4. / for division
+

sum is 12
-----
Process exited after 7.229 seconds with return value 10
Press any key to continue . . .
```

## EXPLANATION:



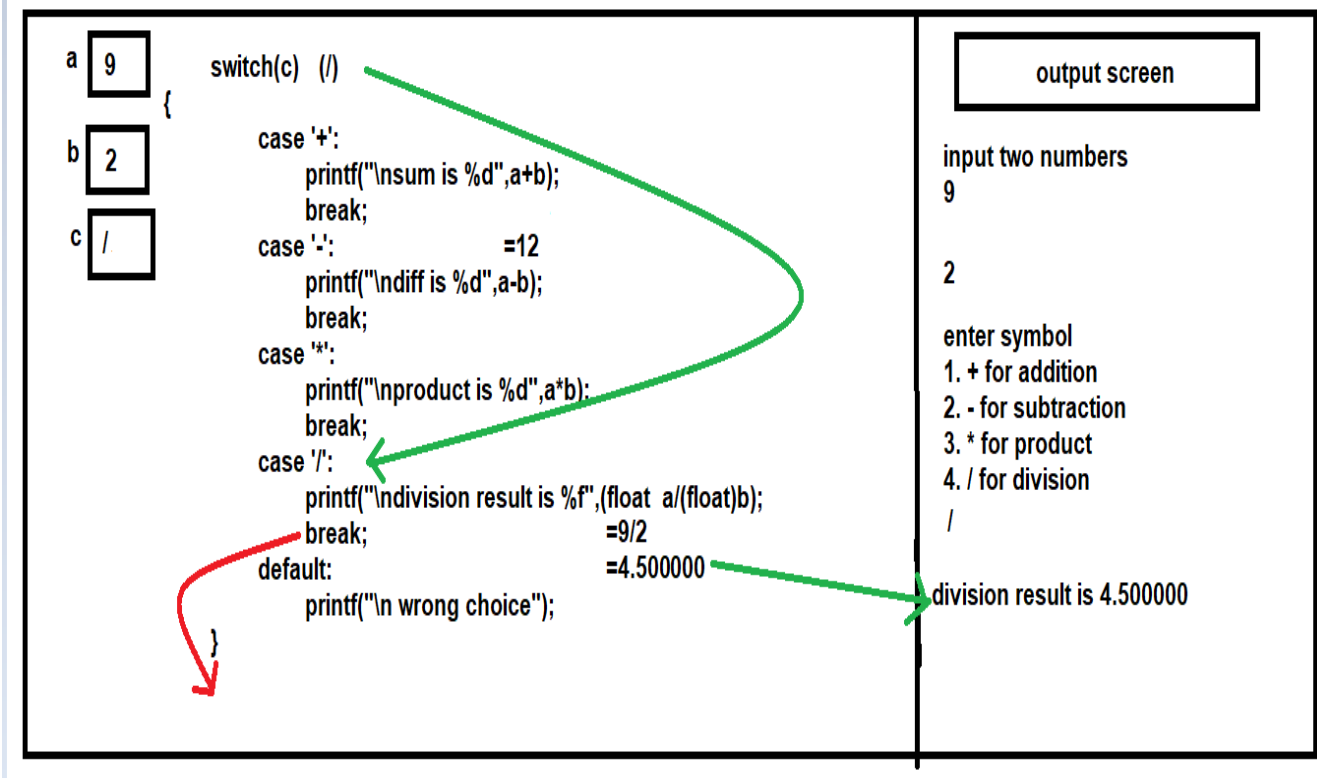
## OUTPUT2:

```
input two numbers
9
2

enter symbol
1. + for addition
2. - for subtraction
3. * for product
4. / for division
/

division result is 4.500000
-----
Process exited after 10.65 seconds with return value 28
Press any key to continue . . .
```

## EXPLANATION:



**P8:** Program to take a input number, and print square of that number if the choice is 2, print cube of the number if the choice is 3, print 4 th power of the number if choice is 4 and wrong choice if the choice is other than that (using switch statement)

**Sample output:**

Enter number : 5

Enter choice : 3

Cube=125

**Lets get back to the jump statements after getting to know about loops in c.**

## LOOPS

Loops are used to repeat a block of statements until the specified condition becomes false. There are 3 types of loops in C. They are

1. while loop
2. do while loop

### 3. for loop

#### 1. while loop:

Syntax:

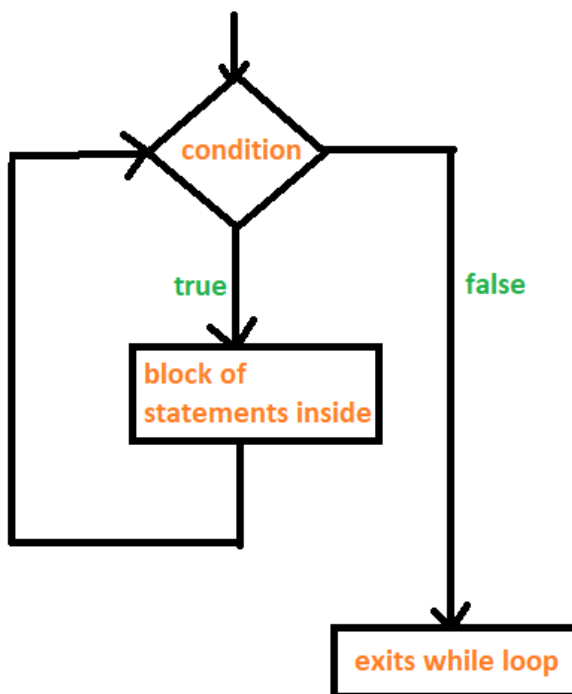
```
while(condition)
```

```
{
```

```
Block of statements;
```

```
}
```

Flow chart:



If the condition is true then the block of statements inside the while loop gets executed and again its repeats till the condition is true and exits the loop when the condition fails.

**Example program 9(ep9):**

```
/*program to print numbers from 1 to 5 */
```

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

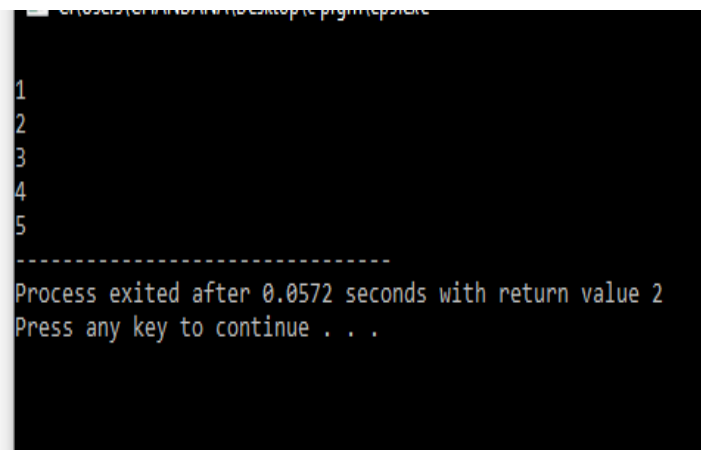
```
int main()
```

```

{
int i=1;
while(i<=5) //block of statements in this loop gets executed until the condition becomes false
{
printf("\n%d",i);
i++; //incrementing i to the next value
}
}

```

## OUTPUT:



```

1
2
3
4
5
-----
Process exited after 0.0572 seconds with return value 2
Press any key to continue . . .

```

## EXPLANATION:

i value ++	Loop condition while(i<=5)		Output printf("\n%d",i);
1	1<=5	yes	1
2	2<=5	yes	2
3	3<=5	yes	3
4	4<=5	yes	4
5	5<=5	yes	5
6	6<=5	no	

## Example program 10(ep10):

/\*program to input a number n and print that nth table using while loop\*/

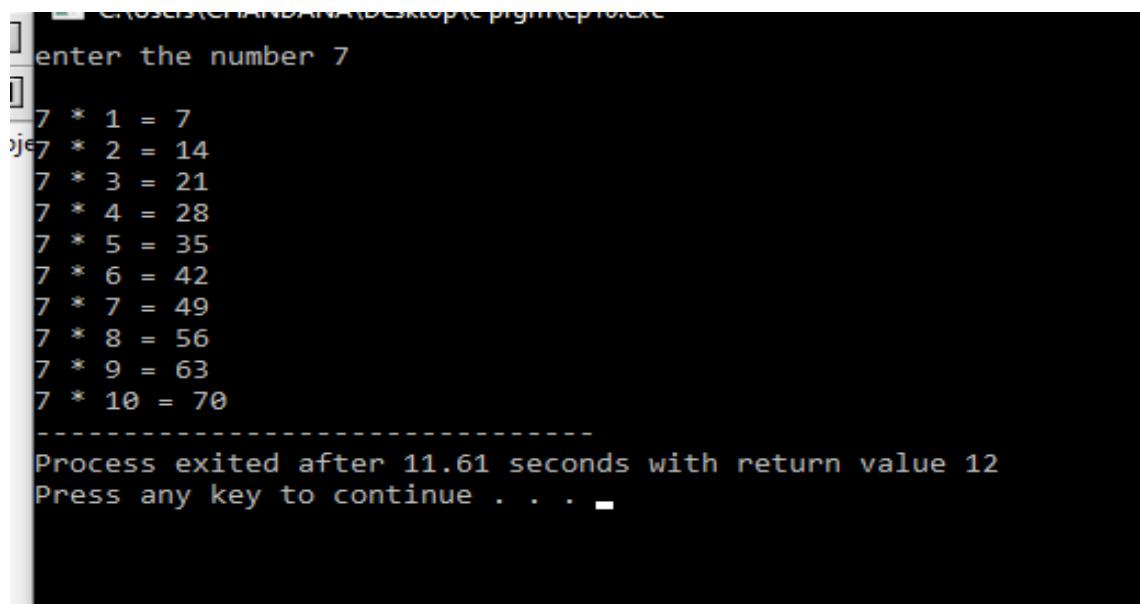
```

#include<stdio.h>

int main()
{
int n;
printf("enter the number ");
scanf("%d",&n);
int i=1;
while(i<=10)//iterates until condition gets false
{
printf("\n%d * %d = %d",n,i,n*i);
i++;
}
}

```

## OUTPUT:



```

C:\Users\reni\OneDrive\Desktop>g++ program10.c
enter the number 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
-----
Process exited after 11.61 seconds with return value 12
Press any key to continue . . .

```

## EXPLANATION:

i value	Condition		Output
i++	while(i<=10)		printf("\n%d * %d = %d",n,i,n*i);



			n=7
1	1<=10	yes	7 * 1 = 7
2	2<=10	yes	7 * 2 = 14
3	3<=10	yes	7 * 3 = 21
4	4<=10	yes	7 * 4 = 28
5	5<=10	yes	7 * 5 = 35
6	6<=10	yes	7 * 6 = 42
7	7<=10	yes	7 * 7 = 49
8	8<=10	yes	7 * 8 = 56
9	9<=10	yes	7 * 9 = 63
10	10<=10	yes	7 * 10 = 70
11	11<=10	no	

### Example program 11(ep11):

*/\*program to check whether the input number is palindrome or not Hint :a number is said to be palindrome if the reverse of the number is equal to original number. Ex:121,12721,9876789 etc..\*/*

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

```
int main()
```

```
{
```

```
int n,n1,i,rev=0;//rev should be initialized with zero
```

```
printf(" enter a number : ");
```

```
scanf("%d",&n);
```

```
n1=n; //storing the number in another variable
```

```
while(n1>0) //logic to rev a number
```

```
{
```

```
rev=rev*10+n1%10;
```

```
n1=n1/10;
```

```
}
```

```

if(rev==n)//checking whether the reversed number is equal to the original number
{
printf("\n palindrome");
}
else
{
printf("\n not a palindrome");
}
}

```

### OUTPUT1:

```

C:\Users\CHANDANA\Desktop\c prgm\ep11.exe
enter a number : 12345
not a palindrome
-----
Process exited after 3.278 seconds with return value 18
Press any key to continue . . .

```

### EXPLANATION:

Initially  $n1=n=12345$

Condition while( $n1>0$ )		$rev=rev*10+n1\%10;$	$n1=n1/10;$
$12345>0$	yes	$=0*10+12345\%10$ $=0+5$ $=5$	$=12345/10$ $=1232$
$1234>0$	yes	$=5*10+1234\%10$ $=50+4$ $=54$	$=1234/10$ $=123$

<b>123&gt;0</b>	<b>yes</b>	<b>=54*10+123%10 =540+3 =543</b>	<b>=123/10 =12</b>
<b>12&gt;0</b>	<b>yes</b>	<b>=543*10+12%10 =5430+2 =5432</b>	<b>=12/10 =1</b>
<b>1&gt;0</b>	<b>yes</b>	<b>=5432*10+1%10 =54320+1 =54321</b>	<b>=1/10 =0</b>
<b>0&gt;0</b>	<b>no</b>		

**rev=54321**

**n=12345**

**So, rev not equal to n**

**So, 12345 is not a palindrome**

## **OUTPUT2:**

```
enter a number : 12321

palindrome
-----
Process exited after 6.502 seconds with return value 12
Press any key to continue . . .
```

## **EXPLANATION:**

Initially **n1=n=12321**

<b>Condition while(n1&gt;0)</b>		<b>rev=rev*10+n1%10;</b>	<b>n1=n1/10;</b>
<b>12321&gt;0</b>	<b>yes</b>	<b>=0*10+12321%10 =0+1</b>	<b>=12321/10 =1232</b>

		=1	
1232>0	yes	=1*10+1232%10 =10+2 =12	=1232/10 =123
123>0	yes	=12*10+123%10 =120+3 =123	=123/10 =12
12>0	yes	=123*10+12%10 =1230+2 =1232	=12/10 =1
1>0	yes	=1232*10+1%10 =12320+1 =12321	=1/10 =0
0>0	no		

**rev=n**

**12321=12321**

**Therefore,12321 is a palindrome**

**P9:**Program to take a input number, and print 12 th table if input number is 12,15 th table if input number is 15 ,27 th table if input number is 27 , 51<sup>st</sup> table if input number is 51 and “sorry,no match” if input number is other than those numbers(using switch and while loops)

**Sample output:**

12

12\*1=12

12\*2=24

12\*3=36

12\*4=48

$$12*5=60$$

$$12*6=72$$

$$12*7=84$$

$$12*8=96$$

$$12*9=108$$

$$12*10=120$$

**P10:**program to check whether the input num is armstrong or not

**Hint:**A number is said to be armstrong number if sum of power of number of digits of the individual number is equal to original number

$$\begin{aligned}\text{Ex1: } 153 &= 1*1*1 + 5*5*5 + 3*3*3 \\ &= 153\end{aligned}$$

$$\begin{aligned}\text{Ex2: } 1634 &= 1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4 \\ &= 1634\end{aligned}$$

**Sample output:**

153

Armstrong number

## **2.do-while loop:**

Syntax:

do

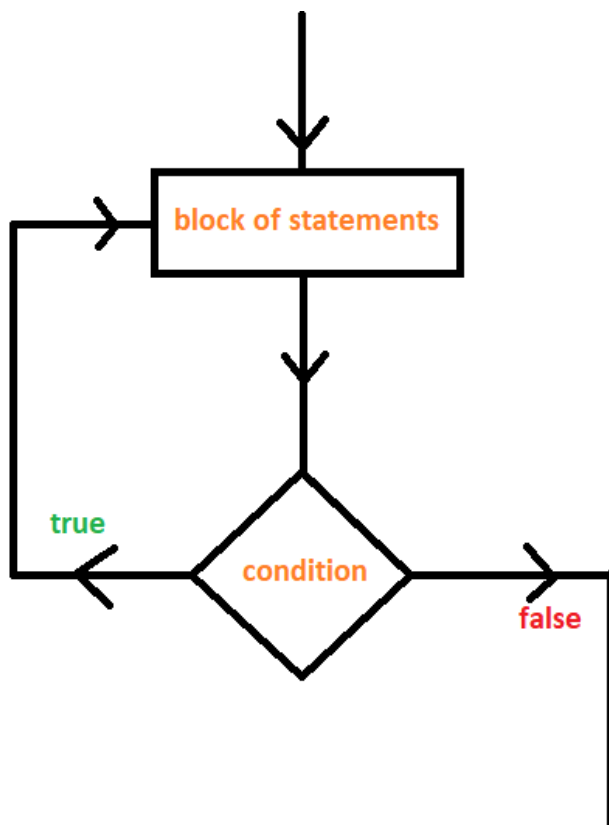
{

Block of statements;

}while(condition);

It ends with a semicolon.

Flow chart:



At the first time it enters the loop and executes the block of statements and then it checks the condition. if the condition is true then it again enters from start of the loop and executes the block of statements and this is repeated till the condition is true and exits the loop when the condition fails.

### Example program 12(ep12):

*/\*program to print even numers from 1 to 10\*/*

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

```
int main()
```

```
{
```

```
int i=2;
```

```
do
```

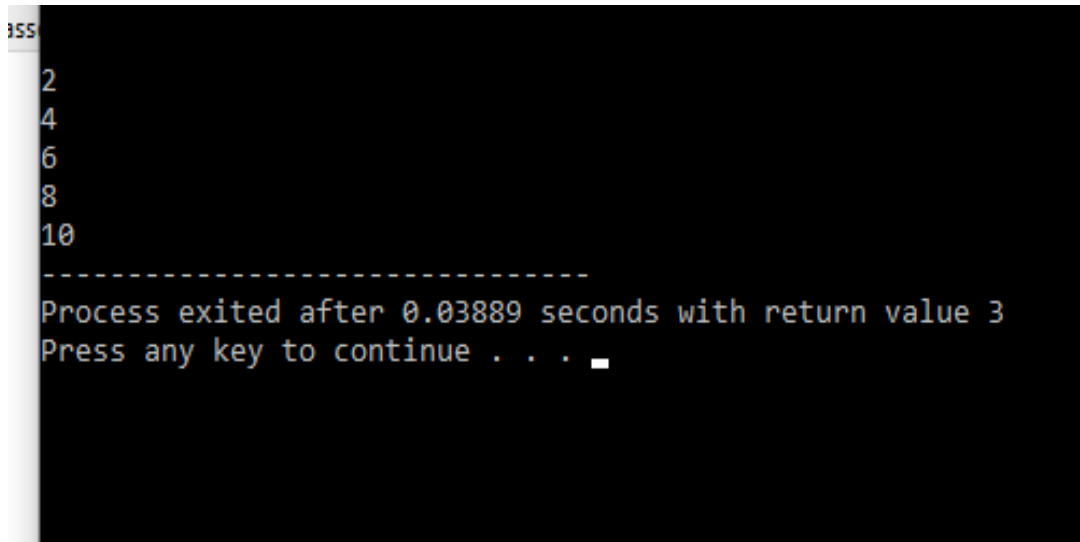
```
{
```

```
printf("\n%d",i);
```

```
i+=2; // incrementing value of i to plus 2
```

```
}while(i<=10);  
}
```

## OUTPUT:



```
2  
4  
6  
8  
10  
-----  
Process exited after 0.03889 seconds with return value 3  
Press any key to continue . . .
```

## EXPLANATION:

<code>printf("\n%d",i);</code> Initially i=2	i value i=i+2;	Condition while(i<=10)	
2	4	4<=10	yes
4	6	6<=10	yes
6	8	8<=10	yes
8	10	10<=10	yes
10	12	12<=10	no

**P11:**Program to input a number n and print sum of n numbers using do while loop

**Hint:** sum must be initialized to zero first(else at the beginning some garbage value will be added) and then run the loop till n while adding each value to sum in each iteration.

sum=sum+i;(i from 1 to n)

**Sample output:**

5

Sum of 5 num is : 15

### **3.for loop:**

Syntax:

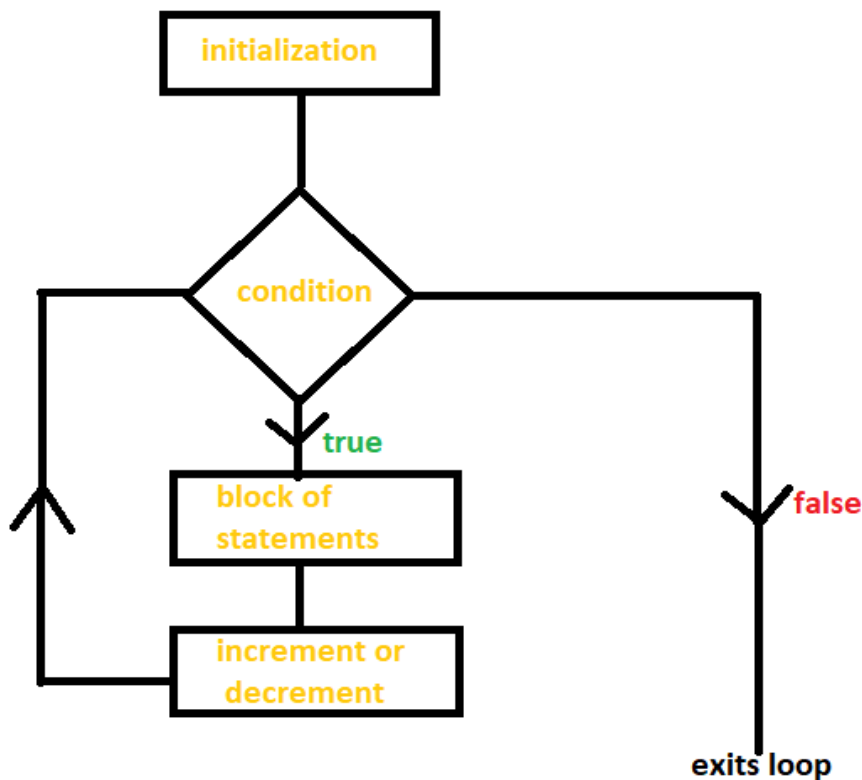
```
for(initializtion ; condition ; increment or decrement)
```

```
{
```

```
Block of statements;
```

```
}
```

Flow chart:



First a value is intialized to a variable and then condition gets checked if it is true then block of statements inside the for loop gets executed and then the variable value gets incremented or decremented and then again the condition gets checked and this process continues till the condition is true and exits the loop when the condition is false.

### **Example program 13(ep13):**

*/\*program to input a number n and print sum of even numbers till n\*/*



```
#include<stdio.h>

int main()
{
    int i,n,sum=0;
    printf("enter number : ");
    scanf("%d",&n);
    for(i=2;i<=n;i=i+2) //incrementing i value to plus 2
    {
        sum=sum+i; //calculating sum
    }
    printf("sum of even numbers : %d",sum);//printing sum
    return 0;
}
```

**OUTPUT:**

```
C:\Users\CHANDANA\Desktop\c\prgm\ep1\exe
enter number : 7
sum of even numbers : 12
-----
Process exited after 6.959 seconds with return value 0
Press any key to continue . . .
```

**EXPLANATION:**logic part

Initially  $i = 2, \text{sum} = 0$

n=7(from input)

Condition		Statement(behind screen)	Increment
-----------	--	--------------------------	-----------

<b>i&lt;=n</b>		<b>sum=sum+i</b>	<b>i=i+2</b>
<b>2&lt;=7</b>	<b>yes</b>	=0+2 =2	=2+2 =4
<b>4&lt;=7</b>	<b>yes</b>	=2+4 =6	=4+2 =6
<b>6&lt;=7</b>	<b>yes</b>	=6+6 =12	=6+2 =8
<b>8&lt;=7</b>	<b>no</b>		

When it exits the loop it prints the value stored in sum i.e sum of even numbers till n.

### Example program 14(ep14):

*/\*program to check whether the input number is prime or not-a prime numbers has only two factors i.e number 1 and itself\*/*

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

```
int main()
```

```
{
```

```
int i,n,count=0;
```

```
printf("enter number : ");
```

```
scanf("%d",&n);
```

```
for(i=1;i<=n;i++) //incrementing i value to plus 2
```

```
{
```

```
if(n%i==0) //checking whether the number is divisible or not
```

```
{
```

```
count++; //incrementing count value when a factor is found
```

```
}
```

```
}
```

```
if(count==2) //if count is 2 then it is prime number
```

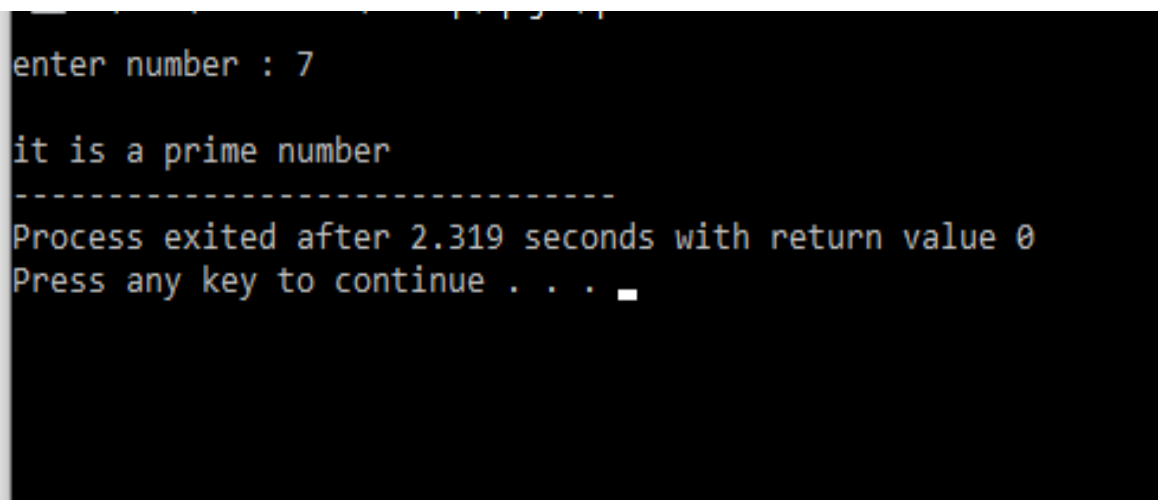
```

{
printf("\nit is a prime number ");
}
else
{
printf("\nit is not a prime number");
}

return 0;
}

```

### **OUTPUT1:**



```

enter number : 7

it is a prime number
-----
Process exited after 2.319 seconds with return value 0
Press any key to continue . . .

```

### **EXPLANATION:**

initially

n=7(from input)

i=1

count=0

<b>For-condition</b> <b>i&lt;=n</b>		<b>If-condition</b> <b>n%i==0</b>		<b>count++</b> <b>(only</b> <b>enters if</b>	<b>i value</b> <b>i++</b>
--	--	--------------------------------------	--	--	------------------------------

				'if' condition is true	
1<=7	yes	7%1==0	yes	1	2
2<=7	yes	7%2==0	no		3
3<=7	yes	7%3==0	no		4
4<=7	yes	7%4==0	no		5
5<=7	yes	7%5==0	no		6
6<=7	yes	7%6==0	no		7
7<=7	yes	7%7==0	yes	2	8
8<=7	no				

By the time it exits the loop count value for the given input number 7 is 2 I.e it has two factors,so the given input number is prime

**P12:**Program to input a number and print all odd numbers in reverse order till 1 using for loop and print their sum in next line

**Hint:** for(i=n;i>=1;i-=2) if n is odd

for(i=n-1;i>=1;i-=2) if n is even

**Sample output:**

Enter a number : 12

11 9 7 5 3 1

36

**Example program 15(ep15):**

*/\*program to input a number and print all the prime numbers till that number\*/*

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

```
int main()
```

```
{
```

```
int i,n,count=0,j;
```

```

printf("enter number : ");
scanf("%d",&n);
for(i=1;i<=n;i++) //running i value from 1 to n
{
    for(j=1;j<=i;j++) //for each value of i checking whether it is a prime number or not
    {
        if(i%j==0) //checking whether the number is divisible or not
        {
            count++; //incrementing count value when a factor is found
        }
    }
    if(count==2) //if count is 2 then it is prime number
    {
        printf("\n%d",i);
    }
    count=0; //initializing count to 0 at end to get used for next value of i
}
return 0;
}

```

**OUTPUT:**

```
enter number : 5
2
3
5
-----
Process exited after 2.722 seconds with return value 0
Press any key to continue . . .
```

**EXPLANATION:**

n=5(from input)

When i=1;

1<=5 **yes** i<=n;

j=1

For-condition j<=i		If-condition i%j==0		count++ (only enters if 'if' condition is true	j value j++
1<=1	<b>yes</b>	1%1==0	<b>yes</b>	1	2
2<=1	<b>no</b>				

By the end of the loop count value is 1-so 1 is not a prime number

Again count=0;

i++ i.e i=2

2<=5 **yes** i<=n;

j=1

For-condition j<=i		If-condition i%j==0		count++ (only enters if 'if'	j value j++
-----------------------	--	------------------------	--	---------------------------------------	----------------

				<b>condition is true</b>	
<b>1&lt;=2</b>	<b>yes</b>	<b>2%1==0</b>	<b>yes</b>	<b>1</b>	<b>2</b>
<b>2&lt;=2</b>	<b>yes</b>	<b>2%2==0</b>	<b>yes</b>	<b>2</b>	<b>3</b>
<b>3&lt;=2</b>	<b>no</b>				

By the end the value of count is 2 . For the i value 2 it has 2 factors so, it is a prime number. It prints the value of i on the screen.

Again, count=0

i++ i.e i=3

3<=5 **yes**

<b>For-condition j&lt;=i</b>		<b>If-condition i%j==0</b>		<b>count++ (only enters if 'if' condition is true</b>	<b>j value j++</b>
<b>1&lt;=3</b>	<b>yes</b>	<b>3%1==0</b>	<b>yes</b>	<b>1</b>	<b>2</b>
<b>2&lt;=3</b>	<b>yes</b>	<b>3%2==0</b>	<b>no</b>		<b>3</b>
<b>3&lt;=3</b>	<b>yes</b>	<b>3%3==0</b>	<b>yes</b>	<b>2</b>	<b>4</b>
<b>4&lt;=3</b>	<b>no</b>				

By the end, the value of count is 2 . For the i value 3 it has 2 factors so, it is a prime number. It prints the value of i on the screen.

Again, count=0

i++ i.e i=4

4<=5 **yes** i<=n;

j=1

<b>For-condition j&lt;=i</b>		<b>If-condition i%j==0</b>		<b>count++ (only enters if</b>	<b>j value j++</b>
----------------------------------	--	--------------------------------	--	--	------------------------

				<b>'if' condition is true</b>	
<b>1&lt;=4</b>	<b>yes</b>	<b>4%1==0</b>	<b>yes</b>	<b>1</b>	<b>2</b>
<b>2&lt;=4</b>	<b>yes</b>	<b>4%2==0</b>	<b>yes</b>	<b>2</b>	<b>3</b>
<b>3&lt;=4</b>	<b>yes</b>	<b>4%3==0</b>	<b>no</b>		<b>4</b>
<b>4&lt;=4</b>	<b>yes</b>	<b>4%4==0</b>	<b>yes</b>	<b>3</b>	<b>5</b>
<b>5&lt;=4</b>	<b>no</b>				

By the end ,the value of count is 3 . For the i value 4 it has 3 factors so,it is not a prime number.

Again,count=0

i++ i.e i=5

5<=5 **yes** i<=n;

j=1

<b>For-condition j&lt;=i</b>		<b>If-condition i%j==0</b>		<b>count++ (only enters if 'if' condition is true</b>	<b>j value j++</b>
<b>1&lt;=5</b>	<b>yes</b>	<b>5%1==0</b>	<b>yes</b>	<b>1</b>	<b>2</b>
<b>2&lt;=5</b>	<b>yes</b>	<b>5%2==0</b>	<b>no</b>		<b>3</b>
<b>3&lt;=5</b>	<b>yes</b>	<b>5%3==0</b>	<b>no</b>		<b>4</b>
<b>4&lt;=5</b>	<b>yes</b>	<b>5%4==0</b>	<b>no</b>		<b>5</b>
<b>5&lt;=5</b>	<b>yes</b>	<b>5%5==0</b>	<b>yes</b>	<b>2</b>	<b>6</b>
<b>6&lt;=5</b>	<b>no</b>				

By the end ,the value of count is 2 . For the i value 5 it has 2 factors so,it is a prime number.It prints the value of i on the screen.

Again,count=0



i++ i.e i=6

6<=5 **no** i<=n;

So, it will exit even from the i loop.

**P13:** Program to input a number and print all armstrong numbers till that number

**Sample output:**

Enter a number : 200

Armstrong numbers till 200 are:

0 1 2 3 4 5 6 7 8 9 153

**P14:** Program to input a number and print all palindrome numbers till that number

**Sample output:**

Enter a number : 30

Palindrome numbers till 200 are:

0 1 2 3 4 5 6 7 8 9 11 22

Now, let's see the jump statements in c

## **JUMP STATEMENTS**

- a. break
- b. continue
- c. goto
- d. return

- a. **break** : It is a keyword in c which is used to exit the loop. It is also used in switch statements.

**Example program 16(ep16):**

/\*program to print all the number till the number which is divisible by  
boh 7 and 5 is found\*/

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

```
int main()
```

```
{
```

```
    int i;
```

```
    i=1;
```

```
    while(i>=0)
```

```
    {
```

```
        if(i%5==0&& i%7==0)//checking whether the number is divisible by both 7 and 5
```

```
        {
```

```
            break; //using break to exit the loop if that number is found
```

```
        }
```

```
        printf("%d ",i);
```

```
        i++;
```

```
    }
```

```
    printf("\n i am out of loop");
```

```
}
```

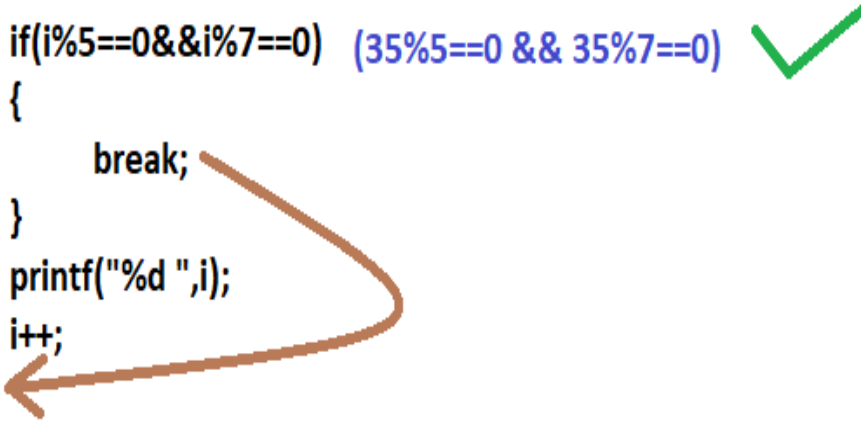
**OUTPUT:**

```
C:\Users\CHANDANA\Desktop\c prgm\ep16.exe
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
i am out of loop
-----
Process exited after 0.1148 seconds with return value 18
Press any key to continue . . .
```

## EXPLANATION:

WHEN **i=35**

```
while(i>=0)
{
    if(i%5==0&& i%7==0) (35%5==0 && 35%7==0) ✓
    {
        break;
    }
    printf("%d ",i);
    i++;
}
printf("\n i am out of loop");
```



In case of nested loops it will only break from the corresponding loop and the process goes as normal.

### **Example program 17(ep17):**

*/\*program to input a number n print all the prime numbers till n except the numbers which end with 3\*/*

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

```

int main()
{
    int i,n,count=0,j;
    printf("enter number : ");
    scanf("%d",&n);
    for(i=1;i<=n;i++) //running i value from 1 to n
    {
        for(j=1;j<=i;j++) //for each value of i checking whether it is a prime number or not
        {

            if(i%j==0) //if the number ends with 3 we are breaking the loop
            {

                break;
            }
            if(i%j==0) //checking whether the number is divisible or not
            {
                count++; //incrementing count value when a factor is found
            }
        }
    }
    if(count==2) //if count is 2 then it is prime number
    {
        printf("%d ",i);
    }
}

```

```

count=0;
}
return 0;
}

```

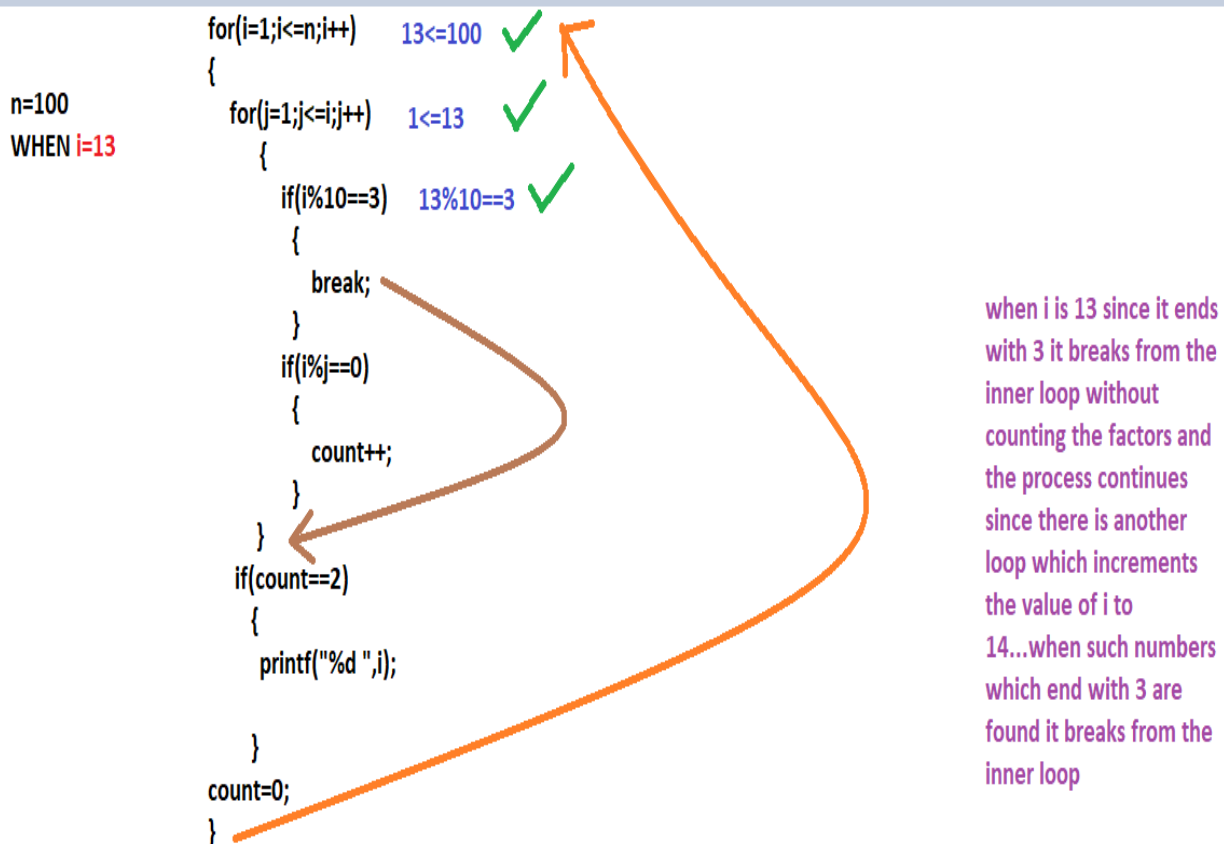
## OUTPUT:

```

enter number : 100
2 5 7 11 17 19 29 31 37 41 47 59 61 67 71 79 89 97
-----
Process exited after 2.312 seconds with return value 0
Press any key to continue . . .

```

## EXPLANATION:



**b. continue** : It is a keyword in c which skips the rest of the statements behind and goes to the beginning of the loop.

**Example program 18(ep18):**

*/\*program to input a number n and print all the numbers till n except the numbers that are divisible by both 5 and 7\*/*

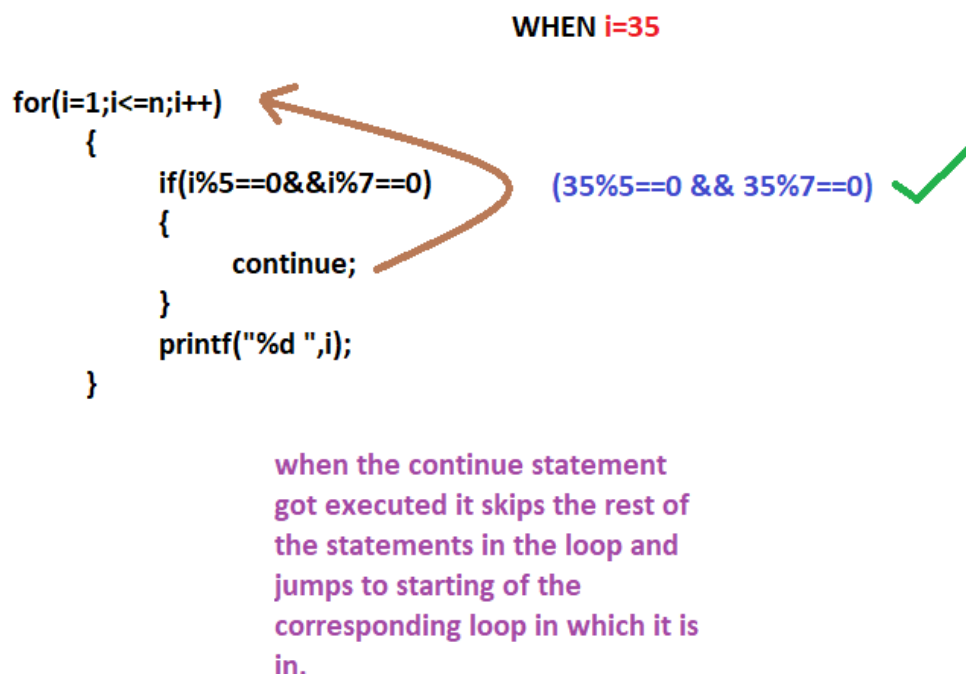
```
#include<stdio.h>

int main()
{
    int i,n;
    printf("\n enter number");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        if(i%5==0&& i%7==0)//checking whether the number is divisible by both 7 and 5
        {
            continue; //using continue to skip rest of satements if condition is true
        }
        printf("%d ",i);
    }
    return 0;
}
```

**OUTPUT:**

```
enter number100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
-----
Process exited after 3.296 seconds with return value 100
Press any key to continue . . .
```

## EXPLANATION:



c. **goto**: The goto statement allows us to transfer control of the program to the specified label.

### Syntax:

goto x;

-----

-----

x:

Statements;

So, here when the goto statement is executed it shifts to x: and continues execution from there.

### **Example program 19(ep19):**

```
/*example goto program-1*/
```

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

```
int main()
```

```
{
```

```
    int n=10;
```

```
    goto y; //it jumps to label y
```

```
    if(n%3==0)
```

```
    {
```

```
        y: printf("hello");
```

```
    }
```

```
        printf("\nbye");
```

```
    return 0;
```

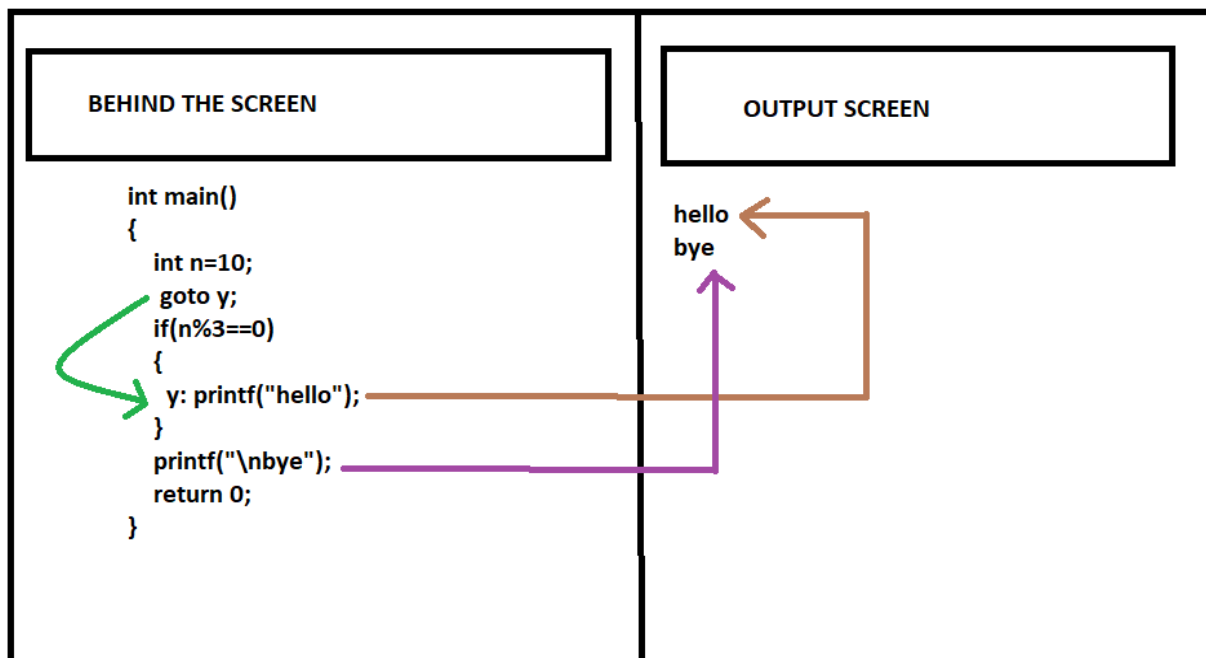
```
}
```

**OUTPUT:**



```
C:\Users\CHANDANA\Desktop\c prgm\ep19.exe
hello
t bye
-----
Process exited after 0.07232 seconds with return value 0
Press any key to continue . . .
```

## EXPLANATION:



### Example program 20(ep20):

*/\*example goto program-2\*/*

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

```
int main()
```

```
{
```

```
    int n=3;
```

```
    y:n++;
```

```
    printf("\nbye");
```

```
if(n==10)
{
    goto x; //it jumps to corresponding label x
}
goto y; //it jumps to corresponding label y
if(n%10==0)
{
    x:printf("hello");
}
printf("\nbye");
return 0;
}
```

```
bye
bye
bye
bye
bye
bye
byehello
bye
-----
Process exited after 0.0337 seconds with return value 0
Press any key to continue . . .
```

- d. **return**: a return statement ends the execution of a function, and returns control to the calling function.

Lets,get back to the return statement while we are learning about functions.

## **Examples on PATTERNS**

### **Example program 21(ep21):**

*/\*Program to input a number ‘ n’ and print the following pattern*

*Where n=no.of lines*

*#  
# #  
# # #  
# # # #*

*Here n=4 \*/*

```
#include<stdio.h>
int main()
{
    int n,i,j;
    scanf("%d",&n);
    for(i=1;i<=n;i++)//consider ‘i’ to be the line number
    {
        for(j=1;j<=i;j++)//consider ‘j’ to print pattern no.of times in a line
        {
            printf("# ");
        }
        printf("\n");
    }
    return 0;
```

```
}
```

## OUTPUT:

```
3
#
# #
# # #

-----
Process exited after 1.784 seconds with return value 3
Press any key to continue . . .
```

## EXPLANATION:

Consider outer loop 'i' to represent the line number and consider inner loop 'j' to represent the no.of times to print in the specific line.

$j \leq i$  since in line 1 only one # is printed

In line 2 ,two # are printed

So,no .of patterns printed are equal to to the line number we took condition as  $j \leq i$

## Logic:

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)
    {
        printf("# ");
    }
    printf("\n");
}
```

Initially

$n=3$

Outerloop iteration 1:

$i=1$ ;

$1 \leq 3 (i \leq n)$  **yes**

So,  $j=1$ ;

$1 \leq 1 (j \leq i)$  **yes**

Prints # ( `printf("# ");`)

$j++$ ;

Now,  $j=2$ ;

$2 \leq 1 (j \leq i)$  **no**

Exits from inner loop

Now, cursor moves to new line( `printf("\n")`)

Output by end of iteration 1:

#

Outerloop iteration 2:

$i++$ ;

Now,  $i=2$ ;

$2 \leq 3 (i \leq n)$  **yes**

So,  $j=1$ ;

$1 \leq 2 (j \leq i)$  **yes**

Prints # ( `printf("# ");`)

$j++$ ;

Now,j=2;

$2 \leq 2$  ( $j \leq i$ ) **yes**

Prints # ( printf("# ");)

j++;

Now,j=3;

$3 \leq 2$  ( $j \leq i$ ) **no**

Exits from inner loop

Now,cursor moves to new line( printf("\n"))

Output by end of iteration2:

#

# #

Outerloop iteration 3:

i++;

Now,i=3;

$3 \leq 3$  ( $i \leq n$ ) **yes**

So,j=1;

$1 \leq 3$  ( $j \leq i$ ) **yes**

Prints # ( printf("# ");)

j++;

Now,j=2;

$2 \leq 3$  ( $j \leq i$ ) **yes**

Prints # ( printf("# ");)

j++;

Now,j=3;

3<=3 (j<=i) **yes**

Prints # ( printf("# ");)

j++;

4<=3 (j<=i) **no**

Exits from inner loop

Now, cursor moves to new line( printf("\n"))

Output by end of iteration3:

#

# #

# # #

Outerloop iteration 4:

i++;

i=4;

4<=3(i<=n) **no**

Exits even from the outer loop.

**Example program 22(ep22):**

**/\*Program to input a number ‘ n’ and print the following pattern**

**Where n=no.of lines**

**@ @ @ @**

**@ @ @ @**

**@ @ @ @**

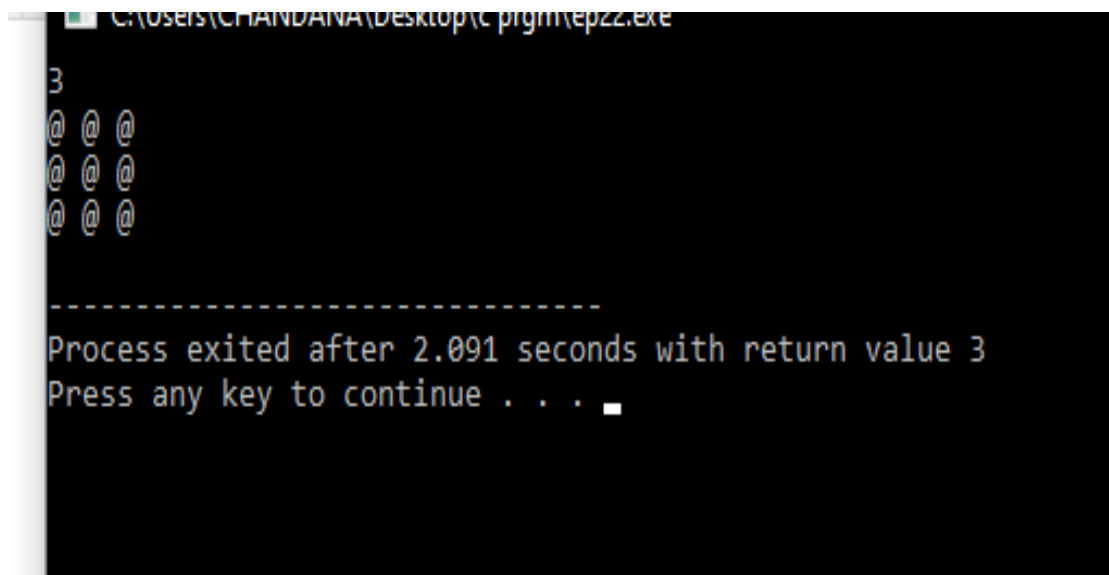
**@ @ @ @**

**Here n=4 \*/**

**#include<stdio.h>**

```
void main()
{
    int n,i,j;
    scanf("%d",&n);
    for(i=1;i<=n;i++) //consider 'i' to be the line number
    {
        for(j=1;j<=n;j++)//consider 'j' to print pattern no.of times in a line
        {
            printf("@ ");
        }
        printf("\n");
    }
}
```

### **OUTPUT:**

A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\CHANDANA\Desktop\c prgm\epzz.exe". The prompt shows the number "3" entered. Below it, a 3x3 grid of '@' symbols is printed. After a dashed line separator, the text "Process exited after 2.091 seconds with return value 3" and "Press any key to continue . . ." are displayed.

```
C:\Users\CHANDANA\Desktop\c prgm\epzz.exe
3
@ @ @
@ @ @
@ @ @

-----
Process exited after 2.091 seconds with return value 3
Press any key to continue . . .
```

### **EXPLANATION:**

Consider outer loop 'i' to represent the line number and consider inner loop 'j' to represent the no.of times to print in the specific line.



$j \leq n$  since in line 1 ,three @ are printed

In line 2 ,three @ are printed

So,no .of patterns printed are equal to to the input number 'n' so, we took condition as  $j \leq n$

Logic:

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
    {
        printf("@ ");
    }
    printf("\n");
}
```

Initially

$n=3$

Outerloop iteration 1:

$i=1;$

$1 \leq 3 (i \leq n)$  yes

So, $j=1;$

$1 \leq 3 (j \leq n)$  yes

Prints @ ( printf("@ ");)

$j++;$

Now, $j=2;$

$2 \leq 3 (j \leq n)$  yes

Prints @ ( printf("@ ");)

j++;

Now,j=3;

3<=3 (j<=n) yes

Prints @ ( printf("@ ");)

j++;

Now,j=4;

4<=3 (j<=n) no

Exits from inner loop

Now,cursor moves to new line( printf("\n"))

Output by end of iteration1:

@ @ @

Outerloop iteration 2:

i++;

i=2;

2<=3(i<=n) yes

So,j=1;

1<=3 (j<=n) yes

Prints @ ( printf("@ ");)

j++;

Now,j=2;

2<=3 (j<=n) yes

Prints @ ( printf("@ ");)

j++;

Now,j=3;

3<=3 (j<=n) **yes**

Prints @ ( printf("@ ");)

j++;

Now,j=4;

4<=3 (j<=n) **no**

Exits from inner loop

Now,cursor moves to new line( printf("\n"))

Output by end of iteration2:

@ @ @

@ @ @

Outerloop iteration 3:

i++;

i=3;

3<=3(i<=n) **yes**

So,j=1;

1<=3 (j<=n) **yes**

Prints @ ( printf("@ ");)

j++;

Now,j=2;

2<=3 (j<=n) **yes**

Prints @ ( printf("@ ");)

j++;

Now,j=3;

3<=3 (j<=n) **yes**

Prints @ ( printf("@ ");)

j++;

Now,j=4;

4<=3 (j<=n) **no**

Exits from inner loop

Now,cursor moves to new line( printf("\n"))

Output by end of iteration3:

@ @ @

@ @ @

@ @ @

Outerloop iteration 4:

i++;

i=4;

4<=3(i<=n) **no**

Exits even from the outer loop.

**Example program 23(ep23):**

**/\*Program to input a number ‘ n’ and print the following pattern**

**Where n=no.of lines**

**A**

**B C**

**D E F**

**G H I J**

**Here n=4 \*/**

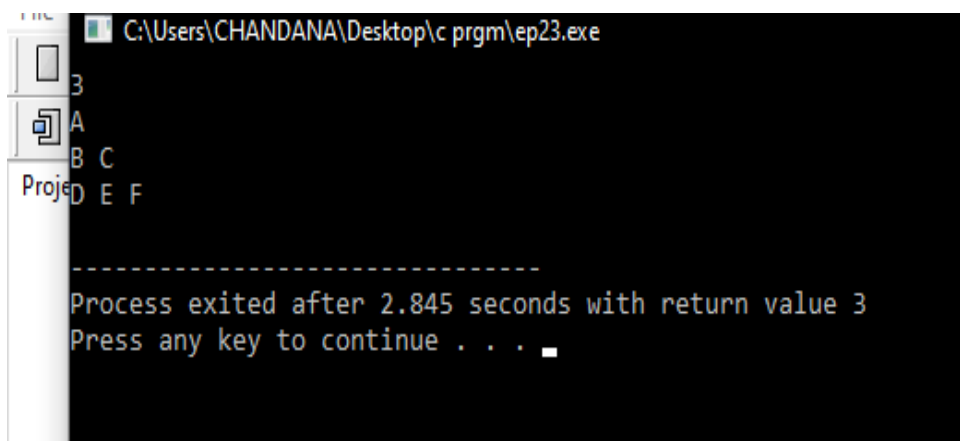
**#include<stdio.h>**

```

int main()
{
    int n,i,j;
    scanf("%d",&n);
    char c='A';
    for(i=1;i<=n;i++) //consider 'i' to be the line number
    {
        for(j=1;j<=i;j++) //consider 'j' to print pattern no.of times in a line
        {
            printf("%c ",c); // printing the character
            c++; // incrementing character to next letter
        }
        printf("\n");
    }
}

```

## **OUTPUT:**



```

C:\Users\CHANDANA\Desktop\c prgm\ep23.exe
3
A
B C
D E F
-----
Process exited after 2.845 seconds with return value 3
Press any key to continue . . .

```

## **EXPLANATION:**

Consider outer loop 'i' to represent the line number and consider inner loop 'j' to represent the no.of times to print in the specific line.

So, no. of letters printed are equal to the line number we took condition as  $j \leq i$

Logic:

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)
    {
        printf("%c ",c);
    }
    printf("\n");
}
```

Initially

$n=3$

$c='A'$

Outerloop iteration 1:

$i=1;$

$1 \leq 3 (i \leq n)$  **yes**

So,  $j=1;$

$1 \leq 1 (j \leq i)$  **yes**

Prints A (  $\text{printf}("%c ",c)$ )

$c++;$

Now,  $c='B'$

$j++;$

Now,  $j=2;$

$2 \leq 1 (j \leq i)$  **no**

Exits from inner loop

Now, cursor moves to new line( printf("\n"))

Output by end of iteration1:

A

Outerloop iteration 2:

i++;

Now, i=2;

2<=3(i<=n) **yes**

So, j=1;

1<=2 (j<=i) **yes**

Prints B ( printf("%c ",c))

c++;

Now, c='C'

j++;

Now, j=2;

2<=2 (j<=i) **yes**

Prints C ( printf("%c ",c))

c++;

Now, c='D'

j++;

Now, j=3;

3<=2 (j<=i) **no**

Exits from inner loop

Now, cursor moves to new line( printf("\n"))

Output by end of iteration2:

A

B C

Outerloop iteration 3:

i++;

Now,i=3;

3<=3(i<=n) yes

So,j=1;

1<=3 (j<=i) yes

Prints D ( printf("%c ",c))

c++;

Now,c='E'

j++;

Now,j=2;

2<=3 (j<=i) yes

Prints E ( printf("%c ",c))

c++;

Now,c='F'

j++;

Now,j=3;

3<=3 (j<=i) yes

Prints F ( printf("%c ",c))

c++;



Now,c='G'

j++;

4<=3 (j<=i) **no**

Exits from inner loop

Now,cursor moves to new line( printf("\n"))

Output by end of iteration3:

A

B C

D E F

Outerloop iteration 4:

i++;

i=4;

4<=3(i<=n) **no**

Exits even from the outer loop.

**P15:**Program to input a number n and print the following output(where n is the number of lines)

**Sample output:**

Enter n value : 4

1

2 3

4 5 6

7 8 9 10

**P16:**Program to input a number n and print the following output(where n is the number of lines)

**Sample output:**

Enter n value : 5

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

**P17:**Program to input a number n and print the following output(where n is the number of lines)

**Sample output:**

Enter n value : 4

1 1 1 1

2 2 2 2

3 3 3 3

4 4 4 4

**P18:**Program to input a number n and print the following output(where n is the number of lines)

**Sample output:**

Enter n value : 5

0 1 0 1 0

1 0 1 0 1

0 1 0 1 0

1 0 1 0 1

0 1 0 1 0

**P19:**Program to input a number n and print the following output(where n is the number of lines)

**Sample output:**

Enter n value : 3

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

**P20:**Program to input a number n and print the following diamond pattern (where n is the no. of lines in first half of diamond)

**Sample output:**

Enter n value : 5

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

**Examples programs:** try to trace the output of the following programs by yourself

**Example program 24(ep24):**

**/\*program to input a number n and print the fibonacci series till n**  
**If n=6**

1 1 2 3 5 8

Print not valid if  $n \leq 2$ \*/

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

```
void main()
```

```
{
```

```
    int a=1,b=1,c,n,i;
```

```
    scanf("%d",&n);
```

```
    if(n<=2)
```

```
    {
```

```
        printf("\n not valid");
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("\n1 1 ");// printing first two values of series
```

```
        for(i=3;i<=n;i++)//initialling i to 3 cause already two values are printed
```

```
        {
```

```
            c=a+b; // adding last two numbers
```

```
            printf("%d ",c);
```

```
            a=b;
```

```
            b=c;
```

```
        }
```

```
    }
```

```
}
```

**OUTPUT:**

```
8
1 1 2 3 5 8 13 21
-----
Process exited after 3.083 seconds with return value 8
Press any key to continue . . .
```

### Example program 25(ep25):

/\*program to input a number n and print the factorial of number n

Ex:5

Factorial of 5 is  $5*4*3*2*1=120$ \*/

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

```
void main()
```

```
{
```

```
    int n,fact=1,i;
```

```
    scanf("%d",&n);
```

```
    for(i=1;i<=n;i++)
```

```
    {
```

```
        fact=fact*i;// multiplying values from 1 to n
```

```
    }
```

```
    printf("\n%d",fact);
```

```
}
```

### OUTPUT:

```

C:\Users\CHANDANA\Desktop\c p\gm\ep24.exe
4
24
-----
Process exited after 4.812 seconds with return value 3
Press any key to continue . . .

```

## ARRAY (DERIVED DATATYPE)

It is a data structure or simply a collection of similar type of pre-defined datatypes, each identified by the array index. There are two types of arrays. They are:

1. Single dimensional array.
2. Multi dimensional array.

### SINGLE DIMENSIONAL ARRAY:

Conceptually you can think of a one-dimensional array as a row, where elements are stored one after another.

Syntax: datatype array\_name[size];

Ex:

```
int b[3]={ 11,12,4};
```

```
int a[5]={ 5,3,5,2,7};
```

5	3	5	2	7
a[0]	a[1]	a[2]	a[3]	a[4]

Memory locations: store in consequent memory locations

4500	4501	4502	4503	4504
------	------	------	------	------

### Example program 25(ep25):

*/\*program to input n elements in array and display them\*/*

```
#include <stdio.h>

int main() {
    int n,i;
    printf("enter no of elements in the array");
    scanf("%d",&n);
    int x[n];
    printf("\n enter the elements into the array ");
    //to input elements into the array
    for(i=0;i<n;i++)
    {
        scanf("%d",&x[i]);
    }
    //to print the elements in the array
    printf("\n the elements in the array are");
    for(i=0;i<n;i++)
    {
        printf("\n %d",x[i]);
    }
    return 0;
}
```

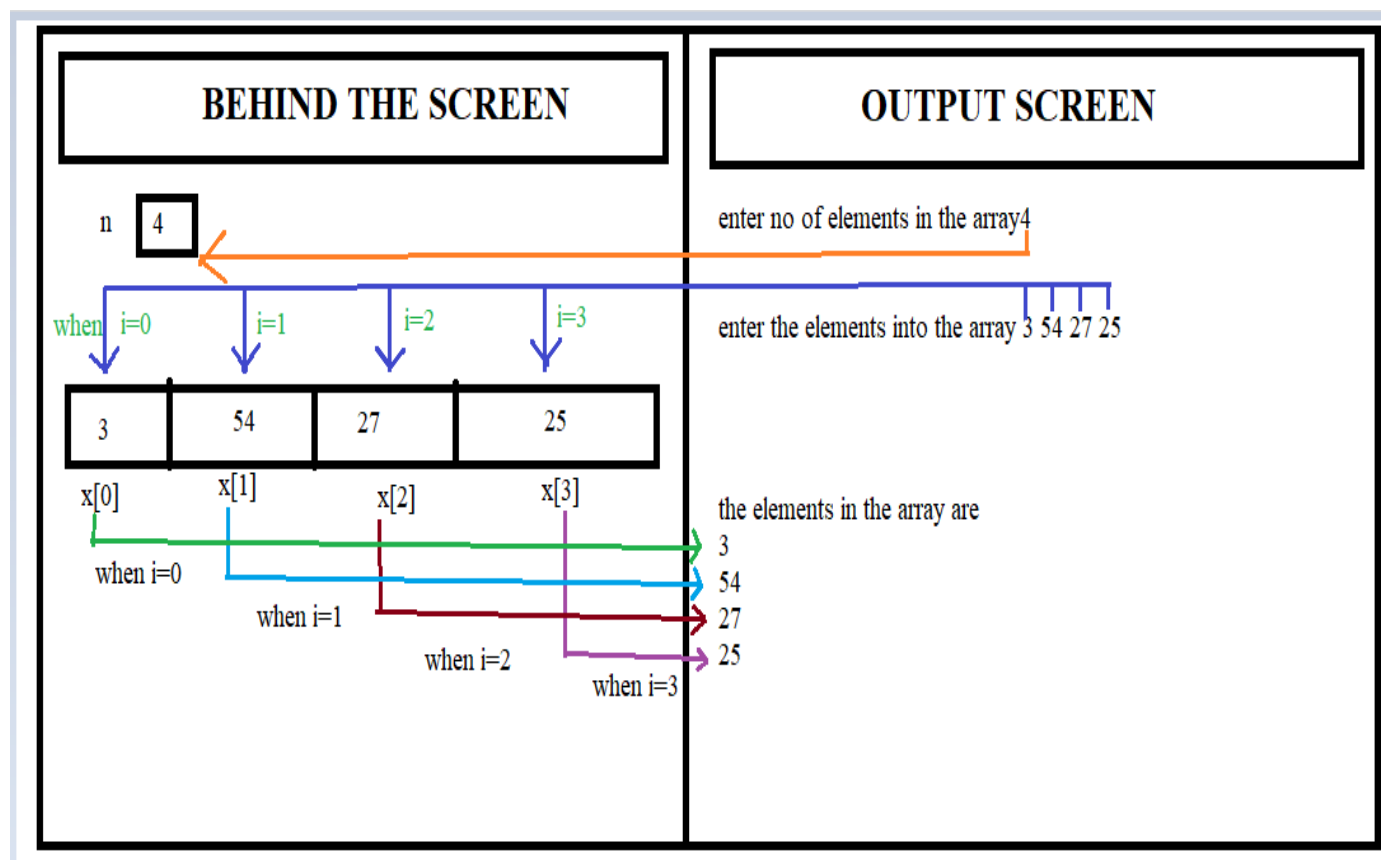
**OUTPUT:**

```
enter no of elements in the array4

enter the elements into the array 3 54 27 25

the elements in the array are
3
54
27
25
```

### EXPLANATION :



### **Example program 26(ep26):**

*/\*program to input n elements in array and print sum of those elements in the array\*/*



```
#include <stdio.h>

int main() {
    int n,i,sum=0;
    printf("enter no of elements in the array");
    scanf("%d",&n);
    int x[n];
    //to input elements into the array
    for(i=0;i<n;i++)
    {
        scanf("%d",&x[i]);
    }
    for(i=0;i<n;i++)
    {
        sum=sum+a[i];
    }
    printf("%d",sum);
}
```

**OUTPUT:**

```
C:\Users\ChandanA\Desktop\others\p1.exe
enter no of elements in the array3
2 3 5
10
-----
Process exited after 11.66 seconds with return value 2
Press any key to continue . . .
```

## **MULTI-DIMENTIONAL ARRAY:**

We can define multi-dimensional arrays in simple word as array of arrays.

### Syntax:

Two dimensional array:

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

Three dimensional array:

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

And so on...

Ex:

Two dimensional array:

```
int a[10][20];
```

Three dimensional array:

```
int a[10][20][10];
```

# HOW TO RUN A “C” PROGRAM IN “DEV C++”

**STEP 1:** install Dev C++

Link : <https://sourceforge.net/projects/orwelldvcpp/>

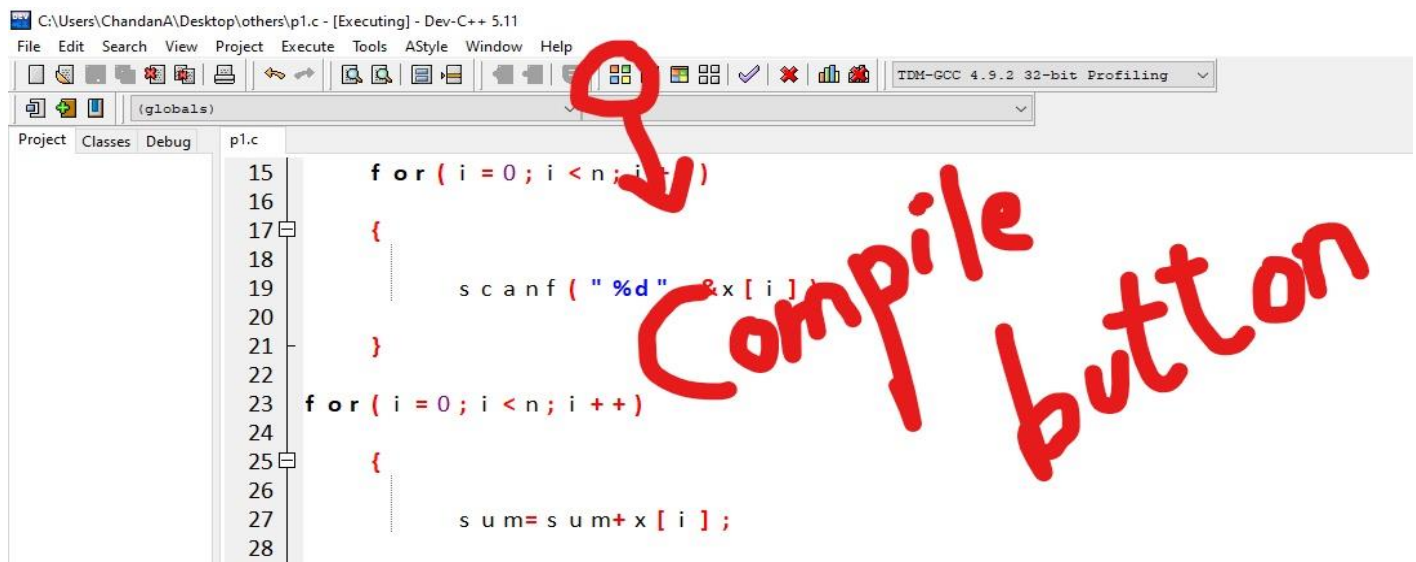
**STEP 2:**click on

File->new->source file

An editor will open

**STEP 3:**Type the code and save as **program\_name.c**

**STEP 4:**click on compile button



If there are errors rectify them till your program is error free.

**STEP 5:**Now click on compile and run button an output screen will be displayed.

C:\Users\ChandanA\Desktop\others\p1.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help



(globals)

Project Classes Debug

p1.c

```
15     for ( i = 0 ; i < n ; i ++ )
16
17 {
18     .....
19     s = printf ( "%d" , &x [ i ] ) ;
20
21 }
22
23 for ( i = 0 ; i < n ; i ++ )
24
```

Compile & Run  
button