C PROGRAMMING FOR PROBLEM SOLVING (18CPS23)

Module 2
Managing Inputs and Outputs
Branching and Looping

Prof. Prabhakara B. K.
Associate Professor
Department of Computer Science and Engineering
Vivekananda College of Engineering and Technology, Puttur.

MANAGING INPUT/OUTPUT STATEMENTS

Input statements

The statements which are used to read values from keyboard (user)

Output statements

The statements are used to display information (results) to user.

Input / Output Statements	
Formatted	**Unformatted**
printf( )	getch( )
scanf( )	getchar( )
etc.	gets( )
	putch( )
	putchar( )
	puts( )
	etc.

FORMATTED OUTPUT STATEMENT printf( )

- This is built-in (library / predefined) function.
- To use this function in a program, we must include stdio.h file.
- Used to display information in user required format.
- Where a letter f in printf stands for formatted.

Syntax

```
printf("Control String",argument1, arguement2,…);
```

The control string is also known as format specifier

Where arguments may be variables / constants / expressions.

If the control string is absent, then the argument is not possible use.

CONTROL STRING SPECIFICATION

%	Flag	Width	Precision	Type	
				Size	Character

% This is a compulsory first character used in control string

Flag It is a optional second character used in control string.

There are five flags

0 This character used to padding zeros.

- This character is used to justify left.

+ This character is used to have sign.

# This character is used to have prefix 0 in octal numbers and to have prefix 0x in hexadecimal numbers.

Space character is used to have space while display a character.

CONTROL STRING SPECIFICATION

Width and Precision

Example 5.2

Where 5 is width.

		.		

It specifies total numbers of reserved spaces for a value.

Where 2 is Precision.

It specifies total numbers of digits after the decimal point.

Example If we store value 8.28 in above width and precision

	8	.	2	8

Type

This character is mandatory in the control string.

It is having two sub parts 1) size 2)character

Where character is mandatory. Examples c d e g i o u x

The size is optional. Examples l h

DATA TYPES WITH CONTROL STRINGS

Data Type		Keyword	Size (in Byte)	Range	Control Strings
Basic Data Types	Void	void	0	Nil	
	Character	char	1	-128 to 127	%c (%s for string)
	Integer	int	2	-32768 to 32767	%d %o %x %i
	Real	float	4	3.4E-38 to 3.4E38	%f %e %g
	Double	double	8	1.7E-308 to 1.7E308	%lf
Basic Data Type's Modifiers	Signed character	signed char	1	-128 to 127	%c
	Unsigned character	unsigned char	1	0 to 255	%c
	Short integer	short int	2	-32768 to 32767	%hi
	Signed short integer	signed short int	2	-32768 to 32767	%hi
	Unsigned short integer	unsigned short int	2	0 to 65535	%hu
	Signed integer	signed int	2	-32768 to 32767	%d %o %x %i
	Unsigned integer	unsigned int	2	0 to 65535	%u
	Long integer	long int	4	-2147483648 to 2147483647	%li %ld
	Signed long integer	signed long int	4	-2147483648 to 2147483647	%li
	Unsigned long integer	unsigned long int	4	0 to 4294967295	%lu
	Long double	long double	10	3.4E-4932 to 1.1E4932	%Lf %Le %Lg

printf( ) EXAMPLES

EXAMPLES	OUTPUTS
printf("C Programing for Problem Solving");	C Programing for Problem Solving
printf("C Programing\nfor Problem\nSolving");	C Programing for Problem Solving
printf("C Programing\tfor Problem Solving");	C Programing for Problem Solving
printf("\"Programing in \'C\' and\\ Problem\\ Solving\"");	"Programing in 'C' and\ Problem\ Solving"
printf("\"I am\nan\"\t\'Engineering\nstudent\'");	"I am an" 'Engineering student'

printf( ) EXAMPLES

If x=4568 for all below printf( )

EXAMPLES	OUTPUTS
printf("%d",x);	4568
printf("Result=%d",x);	Result=4568
printf("Result=%d");	Result=xxxxx
printf("%d",1234);	1234
printf("%d",x/x);	1
printf("Result=%d\t%d",x,1234);	Result=4568 1234
printf("%6d",x);	`  4568`
printf("%06d",x);	`004568`
printf("%0d",x);	4568
printf("%-6d",x);	`4568  `
printf("%6d",-x);	` -4568`
printf("%+d",x);	+4568
printf("%+06d",x);	`+04568`

printf( ) EXAMPLES

If A=1234 for all below printf( )

<table>
<tr><th>EXAMPLES</th><th>OUTPUTS</th></tr>
<tr><td>printf("%x",10);</td><td>a</td></tr>
<tr><td>printf("%#x",10);</td><td>0xa</td></tr>
<tr><td>printf("%o",8);</td><td>10</td></tr>
<tr><td>printf("%#o",8);</td><td>010</td></tr>
<tr><td>printf("%#6x",0xab);</td><td><table><tr><td></td><td></td><td>0</td><td>x</td><td>a</td><td>b</td></tr></table></td></tr>
<tr><td>printf("%6x",0xab);</td><td><table><tr><td></td><td></td><td></td><td></td><td>a</td><td>b</td></tr></table></td></tr>
<tr><td>printf("Value=%d",A);</td><td>Value=1234</td></tr>
<tr><td>printf("Value=%c",'A');</td><td>Value=A</td></tr>
<tr><td>printf("Value=%d",'A');</td><td>Value=65</td></tr>
<tr><td>printf("Value=%c",A);</td><td>Value=xxxxx</td></tr>
<tr><td>printf("%d%c%f",23,'A',4.1);</td><td>23A4.100000</td></tr>
</table>

DIFFERENT NUMBER SYSTEMS

DECIMAL	BINARY	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	a
11	1011	13	b
12	1100	14	c
13	1101	15	d
14	1110	16	e
15	1111	17	f
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

printf( ) EXAMPLES

If x=98.7654 for all below printf( )

EXAMPLES	OUTPUTS
printf("%f",x);	98.7654xx
printf("%.2f",x);	98.77
printf("%7.2f",x);	`  98.77`
printf("%-7.2f",x);	`98.77  `
printf("%e",x);	9.87654xe+01
printf("%11.4e",x);	` 9.8765e+01`
printf("%10.2e",x);	`  9.88e+01`
printf("%13.10f",x);	`98.7654xxxxxx`

FORMATTED INPUT STATEMENT scanf( )

- This is built-in (library / predefine) function.
- To use this function in a program, we must include stdio.h file.
- Used to read data from user.
- Where a letter f in scanf stands for formatted.

Syntax

```
scanf("Control String",argument1, arguement2,…);
```

- Only variables are possible to use in arguments.
- These variables must have prefix &
- If control string is %s prefix & is not required.
- While using more than one control string, avoid spaces between those control strings.

sacnf( ) EXAMPLES

EXAMPLES	DECLARATIONS / INPUTS / OUTPUTS	
scanf("%d",&a);	int a;	
scanf("%f",&b);	float b;	
scanf("%c",&staus);	char status;	
scanf("%s",name);	char name[100];	
scanf("%d%d",&a,&b);	int a, b;	
scanf("%5d%2d",&a,&b);	int a,b;	1234567 128
	AE a=12345 b=67	
scanf("%d%*d%d",&a,&b);	int a, b;	128 246 192
	AE a=128 b=192	

scanf("%d%f%s",&a,&b,name);	int a;
	float b;
	char name[100];
		20 4.1 motor

	AE a=20 b=4.100000 name=motor	

BRANCHING AND LOOPING

- Conditional Branching Statements
  - Two way selections

1. Conditional Operator
2. Simple if
3. if-else
4. Nested if-else
5. Cascade if-else (else-if ladder)
6. while
7. do-while
8. for

Loops

- Multiway selections

switch

- Unconditional Branching Statements

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

if STATEMENT

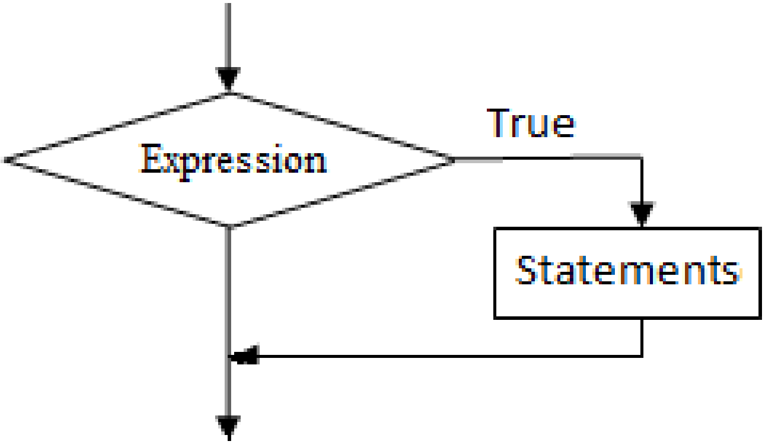
If the expression result is true (Non-zero), then statements are executed.

If the expression result is false (zero), then statements are skipped.

Syntax

```
if(expression)
{
    statements;
}
```

Flowchart



Example

```
if(num>0)
{
    printf(“Number is Positive”);
}
```

```
/*Program to check whether given integer is even or odd
using if statement*/
#include<stdio.h>
void main()
{
    int num;
    printf(“Enter an integer\n”);
    scanf(“%d”, &num);
    if(num%2==0)
    {
        printf(“Number is Even\n”);
    }
    if(num%2!=0)
    {
        printf(“Number is Odd\n”);
    }
}
```

If-else STATEMENT

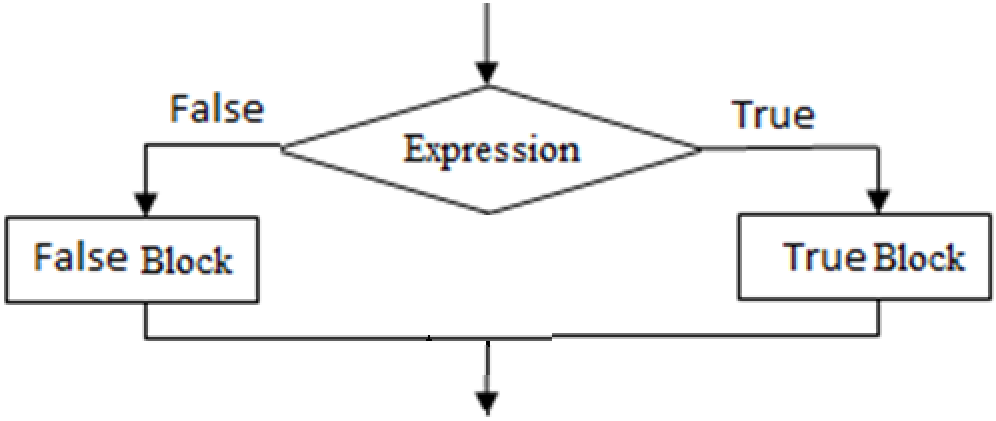
If the expression result is true (Non-zero), then true block is executed.

If the expression result is false (zero), then false block is executed.

Syntax

```
if(expression)
{
        True block;
}
else
{
        False block;
}
```

Flowchart



Example

```
if(num>0)
{
        printf("Number is Positive");
}
else
{
        printf("Number is Negative");
}
```

/*Program to check whether given number is even or odd using if else statement*/

```
#include<stdio.h>
void main()
{
    int num;
    printf(“Enter a number \n”);
    scanf(“%d”,&num);
    if(num%2==0)
    {
        printf(“Number is Even\n”);
    }
    else
    {
        printf(“Number is Odd\n”);
    }
}
```

Write a C program to read the age of a candidate and determine whether he /she is eligible for casting his/her own vote?

```
#include<stdio.h>
void main()
{
    int age;
    printf(“Enter age\n”);
    scanf(“%d”,&age);
    if(age>=18)
    {
       printf(“Eligible\n”);
    }
    else
    {
      printf(“Not Eligible\n”);
    }
}
```

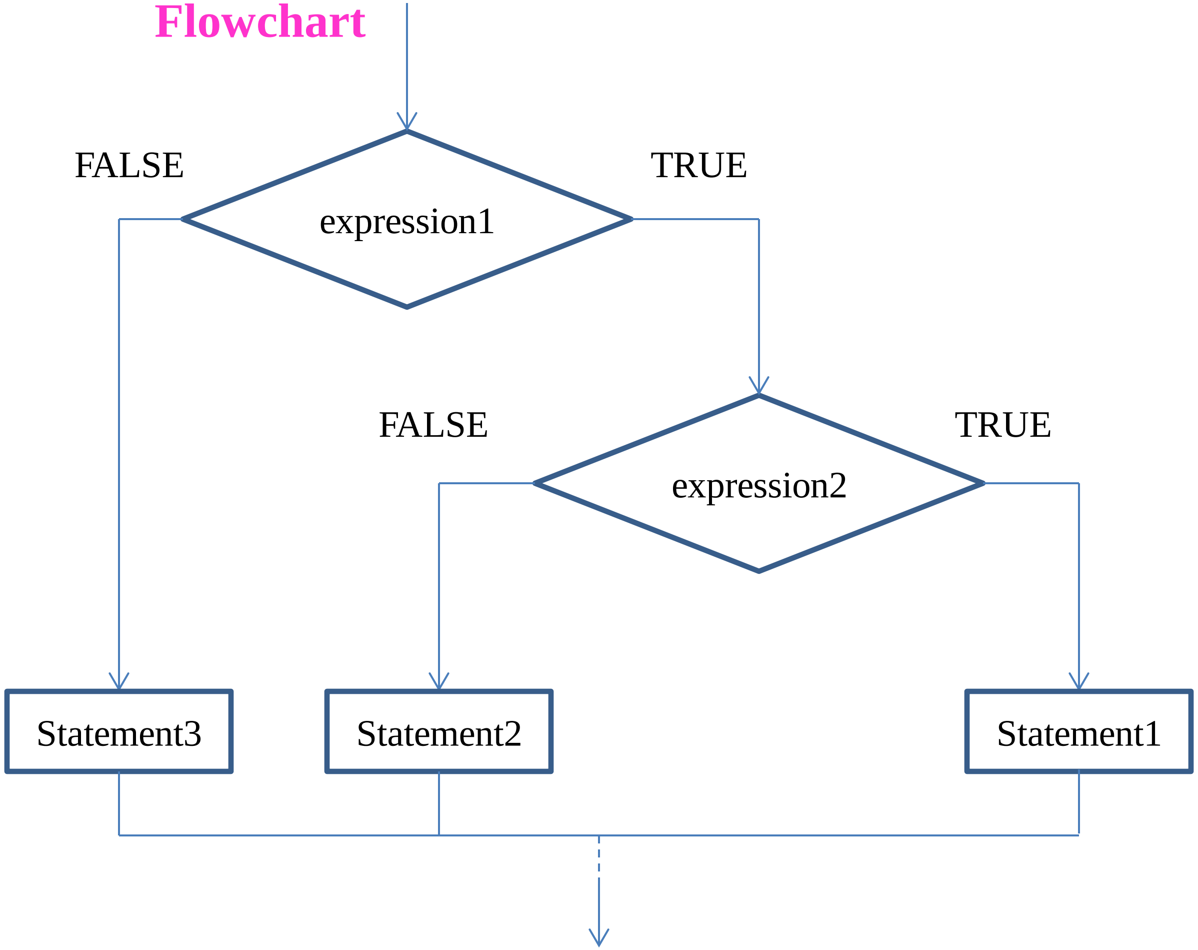
NESTED If-else STATEMENT

if-else within another if-else is known as nested if-else statement.

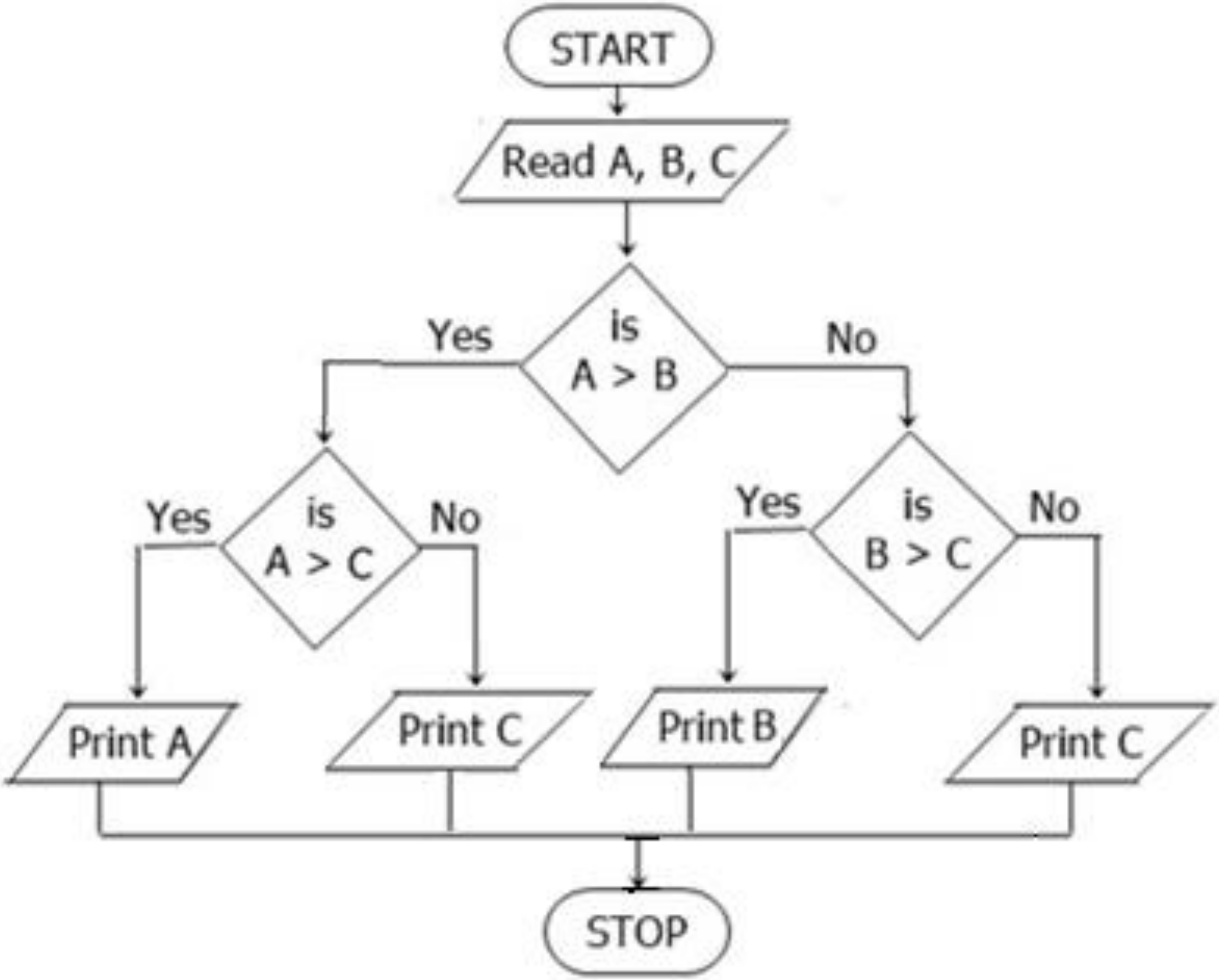
Syntax

```
if(expression 1)
{
    if(expression 2)
    {
          statement 1;
    }
    else
    {
          statement 2;
    }
}
else
{
    statement 3;
}
```

Flowchart



Write a C Program to find biggest of three integers using nested if-else statement.



A=1

B=2

C=3

A=1

B=3

C=2

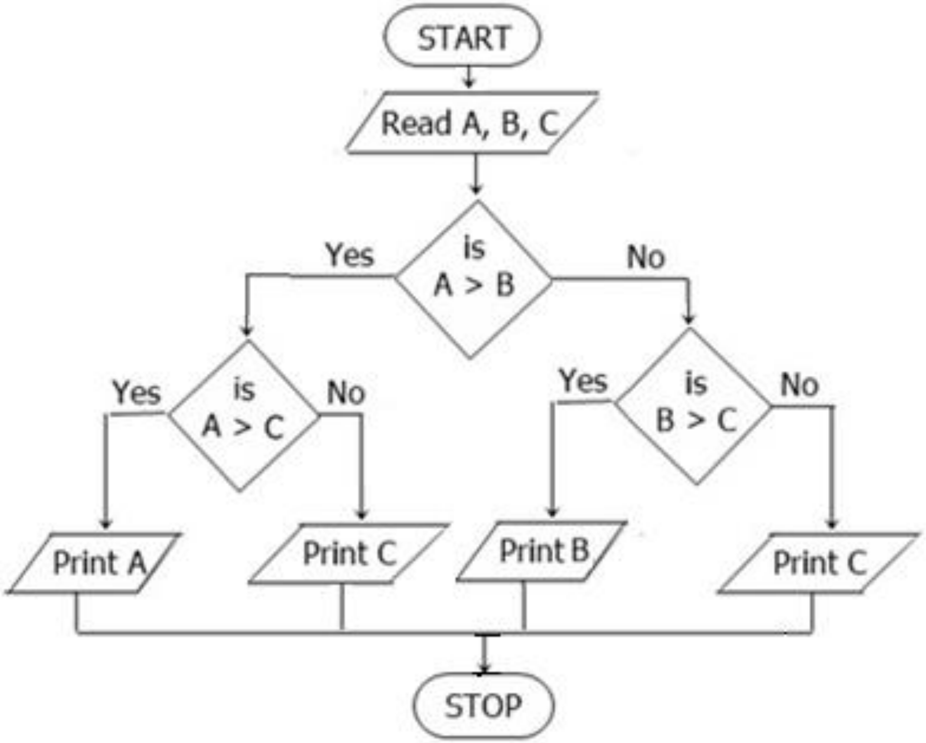
A=3

B=2

C=1

/* Program to find biggest of three integers using nested if-else*/

```
#include<stdio.h>
void main()
{
        int a,b,c;
        printf("Enter three inegers\n");
        scanf("%d%d%d",&a,&b,&c);
        if(a>b)
        {
                if(a>c)
                {
                        printf("%d is big\n",a);
                }
                else
                {
                        printf("%d is big\n",c);
                }
        }
        else
        {
                if(b>c)
                {
                        printf("%d is big\n",b);
                }
                else
                {
                        printf("%d is big\n",c);
                }
        }
}
```



Write a C program to find greatest among three integers using conditional operator.

```
/* Greatest among three integers using conditional operator */
#include<stdio.h>
void main()
{
    int a, b, c, big;
    printf("Enter three integers\n");
    scanf("%d%d%d", &a,&b,&c);
    big = a > b ? a : b;
    big = big > c ? big : c;
    printf(“Biggest is %d\n",big);
}
```

A=1
B=2
C=3

A=1
B=3
C=2

A=3
B=2
C=1

CASCADE If-else (else-if LADDER)

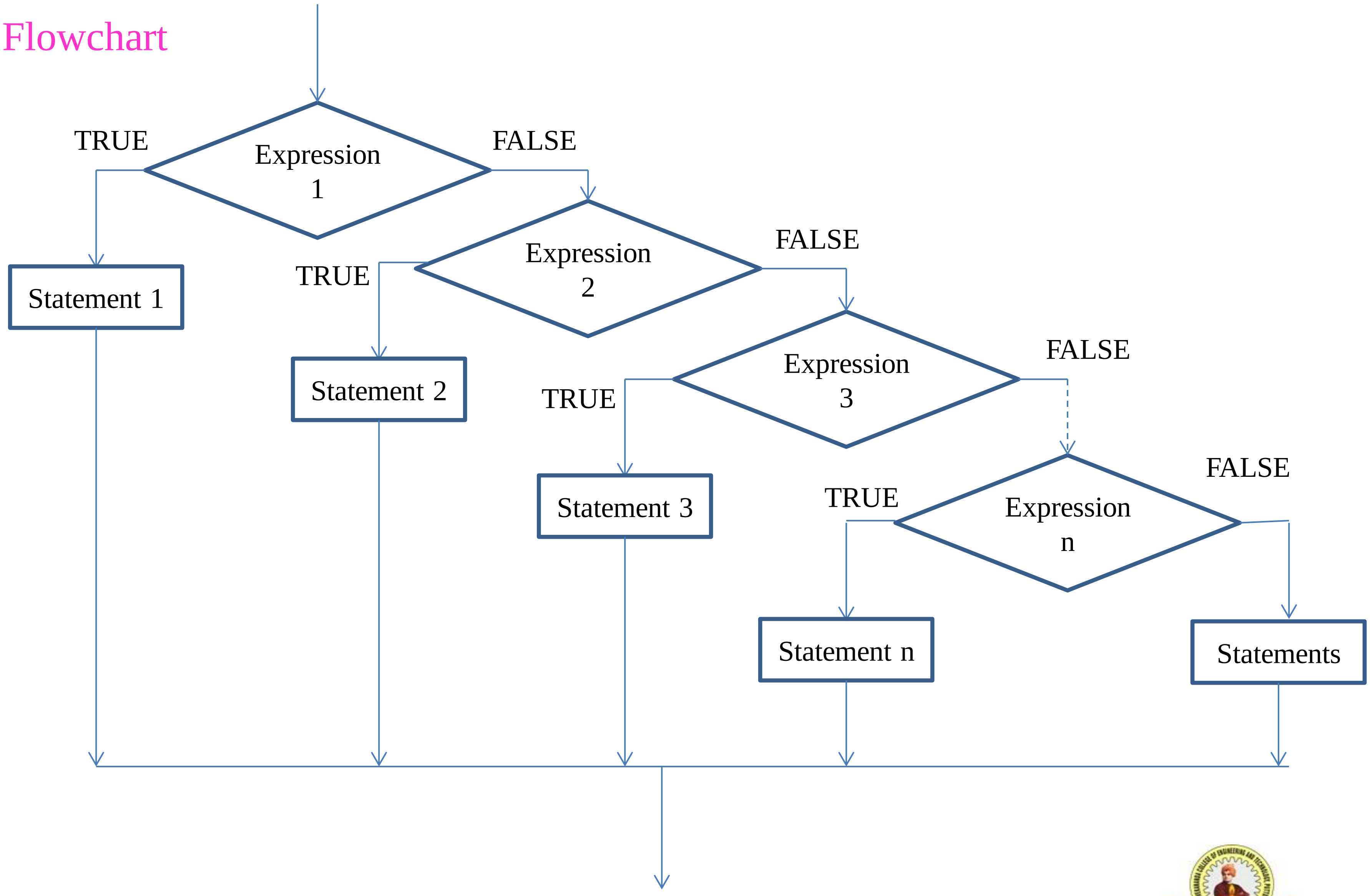
The if-else statement having another if-else statement at **else** part of it, is known as cascade if-else statement or else-if ladder.

Syntax

```
if (expression 1)
{
  statement 1;
}
else if (expression 2)
    {
         statement 2;
     }
     else if (expression 3)
         {
             statement 3;
          }
          •
          •
          •
          else if (expression n)
              {
                  statement n;
               }
```

CASCADE If-else (else-if LADDER)

Flowchart



Write a C program to grade the students based upon following percentage of marks using else-if ladder.

```
#include<stdio.h>
void main()
{
        int marks;
        printf(“Enter the marks obtained\n”);
        scanf(“%d”,&marks);
        if (marks>100)
          printf(“Invalid Marks\n”);
        else if(marks >= 80)
           printf(“S Grade\n”);
        else if(marks >= 70)
           printf(“A Grade\n”);
        else if(marks >= 60)
           printf(“B Grade\n”);
        else if(marks >=50)
           printf(“C Grade\n”);
        else if(marks >= 45)
           printf(“D Grade\n”);
        else if(marks >= 40)
           printf(“E Grade\n”);
        else
           printf(“Fail\n”);
}
```

MARKS	GRADE
80-100	S Grade
70-79	A Grade
60-69	B Grade
50-59	C Grade
45-49	D Grade
40-44	E Grade
00-39	Fail

switch STATEMENT

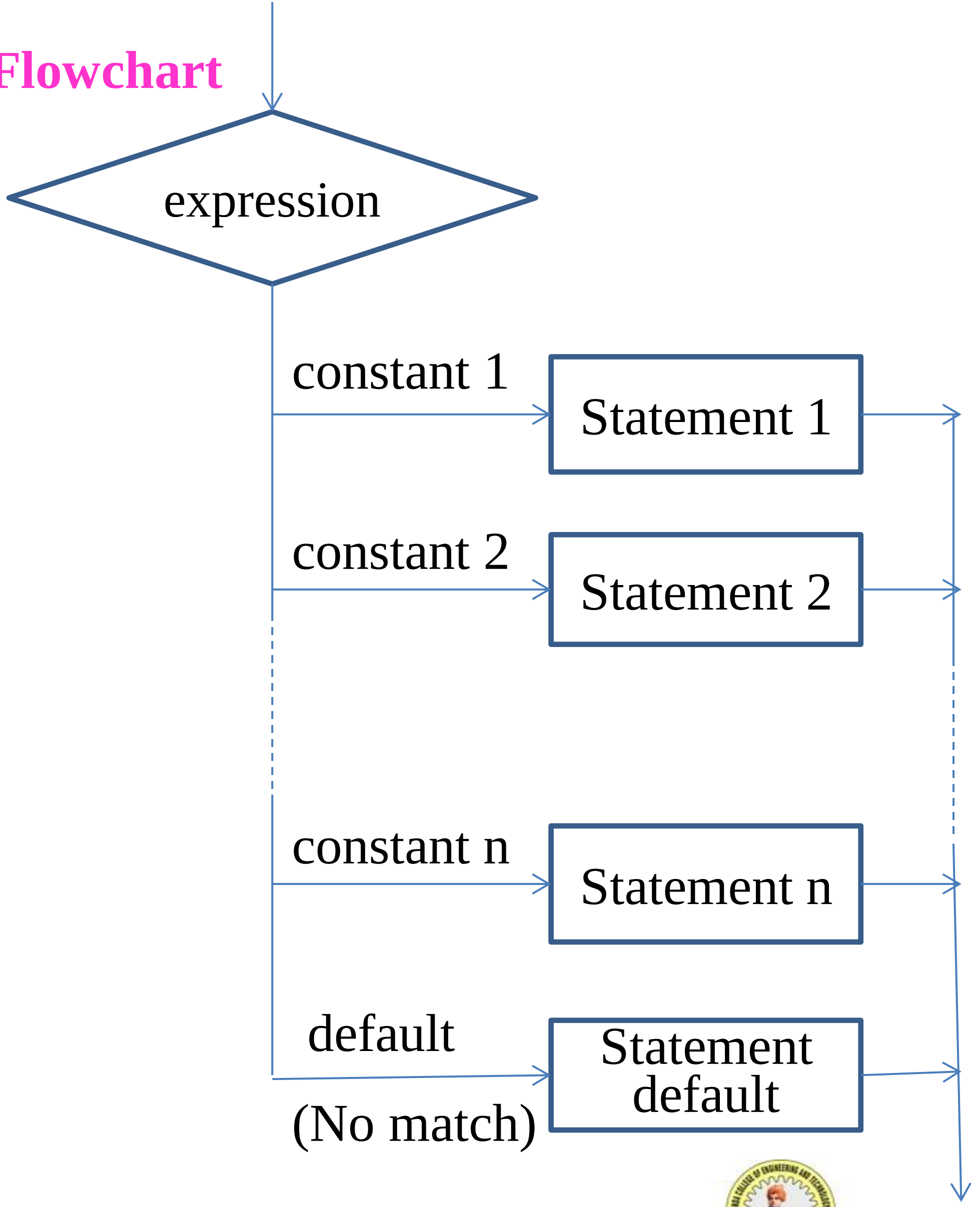
- This is multi-way selection branching statement used to select one out of many branches.
- switch statement verifies the expression result against different cases.
- The expression result is either character constant or integer constant.
- The break statement is used as last statement in each case block to terminate switch statement.
- The default case is executed, if expression result is not matching with any of the cases.
- The break statement is not used in default block.

switch STATEMENT

Syntax

```
switch(expression)
 {
      case constant1: statement 1 ;
                        break;
      case constant2: statement 2 ;
                        break;
                         …..
      case constantn: statement n ;
                        break;
              default: statements;
}
```

Flowchart



Write a C program to print week days based on day number. (Hint: 0-Sunday,1-Monday…6-Saturday)

```
#include <stdio.h>
void main()
{
    int daynumber;
    printf("Enter day number from 0 to 6\n");
    scanf("%d", &daynumber);
    switch(daynumber)
    {
        case 0: printf(“Sunday");
                break;
        case 1: printf("Monday");
                break;
        case 2: printf("Tuesday");
                break;
        case 3: printf("Wednesday");
                break;
        case 4: printf("Thursday");
                break;
        case 5: printf("Friday");
                break;
        case 6: printf("Saturday");
                break;
        default:printf("Invalid Input")
    }
}
```

/*Program to check whether an entered character is a vowel or not*/

```
#include<stdio.h>
void main()
{
        char ch;
        printf(“Enter a character\n ”);
        scanf(“%c”,&ch);
        switch(ch)
        {
                case 'A' :
                case 'a'  :
                case 'E' :
                case 'e'  :
                case 'I'  :
                case 'i'  :
                case 'O':
                case 'o‘ :
                case 'U':
                case 'u' : printf(“%c is an vowel\n”,ch);
                           break;
                 default: printf(“%c is not an vowel\n”,ch)
        }
}
```

LOOPS

Loops allows programmer to execute statements repeatedly until the control expression result is false (zero). (i.e. when the control expression is true, it repeats the execution of the loop)

Loop has three parts

- Initialization where a looping variable gets its first value.
- Control Expression
  - If the expression result is zero then the loop is terminated.
  - If the expression result is non-zero then body of the loop is executed.
- Body contains the statements do a specific task and increment /decrement statements.

Type of Loops

1. **Entry controlled (Pretest)**

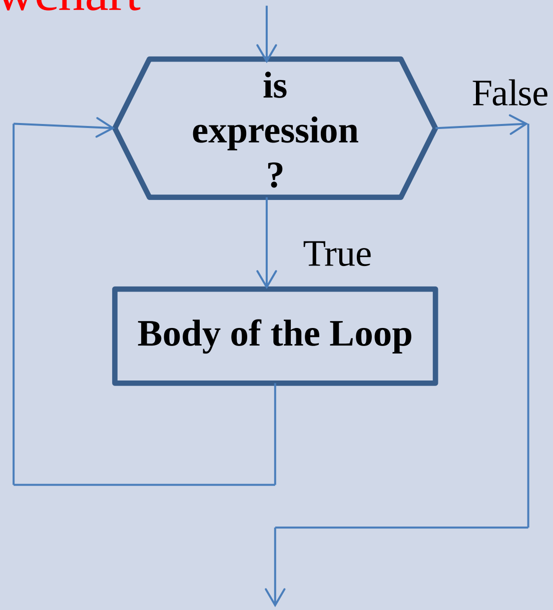
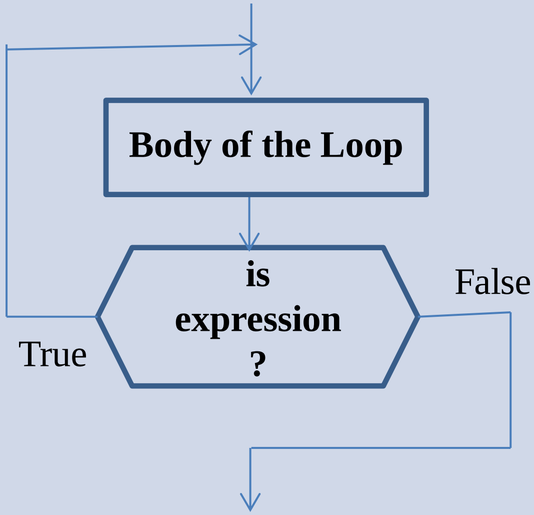
   Where verification of control expression is done before executing the body of the loop. Example while, for

2. **Exit controlled (Post- test)**

   Where verification of control expression is done at the end after executing the body of the loop. Example do-while

while LOOP	do-while LOOP
1. Entry controlled (Pretest) loop	1. Exit controlled (Post-test) loop

2. Syntax
while(expression)
{
    body of the loop;
}
	2. Syntax
	do
	{
	    body of the loop;
	}while(expression);

3. Flowchart  	3. Flowchart 

while LOOP	do-while LOOP
4. If control expression is false, the body of the loop executed zero times.	4. If control expression is false, the body of the loop executed at least once.
5.Control expression is verified n+1 times	5. Control expression is verified n times
6. Semicolon (;) is not required after control expression.	6 Semicolon (;) is required after control expression.

7. Example
i=1;
while(i<3)
{
printf(“C Programming”);
i++;
}
	7. Example
	i=1;
	do
	{
	printf(“C Programming”);
	i++;
	} while(i<3);

while LOOP

Example

```
i=1;
while(i<3)
{
    printf(“C Programming”);
    i++;
}
```

TRACE

```
i=1
---------------------
1<3
C Programming
i=1+1=2
---------------------
2<3
C Programming
i=2+1=3
---------------------
3<3
```

do-while LOOP

Example

```
i=1;
do
{
    printf(“C Programming”);
    i++;
} while(i<3);
```

TRACE

```
i=1
---------------------
C Programming
i=1+1=2
2<3
---------------------
C Programming
i=2+1=3
3<3
```

```
/*Program to calculate the sum of the first n natural
   numbers using while loop */
#include<stdio.h>
void main()
{
    int i=1, n, sum=0;
    printf("Enter the value of n\n");
    scanf("%d",&n);
    while(i<=n)
    {
        sum=sum+i;
        i++;
    }
    printf("Sum  is %d\n",sum);
}
```

TRACE

i=1, sum=0, n=3

1<= 3
sum=0+1=1
i=1+1=2

2<= 3
sum=1+2=3
i=2+1=3

3<= 3
sum=3+3=6
i=3+1=4

4<= 3

```
/*Program to calculate the sum of the first n natural
   numbers using do-while loop */
#include<stdio.h>
void main()
{
      int i=1, n, sum=0;
      printf(“Enter the value of n\n”);
      scanf(“%d”,&n);
      do
      {
            sum=sum+i;
            i++;
      } while(i<=n);
      printf(“Sum   is %d\n”,sum);
}
```

TRACE

i=1, sum=0, n=3

----

sum=0+1=1
i=1+1=2
2<= 3

----

sum=1+2=3
i=2+1=3
3<= 3

----

sum=3+3=6
i=3+1=4
4<= 3

Write a C program to find LCM and GCD of two integers using Euclid’s algorithm

```
#include <stdio.h>
void main()
{
    int m, n, a, b, lcm, rem;
    printf("Enter two integers\n");
    scanf("%d%d",&m,&n);
    a=m;
    b=n;
    while(n!=0)
    {
        rem=m%n;
        m=n;
        n=rem;
    }
    lcm=(a*b)/m;
    printf("LCM is %d\n",lcm);
    printf("GCD is %d\n",m);
}
```

m=15   n=100

6	100	15	1
	90	10	
2	10	5	
	10		
	0		

TRACE

m=15, n=100
a=15, b=100

100≠0
rem=15%100=15
m=100
n=15

15≠0
rem=100%15=10
m=15
n=10

10≠0
rem=15%10=5
m=10
n=5

5≠0
rem=10%5=0
m=5
n=0

0≠0

for LOOP

- The for loop is best loop compared with while and do-while loops.
- All three sections of this loop is available in one line.
- This is entry controlled (Pretest) loop.

Syntax

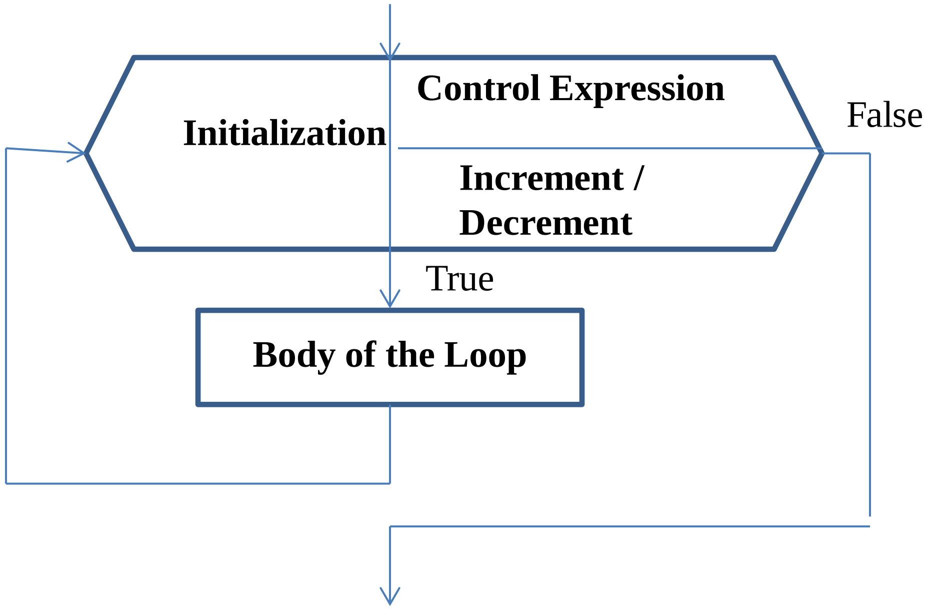
```
for(initialization; control expression; increment/decrement)
{
            body of the loop;
}
```

Working

1. First Initialization of the looping variable is done. This section is executed only once in the beginning of loop execution.
2. Second Control expression is evaluated for true or false.
   If control expression results true, then body of the loop is executed.
   If control expression results false, then loop is terminated.
3. The increment/decrement section is executed after executing the body of loop.

for LOOP

Flowchart



Working

1. First Initialization of the looping variable is done. This section is executed only once in the beginning of loop execution.
2. Second Control expression is evaluated for true or false.
   If control expression results true, then body of the loop is executed.
   If control expression results false, then loop is terminated.
3. The increment/decrement section is executed after executing the body of loop.

/*Program to find factorial of given integer using for loop */

```
#include<stdio.h>
void main()
{
    int i, n, fact=1;
    printf(“Enter the value of n\n”);
    scanf(“%d”,&n);
    for(i=1;i<=n;i++)
    {
        fact=fact*i;
    }
    printf(“Factorial is %d\n”,fact);
}
```

TRACE

fact=1, n=3

----

i=1
1<= 3
fact=1*1=1
i=1+1=2

2<= 3
fact=1*2=2
i=2+1=3

3<= 3
fact=2*3=**6**
i=3+1=4

4<= 3

EXAMPLE	OUTPUT/TRACE

int i, fact=1;
for(i=1; i<=3 ; i++)
{
fact=fact*i;
}
printf("%d",fact);
	6
	fact=1
	-------------------- 	
	i=1
	1<= 3
	fact=1*1=1
	i=1+1=2
	-------------------- 	
	2<= 3
	fact=1*2=2
	i=2+1=3
	-------------------- 	
	3<= 3
	fact=2*3=6
	i=3+1=4
	-------------------- 	
	4<= 3

int i, fact=1;
for(i=3; i >0 ;i--)
{
fact=fact*i;
}
printf("%d",fact);
	6
	fact=1
	-------------------- 	
	i=3
	3>0
	fact=1*3=3
	i=3-1=2
	-------------------- 	
	2>0
	fact=3*2=6
	i=2-1=1
	-------------------- 	
	1> 0
	fact=6*1=6
	i=1-1=0
	-------------------- 	
	0>0

EXAMPLE	OUTPUT/TRACE

int n, m, sum;
for(n=1,m=5; n<=m ; n++,m--)
{
sum=n+m;
}
printf(“%d”,sum);
	6
	n=1,m=5
	1<= 5
	sum=1+5=6
	n=1+1=2
	m=5-1=4
	--------------------- 	
	2<=4
	sum=2+4=6
	n=2+1=3
	m=4-1=3
	-------------------- 	
	3<=3
	sum=3+3=6
	n=3+1=4
	m=3-1=2
	-------------------- 	
	4<=2

int i, sum=0;
for(i=1; i<=10 && sum<10 ;i++)
{
sum=sum+i;
}
printf(“%d”,sum);
	10
	sum=0 , i=1
	1<=10 && 0<10
	sum=0+1=1
	i=1+1=2
	--------------------- 	
	2<=10 && 1<10
	sum=1+2=3
	i=2+1=3
	-------------------- 	
	3<=10 && 3<10
	sum=3+3=6
	i=3+1=4
	-------------------- 	
	4<=10 && 6<10
	sum=6+4=10
	i=4+1=5
	------------------- 	
	5<=10 && 10<10

for LOOP EXAMPLES

EXAMPLE	OUTPUT

for(; ;)
{
printf(“C Programming”);
}
	C Programming
	Displays above Infinite times

for(; ;);
{
printf(“C Programming”);
}
	No Output
	It is infinite loop

Write a C program to print n Fibonacci number.
(0   1   1   2   3   5   8   13   21 34 ...)

```
#include<stdio.h>
void main()
{
	int f, f1=0,f2=1,n,i;
	printf("Enter value of n\n");
	scanf("%d",&n);
	printf("Fibonacci numbers are\n");
	for(i=1;i<=n;i++)
	{
		printf("%d\t",f1);
		f=f1+f2;
		f1=f2;
		f2=f;
	}
}
```

```
Enter value of n
3
Fibonacci numbers are
0     1    1
```

TRACE

```
f1=0, f2=1 n=3
------------------
i=1
1<= 3
0
f=0+1=1
f1=1
f2=1
i=1+1=2
------------------
2<= 3
1
f=1+1=2
f1=1
f2=2
i=2+1=3
------------------
3<= 3
1
f=1+2=3
f1=2
f2=3
i=3+1=4
-----------------
4<= 3
```

NESTING OF LOOPS

- One loop inside another loop is called nested loop.
- They are helpful to produce below pattern of results

To print the pattern

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

To print the pattern

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

To print the pattern

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

NESTING OF LOOPS

```
#include<stdio.h>
void main()
{
        int i, j;
        for(i=1;i<=3;i++)
        {
                for(j=1;j<=i;j++)
                {
                        printf("*\t");
                }
                printf("\n");
        }
}
```

OUTPUT

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

TRACE

```
i=1
1<=3
-------------------
j=1
1<=1
*
j=1+1=2
-------------------
2<=1
-------------------
i=1+1=2
-------------------
2<=3
-------------------
j=1
1<=2
*
j=1+1=2
-------------------
2<=2
*
j=2+1=3
-------------------
3<=2
-------------------
i=2+1=3
-------------------
.....
.....
.....
```

UNCONDITIONAL BRANCHING STATEMENTS OR JUMPS IN LOOPS OR LOOP INTERRUPTION

When a program is under execution, there may be a situation arise

to skip a part of the loop,

to terminate loop completely,

to terminate a program.

To support all above situations C has three types of jump statements

1. Skip a part of the loop
2. Terminate a loop

   To perform above goto, break and continue statements are used.
3. Terminate a program – exit() is used. Including header file stdlib.h

- **TO JUMP OUT OF A LOOP**

  - goto or break statements are used.

- **TO SKIP PART OF A LOOP AND CONTINUE WITH NEXT ITERATION**

  - continue statement is used.

- **TO SPECIFY TYPE OF RETURN DATA IN USER DEFINED FUNCTION**

  - return statement is used.

Disadvantages of using goto statement

1. The compiler generates less efficient code.

2. The program becomes complicated.

The goto statement must be avoided in C programming because it alters the sequence of execution without any condition. This is not allowed in structured programming language like C.

goto STATEMENT

- goto is an unconditional branching statement.
- It contains a label to identify branching location.
- The label is like an identifier can be used without declaration.
- The label must be followed by : (colon).

Syntax

```
goto label;
```

```
label: statement
```

```
#include<stdio.h>
void main()
{
            goto   s;
       a:  printf(“for”);
             goto m;
        s:  printf(“C Programming”);
             goto  a;
       m: printf(“Problem Solving”);
}
```

goto STATEMENT

Forward Branching

If goto statement appears first in the sequence of a program and the label statements appears next in that program is referred as forward branching.

Syntax

```
         goto  label;
         ---------------
         ---------------
label:  statements;
```

Backward Branching

If label statements are appears first in the sequence of a program and the goto statement appears next in that program is referred as forward branching.

Syntax

```
label:  statements;
         ---------------
         ---------------
         goto  label;
```

break STATEMENT	continue STATEMENT
1. In any loop if break statement is executed then all statements after break statement are skipped and loop is terminated.	1. In any loop if continue statement is executed then all statements after continue statement are skipped and loop is continues in next iteration.
2. This statement is used in swtich statement and in loop statements.	2. This statement is used only in loop statements.
3. Syntax break;	3. Syntax continue;

4. Example

```
for( i=1;i<5;i++)
{
      if (i==3)
      {
            break;
       }
       printf("%d\t",i);
}
```

Output

1 2

TRACE

```
i=1  1< 5
1=3
1
i=1+1=2
------------------
2< 5
2=3
2
i=2+1=3
------------------
3< 5
3=3
```

4. Example

```
for( i=1;i<5;i++)
{
      if (i==3)
      {
            continue;
       }
       printf("%d\t",i);
}
```

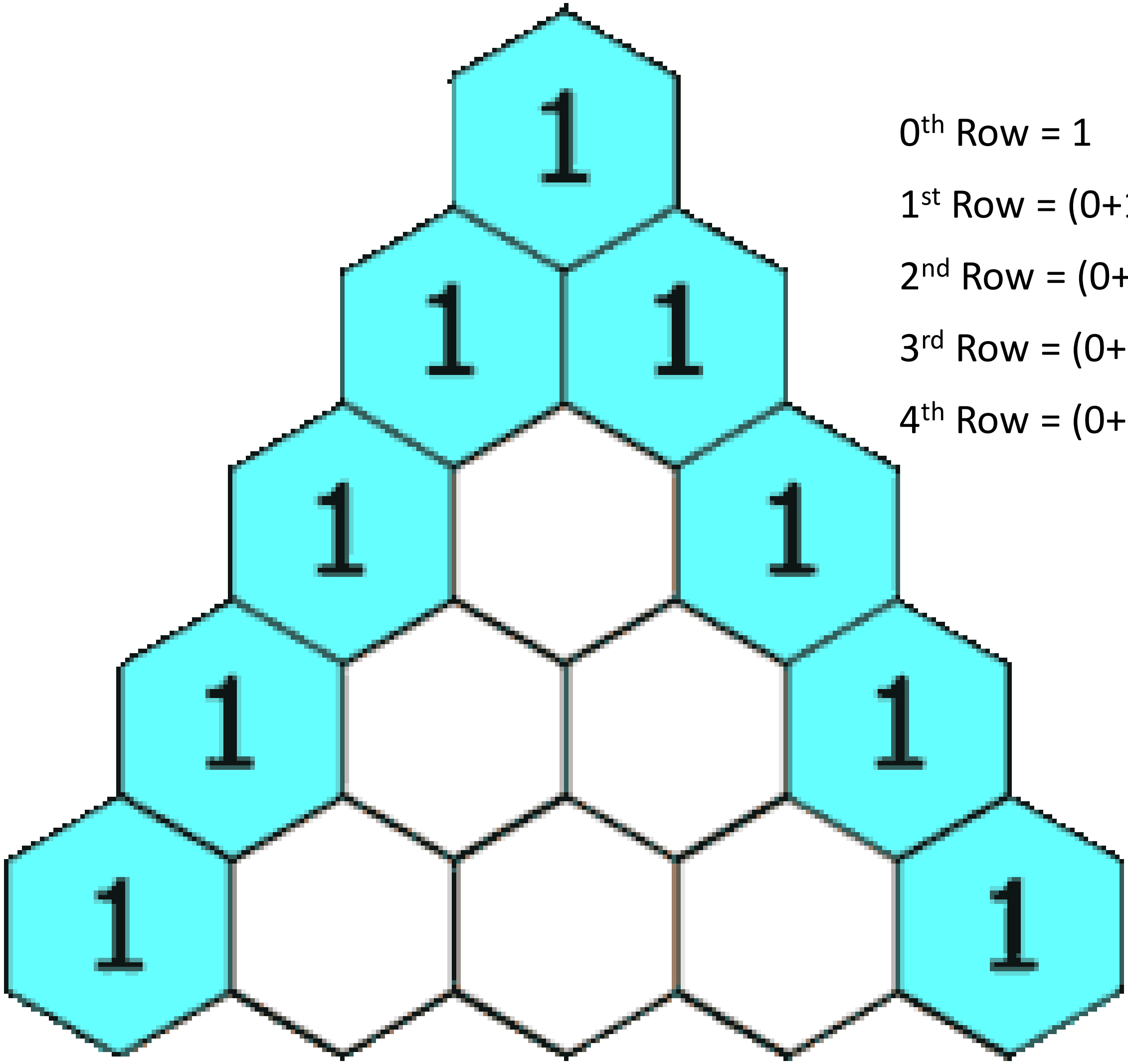
Output

1 2 4

TRACE

```
i=1  1< 5
1=3
1
i=1+1=2
------------------
2< 5
2=3
2
i=2+1=3
------------------
3< 5
3=3
i=3+1=4
------------------
4< 5
4=3
4
i=4+1=5
------------------
5<5
```

PLOTTING PASCAL'S TRIANGLE



0th Row = 1

1st Row = (0+1),(1+0) = 1, 1

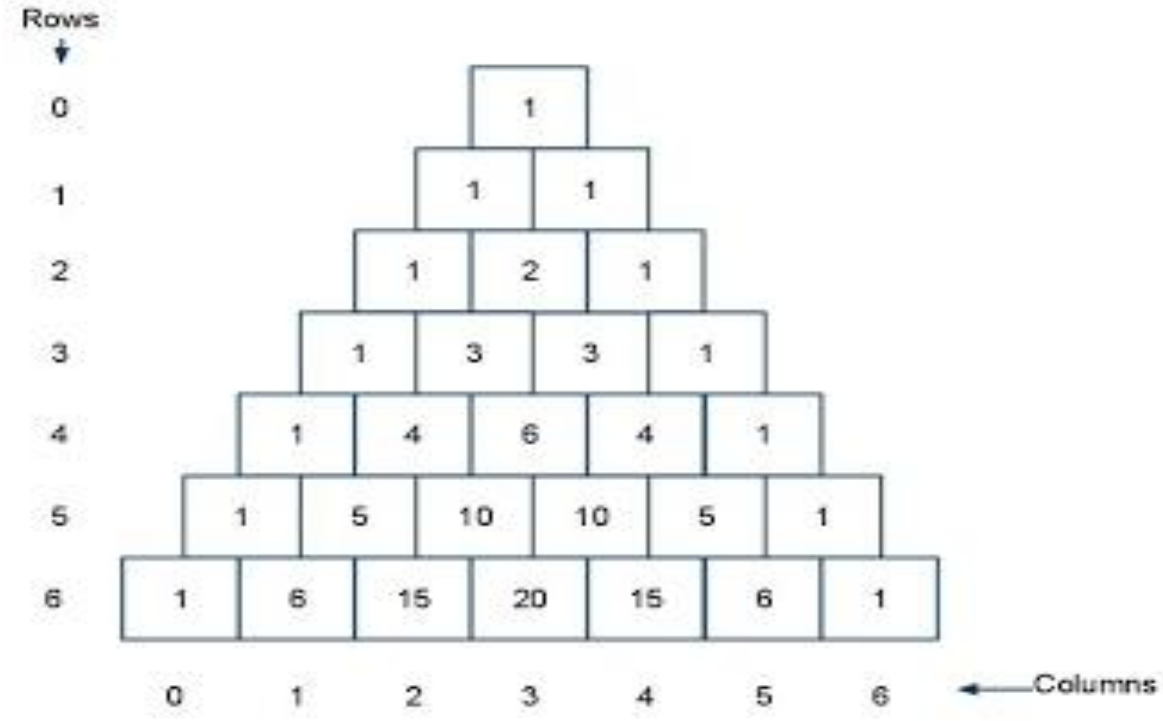
2nd Row = (0+1),(1+1),(1+0) = 1, 2, 1

3rd Row = (0+1),(1+2),(2+1),(1+0)= 1,3,3,1

4th Row = (0+1),(1+3),(3+3),(3+1),(1+0)= 1,4,6,4,1

PROPERTIES OF PASCAL'S TRIANGLE

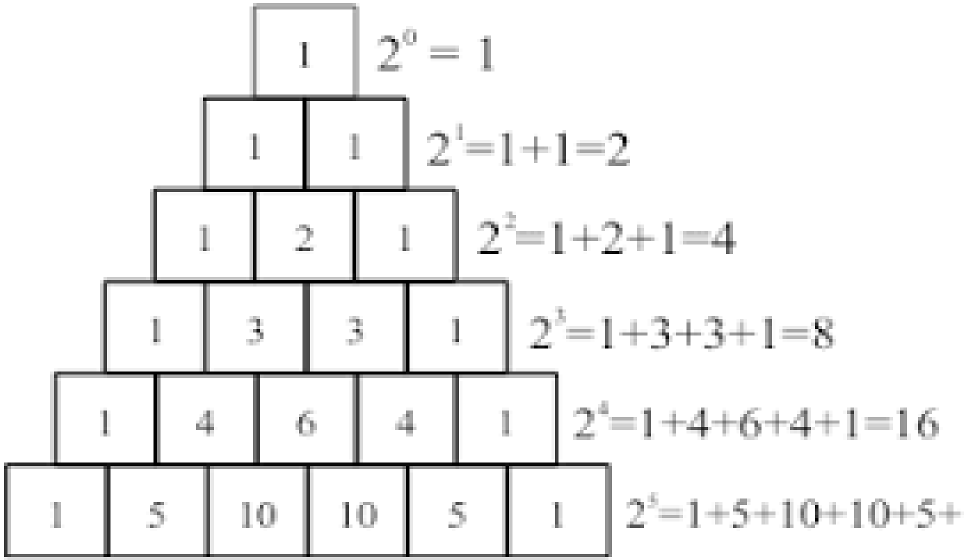
- 0th Row (Top most row) contains unique non-zero entry 1.
- Subsequent row is constructed by adding above left and right numbers treating blank entries as zero.
- Left and right edges always filled by 1.



Rows	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

PROPERTIES OF PASCAL'S TRIANGLE

- The sum of the numbers in each row of Pascal's triangle is equal to 2^n where n represents the row number in Pascal's triangle starting at n=0 for the first row at the top
- Every higher row sum is double than that of lower row.



1 $2^0 = 1$

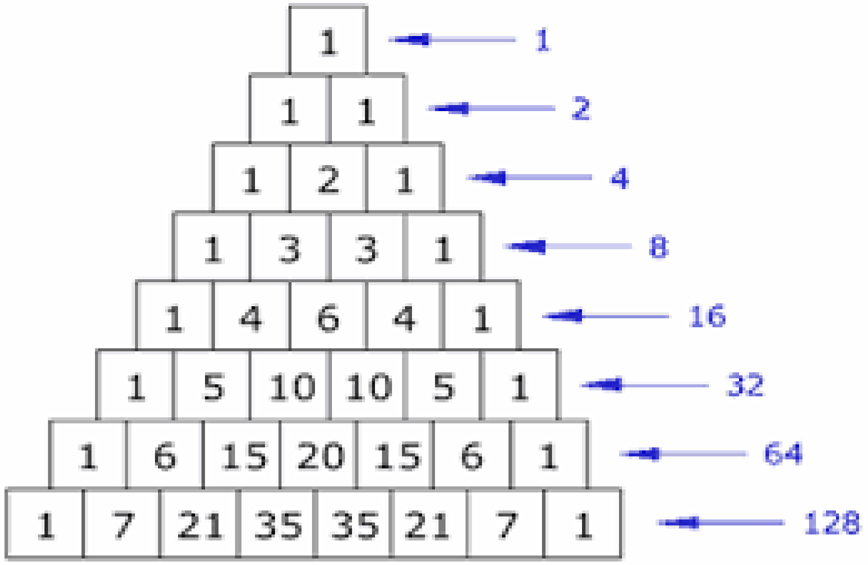
1 1 $2^1=1+1=2$

1 2 1 $2^2=1+2+1=4$

1 3 3 1 $2^3=1+3+3+1=8$

1 4 6 4 1 $2^4=1+4+6+4+1=16$

1 5 10 10 5 1 $2^5=1+5+10+10+5+$



1 → 1

1 1 → 2

1 2 1 → 4

1 3 3 1 → 8

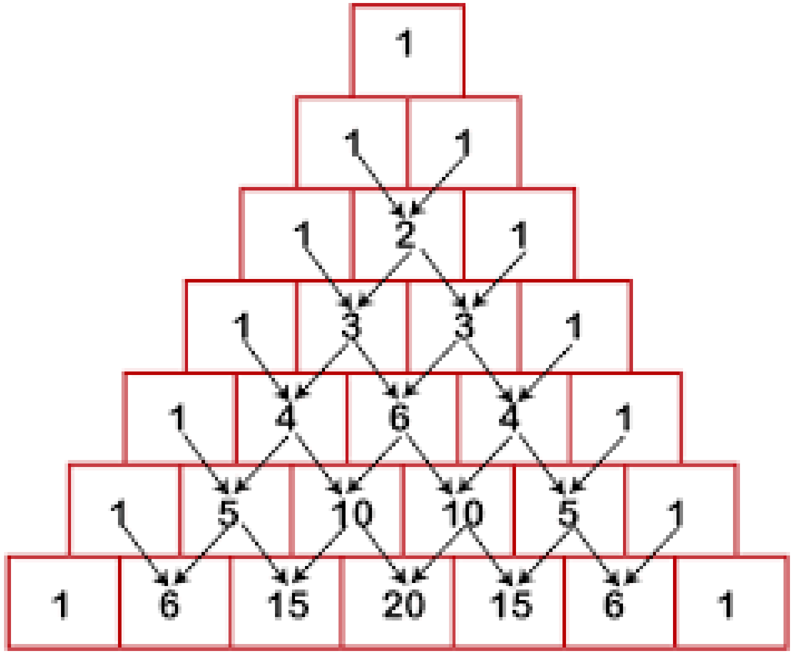
1 4 6 4 1 → 16

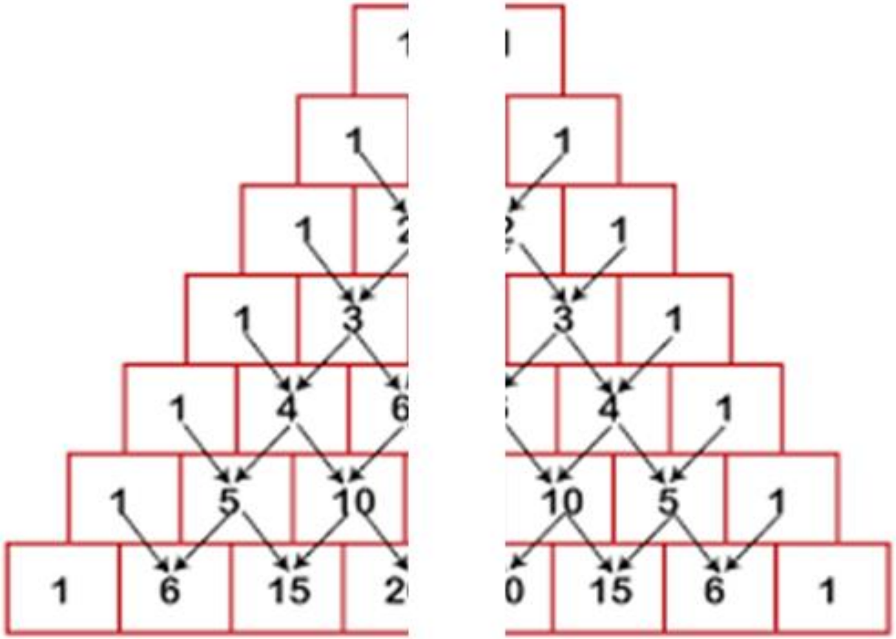
1 5 10 10 5 1 → 32

1 6 15 20 15 6 1 → 64

1 7 21 35 35 21 7 1 → 128

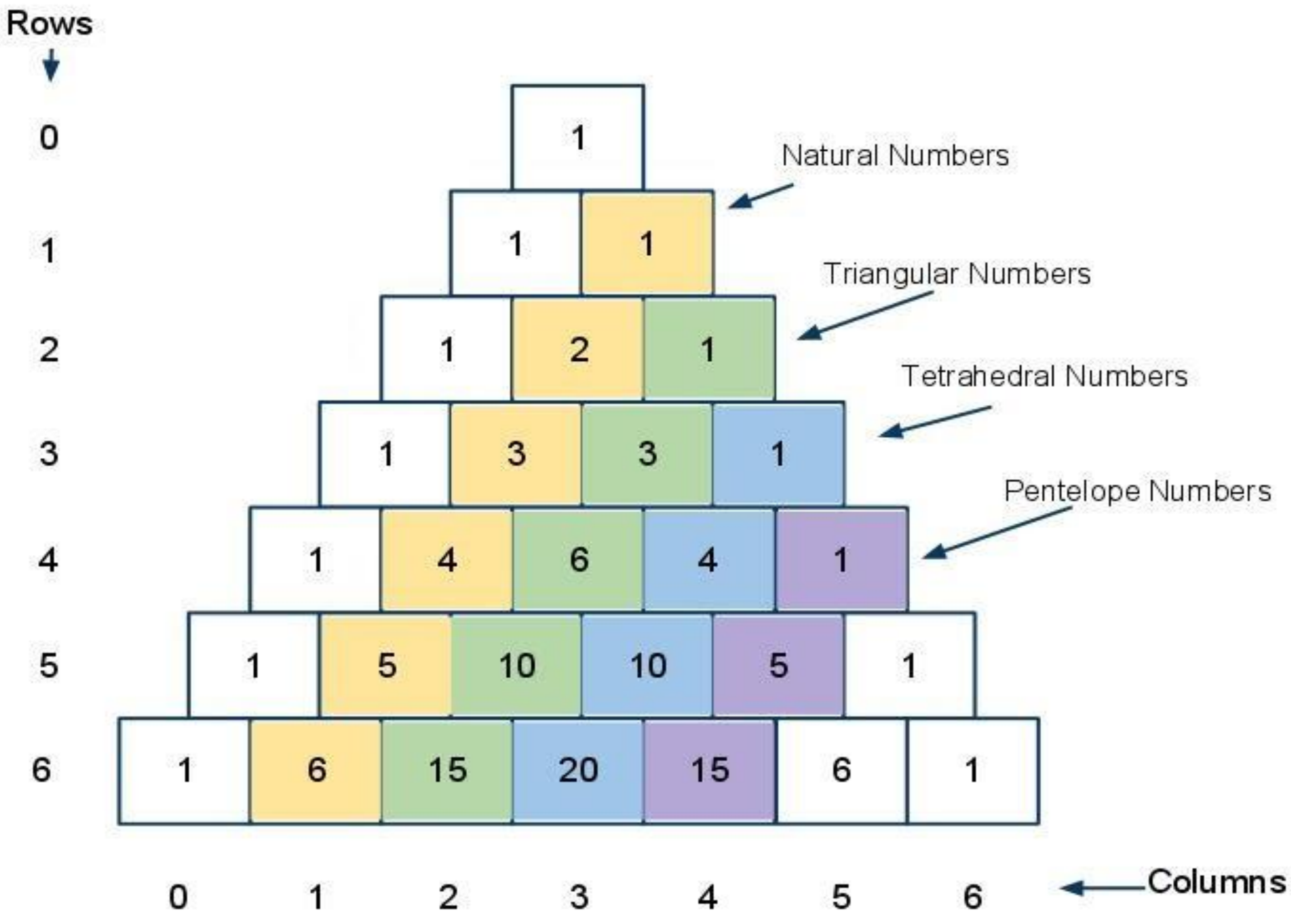
PROPERTIES OF PASCAL'S TRIANGLE





Pascal's triangle is symmetrical; if you cut it in half vertically, the numbers on the left and right side in equivalent positions are equal.

PROPERTIES OF PASCAL'S TRIANGLE



Rows

0 — 1

1 — 1 1

2 — 1 2 1

3 — 1 3 3 1

4 — 1 4 6 4 1

5 — 1 5 10 10 5 1

6 — 1 6 15 20 15 6 1

Columns: 0 1 2 3 4 5 6

Natural Numbers

Triangular Numbers

Tetrahedral Numbers

Pentelope Numbers

Number patterns in diagonals of Pascal's triangle

BINOMIAL COEFFICIENT

The combination of n things taken r at a time is denoted by ${}^{n}C_{r} = \frac{n!}{r!(n-r)!}$ This is known as binomial co-efficient.

It occurs as co-efficient of x in the expansion of $(1+x)^n$

Where $(1+x)^n = {}^{n}C_{0} . X^{0} + {}^{n}C_{1} . X^{1} + {}^{n}C_{2} . X^{2} + \dots + {}^{n}C_{n} . X^{n}$

$(1+x)^0 = {}^{0}C_{0} . X^{0}$

$= 1.1$

$= 1$

$(1+x)^1 = {}^{1}C_{0} . X^{0} + {}^{1}C_{1} . X^{1}$

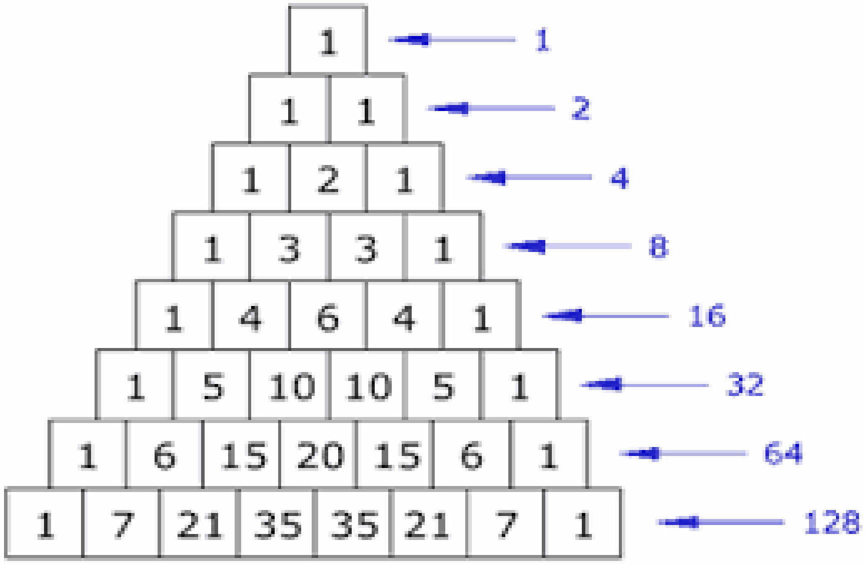
$= 1. 1 + 1. X$

$= 1+X$

$(1+x)^2 = {}^{2}C_{0} . X^{0} + {}^{2}C_{1} . X^{1} + {}^{2}C_{2} . X^{2}$

$= 1.1 + 2.X + 1.X^{2}$

$= 1 + 2X + X^{2}$



1 ← 1

1 1 ← 2

1 2 1 ← 4

1 3 3 1 ← 8

1 4 6 4 1 ← 16

1 5 10 10 5 1 ← 32

1 6 15 20 15 6 1 ← 64

1 7 21 35 35 21 7 1 ← 128

$(1+x)^3 = {}^{3}C_{0} . X^{0} + {}^{3}C_{1} . X^{1} + {}^{3}C_{2} . X^{2} + {}^{3}C_{3} . X^{3}$

$= 1.1 + 3.X + 3.X^{2} + 1.X^{3}$

$= 1 + 3X + 3X^{2} + X^{3}$

Write a C program to compute binomial co-efficient or to plot Pascal’s triangle.

```
#include<stdio.h>
void main()
{
    int n, r, c, s, b;
    printf("Enter number of rows / co-efficient:  ");
    scanf("%d",&n);
    for(r=0;r<n;r++)
    {
         for(s=1;s<=n-r;s++)
         printf("  ");
         b=1;
         for(c=0;c<=r;c++)
         {
             if(c!=0 && r!=0)
             {
                 b=b*(r-c+1)/c;
             }
             printf("%4d",b);
        }
        printf("\n");
    }
}
```

```
Enter number of rows / co-efficient:  5
        1
      1   1
     1   2   1
    1   3   3   1
   1   4   6   4   1
```

Write a C program to plot below triangle.

```
#include<stdio.h>
void main()
{
    int n, r, c, s;
    printf("Enter number of rows:  ");
    scanf("%d",&n);
    for(r=0;r<n;r++)
    {
        for(s=1;s<=n-r;s++)
        printf(" ");
        for(c=0;c<=r;c++)
        {
            printf(“1  ");
        }
        printf("\n");
    }
}
```

```
Enter number of rows:  5
    1
   1 1
  1 1 1
 1 1 1 1
1 1 1 1 1
```

Write a C program to plot below triangle.

```
#include<stdio.h>
void main()
{
    int n, r, c, s;
    printf("Enter number of rows:  ");
    scanf("%d",&n);
    for(r=0;r<n;r++)
    {
        for(s=1;s<=n-r;s++)
        printf(" ");
        for(c=0;c<=r;c++)
        {
            printf(“1");
        }
        printf("\n");
    }
}
```

```
Enter number of rows:  5
         1
       1 1
     1 1 1
   1 1 1 1
 1 1 1 1 1
```

Write a C program to plot below triangle.

```
#include<stdio.h>
void main()
{
    int n, r, c, s;
    printf("Enter number of rows:  ");
    scanf("%d",&n);
    for(r=1;r<n;r++)
    {
        for(s=1;s<=n-r;s++)
        printf(" ");
        for(c=1;c<=r;c++)
        {
            printf(“%d  “,c);
        }
        printf("\n");
    }
}
```

```
Enter number of rows:  5
    1
   1  2
  1  2  3
 1  2  3  4
```