**Right-Angled Triangle of Stars**

**Pattern:**

markdown

CopyEdit

\*

\* \*

\* \* \*

\* \* \* \*

**C Code:**

c

CopyEdit

#include <stdio.h>

int main() {

int i, j, n = 4;

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

for(j = 1; j <= i; j++) {

printf("\* ");

}

printf("\n");

}

return 0;

}

**Number Triangle**

**Pattern:**

CopyEdit

1

1 2

1 2 3

1 2 3 4

**C Code:**

c

CopyEdit

#include <stdio.h>

int main() {

int i, j, n = 4;

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

for(j = 1; j <= i; j++) {

printf("%d ", j);

}

printf("\n");

}

return 0;

}

**3. Inverted Right-Angled Triangle**

**Pattern:**

markdown

CopyEdit

\* \* \* \*

\* \* \*

\* \*

\*

**C Code:**

c

CopyEdit

#include <stdio.h>

int main() {

int i, j, n = 4;

for(i = n; i >= 1; i--) {

for(j = 1; j <= i; j++) {

printf("\* ");

}

printf("\n");

}

return 0;

}

**4. Pyramid Pattern**

**Pattern:**

markdown

CopyEdit

\*

\* \*

\* \* \*

\* \* \* \*

**C Code:**

c

CopyEdit

#include <stdio.h>

int main() {

int i, j, space, n = 4;

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

for(space = 1; space <= n - i; space++)

printf(" ");

for(j = 1; j <= i; j++)

printf("\* ");

printf("\n");

}

return 0;

}

**5. Diamond Pattern**

**Pattern:**

markdown

CopyEdit

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**C Code:**

c

CopyEdit

#include <stdio.h>

int main() {

int i, j, space, n = 4;

// Upper triangle

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

for(space = 1; space <= n - i; space++)

printf(" ");

for(j = 1; j <= i; j++)

printf("\* ");

printf("\n");

}

// Lower triangle

for(i = n - 1; i >= 1; i--) {

for(space = 1; space <= n - i; space++)

printf(" ");

for(j = 1; j <= i; j++)

printf("\* ");

printf("\n");

}

return 0;

}

## Option 1: C Pattern Program for Right-Angled Triangle

### Code:

#include <stdio.h>

int main() {

int i, j, rows = 5;

for(i = 1; i <= rows; i++) {

for(j = 1; j <= i; j++) {

printf("\*");

}

printf("\n");

}

return 0;

}

### Output:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

### Explanation:

This triangle pattern program in C language prints a simple right-angled triangle using stars. It uses nested loops, where the outer loop controls rows and the inner loop prints stars. Such basic C programs on patterns help beginners clearly understand loops.

## Option 2: C Pattern Program for Inverted Right-Angled Triangle

### Code:

#include <stdio.h>

int main() {

int i, j, rows = 5;

for(i = rows; i >= 1; i--) {

for(j = 1; j <= i; j++) {

printf("\*");

}

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

### Explanation:

This triangle pattern program in C language prints an inverted right-angled triangle. The outer loop runs in reverse (decreasing row size), and the inner loop prints stars for each row. Understanding inverted triangles is helpful in creating more complicated patterns. Such simple triangle pattern code in C is asked in beginner-level coding exams.

## Option 3: Pattern Program in C For Equilateral Triangle

### Code:

#include <stdio.h>

int main() {

int i, space, j, rows = 5;

for(i = 1; i <= rows; i++) {

for(space = 1; space <= rows-i; space++)

printf(" ");

for(j = 1; j <= (2\*i-1); j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

### Explanation:

This C program to print a pattern creates an equilateral or full pyramid triangle. It uses nested loops, first printing spaces and then stars. It's a popular triangle pattern in C program for understanding alignment and symmetry. Such C programs on patterns are frequently asked in interviews and tests to assess loop logic.

## Option 4: Pattern Program in C for Pascal's Triangle

### Code:

#include <stdio.h>

int fact(int n) {

if(n==0) return 1;

else return n\*fact(n-1);

}

int main() {

int i, j, rows = 5;

for(i = 0; i < rows; i++) {

for(j = 0; j <= rows-i-1; j++)

printf(" ");

for(j = 0; j <= i; j++)

printf("%d ", fact(i)/(fact(j)\*fact(i-j)));

printf("\n");

}

return 0;

}

### Output:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

### Explanation:

This classic triangle pattern program in C language prints Pascal's triangle using factorial logic. Each number is calculated based on combinations (binomial coefficients). Practicing this triangle pattern code in C helps strengthen understanding of mathematical logic and recursion in programming. Pascal's triangle pattern is common in programming and mathematics.

## Option 5: C Pattern Program for Hollow Triangle

### Code:

#include <stdio.h>

int main() {

int i, j, rows = 5;

for(i = 1; i <= rows; i++) {

for(j = i; j < rows; j++)

printf(" ");

for(j = 1; j <= (2\*i-1); j++) {

if(i == rows || j == 1 || j == (2\*i-1))

printf("\*");

else

printf(" ");

}

printf("\n");

}

return 0;

}

### Output:

\*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*\*

### Explanation:

This C program to print a pattern shows how to create a hollow equilateral triangle. It prints stars only at the triangle's edges using conditional statements within loops. Such a triangle pattern in C program improves your understanding of nested loops and conditionals.

## Option 6: Pattern in C Programming for Number Triangle

### Code:

#include <stdio.h>

int main() {

int i, j, rows = 5;

for(i = 1; i <= rows; i++) {

for(j = 1; j <= i; j++) {

printf("%d ", j);

}

printf("\n");

}

return 0;

}

### Output:

1

1 2

1 2 3

1 2 3 4 1 2 3 4 5

### Explanation:

This easy triangle pattern program in C language demonstrates printing numbers sequentially in each row. Such number patterns strengthen your looping logic and numeric control. This kind of triangle pattern in C programming is common in academic exams and is an excellent practice for beginners learning nested loops and number manipulation.

## Option 7: C Pattern Program for Alphabet Triangle

### Code:

#include <stdio.h>

int main() {

int i, j, rows = 5;

char ch;

for(i = 1; i <= rows; i++) {

ch = 'A';

for(j = 1; j <= i; j++) {

printf("%c ", ch++);

}

printf("\n");

}

return 0;

}

### Output:

A

A B

A B C

A B C D

A B C D E

### Explanation:

This interesting triangle pattern in C programming shows alphabet patterns. The program uses character increments (ASCII values) to print alphabets in sequence. Practicing alphabet patterns helps clearly understand character manipulation in C. Such triangle pattern code in C is useful for beginners to master character handling and nested loops effectively.

## Option 8: C Pattern Program for Inverted Full Pyramid

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

for (i = rows; i >= 1; i--) {

for (space = 1; space <= rows - i; space++)

printf(" ");

for (j = 1; j <= 2 \* i - 1; j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

### Explanation:

This pyramid pattern in C programming prints an inverted pyramid of stars. It reverses the logic of the full pyramid, decreasing the number of stars per row while increasing initial spaces. This C program to print a pattern helps in understanding loop logic clearly, often tested in beginner programming assignments.

## Option 9: Pattern Program in C for Half Pyramid

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j;

for (i = 1; i <= rows; i++) {

for (j = 1; j <= i; j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

### Explanation:

The half pyramid pattern program in C language is one of the simplest pattern programs. It prints stars increasing with each row using nested loops. Such simple pyramid pattern in C programming forms the basic understanding required for advanced patterns. Beginners often practice this pattern first when learning nested loops.

## Option 10: C Pattern Program for Inverted Half Pyramid

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j;

for (i = rows; i >= 1; i--) {

for (j = 1; j <= i; j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

### Explanation:

This simple inverted half pyramid pattern in C programming decreases the number of stars in each subsequent row. It's a beginner-friendly pyramid pattern code in C frequently asked in interviews and tests. Practicing this pattern strengthens your loop-handling logic and helps in understanding decremental loops.

## Option 11: C Pattern Program for Hollow Pyramid Pattern

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

for (i = 1; i <= rows; i++) {

for (space = 1; space <= rows - i; space++)

printf(" ");

for (j = 1; j <= 2 \* i - 1; j++) {

if (j == 1 || j == 2 \* i - 1 || i == rows)

printf("\*");

else

printf(" ");

}

printf("\n");

}

return 0;

}

### Output:

\*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*\*

### Explanation:

This hollow pyramid pattern program in C language prints stars only at pyramid borders, leaving inside empty. It uses nested loops and conditional statements for clear border definition. Such pyramid pattern code in C improves understanding of nested loops and logical conditions, often used in coding exams and interviews.

## Option 12: C Pattern Program for Alphabet Pyramid

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

char ch = 'A';

for (i = 1; i <= rows; i++) {

for (space = 1; space <= rows - i; space++)

printf(" ");

ch = 'A';

for (j = 1; j <= 2 \* i - 1; j++)

printf("%c", ch++);

printf("\n");

}

return 0;

}

### Output:

A

ABC

ABCDE

ABCDEFG

ABCDEFGHI

### Explanation:

This alphabet-based full pyramid pattern program in C language demonstrates incrementing alphabetically across each row. It clearly shows character handling using ASCII values and nested loops. Such alphabet pyramid patterns are valuable to beginners for improving their character and loop management skills, useful in many coding tests.

## Option 13: Solid Diamond Pattern Printing in C

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

// Upper half

for(i = 1; i <= rows; i++) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= 2\*i - 1; j++)

printf("\*");

printf("\n");

}

// Lower half

for(i = rows - 1; i >= 1; i--) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= 2\*i - 1; j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

### Explanation:

This diamond pattern program in C language consists of two pyramids: one upright and one inverted. It clearly shows symmetry and spacing logic using nested loops. This pattern printing in C programming example is useful in strengthening loop logic and is often asked in programming exams to test logical skills.

## Option 14: Hollow Diamond Pattern Printing in C

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

// Upper half

for(i = 1; i <= rows; i++) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= 2\*i - 1; j++)

if(j == 1 || j == 2\*i - 1)

printf("\*");

else

printf(" ");

printf("\n");

}

// Lower half

for(i = rows - 1; i >= 1; i--) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= 2\*i - 1; j++)

if(j == 1 || j == 2\*i - 1)

printf("\*");

else

printf(" ");

printf("\n");

}

return 0;

}

### Output:

\*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\*

### Explanation:

This hollow diamond pattern program in C language prints stars only at the edges, creating a hollow diamond shape. It requires clear conditional logic inside loops.

## Option 15: Rhombus Pattern Printing in C Programming

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

for(i = 1; i <= rows; i++) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= rows; j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

### Explanation:

The rhombus pattern program in C language is a shifted rectangle that forms a rhombus. The program uses loops clearly to manage initial spaces before printing stars.

## Option 16: Hollow Rhombus Pattern Program in C

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

for(i = 1; i <= rows; i++) {

for(space = 1; space <= rows - i; space++)

printf(" ");

for(j = 1; j <= rows; j++) {

if(i == 1 || i == rows || j == 1 || j == rows)

printf("\*");

else

printf(" ");

}

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*

\* \*

\* \*

\* \*

\*\*\*\*\*

### Explanation:

This hollow rhombus pattern program in C language prints stars only on the boundary, creating a clear hollow effect. It demonstrates how nested loops and conditionals manage shapes precisely.

## Option 17: C Program to Print Hourglass Pattern

### Code:

#include <stdio.h>

int main() {

int rows = 5, i, j, space;

// Upper half

for(i = rows; i >= 1; i--) {

for(space = 0; space < rows - i; space++)

printf(" ");

for(j = 1; j <= (2\*i - 1); j++)

printf("\*");

printf("\n");

}

// Lower half

for(i = 2; i <= rows; i++) {

for(space = 0; space < rows - i; space++)

printf(" ");

for(j = 1; j <= (2\*i - 1); j++)

printf("\*");

printf("\n");

}

return 0;

}

### Output:

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

### Explanation:

The hourglass pattern program in C language combines inverted and upright pyramids. It illustrates clear symmetry using nested loops and spacing logic.

### ****Print Multiplication Table of 5****

#include <stdio.h>

int main() {

for (int i = 1; i <= 10; i++) {

printf("5 x %d = %d\n", i, 5 \* i);

}

return 0;

}

#### **Output:**

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

### ****6. Reverse Count from 10 to 1****

#include <stdio.h>

int main() {

for (int i = 10; i >= 1; i--) {

printf("%d ", i);

}

return 0;

}

#### **Output:**

10 9 8 7 6 5 4 3 2 1

### ****Factorial of a Number****

#include <stdio.h>

int main() {

int num = 5, factorial = 1;

for (int i = 1; i <= num; i++) {

factorial \*= i;

}

printf("Factorial: %d", factorial);

return 0;

}

#### **Output:**

Factorial: 120

### ****Traverse and Print Array Elements****

#include <stdio.h>

int main() {

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

for (int i = 0; i < 5; i++) {

printf("%d ", arr[i]);

}

return 0;

}

#### **Output:**

10 20 30 40 50

### ****10. Check Prime Number****

#include <stdio.h>

int main() {

int num = 29, isPrime = 1;

for (int i = 2; i <= num / 2; i++) {

if (num % i == 0) {

isPrime = 0;

break;

}

}

if (isPrime) {

printf("%d is a prime number.", num);

} else {

printf("%d is not a prime number.", num);

}

return 0;

}

#### **Output:**

29 is a prime number.

### ****Example: Print a Pattern****

#include <stdio.h>

int main() {

int rows = 4;

for (int i = 1; i <= rows; i++) { // Outer loop for rows

for (int j = 1; j <= i; j++) { // Inner loop for columns

printf("\*"); // Print a star

}

printf("\n"); // Move to the next line

}

return 0;

}

### ****Output:****

\*

\*\*

\*\*\*

\*\*\*\*

# Compute multiplication table of a given integer

#include <stdio.h> // Include the standard input/output header file.

void main() {

int j, n; // Declare variables 'j' for loop counter and 'n' for user input.

printf("Input the number (Table to be calculated) : "); // Print a message to prompt user input.

scanf("%d", &n); // Read the value of 'n' from the user.

printf("\n"); // Print a newline for formatting.

for (j = 1; j <= 10; j++) { // Start a for loop to calculate the table up to 10.

printf("%d X %d = %d \n", n, j, n \* j); // Print the multiplication expression and result.

}

}

**Output:**

Input the number (Table to be calculated) : 15

15 X 1 = 15

15 X 2 = 30

15 X 3 = 45

15 X 4 = 60

15 X 5 = 75

15 X 6 = 90

15 X 7 = 105

15 X 8 = 120

15 X 9 = 135

15 X 10 = 150

# Display the sum of n number of odd natural number

#include <stdio.h> // Include the standard input/output header file.

void main() {

int i, n, sum = 0; // Declare variables 'i' for loop counter, 'n' for user input, and 'sum' to store the sum.

printf("Input number of terms : "); // Print a message to prompt user input.

scanf("%d", &n); // Read the value of 'n' from the user.

printf("\nThe odd numbers are :"); // Print a message indicating that odd numbers will be displayed.

for (i = 1; i <= n; i++) { // Start a loop to generate odd numbers based on user input.

printf("%d ", 2 \* i - 1); // Print the odd number.

sum += 2 \* i - 1; // Add the odd number to the running sum.

}

printf("\nThe Sum of odd Natural Number upto %d terms : %d \n", n, sum); // Print the sum of odd numbers.

}

**Output:**

Input number of terms : 10

The odd numbers are :1 3 5 7 9 11 13 15 17 19

The Sum of odd Natural Number upto 10 terms : 100

# Display the pattern like right angle using an asterisk

#include <stdio.h> // Include the standard input/output header file.

void main() {

int i, j, rows; // Declare variables 'i' and 'j' for loop counters, 'rows' for user input.

printf("Input number of rows : "); // Print a message to prompt user input.

scanf("%d", &rows); // Read the value of 'rows' from the user.

for (i = 1; i <= rows; i++) { // Start a loop to generate rows of asterisks.

for (j = 1; j <= i; j++) // Nested loop to print asterisks based on the current row.

printf("\*"); // Print an asterisk.

printf("\n"); // Move to the next line for the next row.

}

}

**Output:**

Input number of rows : 10

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

Right Angle Triangle with Sequentially Increasing Numbers

#include <stdio.h> // Include the standard input/output header file.

void main() {

int i, j, rows, k = 1; // Declare variables 'i' and 'j' for loop counters, 'rows' for user input, 'k' for incrementing numbers.

printf("Input number of rows : "); // Print a message to prompt user input.

scanf("%d", &rows); // Read the value of 'rows' from the user.

for (i = 1; i <= rows; i++) { // Start a loop to generate rows.

for (j = 1; j <= i; j++) // Nested loop to print numbers based on the current row.

printf("%d ", k++); // Print the value of 'k' and increment it.

printf("\n"); // Move to the next line for the next row.

}

}

**Output:**

Input number of rows : 4

1

2 3

4 5 6

7 8 9 10

# Display the pattern like a pyramid with numbers increased by 1

#include <stdio.h> // Include the standard input/output header file.

void main() {

int i, j, spc, rows, k, t = 1; // Declare variables 'i' and 'j' for loop counters, 'spc' for spaces, 'rows' for user input, 'k' for loop counter, 't' for incrementing numbers.

printf("Input number of rows : "); // Print a message to prompt user input.

scanf("%d", &rows); // Read the value of 'rows' from the user.

spc = rows + 4 - 1; // Calculate the initial number of spaces.

for (i = 1; i <= rows; i++) { // Start a loop to generate rows.

for (k = spc; k >= 1; k--) { // Loop to print spaces before the numbers.

printf(" ");

}

for (j = 1; j <= i; j++) { // Loop to print numbers based on the current row.

printf("%d ", t++); // Print the value of 't' and increment it.

}

printf("\n"); // Move to the next line for the next row.

spc--; // Decrement the number of spaces for the next row.

}

}

Copy

**Output:**

Input number of rows : 4

1

2 3

4 5 6

7 8 9 10

# Calculate the factorial of a given number

#include <stdio.h> // Include the standard input/output header file.

void main(){

int i, f = 1, num; // Declare variables 'i' for loop counter, 'f' to store factorial, 'num' for user input.

printf("Input the number : "); // Print a message to prompt user input.

scanf("%d", &num); // Read the value of 'num' from the user.

for(i = 1; i <= num; i++) // Start a loop to calculate factorial.

f = f \* i; // Calculate factorial.

printf("The Factorial of %d is: %d\n", num, f); // Print the calculated factorial.

}

**Output:**

Input the number : 5

The Factorial of 5 is: 120

16. Sum of Even Natural Numbers

Write a C program to display the sum of n terms of even natural numbers.  
Test Data :  
Input number of terms : 5  
Expected Output :  
The even numbers are :2 4 6 8 10  
The Sum of even Natural Number upto 5 terms : 30

#include <stdio.h> // Include the standard input/output header file.

void main()

{

int i, n, sum = 0; // Declare variables 'i' for loop counter, 'n' for user input, 'sum' to store the sum.

printf("Input number of terms : "); // Prompt the user to input the number of terms.

scanf("%d", &n); // Read the value of 'n' from the user.

printf("\nThe even numbers are :"); // Print a message.

for(i = 1; i <= n; i++) // Start a loop to generate even numbers.

{

printf("%d ", 2 \* i); // Print the even number.

sum += 2 \* i; // Add the even number to the sum.

}

printf("\nThe Sum of even Natural Number upto %d terms : %d \n", n, sum); // Print the sum of even numbers.

}

Copy

**Output:**

Input number of terms : 5

The even numbers are :2 4 6 8 10

The Sum of even Natural Number upto 5 terms : 30

18. Sum of Series [1 - X²/2! + X⁴/4! - …]

Write a program in C to find the sum of the series [ 1-X^2/2!+X^4/4!- .........].  
Test Data :  
Input the Value of x :2  
Input the number of terms : 5  
Expected Output :  
the sum = -0.415873  
Number of terms = 5  
value of x = 2.000000

#include <stdio.h>

void main()

{

float x,s,t,num=1.00,fac=1.00;

int i,n,pr,y=2,m=1;

printf("Input the Value of x :");

scanf("%f",&x);

printf("Input the number of terms : ");

scanf("%d",&n);

s=1.00; t=1.00;

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

{

for(pr=1;pr<=y;pr++)

{

fac=fac\*pr;

num=num\*x;

}

m=m\*(-1);

num=num\*m;

t=num/fac;

s=s+t;

y=y+2;

num=1.00;

fac=1.00;

}

printf("\nthe sum = %f\nNumber of terms = %d\nvalue of x = %f\n",s,n,x);

}

Copy

**Output:**

Input the Value of x :2

Input the number of terms : 5

the sum = -0.415873

Number of terms = 5

value of x = 2.000000

19. Harmonic Series and Their Sum

Write a program in C to display the n terms of a harmonic series and their sum.  
1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n terms  
Test Data :  
Input the number of terms : 5  
Expected Output :  
1/1 + 1/2 + 1/3 + 1/4 + 1/5 +  
Sum of Series upto 5 terms : 2.283334

#include <stdio.h> // Include the standard input/output header file.

void main()

{

int i, n; // Declare variables to store input and loop control.

float s = 0.0; // Initialize 's' to 0.0 to accumulate the sum.

// Prompt the user to input the number of terms.

printf("Input the number of terms : ");

scanf("%d", &n); // Read the value of 'n' from the user.

printf("\n\n"); // Print extra new lines for formatting.

// Loop to calculate the sum of the series.

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

{

if (i < n)

{

printf("1/%d + ", i); // Print the term with a plus sign.

s += 1 / (float)i; // Calculate and add the term to the sum.

}

if (i == n)

{

printf("1/%d ", i); // Print the last term without a plus sign.

s += 1 / (float)i; // Calculate and add the term to the sum.

}

}

// Print the final result.

printf("\nSum of Series upto %d terms : %f \n", n, s);

}

Copy

**Output:**

Input the number of terms : 5

1/1 + 1/2 + 1/3 + 1/4 + 1/5

Sum of Series upto 5 terms : 2.283334

21. Sum of Series [9 + 99 + 999 + …]

Write a program in C to display the sum of the series [ 9 + 99 + 999 + 9999 ...].  
Test Data :  
Input the number or terms :5  
Expected Output :  
9 99 999 9999 99999  
The sum of the saries = 111105

#include <stdio.h> // Include the standard input/output header file.

void main()

{

long int n, i, t = 9; // Declare variables to store input, control loop indices, and temporary value.

int sum = 0; // Initialize a variable to store the sum.

printf("Input the number or terms :"); // Prompt the user for input.

scanf("%ld", &n); // Read the value of 'n' from the user.

for (i = 1; i <= n; i++) // Loop for the number of terms.

{

sum += t; // Add 't' to the sum.

printf("%ld ", t); // Print the current value of 't'.

t = t \* 10 + 9; // Update 't' for the next iteration.

}

printf("\nThe sum of the series = %d \n", sum); // Print the sum of the series.

}

**Output:**

Input the number or terms :5

9 99 999 9999 99999

The sum of the series = 111105

22. Floyd's Triangle

Write a program in C to print Floyd's Triangle.

1

01

101

0101

10101

#include <stdio.h> // Include the standard input/output header file.

void main()

{

int i, j, n, p, q; // Declare variables to store input and control loop indices.

printf("Input number of rows : "); // Prompt the user for input.

scanf("%d", &n); // Read the value of 'n' from the user.

for (i = 1; i <= n; i++) // Loop for the number of rows.

{

if (i % 2 == 0) // Check if 'i' is even.

{

p = 1;

q = 0;

}

else // If 'i' is odd.

{

p = 0;

q = 1;

}

for (j = 1; j <= i; j++) // Loop for each element in the row.

{

if (j % 2 == 0) // Check if 'j' is even.

printf("%d", p); // Print 'p' if 'j' is even.

else

printf("%d", q); // Print 'q' if 'j' is odd.

}

printf("\n"); // Move to the next line after printing a row.

}

}

Copy

**Output:**

Input number of rows : 5

1

01

101

0101

10101

2. Sum of First 10 Natural Numbers

Write a C program to compute the sum of the first 10 natural numbers.  
Expected Output :  
The first 10 natural number is :  
1 2 3 4 5 6 7 8 9 10  
The Sum is : 55

#include <stdio.h> // Include the standard input/output header file.

int main() {

int j, sum = 0; // Declare variables 'j' for the loop counter and 'sum' to store the sum.

printf("The first 10 natural numbers are :\n"); // Print a message to indicate the output.

for (j = 1; j <= 10; j++) // Start a for loop to iterate from 1 to 10.

{

sum = sum + j; // Add the current value of 'j' to the sum.

printf("%d ", j); // Print the current value of 'j'.

}

printf("\nThe Sum is : %d\n", sum); // Print the sum of the first 10 natural numbers.

return 0; // Indicate successful execution of the program.

}

Copy

**Output:**

The first 10 natural number is :

1 2 3 4 5 6 7 8 9 10

The Sum is : 55

Odd Natural Numbers and Their Sum

Write a C program to display the n terms of odd natural numbers and their sum.  
Test Data  
Input number of terms : 10  
Expected Output :  
The odd numbers are :1 3 5 7 9 11 13 15 17 19  
The Sum of odd Natural Number upto 10 terms : 100

#include <stdio.h> // Include the standard input/output header file.

void main() {

int i, n, sum = 0; // Declare variables 'i' for loop counter, 'n' for user input, and 'sum' to store the sum.

printf("Input number of terms : "); // Print a message to prompt user input.

scanf("%d", &n); // Read the value of 'n' from the user.

printf("\nThe odd numbers are :"); // Print a message indicating that odd numbers will be displayed.

for (i = 1; i <= n; i++) { // Start a loop to generate odd numbers based on user input.

printf("%d ", 2 \* i - 1); // Print the odd number.

sum += 2 \* i - 1; // Add the odd number to the running sum.

}

printf("\nThe Sum of odd Natural Number upto %d terms : %d \n", n, sum); // Print the sum of odd numbers.

}

Copy

**Output:**

Input number of terms : 10

The odd numbers are :1 3 5 7 9 11 13 15 17 19

The Sum of odd Natural Number upto 10 terms : 100