**Lab Sheet 5**

1. WAP to read 10 numbers from user and find their sum and average.

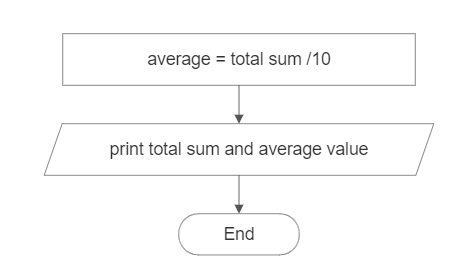
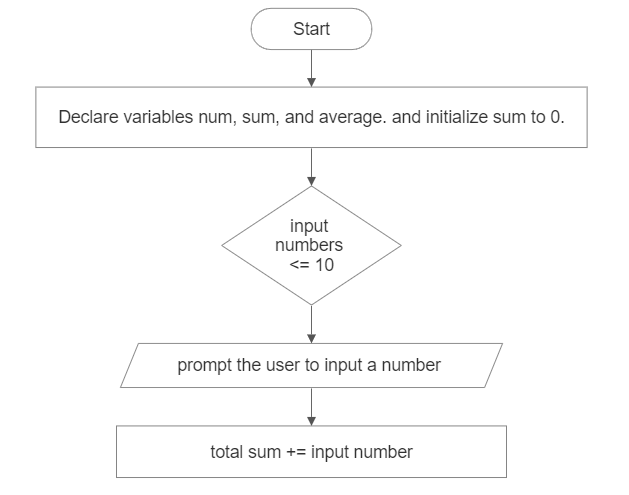
* **Objective**

The objective of this program is to read 10 numbers from the user, calculate their sum and average, and display the results.

* **Algorithm**

1. Start.
2. Declare variables num, sum, and average.
3. Initialize sum to 0.
4. Prompt the user to enter 10 numbers one by one and read them using scanf.
5. Add each number to the sum variable.
6. Calculate the average by dividing the sum by 10.
7. Display the sum and average using printf.
8. Stop.

* **Flowchart**



False

True

* **Code**

#include <stdio.h>

int main() {

int num, sum = 0;

float average;

printf("Enter 10 numbers:\n");

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

printf("Number %d: ", i);

scanf("%d", &num);

sum += num;

}

average = (float) sum / 10;

printf("Sum: %d\n", sum);

printf("Average: %.2f\n", average);

return 0;

}

* **Output**

Enter 10 numbers:

Number 1: 5

Number 2: 8

Number 3: 12

Number 4: 3

Number 5: 6

Number 6: 10

Number 7: 7

Number 8: 9

Number 9: 4

Number 10: 11

Sum: 75

Average: 7.50

* **Discussion and Conclusion**

This program successfully reads 10 numbers from the user, calculates their sum and average, and displays the results. It uses a for loop to iterate 10 times and prompts the user to enter a number on each iteration. The numbers are then added to the sum variable. After the loop, the average is calculated by dividing the sum by 10. Finally, the sum and average are displayed using printf.

1. WAP to display the multiplication table of integer given by the user.

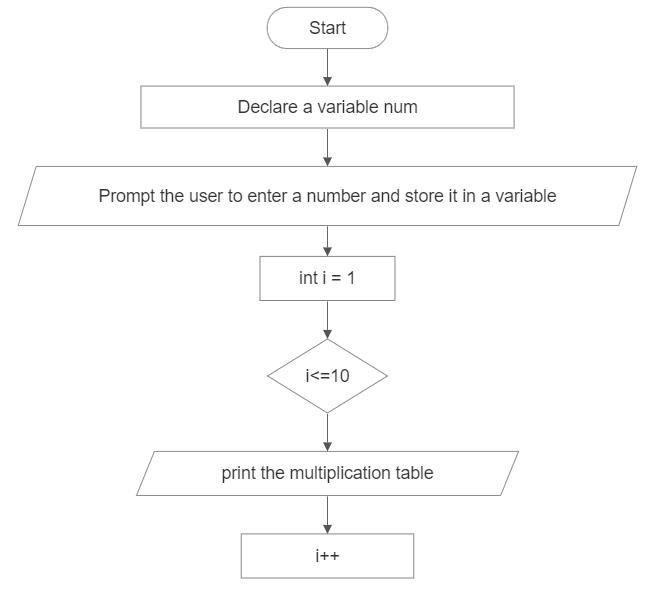
* **Objective**

The objective of this program is to take an integer input from the user and display its multiplication table.

* **Algorithm**

1. Start.
2. Declare variables num and i.
3. Prompt the user to enter a number and read it using scanf.
4. Print the multiplication table heading.
5. Use a loop to iterate from 1 to 10.
6. Inside the loop, calculate the product of num and i.
7. Print the multiplication expression and the product using printf.
8. Stop.

* **Flowchart**



True

False

* **Code**

#include <stdio.h>

int main( )

{

int num, i;

printf("Enter a number: ");

scanf("%d", &num);

printf("Multiplication table of %d:\n", num);

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

{

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

}

return 0;

}

* **Output**

Enter a number: 5

Multiplication table of 5:

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

* **Discussion and Conclusion**

This program takes an integer input from the user and displays its multiplication table up to 10. It uses a for loop to iterate from 1 to 10 and calculates the product of the input number and the current iteration value. The multiplication expression and the product are then printed using printf

1. WAP to input two integer values from the user and print the even number between the range of integers. Also count the even number and display the count as well [Hint: if user enters 10 and 100. The program should print and count even numbers between 10 and 100].

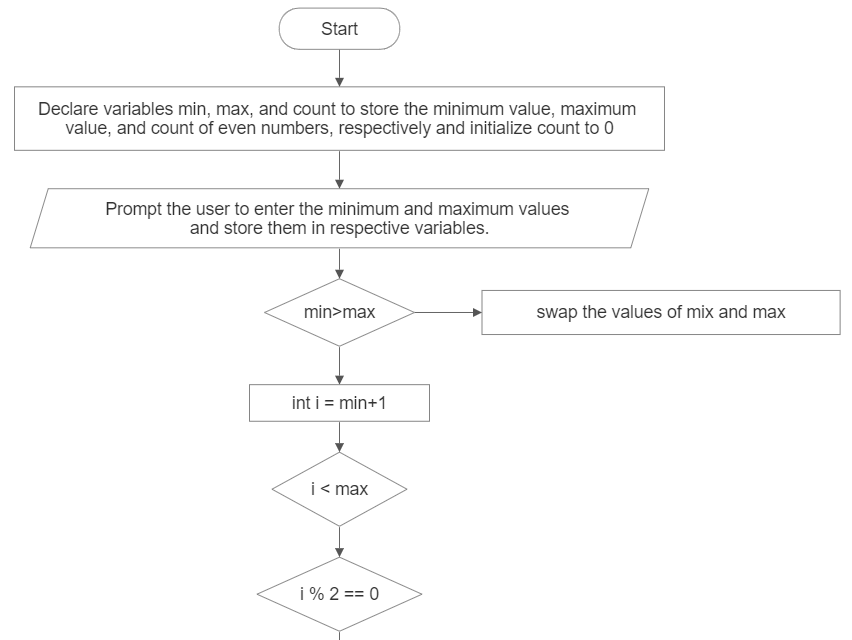
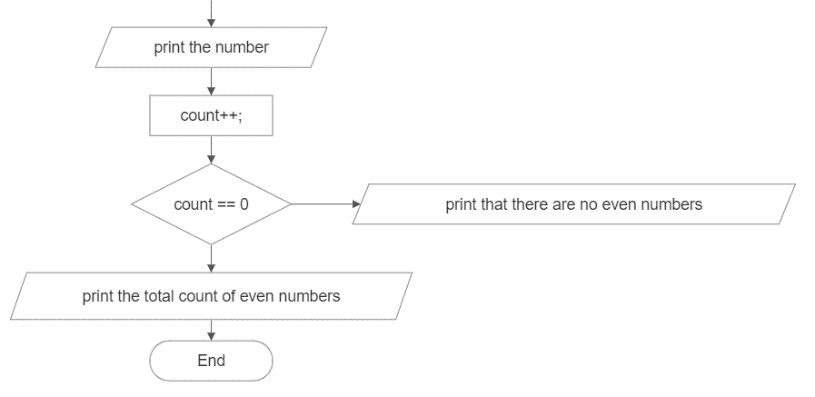
* **Objective**

The objective of this program is to take two integer values from the user and print the even numbers between the range of integers. Additionally, the program counts the even numbers and displays the count.

* **Algorithm**

1. Start.
2. Declare variables min, max, and count to store the minimum value, maximum value, and count of even numbers, respectively and initialize count to 0.
3. Prompt the user to enter the minimum and maximum values and store them in respective variables.
4. If min is greater than max, swap their values.
5. Print the message indicating the range of numbers.
6. Initialize a loop with a variable i starting from min + 1 and ending at max - 1.
7. Within the loop, check if i is divisible by 2 (i.e., an even number).
8. If i is even, print its value and increment the count variable.
9. After the loop, check if the count variable is 0.
10. If count is 0, print "none" to indicate that there are no even numbers in the range.
11. Print the total count of even numbers.
12. End.

* **Flowchart**



False

True

False

False

True

True

True

False

* **Code**

#include <stdio.h>

int main( ) {

int min, max, count = 0;

printf("Enter the minimum and maximum value: ");

scanf("%d%d", &min, &max);

if (min > max) {

int temp = min;

min = max;

max = temp;

}

printf("The even numbers between %d and %d are: ", min, max);

for (int i = min + 1; i < max; i++) {

if (i % 2 == 0) {

printf("%d ", i);

count++;

}

} if (count == 0) {

printf("none ");

}

printf("\nThere are %d even numbers.\n", count);

return 0;

}

* **Output**

Enter the minimum and maximum value: 15 25

The even numbers between 15 and 25 are: 16 18 20 22 24

There are 5 even numbers.

* **Discussion and Conclusion**

This program takes two integer values from the user representing the minimum and maximum values. It then finds and prints all the even numbers between the given range, along with the count of even numbers. If no even numbers are found, it displays "none" as the output. The program is implemented using formatted input/output in the C programming language. By using a loop and checking divisibility by 2, the program efficiently determines the even numbers within the specified range.

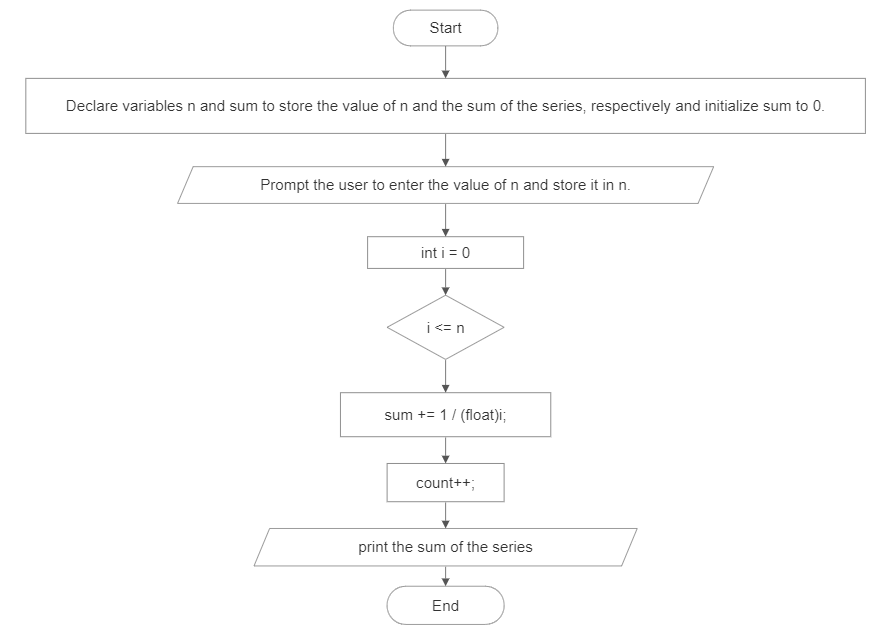
1. WAP to display sum of series: 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n

* **Objective**

The objective of this program is to calculate and display the sum of the given series: 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n.

* **Algorithm**

1. Start.
2. Declare variables n and sum to store the value of n and the sum of the series, respectively and initialize sum to 0.
3. Prompt the user to enter the value of n and store it in n.
4. Use a loop with a variable i starting from 1 and ending at n.
5. Within the loop, add 1 / (float)i to the sum variable. The (float) conversion is used to ensure floating-point division.
6. After the loop, print the value of sum as the sum of the series.
7. End.

* **Flowchart**

i++

False

True

* **Code**

#include <stdio.h>

int main()

{

int n;

float sum = 0;

printf("Enter the value of n: ");

scanf("%d", &n);

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

{

sum += 1 / (float)i;

}

printf("The sum of series: 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n is %f", sum);

return 0;

}

* **Output**

Enter the value of n: 5

The sum of the series: 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n is 2.283334

* **Discussion and Conclusion**

This program calculates the sum of the given series by taking the value of n from the user. It uses a loop to iterate from 1 to n and adds the reciprocal of each number to the sum. The program utilizes the (float) type conversion to perform floating-point division and accurately calculate the sum. Finally, it displays the result using formatted output. The program is implemented in the C programming language.

1. WAP to display sum of series: 1 + 1/2! + 1/3! + 1/4! + 1/5! ... 1/n!

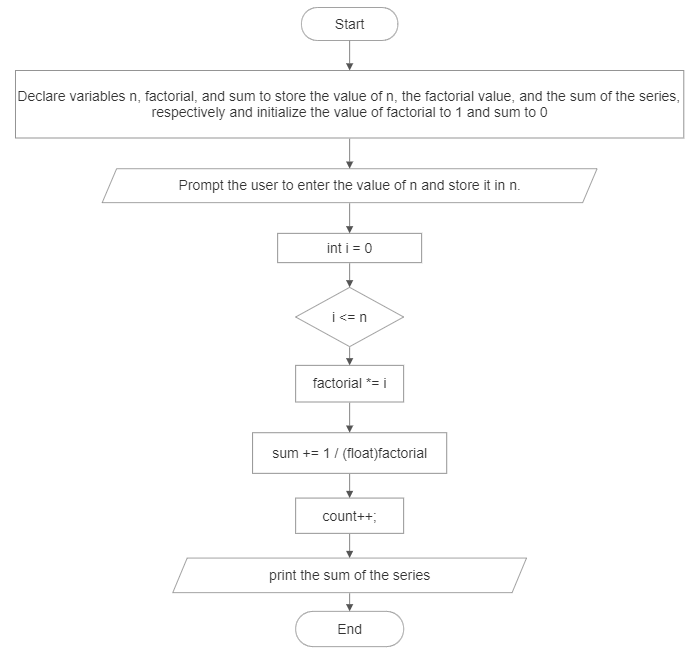
* **Objective**

The objective of this program is to calculate and display the sum of the series: 1 + 1/2! + 1/3! + 1/4! + 1/5! ... 1/n!.

* **Algorithm**

1. Start.
2. Declare variables n, factorial, and sum to store the value of n, the factorial value, and the sum of the series, respectively and initialize the value of factorial to 1 and sum to 0.
3. Prompt the user to enter the value of n and store it to the respective variable.
4. Use a loop with a variable i starting from 1 and ending at n.
5. Within the loop, calculate the factorial of i by multiplying it with the current value of factorial and add 1 / (float)factorial to the sum variable.
6. After the loop, print the value of sum as the sum of the series.
7. End.

* **Flowchart**



i++

True

False

* **Code**

#include <stdio.h>

int main()

{

int n, factorial= 1;

float sum = 0;

printf("Enter the value of n: ");

scanf("%d", &n);

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

{

factorial \*= i;

sum += 1 / (float)factorial;

}

printf("The sum of series: 1 + 1/2! + 1/3! + 1/4! + 1/5! ... 1/n! is %f", sum);

return 0;

}

* **Output**

Enter the value of n: 5

The sum of the series: 1 + 1/2! + 1/3! + 1/4! + 1/5! ... 1/n! is 1.716667

* **Discussion and Conclusion**

This program calculates the sum of the given series by taking the value of n from the user. It uses a loop to iterate from 1 to n, calculating the factorial of each number and adding the reciprocal of the factorial to the sum. The program utilizes the (float) type conversion to perform floating-point division and accurately calculate the sum. Finally, it displays the result using formatted output. The program is implemented in the C programming language.

1. WAP to display sum of series: x + x2/2! + x3/3! + x4/4! + x5/5! ... xn/n!

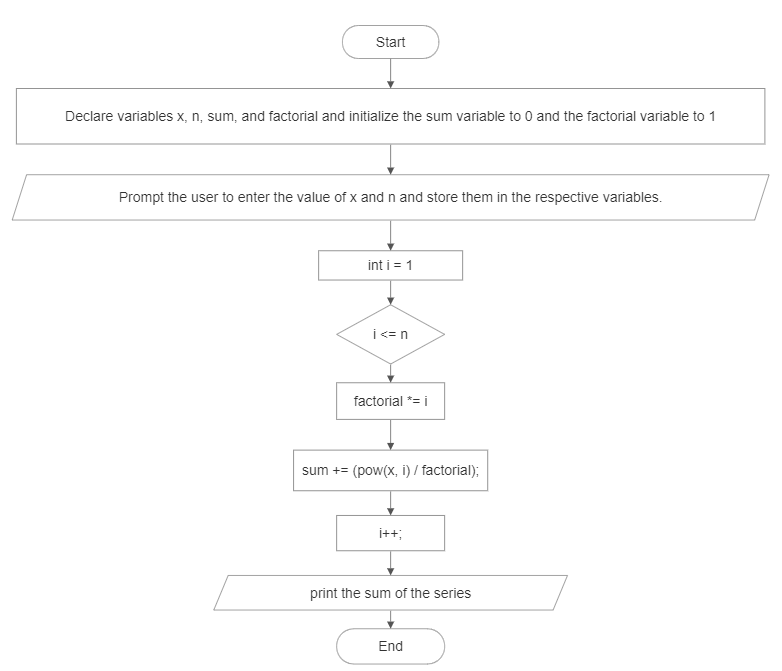
* **Objective**

The objective of this program is to calculate and display the sum of the series: x + x^2/2! + x^3/3! + x^4/4! + x^5/5! ... + x^n/n!.

* **Algorithm**

1. Start.
2. Declare variables x, n, sum, and factorial and initialize the sum variable to 0 and the factorial variable to 1.
3. Prompt the user to enter the value of x and n and store them in the respective variables.
4. Use a loop with a variable i starting from 1 and ending at n.
5. Within the loop, calculate the factorial of i by multiplying it with the current value of factorial.
6. Calculate x raised to the power of i using the pow() function from the math.h library.
7. Add (x^i / factorial) to the sum variable.
8. After the loop, print the value of sum as the sum of the series.
9. End.

* **Flowchart**



True

False

* **Code**

#include <stdio.h>

#include <math.h>

int main()

{

float x, sum = 0;

int n, factorial = 1;

printf("Enter the value of x: ");

scanf("%f", &x);

printf("Enter the value of n: ");

scanf("%d", &n);

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

{

factorial \*= i;

sum += (pow(x, i) / factorial);

}

printf("The sum of series: x + x2/2! + x3/3! + x4/4! + x5/5! ... xn/n! is %f.", sum);

return 0;

}

* **Output**

Enter the value of x: 2.5

Enter the value of n: 5

The sum of the series: x + x^2/2! + x^3/3! + x^4/4! + x^5/5! ... + x^n/n! is 24.041666.

* **Discussion and Conclusion**

This program calculates the sum of the given series by taking the value of x and n from the user. It uses a loop to iterate from 1 to n, calculating the factorial of each number and raising x to the power of i. It then adds (x^i / factorial) to the sum. The program utilizes the pow() function from the math.h library to perform the power calculation. Finally, it displays the result using formatted output. The program is implemented in the C programming language.

1. WAP to find the value cos(x) without using cos(x) library function.

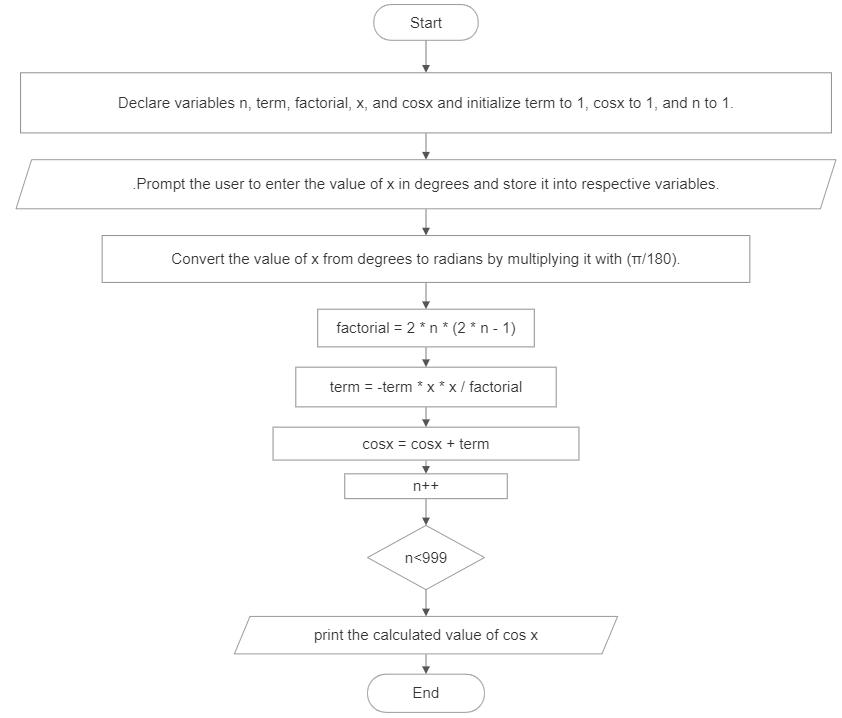
* **Objective:**

The objective of this program is to calculate the value of cos(x) without using the cos(x) library function. It uses the Taylor series expansion of cos(x) to approximate the value with a given accuracy.

* **Algorithm**

1. Start.
2. Declare variables n, term, factorial, x, and cosx and initialize term to 1, cosx to 1, and n to 1.
3. Prompt the user to enter the value of x in degrees and store it into respective variables.
4. Convert the value of x from degrees to radians by multiplying it with (π/180).
5. Use a do-while loop to calculate the terms of the Taylor series expansion.
6. Inside the loop, calculate the factorial of (2 \* n) \* (2 \* n - 1).
7. Calculate the next term of the series using the previous term, x, and factorial.
8. Add the term to the cosx variable.
9. Increment n by 1.
10. Repeat steps 8-11 until n reaches a sufficiently large value (e.g., 999).
11. Print the calculated value of cos(x)
12. End.

* **Flowchart**



False

True

* **Code**

#include <stdio.h>

int main()

{

int n = 1;

float term = 1, factorial, x, cosx = 1;

printf("Enter the value of x (in degrees): ");

scanf("%f", &x);

float x\_in\_deg = x;

/\* Converting degrees to radians \*/

x = x \* (3.142 / 180.0);

do

{

factorial = 2 \* n \* (2 \* n - 1);

term = -term \* x \* x / factorial;

cosx = cosx + term;

n++;

} while (n < 999);

printf("Cos(%.2fdeg): %.4f\n", x\_in\_deg, cos);

return 0;

}

* **Output**

Enter the value of x (in degrees): 60

Cos(60.00deg): 0.50

* **Discussion and Conclusion**

In this program, we utilize the Taylor series expansion to calculate the value of cos(x) without relying on the built-in cos(x) library function. The Taylor series for cos(x) is given by:

cos(x) = 1 - (x^2 / 2!) + (x^4 / 4!) - (x^6 / 6!) + ...

It prompts the user to enter the value of x in degrees, converts it to radians, and then calculates the series terms and the approximate value of cos(x). The accuracy of the approximation depends on the number of terms used in the series. The program prints the calculated value of cos(x) along with the entered value of x in degrees. The implementation is in the C programming language.

1. WAP to display weather a number is Armstrong or not.

* **Objective**

The objective of this program is to determine whether a given number is an Armstrong number or not. An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits.

* **Algorithm**

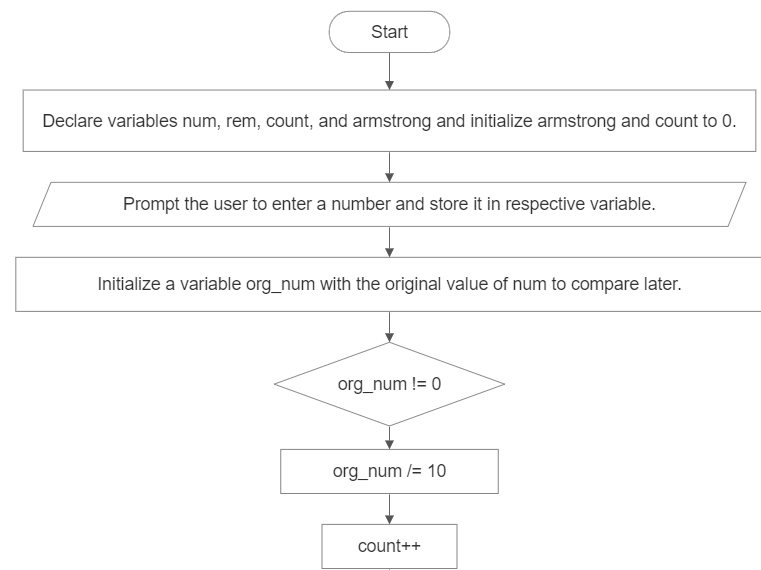
True

False

False

True

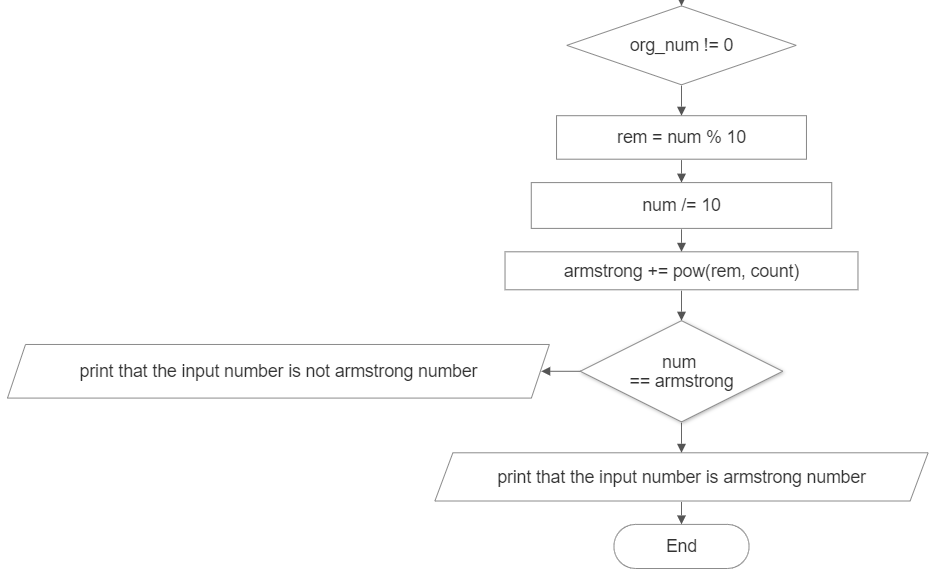
1. Start.
2. Declare variables num, rem, armstrong, and count and initialize armstrong and count to 0.
3. Prompt the user to enter a number and store it in the respective variable.
4. Initialize a variable org\_num with the value of num.
5. Calculate the number of digits in num by repeatedly dividing org\_num by 10 and incrementing count until org\_num becomes 0.
6. Reset org\_num to its original value.
7. Use a while loop to calculate the sum of individual digits raised to power count of org\_num.
8. Check if num is equal to armstrong.
9. If they are equal, print that num is an Armstrong number.
10. If they are not equal, print that num is not an Armstrong number.
11. End.



* **Flowchart**

False

True



False

True

* **Code**

#include <stdio.h>

#include <math.h>

int main()

{

int num, rem, armstrong = 0, count = 0;

printf("Enter a number: ");

scanf("%d", &num);

int org\_num = num;

while (org\_num != 0)

{

org\_num /= 10;

count++;

}

org\_num = num;

while (org\_num != 0)

{

rem = org\_num % 10;

org\_num /= 10;

armstrong += pow(rem, count);

}

if (num == armstrong)

{

printf("%d is an Armstrong number.", num);

}

else

{

printf("%d is not an Armstrong number.", org\_num);

}

return 0;

}

* **Output**

Enter a number: 153

153 is an Armstrong number.

* **Discussion and Conclusion**

This program determines whether a given number is an Armstrong number or not. It calculates the number of digits in the given number and then calculates the sum of the digits raised to the power of the number of digits. If the calculated sum is equal to the original number, then it is an Armstrong number. The program prints the appropriate message based on the result. The implementation is in the C programming language.

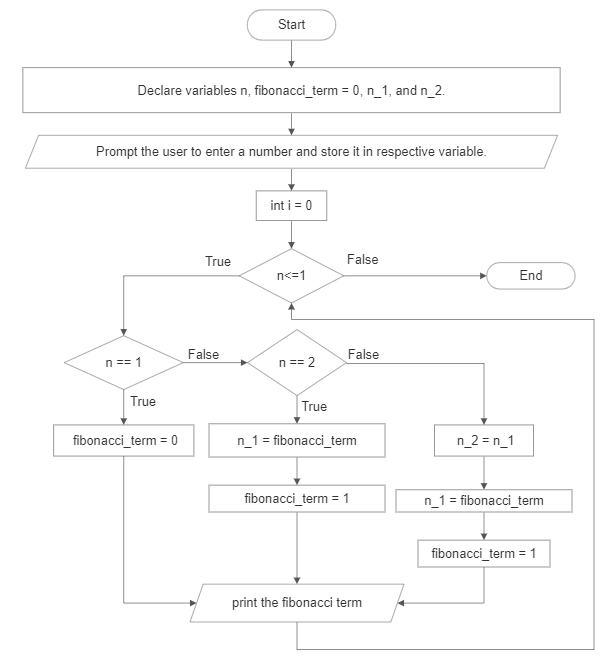
1. WAP to display the first n terms of Fibonacci series.

* **Objective**

The objective of this program is to display the first n terms of the Fibonacci series.

* **Algorithm**

1. Start.
2. Declare variables n, fibonacci\_term = 0, n\_1, and n\_2.
3. Prompt the user to enter a number and store it in respective variable.
4. Use a for loop to iterate from 1 to n.
5. Initialize fibonacci\_term as 0 for the first term.
6. If i is equal to 2, set n\_1 as fibonacci\_term and update fibonacci\_term to 1.
7. For subsequent terms (i > 2), update n\_2 as n\_1, n\_1 as fibonacci\_term, and calculate fibonacci\_term as the sum of n\_2 and n\_1.
8. Print the value of fibonacci\_term.
9. End.

* **Flowchart**
* **Code**

#include <stdio.h>

int main()

{

// n\_1 is n-1th term and n\_2 is n-2th term of fibonacci series

int n, fibonacci\_term = 0, n\_1, n\_2;

printf("Enter a number: ");

scanf("%d", &n);

printf("Fibonacci Series to %dth term: \n", n);

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

{

if (i == 1)

{

fibonacci\_term = 0;

}

else if (i == 2)

{

n\_1 = fibonacci\_term;

fibonacci\_term = 1;

}

else

{

n\_2 = n\_1;

n\_1 = fibonacci\_term;

fibonacci\_term = n\_2 + n\_1;

}

printf("%d ", fibonacci\_term);

}

return 0;

}

* **Output**

Enter a number: 10

Fibonacci Series to 10th term:

0 1 1 2 3 5 8 13 21 34

* **Discussion and Conclusion**

This program displays the first n terms of the Fibonacci series. It takes a number as input from the user and uses a for loop to calculate and print each term of the series. The Fibonacci series starts with 0 and 1, and each subsequent term is the sum of the previous two terms. The program utilizes variables and a loop to generate the desired number of terms. The implementation is in the C programming language.

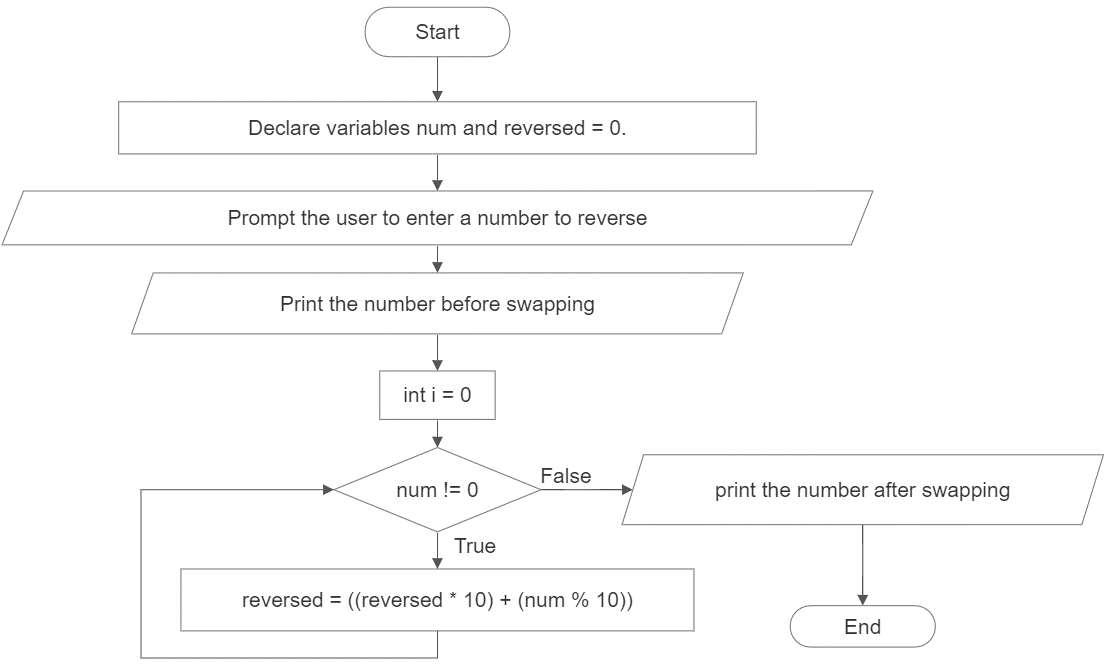
1. WAP to display the number in reverse order.

* **Objective**

The objective of this program is to display a number in reverse order.

* **Algorithm**

1. Start.
2. Declare variables num and reversed = 0.
3. Prompt the user to enter a number and store the input in respective variable.
4. Print the number before swapping.
5. Use a for loop to reverse the number.
6. Repeat the loop until num becomes 0.
7. In each iteration, calculate the reversed number by multiplying it by 10 and adding the last digit of num using the modulus operator.
8. Divide num by 10 to remove the last digit.
9. Print the reversed number
10. End.

* **Flowchart**
* **Code**

#include <stdio.h>

int main()

{

int num, reversed = 0;

printf("Enter a number: ");

scanf("%d", &num);

printf("Before Swapping: %d\n", num);

for (int i = 0; num != 0; i++)

{

reversed = ((reversed \* 10) + (num % 10));

num /= 10;

}

printf("After Swapping: %d\n", reversed);

return 0;

}

* **Output**

Enter a number: 12345

Before Swapping: 12345

After Swapping: 54321

* **Discussion and Conclusion**

This program takes a number as input from the user and uses a for loop to reverse the number. It iteratively extracts the last digit of the number and adds it to the reversed number by multiplying it by 10. The loop continues until the original number becomes 0. Finally, it prints the reversed number. The implementation is in the C programming language.

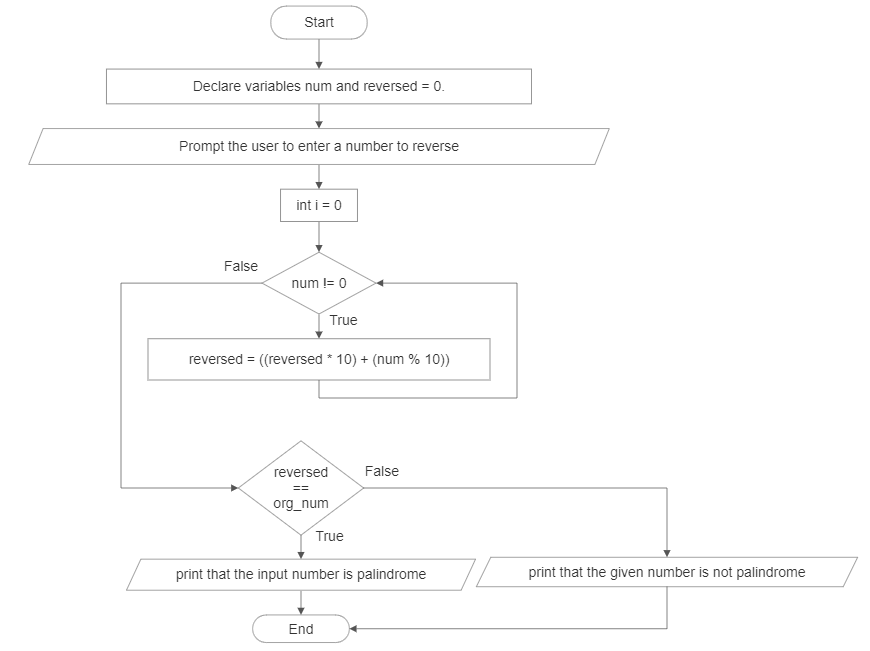
1. WAP to check whether a number is a palindrome or not.

* **Objective**

The objective of this program is to check whether a number is a palindrome or not.

* **Algorithm**

1. Start the program.
2. Declare variables num, reversed = 0, and org\_num.
3. Prompt the user to enter a number.
4. Read the value of num from the user.
5. Set org\_num equal to num.
6. Use a for loop to reverse the number.
7. Repeat the loop until num becomes 0.
8. In each iteration, calculate the reversed number by multiplying it by 10 and adding the last digit of num using the modulus operator.
9. Divide num by 10 to remove the last digit.
10. Check if org\_num is equal to reversed.
11. If they are equal, print the message "num is a palindrome number."
12. If they are not equal, print the message "num is not a palindrome number."
13. End.

* **Flowchart**
* **Code**

#include <stdio.h>

int main()

{

int num, reversed = 0, org\_num;

printf("Enter a number: ");

scanf("%d", &num);

org\_num = num;

for (int i = 0; num != 0; i++)

{

reversed = ((reversed \* 10) + (num % 10));

num /= 10;

}

if (org\_num == reversed){

printf("%d is palindrome number.", org\_num);

}

else{

printf("%d is not a palindrome number.", org\_num);

}

return 0;

}

* **Output**

Enter a number: 12321

12321 is a palindrome number.

* **Discussion and Conclusion**

This program takes a number as input from the user and checks if it is a palindrome number or not. It uses a for loop to reverse the number and then compares the reversed number with the original number. If they are equal, the number is a palindrome. Otherwise, it is not a palindrome. The implementation is in the C programming language.

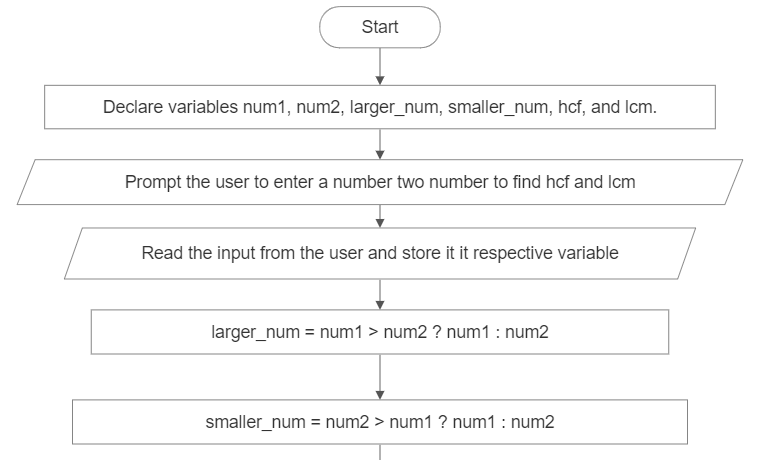
1. WAP to find HCF and LCM of two numbers.

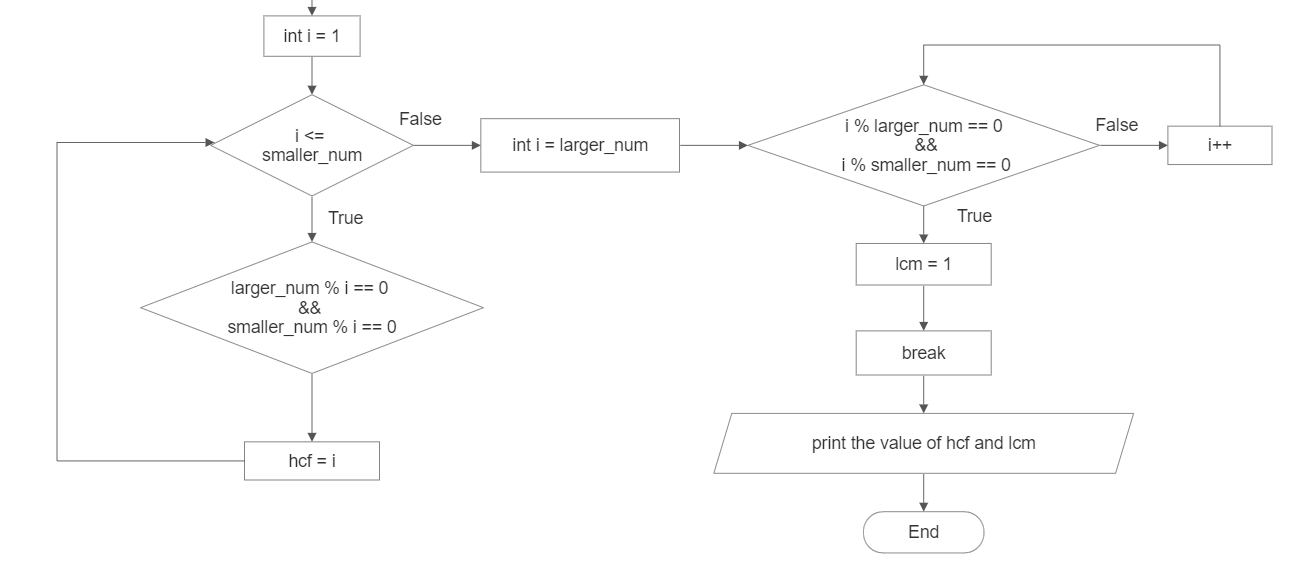
* **Objective**

The objective of this program is to find the highest common factor (HCF) and least common multiple (LCM) of two numbers.

* **Algorithm**

1. Start.
2. Declare variables num1, num2, larger\_num, smaller\_num, hcf, and lcm.
3. Prompt the user to enter two numbers.
4. Read the values of num1 and num2 from the user.
5. Determine the larger and smaller numbers by comparing num1 and num2.
6. Use a for loop to find the HCF.
7. Start the loop from 1 and iterate up to the value of the smaller number.
8. Check if both the larger and smaller numbers are divisible by the current iteration value.
9. If they are divisible, update the value of hcf to the current iteration value.
10. Use an infinite for loop to find the LCM.
11. Start the loop from the larger number and increment by the value of the larger number in each iteration.
12. Check if both the current iteration value and the smaller number are divisible by the larger number.
13. If they are divisible, update the value of lcm to the current iteration value and break the loop.
14. Print the values of hcf and lcm.
15. End.

* **Flowchart**



* **Code**

#include <stdio.h>

int main()

{

int num1, num2, larger\_num, smaller\_num, hcf, lcm;

printf("Enter two numbers: ");

scanf("%d%d", &num1, &num2);

larger\_num = num1 > num2 ? num1 : num2;

smaller\_num = num2 > num1 ? num1 : num2;

for (int i = 1; i <= smaller\_num; i++)

{

if (larger\_num % i == 0 && smaller\_num % i == 0)

{

hcf = i;

}

}

for (int i = larger\_num;; i++)

{

if (i % larger\_num == 0 && i % smaller\_num == 0)

{

lcm = i;

break;

}

}

printf("HCF of %d and %d is: %d\n", num1, num2, hcf);

printf("LCM of %d and %d is: %d\n", num1, num2, lcm);

return 0;

}

* **Output**

Enter two numbers: 12 18

HCF of 12 and 18 is: 6

LCM of 12 and 18 is: 36

* **Discussion and Conclusion**

This program takes two numbers as input from the user and calculates their highest common factor (HCF) and least common multiple (LCM). It uses loops to iterate through the numbers and find the HCF and LCM based on the given conditions. The implementation is in the C programming language.

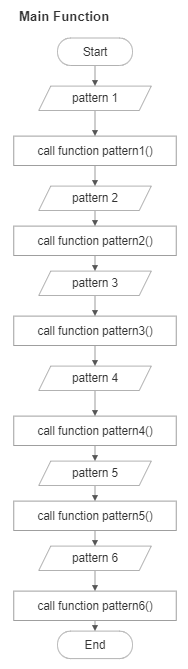
1. WAP to print the following patterns:

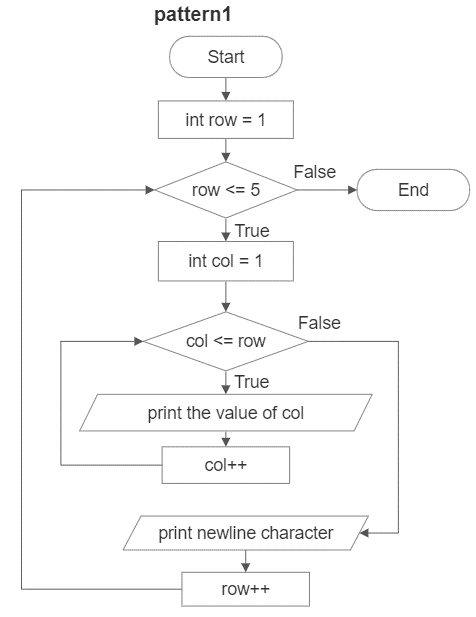
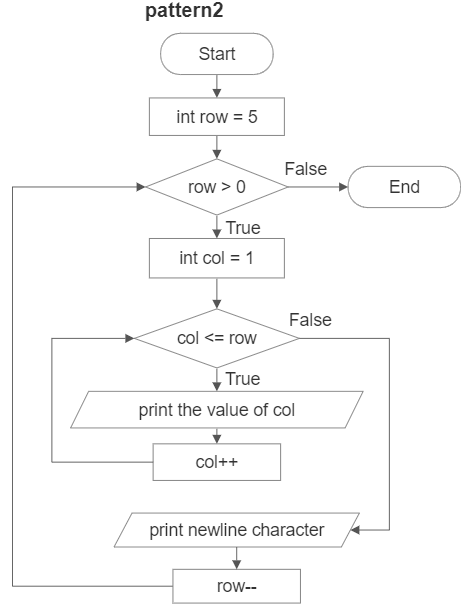
* **Objective**

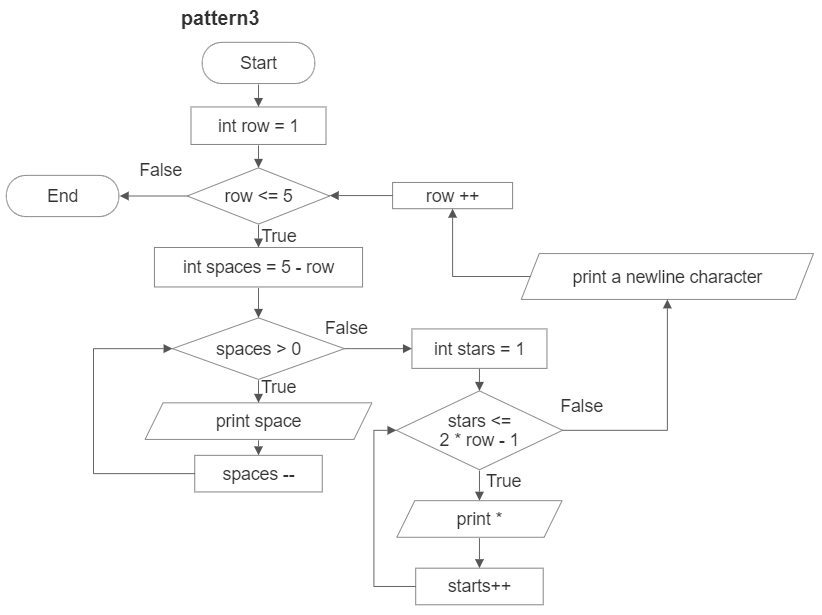
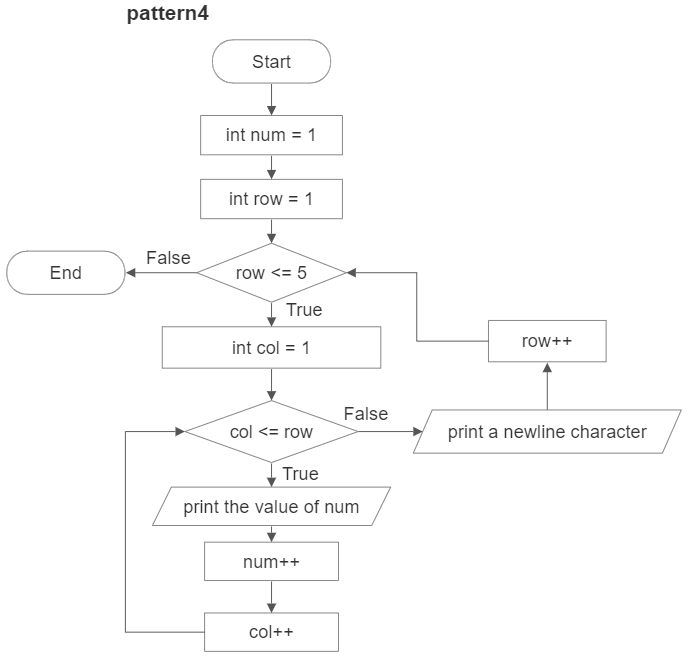
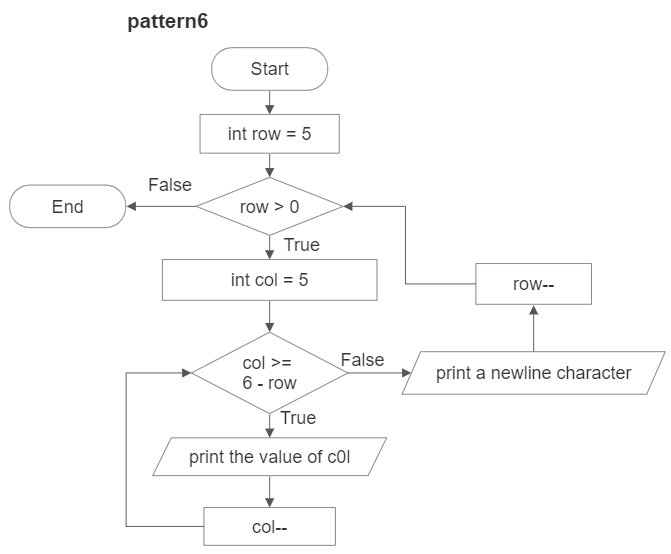
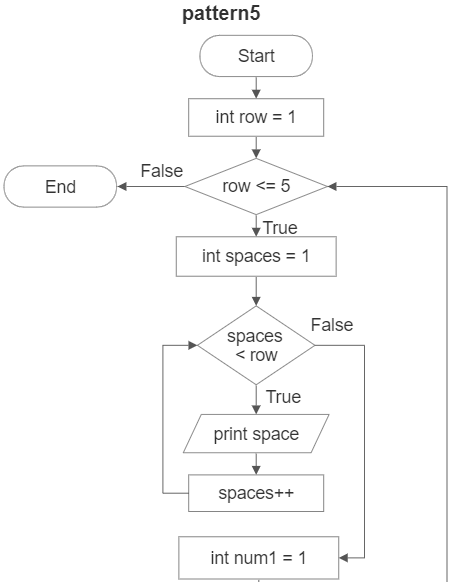
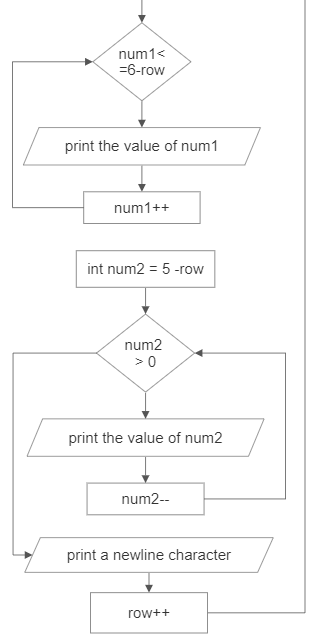
The objective of this program is to print various patterns using loops and nested loops.

* **Algorithm**

1. **main():**
2. Start.
3. Print a newline and the string "Pattern 1: ".
4. Call the pattern1 function.
5. Print a newline and the string "Pattern 2: ".
6. Call the pattern2 function.
7. Print a newline and the string "Pattern 3: ".
8. Call the pattern3 function.
9. Print a newline and the string "Pattern 4: ".
10. Call the pattern4 function.
11. Print a newline and the string "Pattern 5: ".
12. Call the pattern5 function.
13. Print a newline and the string "Pattern 6: ".
14. Call the pattern6 function.
15. End.
16. **pattern1():**
17. Start.
18. Use a loop to iterate over the rows from 1 to 5.
19. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
20. Print the column number for each iteration.
21. Move to the next line after printing the columns.
22. End.
23. **pattern2():**
24. Start.
25. Use a loop to iterate over the rows from 5 to 1 in a descending order.
26. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
27. Print the column number for each iteration.
28. Move to the next line after printing the columns.
29. End.
30. **pattern3():**
31. Start.
32. Use a loop to iterate over the rows from 1 to 5.
33. Inside the row loop, use a loop to print spaces based on the row number (5 - current row).
34. Use another loop to print stars based on the row number (2 \* current row - 1).
35. Move to the next line after printing the spaces and stars.
36. End.
37. **pattern4():**
38. Start the pattern4 function.
39. Initialize a variable num to 1.
40. Use a loop to iterate over the rows from 1 to 5.
41. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
42. Print the current value of num and increment it by 1 for each iteration.
43. Move to the next line after printing the columns.
44. End the function.
45. **patter5():**
46. Start.
47. Use a loop to iterate over the rows from 1 to 5.
48. Inside the row loop, use a loop to print spaces based on the row number.
49. Use another loop to print numbers in an increasing order from 1 to (6 - row).
50. Use a third loop to print numbers in a decreasing order from (5 - row) to 1.
51. Move to the next line after printing the spaces and numbers.
52. End.
53. **pattern6():**
54. Start.
55. Use a loop to iterate over the rows from 5 to 1 in a descending order.
56. Inside the row loop, use another loop to iterate over the columns from 5 to (6 - row).
57. Print the column number for each iteration.
58. Move to the next line after printing the columns.
59. End.

* **Flowchart**





* **Code**

#include <stdio.h>

void pattern1();

void pattern2();

void pattern3();

void pattern4();

void pattern5();

void pattern6();

int main( )

{

printf("\nPattern 1: \n");

pattern1();

printf("\nPattern 2: \n");

pattern2();

printf("\nPattern 3: \n");

pattern3();

printf("\nPattern 4: \n");

pattern4();

printf("\nPattern 5: \n");

pattern5();

printf("\nPattern 6: \n");

pattern6();

return 0;

}

void pattern1()

{

for (int row = 1; row <= 5; row++)

{

for (int col = 1; col <= row; col++)

{

printf("%d ", col);

}

printf("\n");

}

}

void pattern2()

{

for (int row = 5; row > 0; row--)

{

for (int col = 1; col <= row; col++)

{

printf("%d ", col);

}

printf("\n");

}

}

void pattern3()

{

for (int row = 1; row <= 5; row++)

{

for (int spaces = 5 - row; spaces > 0; spaces--)

{

printf(" ");

}

for (int stars = 1; stars <= 2 \* row - 1; stars++)

{

printf("\*");

}

printf("\n");

}

}

void pattern4()

{

int num = 1;

for (int row = 1; row <= 5; row++)

{

for (int col = 1; col <= row; col++)

{

printf("%d ", num);

num++;

}

printf("\n");

}

}

void pattern5()

{

for (int row = 1; row <= 5; row++)

{

for (int spaces = 1; spaces < row; spaces++)

{

printf(" ");

}

for (int num1 = 1; num1 <= 6 - row; num1++)

{

printf("%d", num1);

}

for (int num2 = 5 - row; num2 > 0; num2--)

{

printf("%d", num2);

}

printf("\n");

}

}

void pattern6()

{

for (int row = 5; row > 0; row--)

{

for (int col = 5; col >= 6 - row; col--)

{

printf("%d ", col);

}

printf("\n");

}

}

* **Output**

Pattern 1:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Pattern 2:

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

Pattern 3:

\*

\*\*\*

\*\*\*\*\*

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

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

Pattern 4:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

Pattern 5:

123454321

1234321

12321

121

1

Pattern 6:

5 4 3 2 1

5 4 3 2

5 4 3

5 4

5

* **Discussion and Conclusion**

This program demonstrates the use of loops and nested loops to print various patterns. Each pattern is implemented as a separate function, making the code modular and easy to understand. The patterns range from simple number patterns to more complex patterns involving spaces and stars. The implementation is in the C programming language.