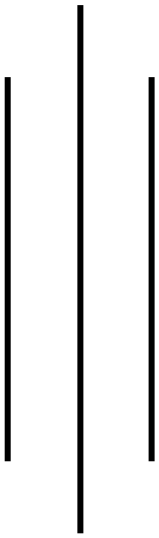
Tribhuwan University
Institute of Engineering
Purbanchal Campus, Dharan

C Lab Report



Submitted By:

Name: Tilak Thapa

Roll No: PUR079BCT094

Submitted To:

Department of Electronic and Computer Engineering

Lab Date:

Submission Date:

Signature:

Table of Contents

1. Lab Sheet 1 [To be familiar with C Programming] ……
2. Lab Sheet 2 [To be familiar Data types, Constants, Operators and Expressions] ……..
3. Lab Sheet 3 [To be familiar with selective structure (branching)] ……..…..
4. Lab Sheet 4 [To be familiar with Unformatted and Formatted I/0] ……
5. Lab Sheet 5 [To be familiar with LOOPS] ……
6. Lab Sheet 6 [To be familiar with FUNCTIONS:] ……..
7. Lab Sheet 7 [To be familiar with Array] ……...
8. Lab Sheet 8 [To be familiar with Pointers] ……
9. Lab Sheet 9 [To be familiar with Structure] ……..
10. Lab Sheet 10 [To be familiar with String] ……..
11. Lab Sheet 11 [To be familiar with File Handling] ……

Lab Sheet 1

1. WAP to display hello world.

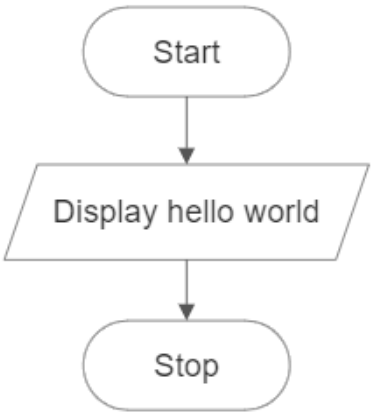
➢ Objective

The objective of this program is to write a code that displays the message "Hello, World!" on the console.

➢ Algorithm

i. Start
ii. Display "Hello, World!"
iii. Stop

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ) {
   printf("Hello, World!\n");
   return 0;
}
```

➢ Output

```
Hello, World!
```

➢ Discussion and Conclusion

This program displays the message "Hello, World!" on the console. The printf function is used to print the message. The \n is used to add a new line after the message. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

2. WAP to display your name, roll number and address

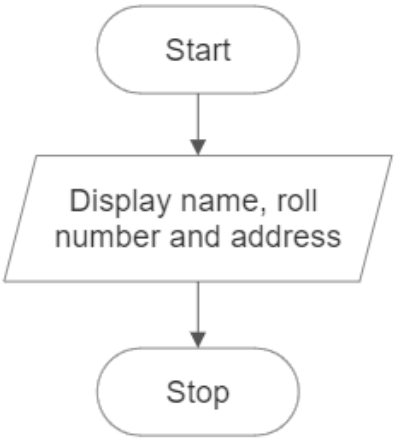
➢ Objective

The objective of this program is to write a code that displays your name, roll number, and address on the console.

➢ Algorithm

i. Start
ii. Display name, roll number and address
iii. Stop

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   printf("Name: Tilak Thapa\n");
   printf("Roll Number: PUR079BCT094\n");
   printf("Address: Tulsipur - 4, Dang\n");
   return 0;
}
```

➢ Output

```
Name: Tilak Thapa
Roll Number: PUR079BCT094
Address: Tulsipur - 4, Dang
```

➢ Discussion and Conclusion

This program displays my name, roll number, and address on the console. The printf function is used to print each piece of information. The newline character \n is used to add a new line after each line of output. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

3. WAP to add two integer variables and print sum

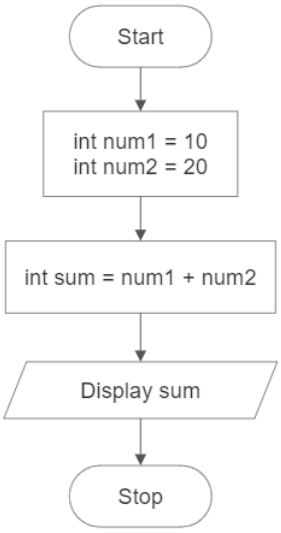
➢ Objective

The objective of this program is to write a code that adds two predefined integer variables and prints their sum.

➢ Algorithm

i. Start.
ii. Declare two integer variables, num1 and num2, and initialize them with predefined values.
iii. Calculate the sum of num1 and num2 and store it in a variable called sum.
iv. Print the value of sum.
v. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   int num1 = 10;
   int num2 = 20;
   int sum = num1 + num2;
   printf("Sum: %d\n", sum);
   return 0;
}
```

➢ Output

```
Sum: 30
```

➢ Discussion and Conclusion

This program adds two predefined integer variables, num1 and num2, and prints their sum. The values of num1 and num2 are initialized with the numbers 10 and 20, respectively. The sum of num1 and num2 is calculated and stored in the sum variable using the addition operator (+). The printf function is used to display the value of sum. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file. The objective of the program was achieved, and the code executed successfully.

4. WAP to multiply two integer variables and print product

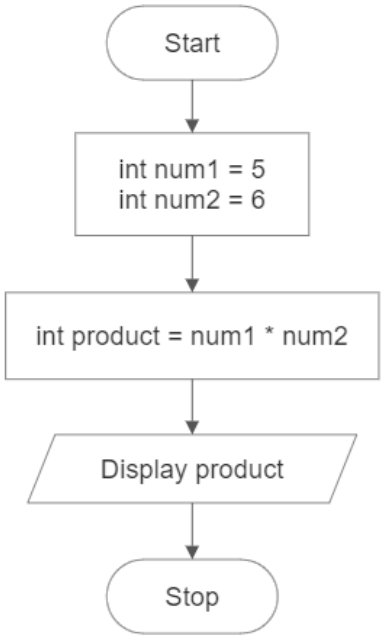
➢ Objective

The objective of this program is to write a code that multiplies two integer variables and prints their product.

➢ Algorithm

i. Start.
ii. Declare two integer variables, num1 and num2, and initialize them with predefined values.
iii. Calculate the product of num1 and num2 and store it in a variable called product.
iv. Print the value of product.
v. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main()
{
   int num1 = 5;
   int num2 = 6;
   int product = num1 * num2;
   printf("Product: %d\n", product);
   return 0;
}
```

➢ Output

```
Product: 30
```

➢ Discussion and Conclusion

This program multiplies two integer variables, num1 and num2, and prints their product. The values of num1 and num2 are assigned as 5 and 6, respectively. The product of num1 and num2 is calculated and stored in the product variable using the multiplication operator (*). The printf function is used to display the value of product. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file. The objective of the program was achieved, and the code executed successfully.

5. WAP to calculate and display the simple interest.

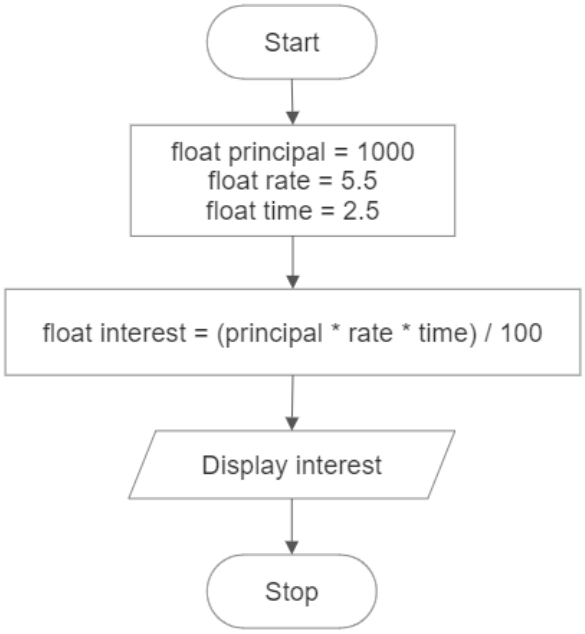
➢ Objective

The objective of this program is to write a code that calculates and displays the simple interest based on predefined values for principal amount, rate, and time.

➢ Algorithm

i. Start.
ii. Declare and initialize three variables: principal, rate, and time with predefined values.
iii. Calculate the simple interest using the formula: interest = (principal * rate * time) / 100 and assign the value to variable called interest.
iv. Print the value of the interest.
v. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main()
{
   float principal = 1000;
   float rate = 5.5;
   float time = 2.5;
   float interest = (principal * rate * time) / 100;
   printf("Simple Interest: Rs %f\n", interest);
   return 0;
}
```

➢ Output

```
Interest: Rs 137.500000
```

➢ Discussion and Conclusion

This program calculates and displays the simple interest based on predefined values for the principal amount, rate of interest, and time period. The values of principal, rate, and time are initialized as 1000, 5.5, and 2.5, respectively. The simple interest is calculated using the formula: interest = (principal * rate * time) / 100. The calculated interest value is then printed using the printf function. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

6. WAP to calculate the area of the circle

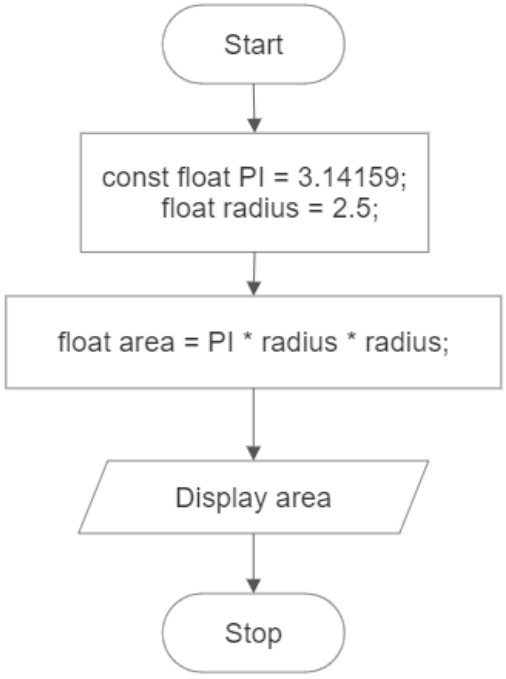
➢ Objective

The objective of this program is to write a code that calculates the area of a circle based on a predefined radius.

➢ Algorithm

i. Start.
ii. Declare a constant variable for PI and a variable for the radius and assign their values.
iii. Calculate the area of the circle using the formula: area = pi * radius * radius and assign the value to a variable called area.
iv. Print the value of the area.
v. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   const float PI = 3.14159;
   float radius = 2.5;
   float area = PI * radius * radius;
   printf("Area of the circle: %.2f sq unit.\n", area);
   return 0;
}
```

➢ Output

```
Area of the circle: 19.63 sq unit.
```

➢ Discussion and Conclusion:

This program calculates the area of a circle based on a predefined radius. The value of radius is assigned as 2.5. The area of the circle is calculated using the formula: area = PI * radius * radius, where PI is a constant value representing the mathematical constant pi (approximately 3.14159). The calculated area value is then printed using the printf function. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file

Lab Sheet 2

1. WAP to declare integer, float and character variable. Initialize them with certain value and print those values. Also display the size of variables.

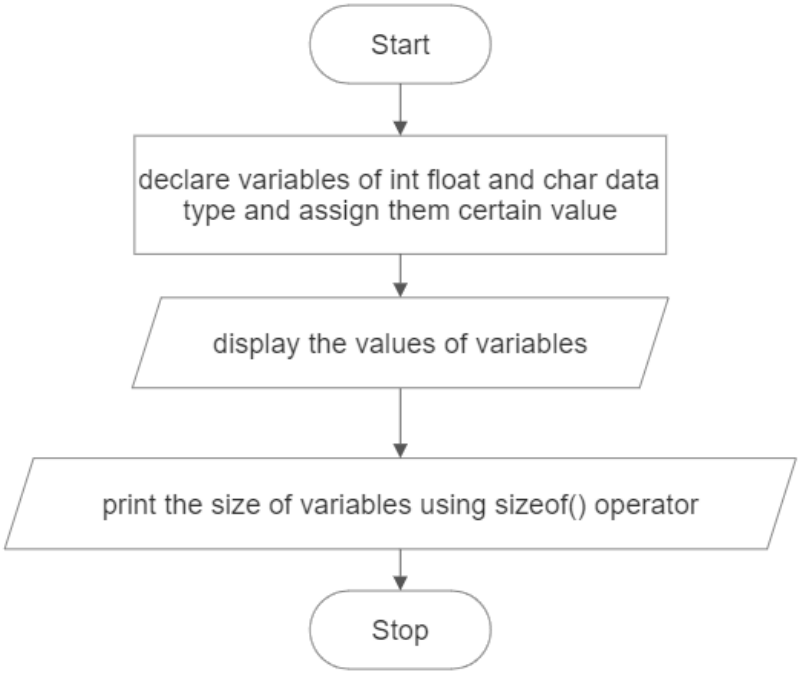
➢ **Objective**

The objective of this program is to write a code that declares integer, float, and character variables, initializes them with certain values, and prints the values. Additionally, the program will display the size of each variable.

➢ **Algorithm**

i. Start.
ii. Declare an integer variable and assign certain value.
iii. Declare a float variable and assign certain value.
iv. Declare a character variable and assign certain value.
v. Print the values of the variables using the printf function.
vi. Use the sizeof() operator to determine the size of each variable and print the sizes of the variables.
vii. Stop.

➢ **Flowchart**



➢ Code

```
#include <stdio.h>

int main( )
{
   int integerVariable = 10;
   float floatVariable = 3.14;
   char charVariable = 'A';

   printf("Integer Variable: %d\n", integerVariable);
   printf("Float Variable: %f\n", floatVariable);
   printf("Character Variable: %c\n\n", charVariable);

   printf("Size of Integer Variable: %d bytes\n", sizeof(integerVariable));
   printf("Size of Float Variable: %d bytes\n", sizeof(floatVariable));
   printf("Size of Character Variable: %d bytes\n", sizeof(charVariable));

   return 0;
}
```

➢ Output

```
Integer Variable: 10
Float Variable: 3.140000
Character Variable: A

Size of Integer Variable: 4 bytes
Size of Float Variable: 4 bytes
Size of Character Variable: 1 bytes
```

➢ Discussion and Conclusion

This program declares an integer variable, a float variable, and a character variable. The variables are initialized with certain values. The values of the variables are printed using the printf function. The sizeof operator is used to determine the size of each variable, and the sizes are printed. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

2. WAP to swap the values of the variable with and without using third variable.

➢ **Objective**

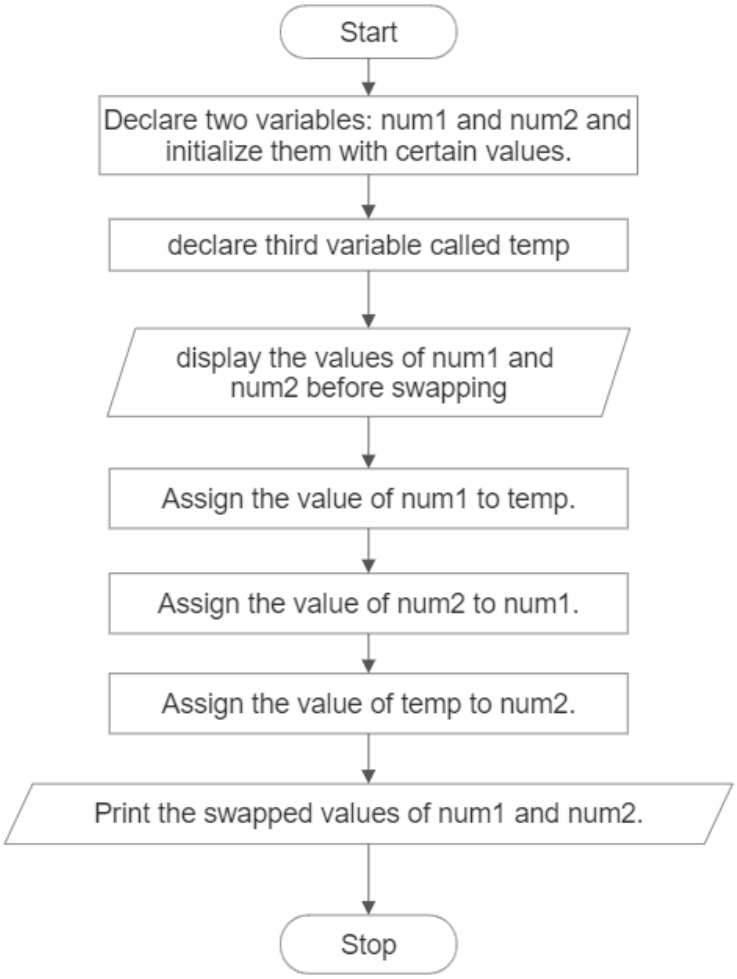
The objective of this program is to write a code that swaps the values of two variables, both with and without using a third variable.

I. Approach 1 (Using third variable)

➢ **Algorithm**

i. Start.
ii. Declare two variables: num1 and num2 and initialize them with certain values.
iii. Declare a third variable, temp.
iv. Display the value of num1 and num2 before swapping.
v. Assign the value of num1 to temp.
vi. Assign the value of num2 to num1.
vii. Assign the value of temp to num2.
viii. Print the swapped values of num1 and num2.
ix. Stop.

➢ **Flowchart**



➢ Code

```
#include <stdio.h>

int main( )
{
   int num1 = 10;
   int num2 = 20;
   int temp;

   printf("Before swapping:\n");
   printf("num1 = %d\n", num1);
   printf("num2 = %d\n", num2);

   temp = num1;
   num1 = num2;
   num2 = temp;

   printf("After swapping (using third variable):\n");
   printf("num1 = %d\n", num1);
   printf("num2 = %d\n", num2);

   return 0;
}
```

➢ Output

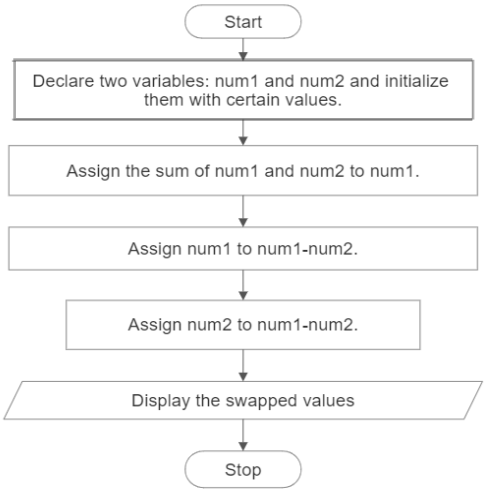
```
Before swapping:
num1 = 10
num2 = 20
After swapping (using third variable):
num1 = 20
num2 = 10
```

II. Approach 2 (Without Using Third Variable)

➢ Algorithm

i. Start.
ii. Declare two variables: num1 and num2 and initialize them with certain values.
iii. Display the value of num1 and num2 before swapping.
iv. Assign the sum of num1 and num2 to num1.
v. Assign num1 to num1-num2.
vi. Again, assign num2 to num1-num2.
vii. Print the swapped values of num1 and num2.
viii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main()
{
   int num1 = 10;
   int num2 = 20;

   printf("Before swapping:\n");
   printf("num1 = %d\n", num1);
   printf("num2 = %d\n", num2);

   num1 = num1 + num2;
   num2 = num1 - num2;
   num1 = num1 - num2;

   printf("After swapping (without using third variable):\n");
   printf("num1 = %d\n", num1);
   printf("num2 = %d\n", num2);

   return 0;
}
```

➢ Output

```
Before swapping:
num1 = 10
num2 = 20
After swapping (without using third variable):
num1 = 20
num2 = 10
```

➢ Discussion and Conclusion

In the first part of the program, the values of num1 and num2 are swapped using a third variable. The values are stored in a temporary variable, temp, before swapping. Then, the values are exchanged by assigning num2 to num1 and temp to num2.

In the second part of the program, the values of num1 and num2 are swapped without using a third variable. This is achieved using the simple addition and subtraction operation. By performing that operations on the two variables, the original values are swapped without the need for an additional variable.

Both cases print the values of num1 and num2 before and after swapping to demonstrate the results. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

3. WAP to calculate the area and volume of a cylinder using pre-processor directive for value of PI.

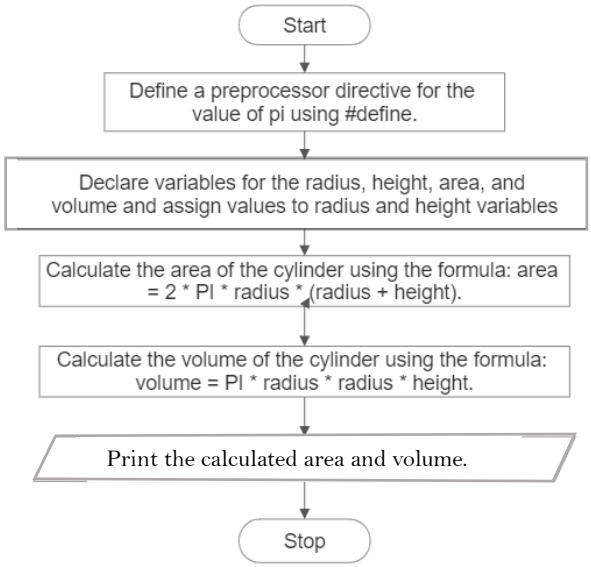
➢ Objective

The objective of this program is to write a code that calculates the area and volume of a cylinder using a preprocessor directive for the value of pi.

➢ Algorithm

i. Start.
ii. Define a preprocessor directive for the value of pi using #define.
iii. Declare variables for the radius, height, area, and volume.
iv. Assign predetermined values to the radius and height variables.
v. Calculate the area of the cylinder using the formula: area = 2 * PI * radius * (radius + height).
vi. Calculate the volume of the cylinder using the formula: volume = PI * radius * radius * height.
vii. Print the calculated area and volume.
viii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

#define PI 3.14159

int main()
{
   float radius = 2.5;
   float height = 5.0;
   float area, volume;

   area = 2 * PI * radius * (radius + height);
   volume = PI * radius * radius * height;

   printf("Area of the cylinder: %.2f\n", area);
   printf("Volume of the cylinder: %.2f\n", volume);

   return 0;
}
```

➢ Output

```
Area of the cylinder: 117.81
Volume of the cylinder: 98.17
```

➢ Discussion and Conclusion

This program calculates the area and volume of a cylinder using a preprocessor directive for the value of pi. The values of the radius and height are predetermined and assigned to the respective variables. The area of the cylinder is calculated using the formula: area = 2 * PI * radius * (radius + height), and the volume is calculated using the formula: volume = PI * radius * radius * height. The calculated values are then printed using the printf function. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

4. WAP to input two numbers from user and display the minimum using conditional operator.

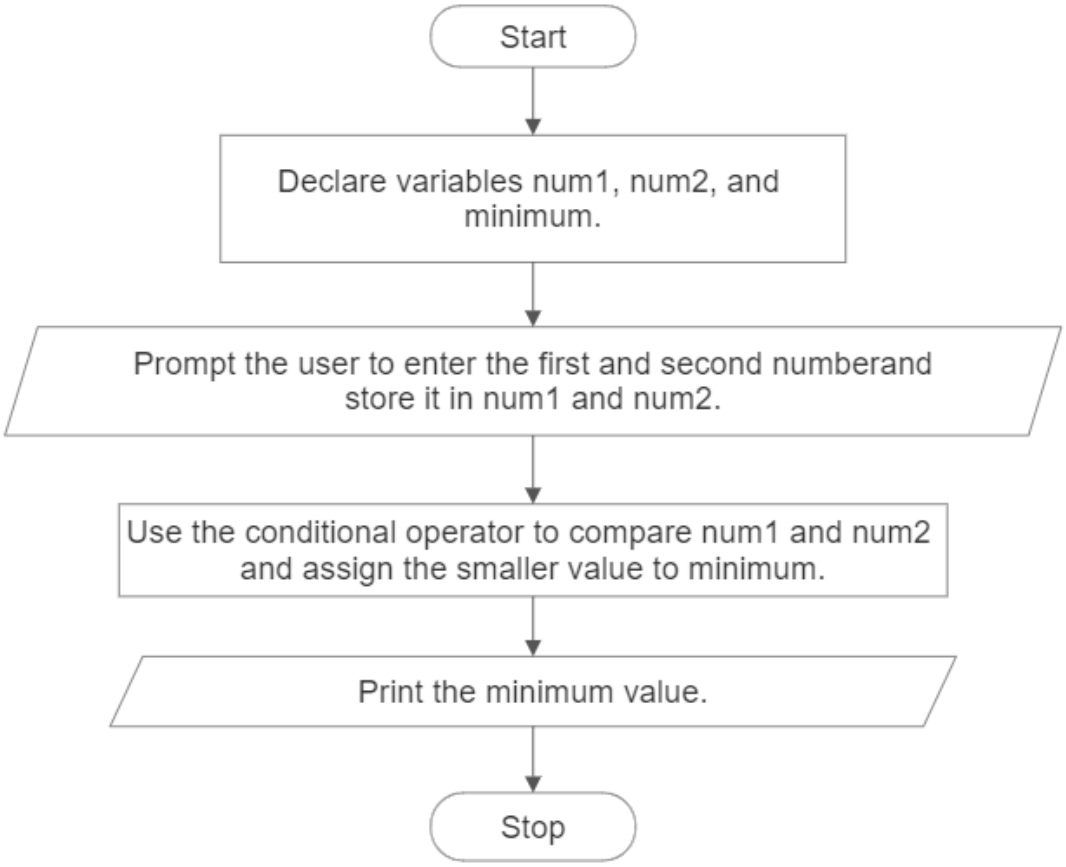
➢ **Objective**

The objective of this program is to write a code that takes two numbers as input from the user and displays the minimum of the two numbers using the conditional operator.

➢ **Algorithm**

i. Start.
ii. Declare variables num1, num2, and minimum.
iii. Prompt the user to enter the first number and store it in num1.
iv. Prompt the user to enter the second number and store it in num2.
v. Use the conditional operator to compare num1 and num2 and assign the smaller value to minimum.
vi. Print the value of minimum.
vii. Stop.

➢ **Flowchart**



➢ Code

```
#include <stdio.h>

int main( )
{
   int num1, num2, minimum;

   printf("Enter the first number: ");
   scanf("%d", &num1);

   printf("Enter the second number: ");
   scanf("%d", &num2);

   minimum = (num1 < num2) ? num1 : num2;

   printf("The minimum number is: %d\n", minimum);

   return 0;
}
```

➢ Output

```
Enter the first number: 4
Enter the second number: 5
The minimum number is: 4
```

➢ Discussion and Conclusion

This program takes two numbers as input from the user and determines the minimum of the two numbers using the conditional operator. The user is prompted to enter the first number and the second number, which are stored in num1 and num2 variables, respectively. The conditional operator (num1 < num2) ? num1 : num2 compares the two numbers and assigns the smaller value to the minimum variable. Finally, the minimum value is printed using the printf function. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

5. WAP to display whether a number is even or odd using conditional operator

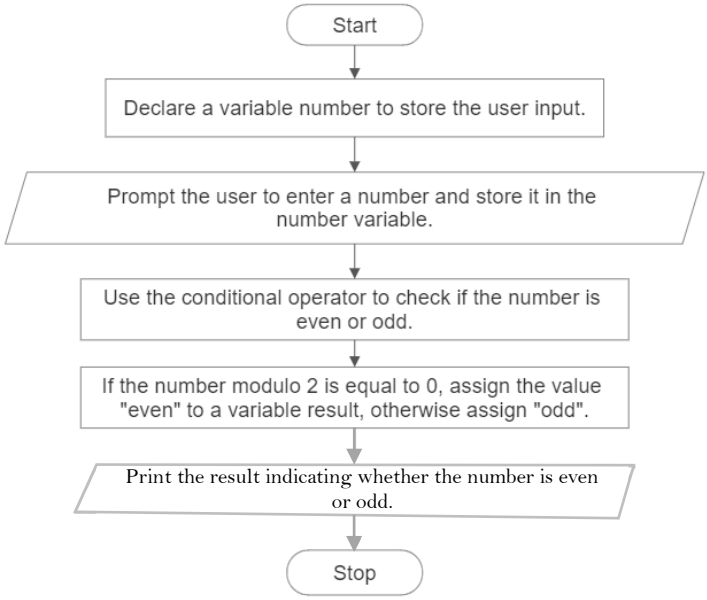
➢ Objective

The objective of this program is to write a code that takes a number as input from the user and displays whether the number is even or odd using the conditional operator.

➢ Algorithm

i. Start.
ii. Declare a variable number to store the user input.
iii. Prompt the user to enter a number and store it in the number variable.
iv. Use the conditional operator to check if the number is even or odd.
v. If the number % 2 is equal to 0, assign the value "even" to a variable result, otherwise assign "odd".
vi. Print the result indicating whether the number is even or odd.
vii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   int number;
   char *result;
   printf("Enter a number: ");
   scanf("%d", &number);
   result = (number % 2 == 0) ? "even" : "odd";
   printf("The number is %s.\n", result);
   return 0;
}
```

➢ Output

```
Enter a number: 6
The number is even.
```

➢ Discussion and Conclusion

This program takes a number as input from the user and determines whether the number is even or odd using the conditional operator. The user is prompted to enter a number, which is stored in the number variable. The conditional operator (number % 2 == 0) ? "even" : "odd" checks if the number modulo 2 is equal to 0. If it is, the value "even" is assigned to the result variable; otherwise, the value "odd" is assigned. Finally, the program prints the result indicating whether the number is even or odd. The program was implemented using the VS Code IDE and compiled using gcc to generate an executable file.

6. What are the output of the following programs:

➢ Objective

The objective is to find the input of given code.

➢ Code

```
#include <stdio.h>
int main()
{
   int a = 5, b = 9;
   printf("a = %d, b = %d\n", a, b);
   printf("a&b = %d\n", a & b);
   printf("a|b = %d\n", a | b);
   printf("a^b = %d\n", a ^ b);
   printf("~a = %d\n", ~a);
   printf("(b<<2)+(a<<1) = %d\n", (b << 2) + (a << 1));
   printf("(b>>1)+(a>>1) = %d\n", (b >> 1) + (a >> 1));
   return 0;
}
```

➢ Output

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = -6
(b<<2)+(a<<1) = 46
(b>>1)+(a>>1) = 6
```

➢ Discussion and Conclusion

The program displays the output showing the effects of the bitwise operations on the given variables.

- a & b performs a bitwise AND operation between a and b. In binary, 5 is 0101 and 9 is 1001. The result of a & b is 0001, which is equal to 1 in decimal.
- a | b performs a bitwise OR operation between a and b. In binary, the result is 1101, which is equal to 13 in decimal.
- a ^ b performs a bitwise XOR operation between a and b. In binary, the result is 1100, which is equal to 12 in decimal.
- ~a performs a bitwise complement operation on a. In binary, the result is 11111111111111111111111111111010, which is equal to -6 in decimal due to two's complement representation.
- (b<<2)+(a<<1) performs a left shift by 2 bits on b and a left shift by 1 bit on a, then adds the results. The value of b after the left shift is 36, and the value of a after the left shift is 10. The final result is 46.
- (b>>1)+(a>>1) performs a right shift by 1 bit on both b and a, then adds the results. The value of b after the right shift is 4, and the value of a after the right shift is 2. The final result is 7.

1. Write a C program to check whether a number is negative, positive, or zero.

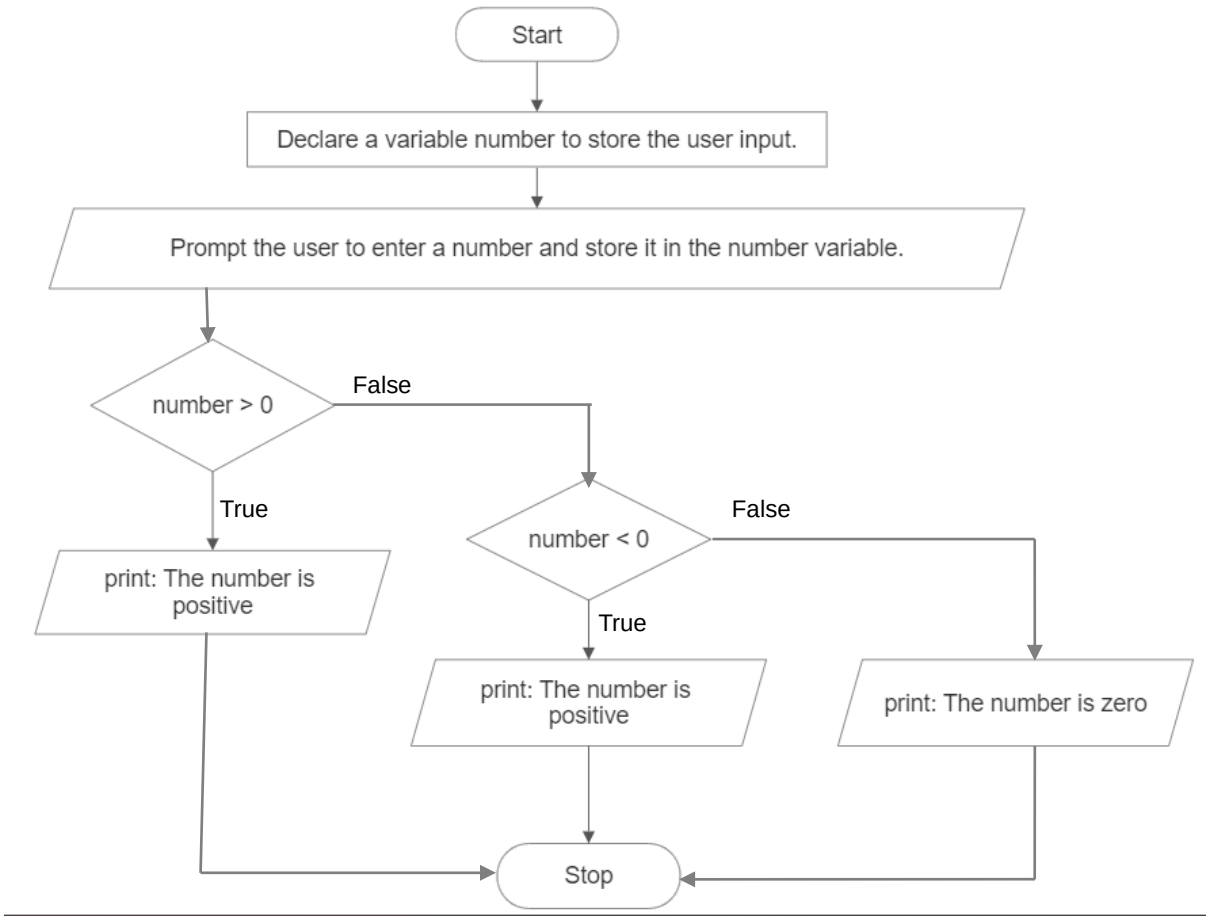
➢ Objective

The objective of this program is to write a code that takes a number as input from the user and determines whether the number is negative, positive, or zero. The program will use conditional statements to check the value of the number and display the appropriate message indicating its classification.

➢ Algorithm

i. Start.
ii. Declare a variable number to store the user input.
iii. Prompt the user to enter a number and store it in the number variable.
iv. Use conditional statements to check the value of the number:
v. If number is greater than 0, print "The number is positive."
vi. If number is less than 0, print "The number is negative."
vii. If number is equal to 0, print "The number is zero."
viii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   int number;

   printf("Enter a number: ");
   scanf("%d", &number);

   if (number > 0)
      printf("The number is positive.\n");
   else if (number < 0)
      printf("The number is negative.\n");
   else
      printf("The number is zero.\n");

   return 0;
}
```

➢ Output

```
Enter a number: 5
The number is positive.
```

➢ Discussion and Conclusion

This program takes a number as input from the user and determines whether the number is negative, positive, or zero using conditional statements. The user is prompted to enter a number, which is stored in the number variable. The program checks the value of the number using conditional statements (if, else if, and else) and prints the appropriate message indicating whether the number is positive, negative, or zero. The program was implemented using the VS Code IDE and compiled using GCC to generate an executable file. By using conditional statements, the program accurately classifies the input number based on its value.

2. WAP to find maximum between three numbers entered by the user.

- **Objective**

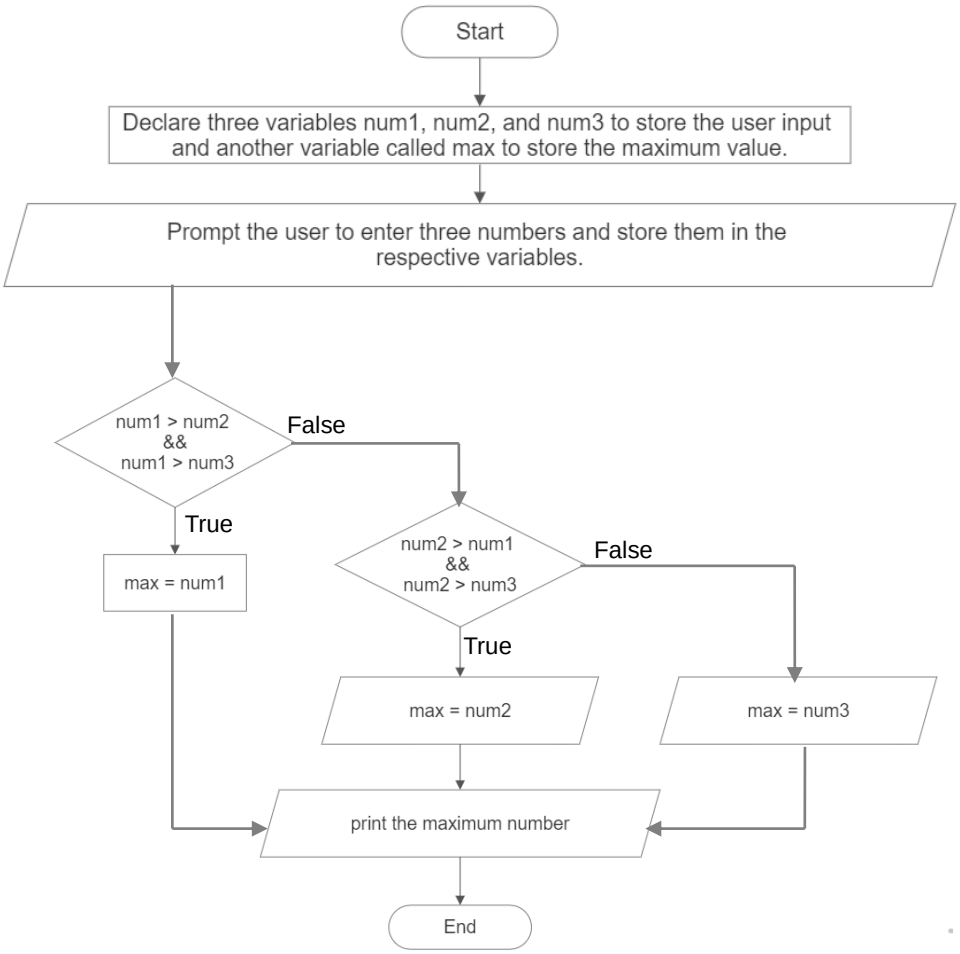
  The objective of this program is to write a code that takes three numbers as input from the user and determines the maximum among them. The program will use conditional statements to compare the values of the three numbers and display the maximum value.

- **Algorithm**

  i. Start.
  ii. Declare three variables num1, num2, and num3 to store the user input and another variable called max to store maximum number.
  iii. Prompt the user to enter three numbers and store them in the respective variables.
  iv. Use conditional statements to compare the values of the three numbers:
  - If num1 is greater than both num2 and num3, it is the maximum.
  - If num2 is greater than both num1 and num3, it is the maximum.
  - If num3 is greater than both num1 and num2, it is the maximum.

  v. Print the maximum value.
  vi. End the program.

- **Flowchart**



➢ Code

```
#include <stdio.h>

int main()
{
   int num1, num2, num3, max;

   printf("Enter three numbers: ");
   scanf("%d %d %d", &num1, &num2, &num3);

   if (num1 > num2 && num1 > num3)
      max = num1;
   else if (num2 > num1 && num2 > num3)
      max = num2;
   else
      max = num3;

   printf("The maximum number is: %d\n", max);

   return 0;
}
```

➢ Output

```
Enter three numbers: 1 2 2
The maximum number is: 2
```

➢ Discussion and Conclusion

This program takes three numbers as input from the user and determines the maximum among them using conditional statements. The user is prompted to enter three numbers, which are stored in the variables num1, num2, and num3. The program compares the values of these numbers using conditional statements (if and else if) and assigns the maximum value to the variable max. Finally, the program prints the maximum number. The program was implemented using the VS Code IDE and compiled using GCC to generate an executable file. By comparing the values of the three numbers, the program accurately determines the maximum value.

3. WAP to input a character from the user and check whether the character is vowel or consonant.

➢ Objective

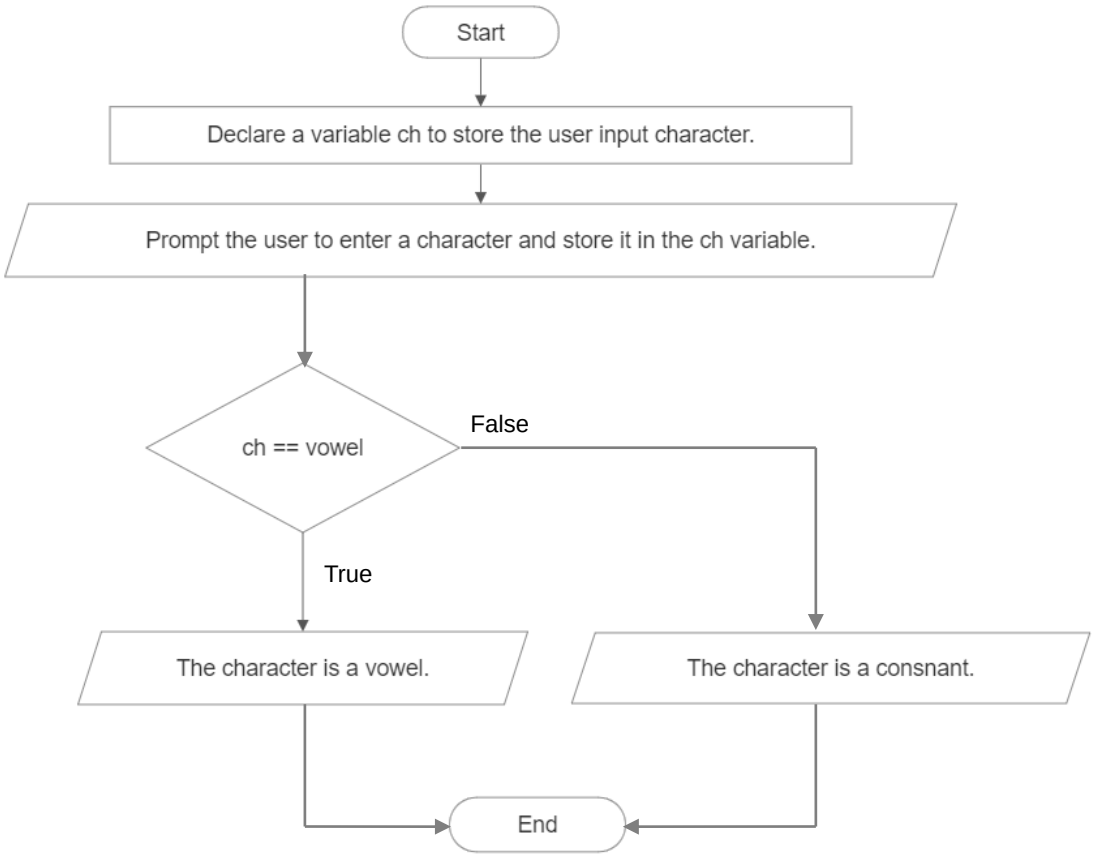
The objective of this program is to write a code that takes a character as input from the user and determines whether the character is a vowel or a consonant.

➢ Algorithm

i. Start.
ii. Declare a variable ch to store the user input character.
iii. Prompt the user to enter a character and store it in the ch variable.
iv. Use conditional statements to check if the character is a vowel or a consonant:
- If ch is equal to 'a', 'e', 'i', 'o', or 'u', it is a vowel.

v. If the character is a vowel, print a message indicating that it is a vowel.
vi. If the character is not a vowel, print a message indicating that it is a consonant.
vii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( )
{
   char ch;

   printf("Enter a character: ");
   scanf(" %c", &ch);

   if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
      ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U')
   {
      printf("The character is a vowel.\n");
   }
   else
   {
      printf("The character is a consonant.\n");
   }

   return 0;
}
```

➢ Output

```
Enter a character: r
The character is a consonant.
```

➢ Discussion

This program takes a character as input from the user and checks whether the character is a vowel or a consonant using conditional statements. The user is prompted to enter a character, which is stored in the variable ch. The program compares the value of ch with the vowels ('a', 'e', 'i', 'o', 'u') in both lowercase and uppercase forms to determine whether it is a vowel. If it is a vowel, the program prints a message indicating that it is a vowel. If it is not a vowel, the program considers it a consonant and prints a corresponding message. The program was implemented using the VS Code IDE and compiled using GCC to generate an executable file. By comparing the value of the character with the predefined vowels, the program accurately determines whether it is a vowel or a consonant.

4. WAP to input a character from the user and check whether the character is Alphabet or not. If the character is alphabet then show whether it is uppercase or lowercase

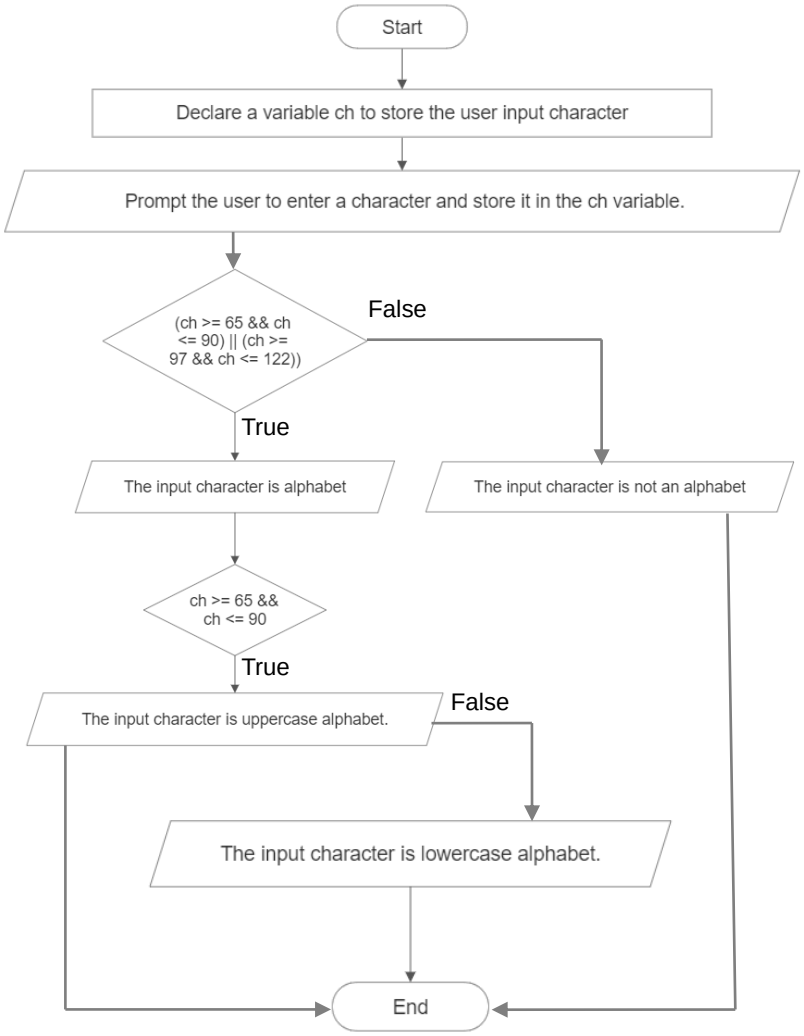
➢ Objective

The objective of this program is to write a code that takes a character as input from the user and determines whether the character is an alphabet or not. If the character is an alphabet, the program will further identify whether it is in uppercase or lowercase.

➢ Algorithm

i. . Start.
ii. Declare a variable ch to store the user input character.
iii. Prompt the user to enter a character and store it in the ch variable.
iv. Use conditional statements to check if the character is an alphabet:
v. If the ASCII value of ch is within the range of uppercase letters (65 to 90) or lowercase letters (97 to 122), it is an alphabet.
vi. If the character is an alphabet, further check whether it is in uppercase or lowercase:
vii. If the ASCII value of ch is within the range of uppercase letters (65 to 90), it is an uppercase letter and if the ASCII value of ch is within the range of lowercase letters (97 to 122), it is a lowercase letter.
viii. Print the appropriate message indicating the result.
ix. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main()
{
   char ch;

   printf("Enter a character: ");
   scanf(" %c", &ch);

   if ((ch >= 65 && ch <= 90) || (ch >= 97 && ch <= 122))
   {
      printf("The character is an alphabet.\n");

      if (ch >= 65 && ch <= 90)
         printf("It is in uppercase.\n");
      else
         printf("It is in lowercase.\n");
   }
   else
   {
      printf("The character is not an alphabet.\n");
   }

   return 0;
}
```

➢ Output

```
Enter a character: e
The character is an alphabet.
It is in lowercase.
```

➢ Discussion and Conclusion

This program takes a character as input from the user and checks whether the character is an alphabet or not using conditional statements. The user is prompted to enter a character, which is stored in the variable ch. The program compares the ASCII value of the character to determine if it falls within the range of uppercase or lowercase letters. If it is an alphabet, the program prints the appropriate messages indicating that it is an alphabet and whether it is in uppercase or lowercase. If it is not an alphabet, the program notifies the user accordingly. The program was implemented using the VS Code IDE and compiled using GCC to generate an executable file. By checking the ASCII value of the character, the program accurately determines whether it is an alphabet and its case.

5. WAP to check whether the year entered by the user is leap year or not

➢ Objective

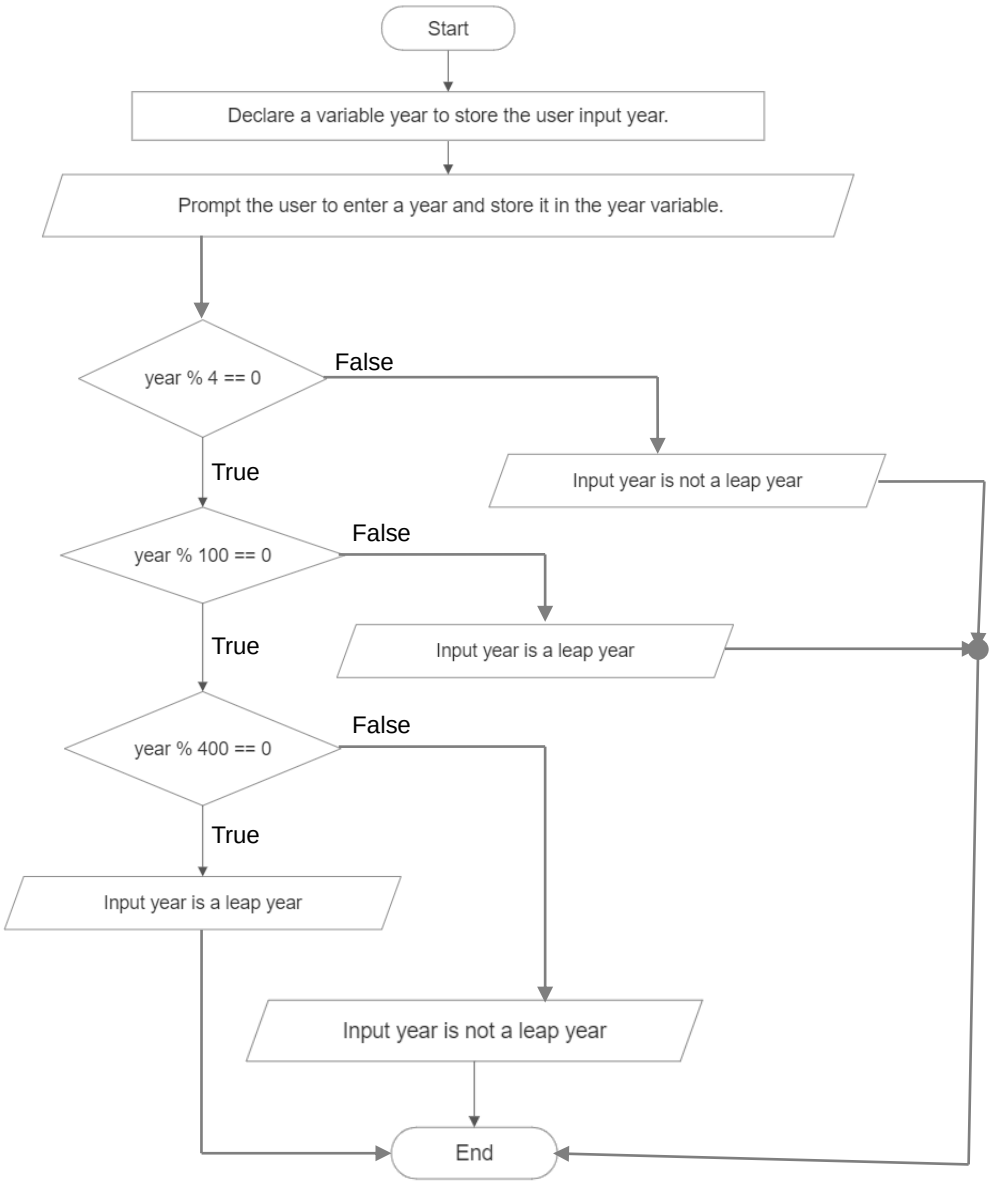
The objective of this program is to write a code that takes a year as input from the user and determines whether the year is a leap year or not.

➢ Algorithm

i. . Start.
ii. Declare a variable year to store the user input year.
iii. Prompt the user to enter a year and store it in the year variable.
iv. Use conditional statements to check if the year is a leap year:
- If the year is evenly divisible by 4, it is a potential leap year.
- If the year is also divisible by 100, it must be divisible by 400 to be considered a leap year.

v. If the year satisfies the leap year conditions, print a message indicating that it is a leap year.
vi. If the year does not satisfy the leap year conditions, print a message indicating that it is not a leap year.
vii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ) {
   int year;

   printf("Enter a year: ");
   scanf("%d", &year);

   if (year % 4 == 0) {
      if (year % 100 == 0){
         if (year % 400 == 0){
            printf("%d is a leap year.\n", year);
         } else {
            printf("%d is not a leap year.\n", year);
         }
      } else {
         printf("%d is a leap year.\n", year);
      }
   } else {
      printf("%d is not a leap year.\n", year);
   }

   return 0;
}
```

➢ Output

```
Enter a year: 2022
2022 is not a leap year.
```

➢ Discussion and Conclusion

The program checks whether a year entered by the user is a leap year or not. It uses nested if statements to evaluate the leap year conditions. First, it checks if the year is divisible by 4. If it is, it further checks if the year is divisible by 100. If it is divisible by 100, it checks if it is also divisible by 400. If it satisfies all these conditions, it is considered a leap year. Otherwise, it is not a leap year. The program then displays the result accordingly. The program successfully determines whether a given year is a leap year or not using nested if statements. It prompts the user for a year and applies the leap year conditions to determine the result. The program was implemented using the VS Code IDE and compiled with GCC. By using the nested if statements, the program accurately identifies leap years based on the defined rules.

6. WAP to check whether the number entered by the user is divisible by 5 and 11 or not

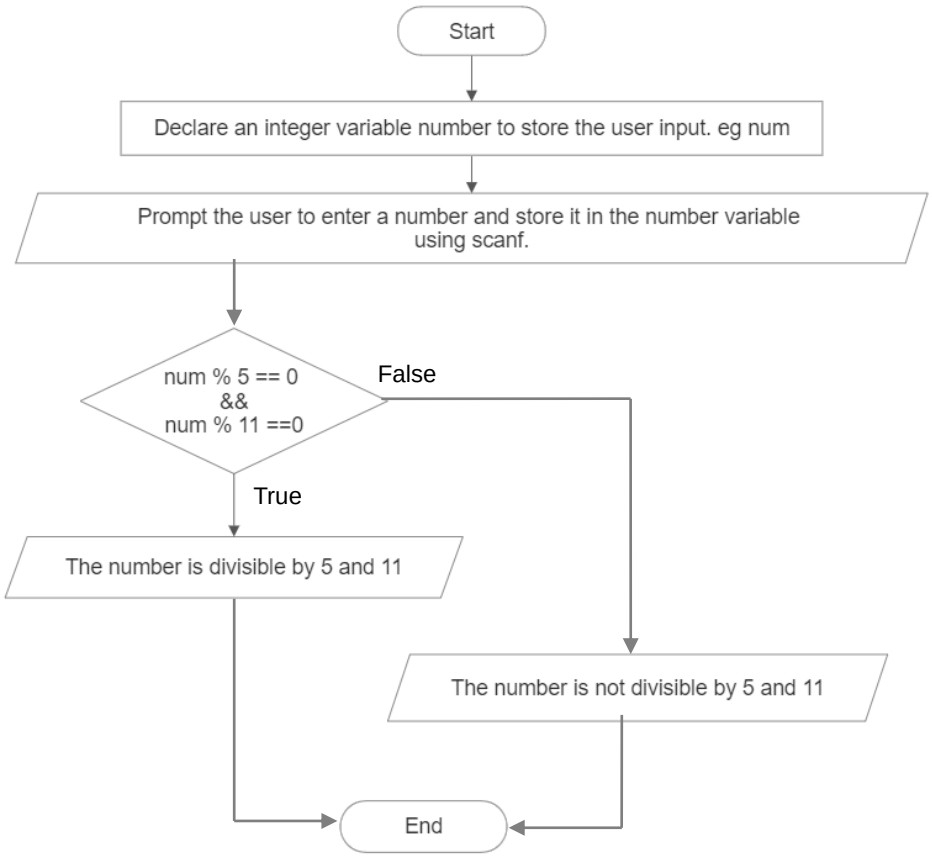
➢ **Objective**

The objective of this program is to write a code that takes a number as input from the user and checks whether the number is divisible by both 5 and 11.

➢ **Algorithm**

i. Start.
ii. Declare an integer variable number to store the user input.
iii. Prompt the user to enter a number and store it in the number variable using scanf.
iv. Use the modulo operator % to check if the number is divisible by 5 and 11.
v. If the remainder of the number divided by both 5 and 11 is 0, display a message indicating that the number is divisible by both 5 and 11.
vi. If the condition in step v is not met, display a message indicating that the number is not divisible by both 5 and 11.
vii. Stop.

➢ **Flowchart**



➢ Code

```
#include <stdio.h>

int main( ){
   int number;

   printf("Enter a number: ");
   scanf("%d", &number);

   if (number % 5 == 0 && number % 11 == 0) {
      printf("%d is divisible by 5 and 11.\n", number);
   } else {
      printf("%d is not divisible by 5 and 11.\n", number);
   }
   return 0;
}
```

➢ Output

```
Enter a number: 55
55 is divisible by 5 and 11.
```

➢ Discussion and Conclusion

This program takes a number as input from the user and checks whether the number is divisible by both 5 and 11 using the modulo operator. If the remainder of the number divided by 5 and 11 is 0, it indicates that the number is divisible by both. Otherwise, it is not divisible by both. The program displays the appropriate message based on the divisibility result. The program provides the expected output.

7. WAP to find the all the roots of a quadratic equation.

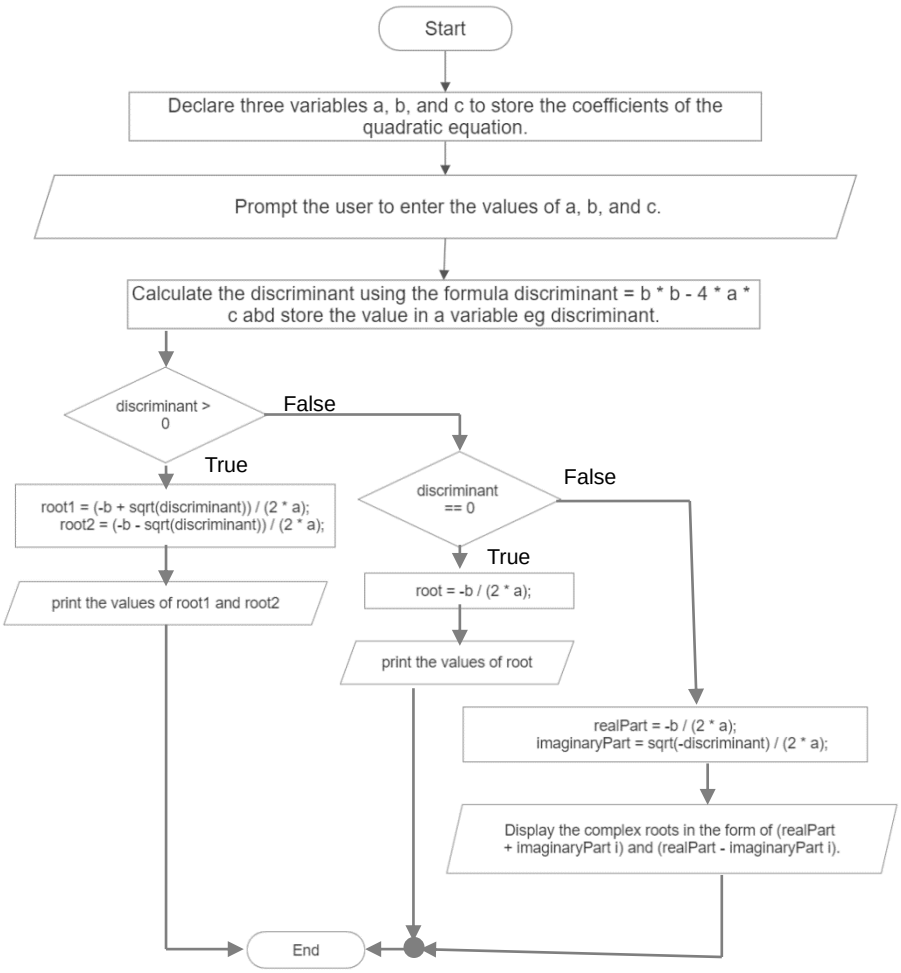
➢ Objective

The objective of this program is to write a code that calculates and displays all the roots of a quadratic equation.

➢ Algorithm

i. Start.
ii. Declare three variables a, b, and c to store the coefficients of the quadratic equation.
iii. Prompt the user to enter the values of a, b, and c.
iv. Calculate the discriminant using the formula discriminant = b * b - 4 * a * c.
v. Check the value of the discriminant:
If the discriminant is greater than 0, calculate the two roots using the formulas:
root1 = (-b + sqrt(discriminant)) / (2 * a)
root2 = (-b - sqrt(discriminant)) / (2 * a)
vi. Display the roots.
vii. If the discriminant is equal to 0, calculate the single root using the formula:
- root = -b / (2 * a)
viii. Display the root.
ix. If the discriminant is less than 0, calculate the complex roots using the formulas:
realPart = -b / (2 * a)
imaginaryPart = sqrt(-discriminant) / (2 * a)
x. Display the complex roots in the form of (realPart + imaginaryPart i) and (realPart - imaginaryPart.
xi. Stop.

➢ Flowchart



➢ Code

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

int main( ) {
   double a, b, c;
   double discriminant, root1, root2, realPart, imaginaryPart;
   printf("Enter the coefficients of the quadratic equation (a, b, c): ");
   scanf("%lf %lf %lf", &a, &b, &c);
   discriminant = b * b - 4 * a * c;
   if (discriminant > 0) {
      root1 = (-b + sqrt(discriminant)) / (2 * a);
      root2 = (-b - sqrt(discriminant)) / (2 * a);
      printf("Roots are real and different:\n");
      printf("Root 1 = %.2lf\n", root1);
      printf("Root 2 = %.2lf\n", root2);
   } else if (discriminant == 0)  {
      root1 = -b / (2 * a);
      printf("Roots are real and same:\n");
      printf("Root = %.2lf\n", root1);
   }  else  {
      realPart = -b / (2 * a);
      imaginaryPart = sqrt(-discriminant) / (2 * a);
      printf("Roots are complex and different:\n");
      printf("Root 1 = %.2lf + %.2lfi\n", realPart, imaginaryPart);
      printf("Root 2 = %.2lf - %.2lfi\n", realPart, imaginaryPart);
   }
   return 0;
}
```

➢ Output

```
Enter the coefficients of the quadratic equation (a, b, c): 1 -3 10
Roots are complex and different:
Root 1 = 1.50 + 2.96i
Root 2 = 1.50 - 2.96i
```

➢ Discussion and Conclusion

This program calculates and displays all the roots of a quadratic equation, including real and complex roots. It calculates the discriminant and checks its value to determine the nature of the roots. If the discriminant is positive, it calculates and displays two distinct real roots. If the discriminant is zero, it calculates and displays a single real root. If the discriminant is negative, it calculates and displays two complex roots in the form of (realPart + imaginaryPart i) and (realPart - imaginaryPart i). The program provides the expected output for different input cases, including non-real roots.

8. WAP to input two numbers and operator among [+ , - ,* , /]. If user enters + then the program should perform the addition of the number and display the sum. If user enters – then the program should perform subtraction of number and display the difference and so on for * and /.

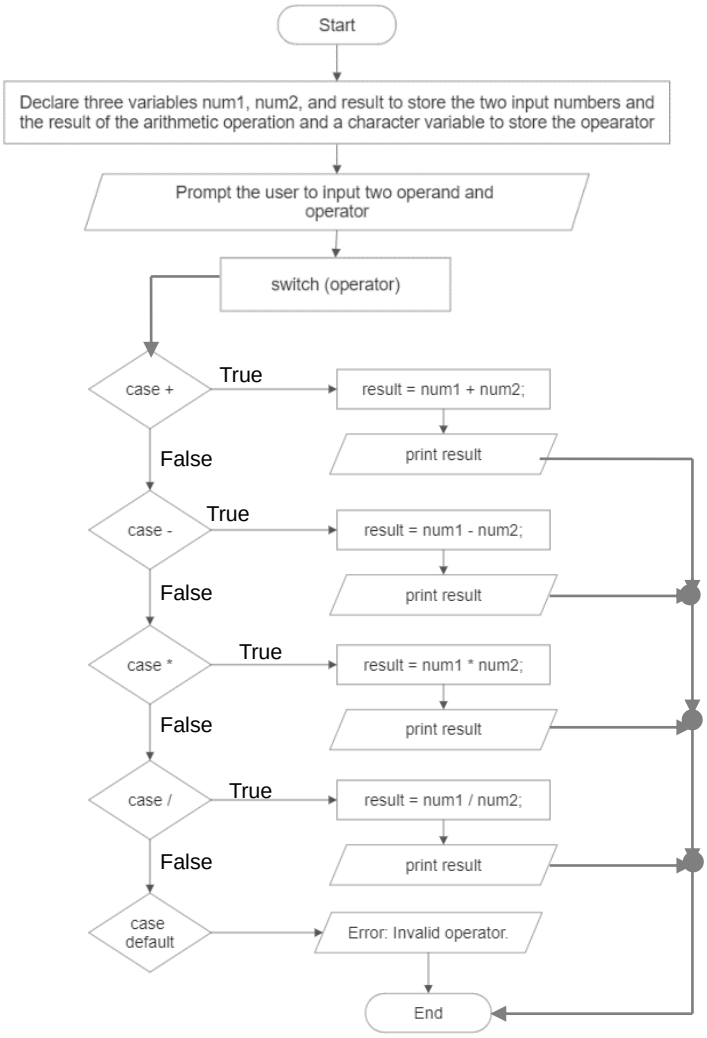
➢ Objective

The objective of this program is to write a code that takes two numbers and an operator (+, -, *, /) as input from the user and performs the corresponding arithmetic operation, displaying the result.

➢ Algorithm

i. Start.
ii. Declare three variables num1, num2, and result to store the two input numbers and the result of the arithmetic operation, respectively.
iii. Declare a variable operator to store the input operator.
iv. Prompt the user to enter the values of num1, num2, and operator.
v. Use a switch statement to perform the arithmetic operation based on the input operator:
vi. If the operator is '+', add num1 and num2 and store the result in the result variable.
vii. If the operator is '-', subtract num2 from num1 and store the result in the result variable.
viii. If the operator is '*', multiply num1 and num2 and store the result in the result variable.
ix. If the operator is '/', divide num1 by num2 and store the result in the result variable.
x. If none of the above cases match, display an error message.
xi. Display the result of the arithmetic operation.
xii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ){
   double num1, num2, result;
   char operator;
   printf("Enter the first number: ");
   scanf("%lf", &num1);
   printf("Enter the second number: ");
   scanf("%lf", &num2);
   printf("Enter the operator (+, -, *, /): ");
   scanf(" %c", &operator);
   switch (operator) {
   case '+':
      result = num1 + num2;
      printf("Result: %.2lf\n", result);
      break;
   case '-':
      result = num1 - num2;
      printf("Result: %.2lf\n", result);
      break;
   case '*':
      result = num1 * num2;
      printf("Result: %.2lf\n", result);
      break;
   case '/':
      if (num2 != 0){
         result = num1 / num2;
         printf("Result: %.2lf\n", result);
      } else{
         printf("Error: Division by zero is not allowed.\n");
      }
      break;
   default:
      printf("Error: Invalid operator.\n");
   }
   return 0;
}
```

➢ Output

```
Enter the first number: 10
Enter the second number: 5
Enter the operator (+, -, *, /): +
Result: 15.00
```

➢ Discussion and Conclusion

This program allows the user to input two numbers and an operator (+, -, *, /). Based on the operator provided, the program performs the corresponding arithmetic operation and displays the result. It handles different scenarios such as addition, subtraction, multiplication, and division, while also checking for division by zero. The program provides the expected output for different input cases and ensures proper error handling. It was implemented using the VS Code IDE and compiled using GCC to generate an executable file.

9. WAP in C to input marks of five subjects C-programming, Physics, Maths, Applied Mechanics and Basic electrical. Display whether the student passed or failed. Take F.M=100 and P.M.=40 For passed students calculate percentage and grade according to following: Percentage >= 90% : A, Percentage >=80% : B , Percentage >= 70% : C , Percentage >= 60% : D , Percentage >= 40% : E

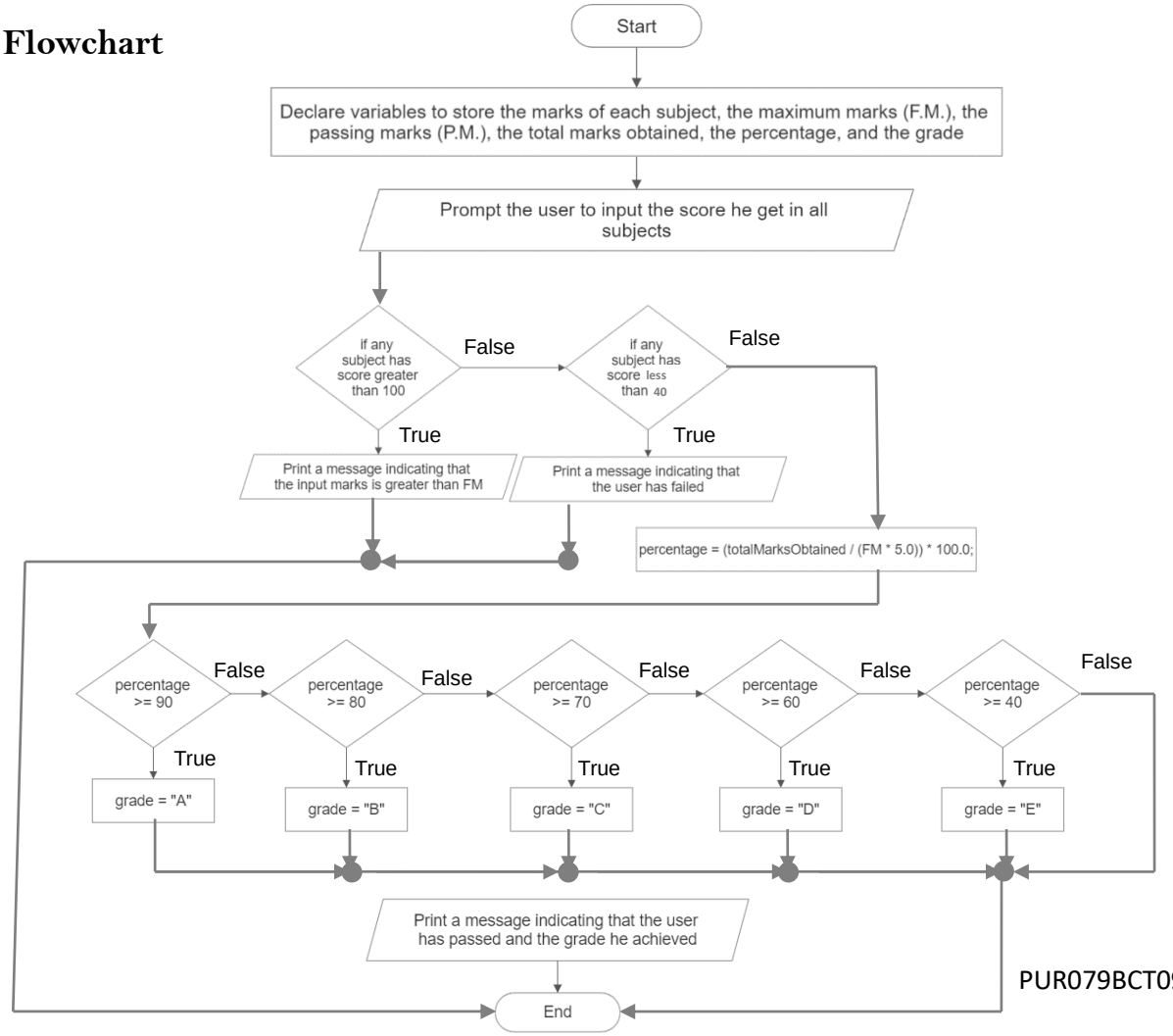
➢ Objective

The objective of this program is to write a code that takes marks of five subjects from the user and determines whether the student passed or failed. It also calculates the percentage and assigns a grade based on the given criteria.

➢ Algorithm

i. Start.
ii. Declare variables to store the marks of each subject, the maximum marks (F.M.), the passing marks (P.M.), the total marks obtained, the percentage, and the grade.
iii. Prompt the user to enter the marks of each subject (C-programming, Physics, Maths, Applied Mechanics, Basic Electrical) and store them in their respective variables.
iv. Ensure the marks provided by student is <= 100 because FM= 100.
v. If the user has scored less than 40 in any of the subjects he failed.
vi. If the user has not failed calculate his percentage as (total marks obtained / (F.M. * 5)) * 100 and assign him respective grades as Percentage >= 90%: Grade A Percentage >= 80%: Grade B Percentage >= 70%: Grade C Percentage >= 60%: Grade D Percentage >= 40%: Grade E
vii. Display the result indicating whether the student passed or failed, the percentage obtained, and the grade assigned.
viii. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ) {
   float cProgramming, physics, maths, appliedMechanics, basicElectrical, totalMarksObtained, percentage;
   float FM = 100.0, PM = 40.0;
   char grade;
   printf("Enter marks for C Programming, Physics, Maths, Applied Mechanics, and Basic Electrical: ");
   scanf("%f%f%f%f%f", &cProgramming, &physics, &maths, &appliedMechanics, &basicElectrical);
   totalMarksObtained = cProgramming + physics + maths + appliedMechanics + basicElectrical;

   if (cProgramming > FM || physics > FM || maths > FM || appliedMechanics > FM || basicElectrical > FM){
      printf("Error: Marks cannot be greater than 100.\n");
   } else if (cProgramming < PM || physics < PM || maths < PM || appliedMechanics < PM || basicElectrical < PM){
      printf("Sorry, you have failed.\n");
   } else {
      percentage = (totalMarksObtained / (FM * 5.0)) * 100.0;
      if (percentage >= 90.0){
         grade = 'A';
      }else if (percentage >= 80.0) {
         grade = 'B';
      }
      else if (percentage >= 70.0) {
         grade = 'C';
      } else if (percentage >= 60.0) {
         grade = 'D';
      } else if (percentage >= 40.0) {
         grade = 'E';
      }
      printf("Congratulations! You have passed.\n");
      printf("Percentage: %.2f%%\n", percentage);
      printf("Grade: %c\n", grade);
   }
   return 0;
}
```

➢ Output

```
Enter marks for C Programming, Physics, Maths, Applied Mechanics, and Basic Electrical: 85.5 78.2 92.6 81.8
73.4
Congratulations! You have passed.
Percentage: 82.70%
Grade: B
```

➢ Discussion and Conclusion

This program allows the user to input marks for five subjects (C Programming, Physics, Maths, Applied Mechanics, Basic Electrical) and determines whether the student passed or failed. It calculates the total marks obtained and checks if it is greater than or equal to the passing marks. If passed, it calculates the percentage and assigns a grade based on the given criteria. Finally, it displays the result indicating whether the student passed or failed, the percentage obtained, and the assigned grade. The program was implemented using the VS Code IDE and compiled using GCC to generate an executable file.

10. WAP to input a number from user. If user enters a number less than or equal to zero, then program should just display the number. If user enters 1 the program should display output as neither prime nor composite, if user enters 2 the program should display output as smallest and only even prime number. If user enters any number greater than 2 the program should check whether the number is prime or not, also if the number is not prime the program should display whether it is even or odd.

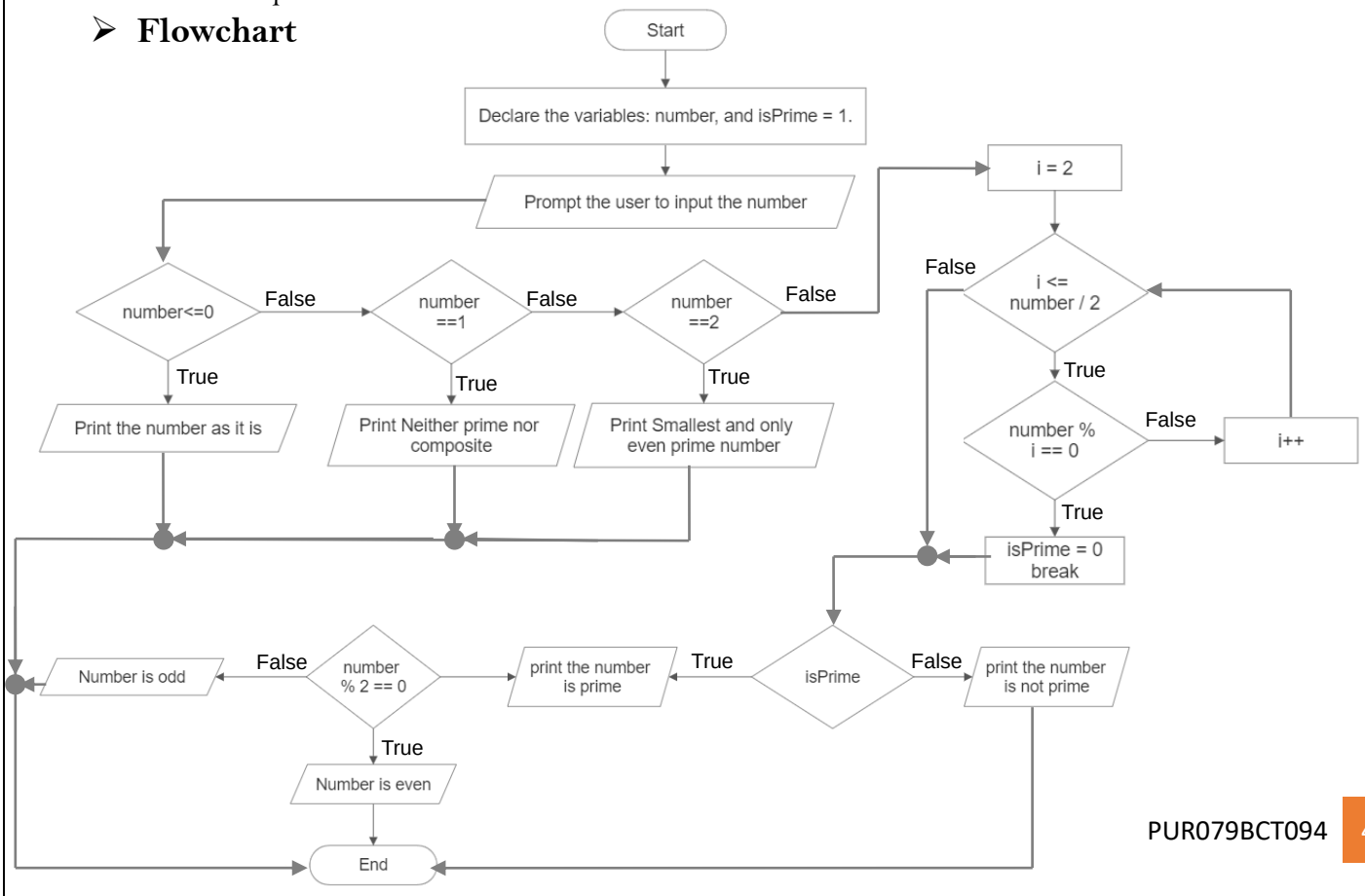
- **Objective**

The objective of this program is to take a number as input from the user and determine its properties. It checks if the number is less than or equal to zero and displays it as is. It also identifies if the number is 1, 2, or greater than 2, and performs specific checks accordingly. For numbers greater than 2, it determines whether the number is prime or composite and additionally checks if the number is even or odd.

- **Algorithm**

i. Start.
ii. Declare the variables: number, and isPrime = 1.
iii. Input a number from the user and store it in the number variable.
iv. If number is less than or equal to zero, display the number.
v. If number is equal to 1, display "Neither prime nor composite".
vi. If number is equal to 2, display "Smallest and only even prime number".
vii. If number is greater than 2, check whether it is prime as follows:
   a) Iterate from 2 to number / 2 using the variable i.
   b) If number is divisible by i, set isPrime to 0 and break the loop.
   c) If isPrime is 1, display "Number is prime" and stop.
   d) If isPrime is 0, display "Number is not prime".
   e) If number is even, display "Number is even".
   f) If number is odd, display "Number is odd".
viii. Stop.

- **Flowchart**



Code

```
#include <stdio.h>

int main( ) {
   int number, isPrime = 1;
   printf("Enter a number: ");
   scanf("%d", &number);

   if (number <= 0){
      printf("Number: %d\n", number);
   } else if (number == 1) {
       printf("Neither prime nor composite\n");
   } else if (number == 2) {
      printf("Smallest and only even prime number\n");
   } else {
      for (int i = 2; i <= number / 2; i++) {
         if (number % i == 0) {
            isPrime = 0;
            break;
         }
      }
      if (isPrime)  {
         printf("Number is prime\n");
      } else{
         printf("Number is not prime\n");
          if (number % 2 == 0){
            printf("Number is even\n");
         }  else {
            printf("Number is odd\n");
         }
      }
   }
   return 0;
}
```

➢ Output

```
Enter a number: 9
Number is not prime
Number is odd
```

➢ Discussion and Conclusion

This program accomplishes the objective of determining the properties of a given number. It correctly identifies numbers less than or equal to zero, 1, 2, and numbers greater than 2. It checks whether a number greater than 2 is prime or composite, and determines whether it is even or odd. The program uses a combination of conditional statements and loops to implement the required logic. It takes user input, performs the necessary calculations, and displays the output accordingly. The code is written in the C programming language using standard input/output functions and basic arithmetic operations.

In conclusion, this program provides an effective solution to determine the properties of a given number. It showcases the use of conditional statements, loops, and arithmetic operations in C programming.

Lab Sheet 4

1. WAP to get your name, address and display using unformatted I/O.

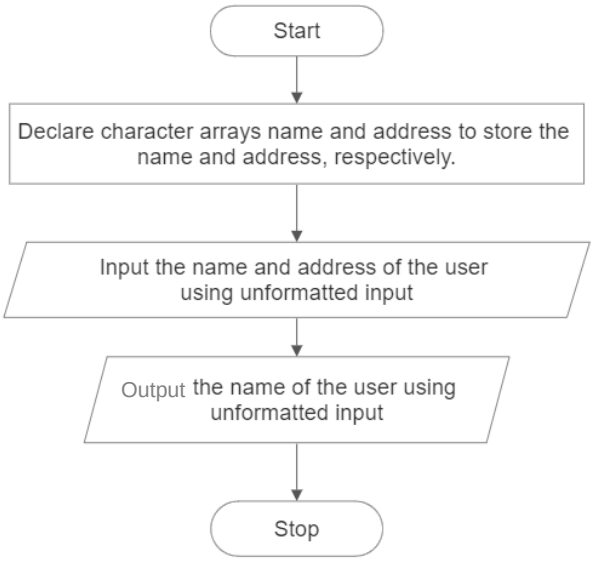
➢ Objective

The objective of this program is to input the name and address of the user and display them using unformatted I/O. Unformatted I/O allows reading and writing raw data without any specific formatting.

➢ Algorithm

i. Start.
ii. Declare character arrays name and address to store the name and address, respectively.
iii. Input the name of the user using unformatted input.
iv. Input the address from the user using unformatted input.
v. Output the name and address using unformatted output.
vi. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ){
   char name[100], address[100];
   puts("Enter your name: ");
   gets(name);
   puts("Enter your address: ");
   gets(address);
   puts("\nYour details:\n");
   puts("Name: ");
   puts(name);
   puts("Address: ");
   puts(address);
   return 0;
}
```

➢ Output

```
Enter your name: Tilak Thapa
Enter your address: Tulsipur – 4, Dang

Your details:
Name: Tilak Thapa
Address: Tulsipur – 4, Dang
```

➢ Discussion and Conclusion

This program allows the user to input their name and address using unformatted I/O, specifically the gets() function. Unformatted I/O allows reading and writing raw data without any specific formatting.

The program uses character arrays to store the name and address provided by the user. It prompts the user to enter their name and address, reads them using gets(), and then outputs the details using unformatted output with the puts() function. The code effectively demonstrates the use of unformatted I/O in C programming for reading and writing character data. It provides a simple and straightforward solution to input and display name and address using unformatted I/O.

In conclusion, this program successfully achieves its objective of obtaining and displaying the user's name and address using unformatted I/O. It showcases the usage of unformatted input and output functions in C programming.

2. WAP to get a character form the user using unformatted I/O and display the ASCII value of the entered character.

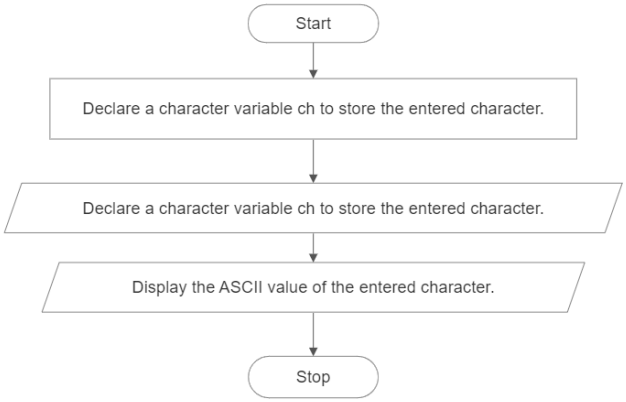
➢ Objective

The objective of this program is to input a character from the user using unformatted I/O and display the ASCII value of the entered character.

➢ Algorithm

i. Start.
ii. Declare a character variable ch to store the entered character.
iii. Prompt the user to enter a character using unformatted input.
iv. Read the character using unformatted input.
v. Display the ASCII value of the entered character.
vi. Stop

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main() {
   char ch;
   puts("Enter a character: ");
   ch = getch();
   puts("ASCII value of the entered character is: %d\n", ch);
   return 0;
}
```

➢ Output

Enter a character: A
ASCII value of the entered character is: 65

➢ Discussion and Conclusion

This program uses unformatted I/O to input a character from the user and then displays the ASCII value of the entered character.

3. WAP to display the output as [take a=15, b=20.43, c=35] as given.

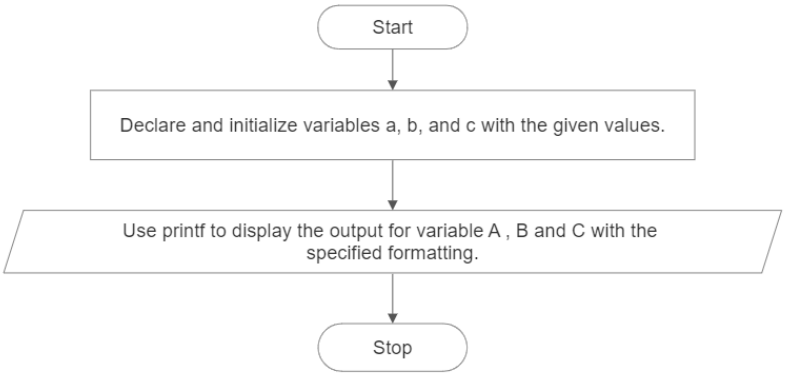
- **Objective**

  The objective of this program is to display the output as specified, with the given values of variables a, b, and c, using formatted I/O.

- **Algorithm**

  i. Start.
  ii. Declare and initialize variables a, b, and c with the given values.
  iii. Use printf to display the output for variable A , B and C with the specified formatting.
  iv. Stop.

- **Flowchart**



- **Code**

```
#include <stdio.h>

int main( )
{
   int a = 15;
   float b = 20.43;
   int c = 35;

   printf("A = %6d|%-6d|%6d|%-6d|%6d|%-6d|%6d|%-6d|%6d|%-6d|", a, a, a, a, a, a, a, a, a, a);
   printf("\nB = %6.2f|%-6.2f|%6.2f|%-6.2f|%6.2f|%-6.2f|%6.2f|%-6.2f|%6.2f|%-6.2f|", b, b, b, b, b, b, b, b, b, b);
   printf("\nC = %6d|%-6d|%6d|%-6d|%6d|%-6d|%6d|%-6d|%6d|%-6d|", c, c, c, c, c, c, c, c, c, c);
   return 0;
}
```

- **Output**

```
A =     15|15     |     15|15     |     15|15     |     15|15     |   15|15       |
B =  20.43|20.43 | 20.43|20.43 | 20.43|20.43 | 20.43|20.43 | 20.43|20.43 |
C =     35|35     |     35|35     |     35|35     |     35|35     |   35|35       |
```

- **Discussion and Conclusion**

  This program successfully displays the output as specified in the objective using formatted I/O with the specified formatting. It initializes the variables a, b, and c with the given values and then uses printf to display them in the desired format.The code uses format specifiers such as %d for integers (a and c) and %6d for right-aligned integers, %f for floating-point numbers (b) and %6.2f for right-aligned floating-point numbers with two decimal places. The - flag is used to left-align the values.

4. WAP to display the output as below using formatted I/O [take char a[]="I Love Nepal"].

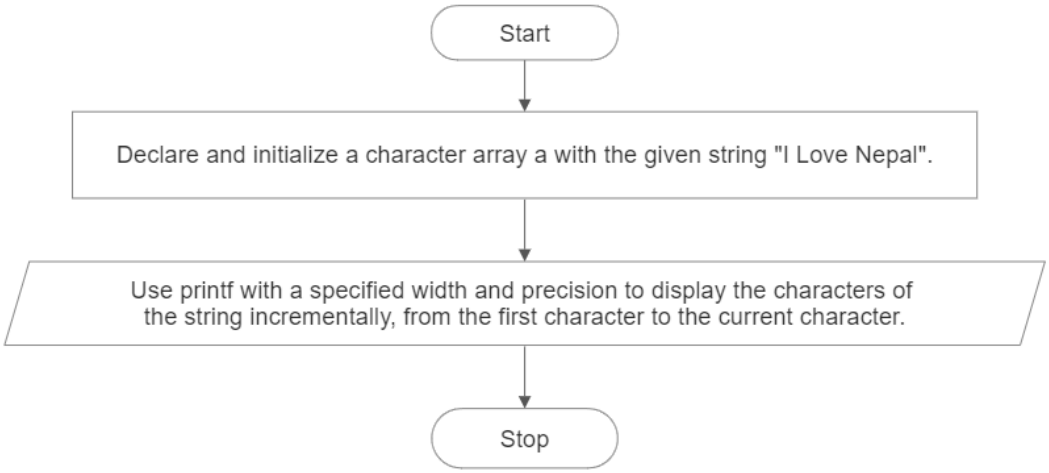
➢ Objective

The objective of this program is to display the output as specified, which is a pyramid of characters obtained from the given string "I Love Nepal" using formatted I/O.

➢ Algorithm

i. Start
ii. Declare and initialize a character array a with the given string "I Love Nepal".
iii. Use printf with a specified width and precision to display the characters of the string incrementally, from the first character to the current character.
iv. Stop.

➢ Flowchart



➢ Code

```
#include <stdio.h>

int main( ) {
   char a[] = "I Love Nepal";
   printf("%1.1s\n", a);
   printf("%1.3s\n", a);
   printf("%1.4s\n", a);
   printf("%1.5s\n", a);
   printf("%1.7s\n", a);
   printf("%1.8s\n", a);
   printf("%1.9s\n", a);
   printf("%1.10s\n", a);
   printf("%1.11s\n", a);
   printf("%1.12s\n", a);
   return 0;
}
```

➢ Output

```
I
I L
I Lo
I Lov
I Love
I Love N
```

```
I Love Ne
I Love Nep
I Love Nepa
I Love Nepal
```

➢ Discussion and Conclusion

This program successfully displays the desired output using formatted I/O. It initializes a character array a with the given string "I Love Nepal" and then uses printf with a specified width and precision to display the characters incrementally and create a pyramid pattern.

By specifying the width and precision in the format string of printf, we can control the number of characters to be displayed. The width represents the minimum field width, and the precision represents the maximum number of characters to be displayed.

Lab Sheet 5

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

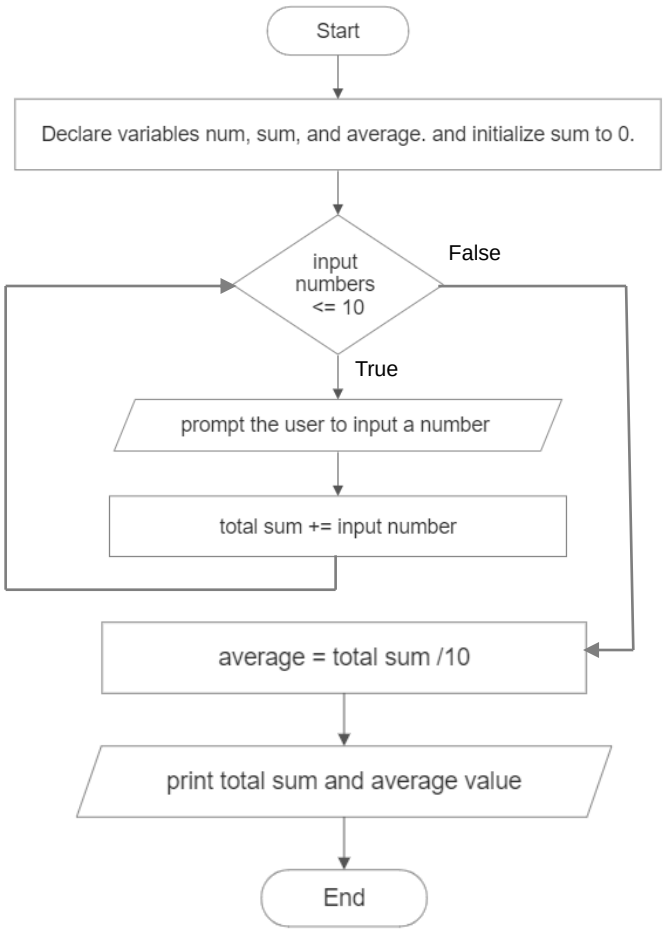
➢ Objective

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

➢ Algorithm

i. Start.
ii. Declare variables num, sum, and average.
iii. Initialize sum to 0.
iv. Prompt the user to enter 10 numbers one by one and read them using scanf.
v. Add each number to the sum variable.
vi. Calculate the average by dividing the sum by 10.
vii. Display the sum and average using printf.
viii. Stop.

➢ Flowchart



➢ 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.

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

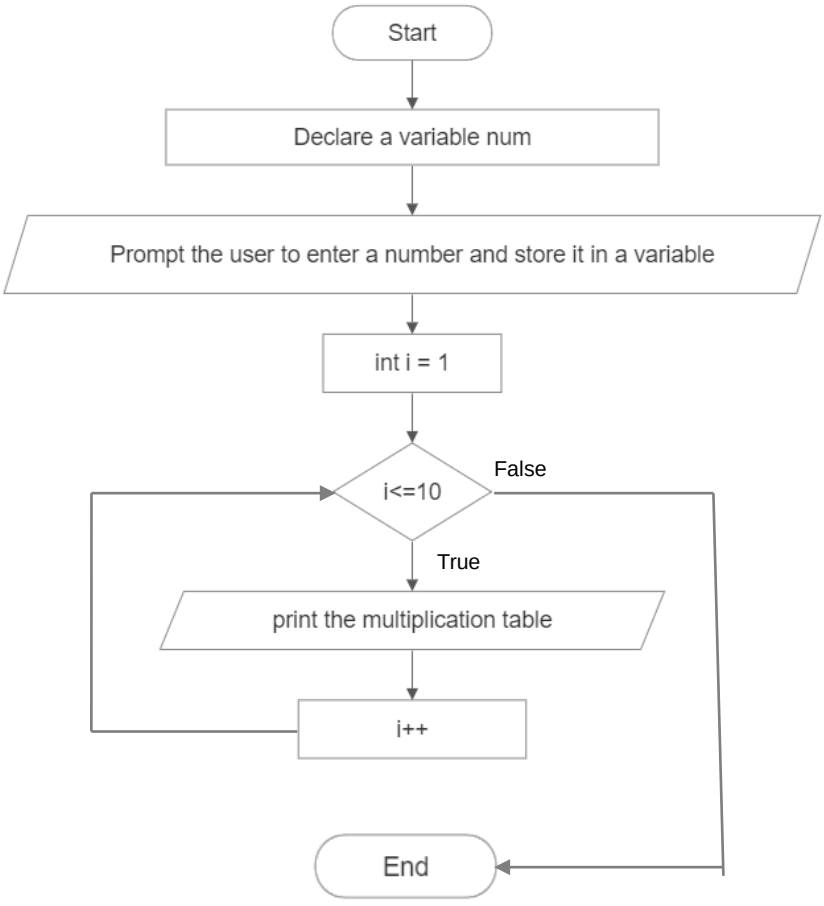
- **Objective**

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

- **Algorithm**

i. Start.
ii. Declare variables num and i.
iii. Prompt the user to enter a number and read it using scanf.
iv. Print the multiplication table heading.
v. Use a loop to iterate from 1 to 10.
vi. Inside the loop, calculate the product of num and i.
vii. Print the multiplication expression and the product using printf.
viii. Stop.

- **Flowchart**



➢ 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

3. 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].

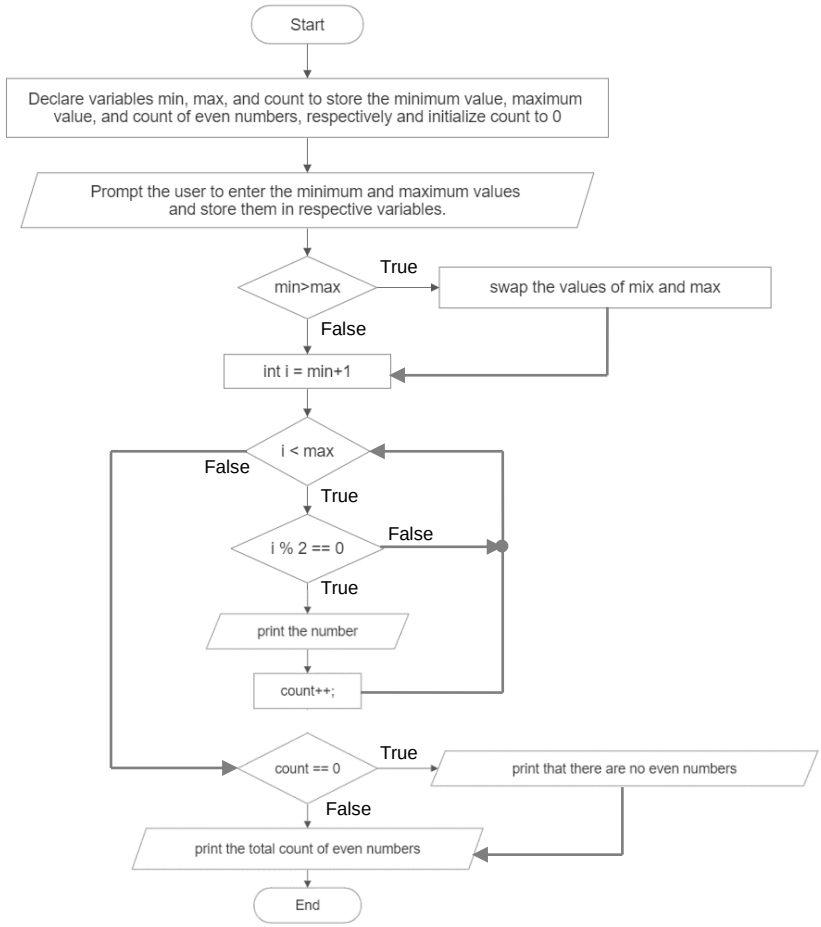
- **Objective**

  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**

  i. Start.
  ii. Declare variables min, max, and count to store the minimum value, maximum value, and count of even numbers, respectively and initialize count to 0.
  iii. Prompt the user to enter the minimum and maximum values and store them in respective variables.
  iv. If min is greater than max, swap their values.
  v. Print the message indicating the range of numbers.
  vi. Initialize a loop with a variable i starting from min + 1 and ending at max - 1.
  vii. Within the loop, check if i is divisible by 2 (i.e., an even number).
  viii. If i is even, print its value and increment the count variable.
  ix. After the loop, check if the count variable is 0.
  x. If count is 0, print "none" to indicate that there are no even numbers in the range.
  xi. Print the total count of even numbers.
  xii. End.

- **Flowchart**



➢ 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.

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

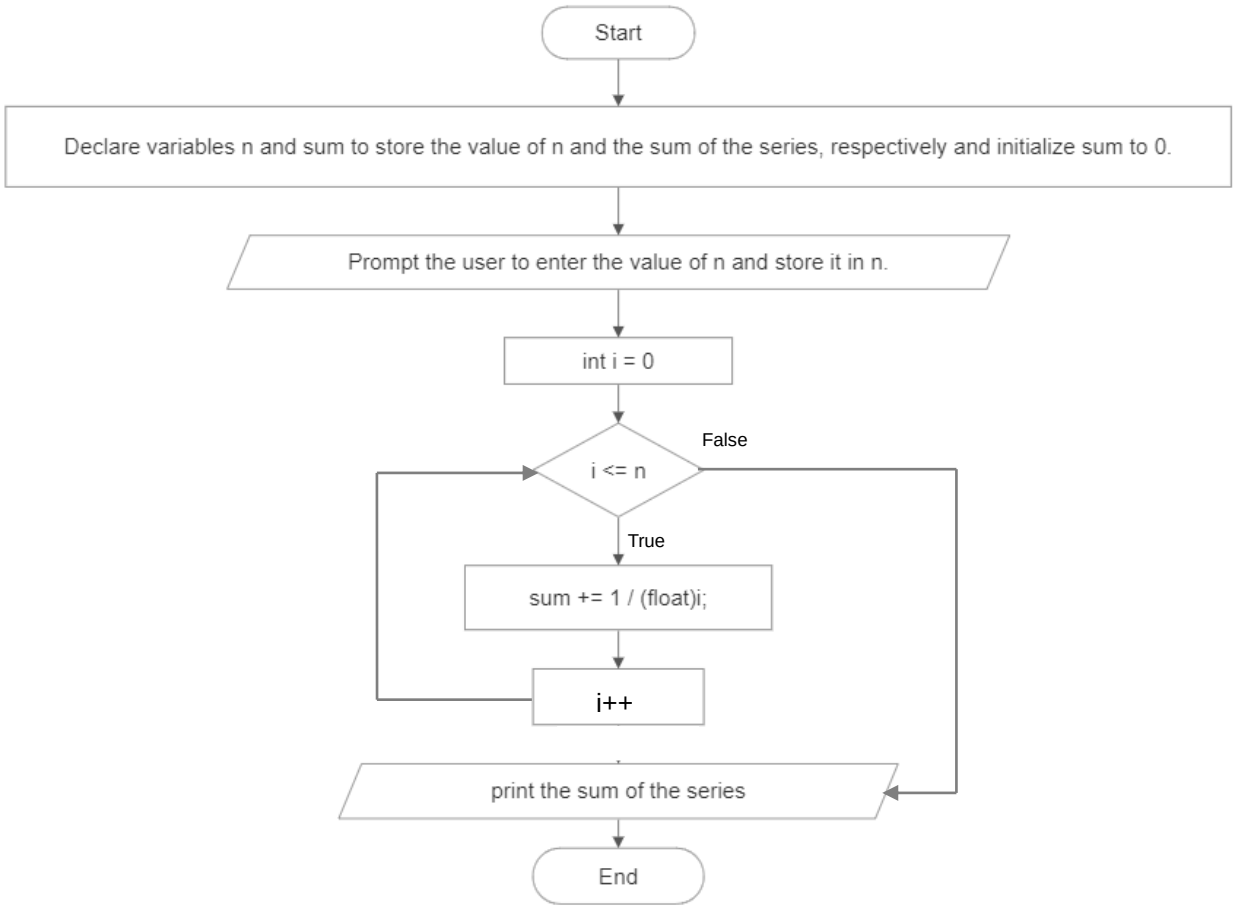
➢ Objective

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

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

➢ Flowchart



➢ 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.

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

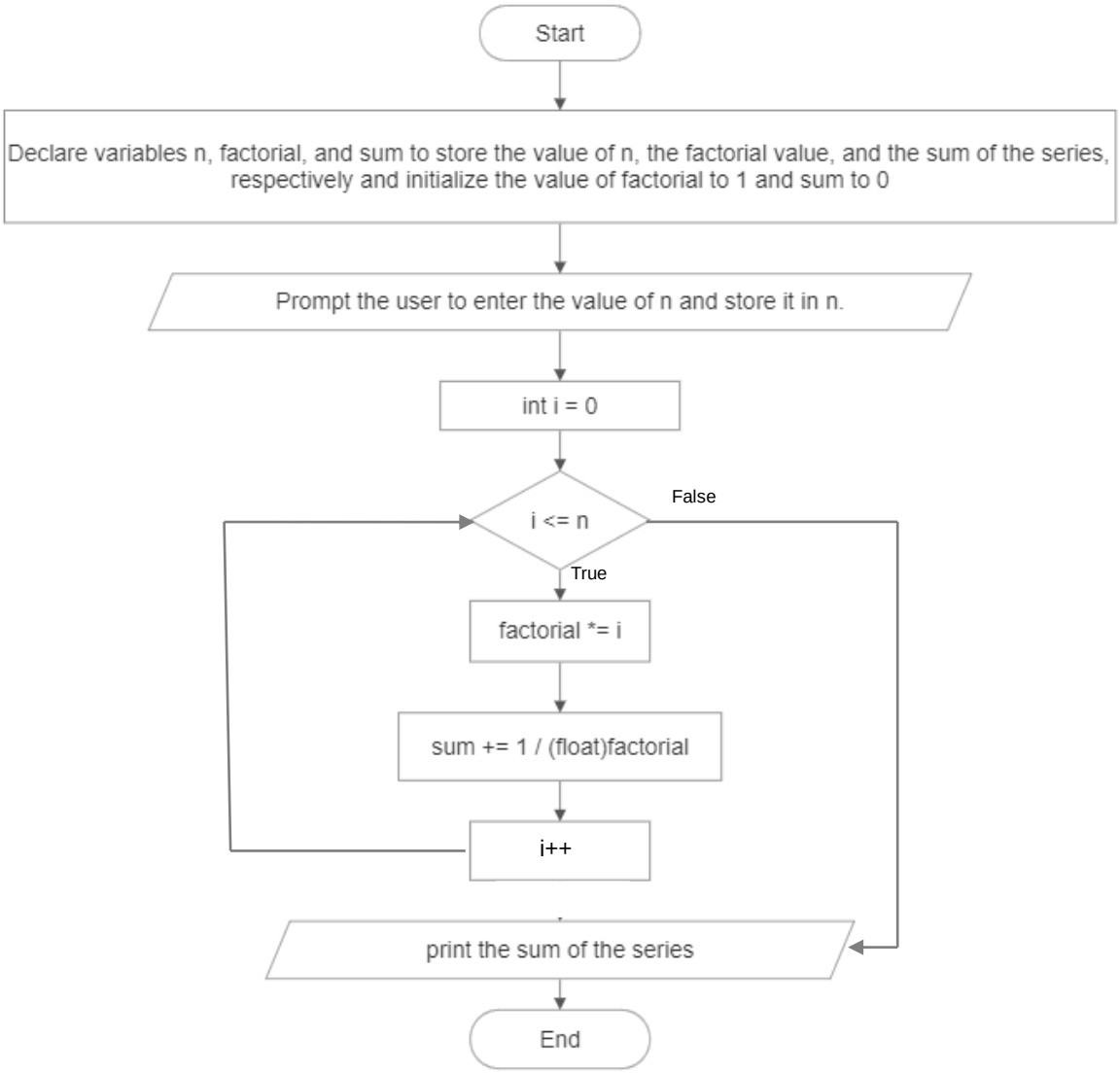
➢ Objective

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

i. Start.
ii. 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.
iii. Prompt the user to enter the value of n and store it to the respective variable.
iv. Use a loop with a variable i starting from 1 and ending at n.
v. 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.
vi. After the loop, print the value of sum as the sum of the series.
vii. End.

➢ Flowchart



➢ 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.

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

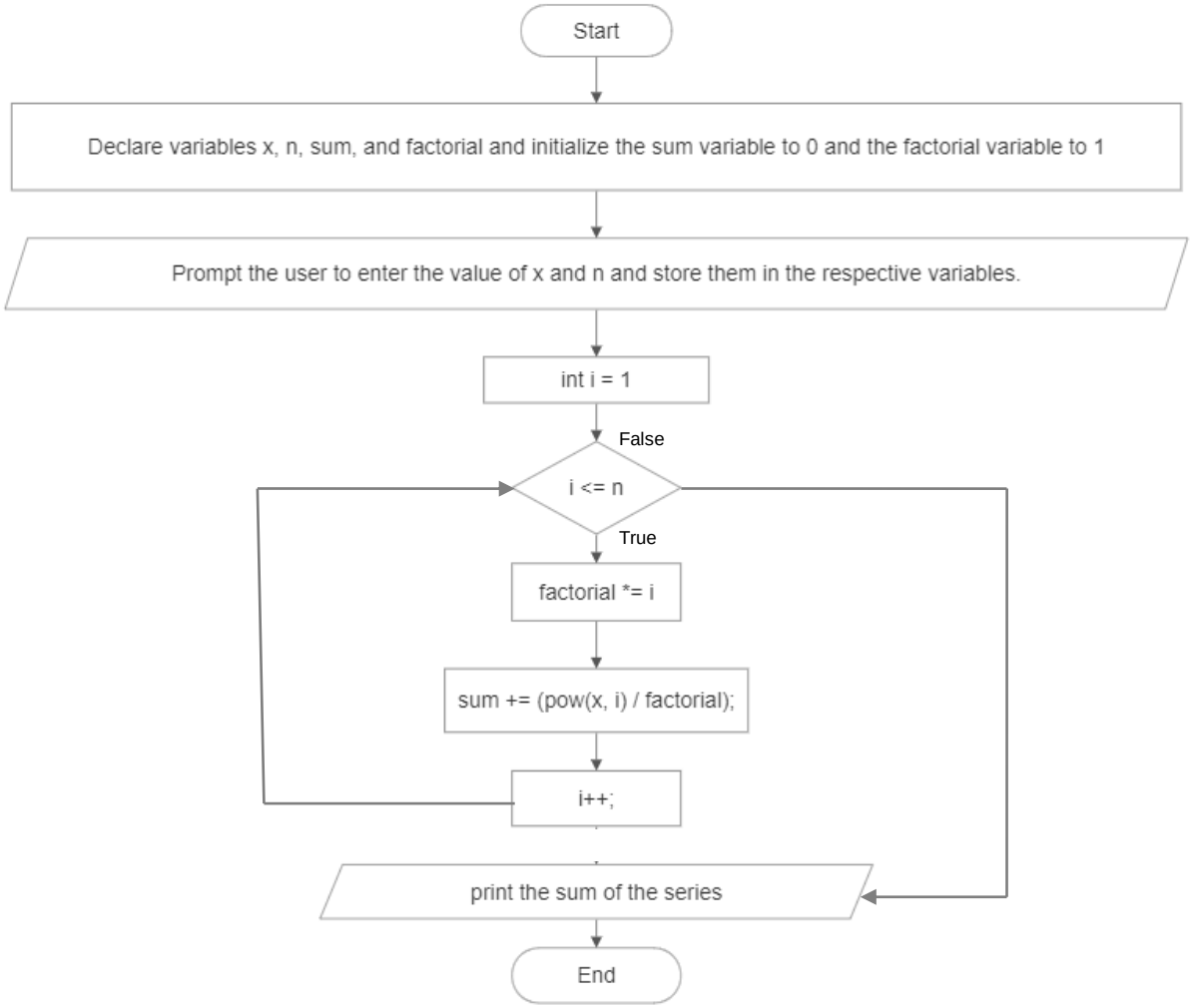
➢ Objective

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

i. Start.
ii. Declare variables x, n, sum, and factorial and initialize the sum variable to 0 and the factorial variable to 1.
iii. Prompt the user to enter the value of x and n and store them in the respective variables.
iv. Use a loop with a variable i starting from 1 and ending at n.
v. Within the loop, calculate the factorial of i by multiplying it with the current value of factorial.
vi. Calculate x raised to the power of i using the pow() function from the math.h library.
vii. Add (x^i / factorial) to the sum variable.
viii. After the loop, print the value of sum as the sum of the series.
ix. End.

➢ Flowchart



➢ 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.

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

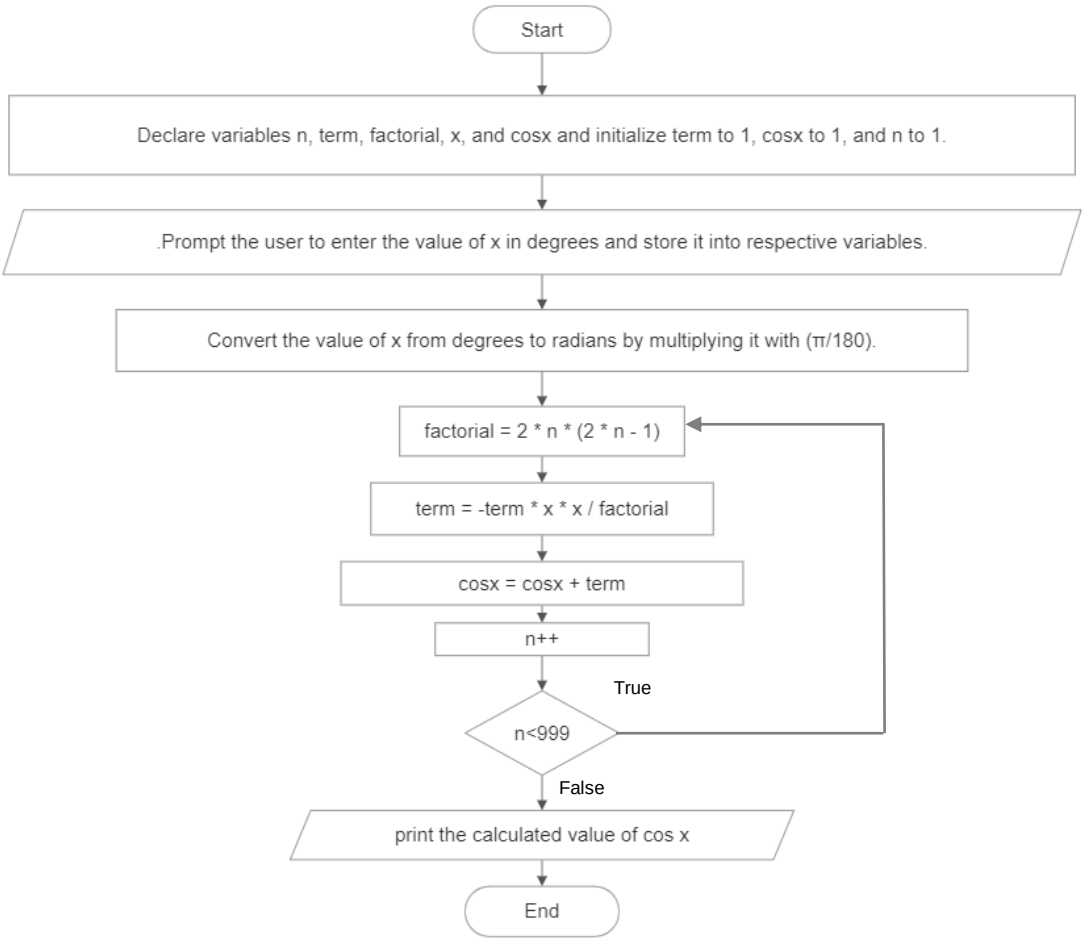
➢ **Objective:**

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**

i. Start.
ii. Declare variables n, term, factorial, x, and cosx and initialize term to 1, cosx to 1, and n to 1.
iii. Prompt the user to enter the value of x in degrees and store it into respective variables.
iv. Convert the value of x from degrees to radians by multiplying it with (π/180).
v. Use a do-while loop to calculate the terms of the Taylor series expansion.
vi. Inside the loop, calculate the factorial of (2 * n) * (2 * n - 1).
vii. Calculate the next term of the series using the previous term, x, and factorial.
viii. Add the term to the cosx variable.
ix. Increment n by 1.
x. Repeat steps 8-11 until n reaches a sufficiently large value (e.g., 999).
xi. Print the calculated value of cos(x)
xii. End.

➢ **Flowchart**



➢ 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.

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

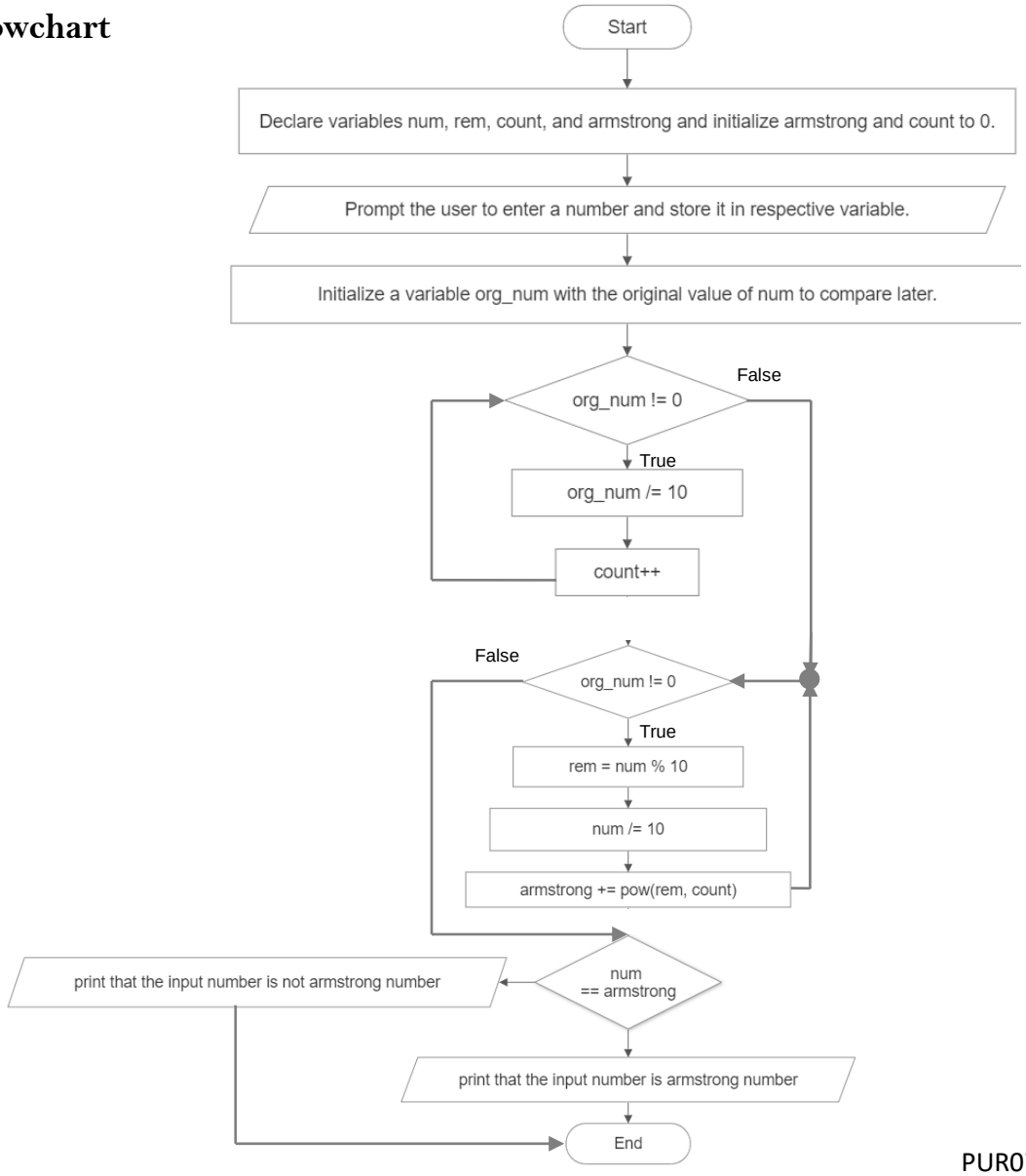
➢ Objective

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

i. Start.
ii. Declare variables num, rem, armstrong, and count and initialize armstrong and count to 0.
iii. Prompt the user to enter a number and store it in the respective variable.
iv. Initialize a variable org_num with the value of num.
v. Calculate the number of digits in num by repeatedly dividing org_num by 10 and incrementing count until org_num becomes 0.
vi. Reset org_num to its original value.
vii. Use a while loop to calculate the sum of individual digits raised to power count of org_num.
viii. Check if num is equal to armstrong.
ix. If they are equal, print that num is an Armstrong number.
x. If they are not equal, print that num is not an Armstrong number.
xi. End.

➢ Flowchart



➢ 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.

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

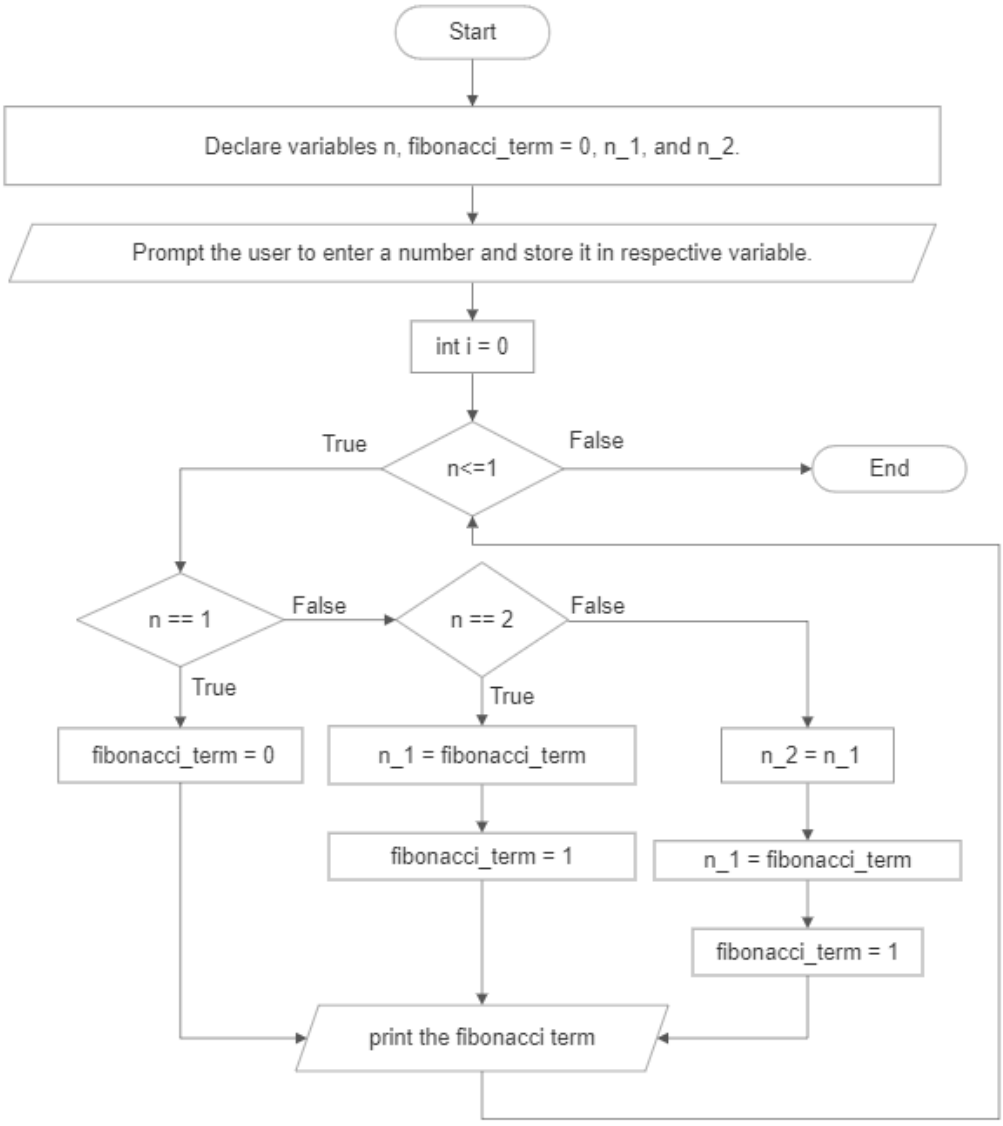
➢ **Objective**

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

➢ **Algorithm**

i. Start.
ii. Declare variables n, fibonacci_term = 0, n_1, and n_2.
iii. Prompt the user to enter a number and store it in respective variable.
iv. Use a for loop to iterate from 1 to n.
v. Initialize fibonacci_term as 0 for the first term.
vi. If i is equal to 2, set n_1 as fibonacci_term and update fibonacci_term to 1.
vii. 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.
viii. Print the value of fibonacci_term.
ix. 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.

10. WAP to display the number in reverse order.

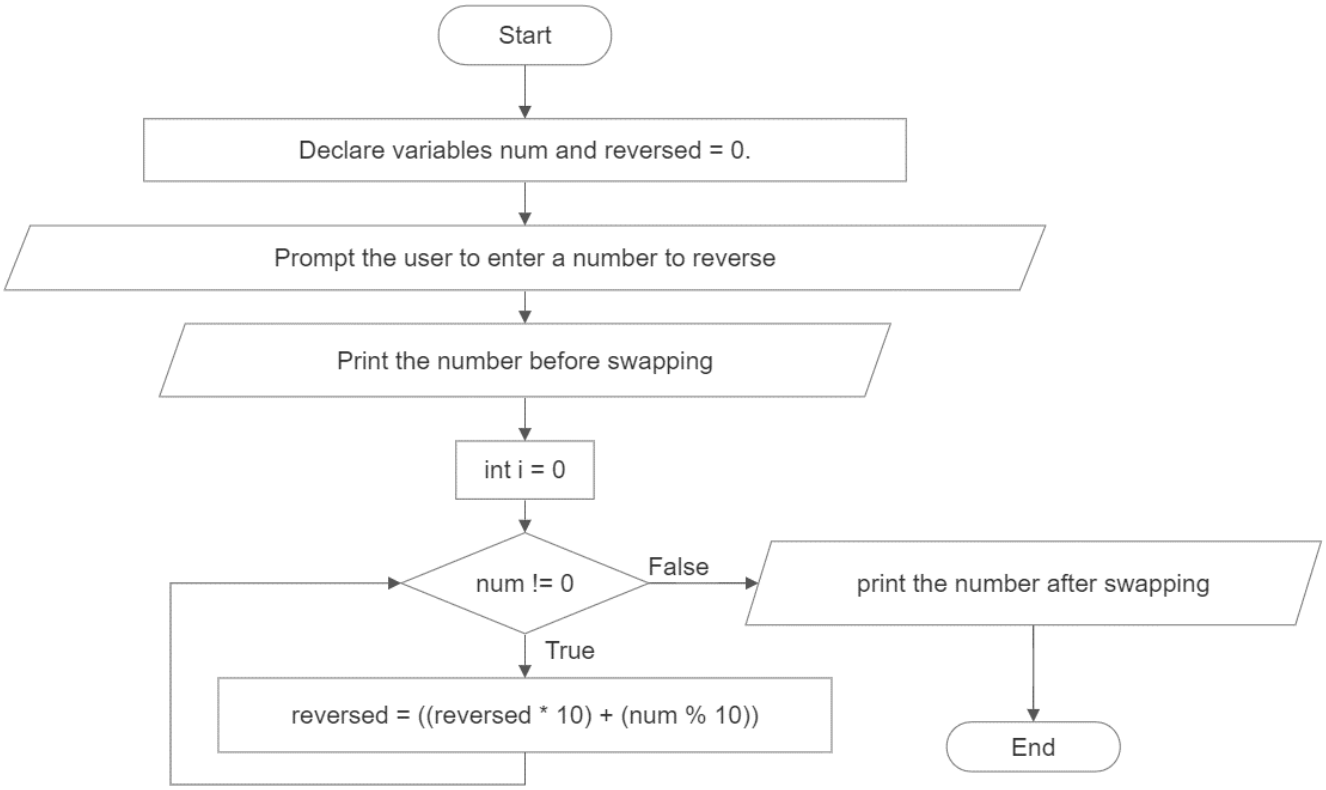
➢ Objective

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

➢ Algorithm

i. Start.
ii. Declare variables num and reversed = 0.
iii. Prompt the user to enter a number and store the input in respective variable.
iv. Print the number before swapping.
v. Use a for loop to reverse the number.
vi. Repeat the loop until num becomes 0.
vii. In each iteration, calculate the reversed number by multiplying it by 10 and adding the last digit of num using the modulus operator.
viii. Divide num by 10 to remove the last digit.
ix. Print the reversed number
x. 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.

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

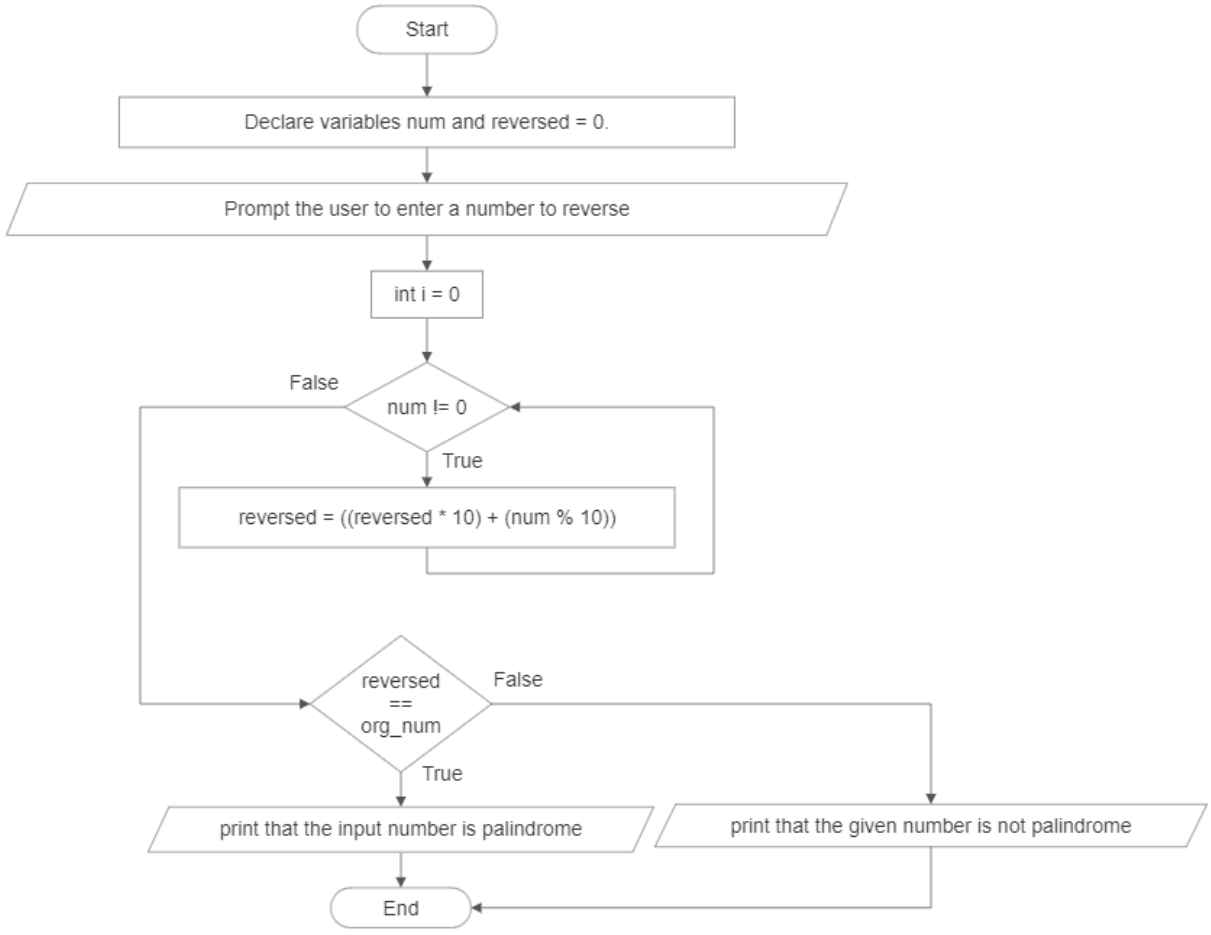
➢ Objective

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

➢ Algorithm

i. Start the program.
ii. Declare variables num, reversed = 0, and org_num.
iii. Prompt the user to enter a number.
iv. Read the value of num from the user.
v. Set org_num equal to num.
vi. Use a for loop to reverse the number.
vii. Repeat the loop until num becomes 0.
viii. In each iteration, calculate the reversed number by multiplying it by 10 and adding the last digit of num using the modulus operator.
ix. Divide num by 10 to remove the last digit.
x. Check if org_num is equal to reversed.
xi. If they are equal, print the message "num is a palindrome number."
xii. If they are not equal, print the message "num is not a palindrome number."
xiii. 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.

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

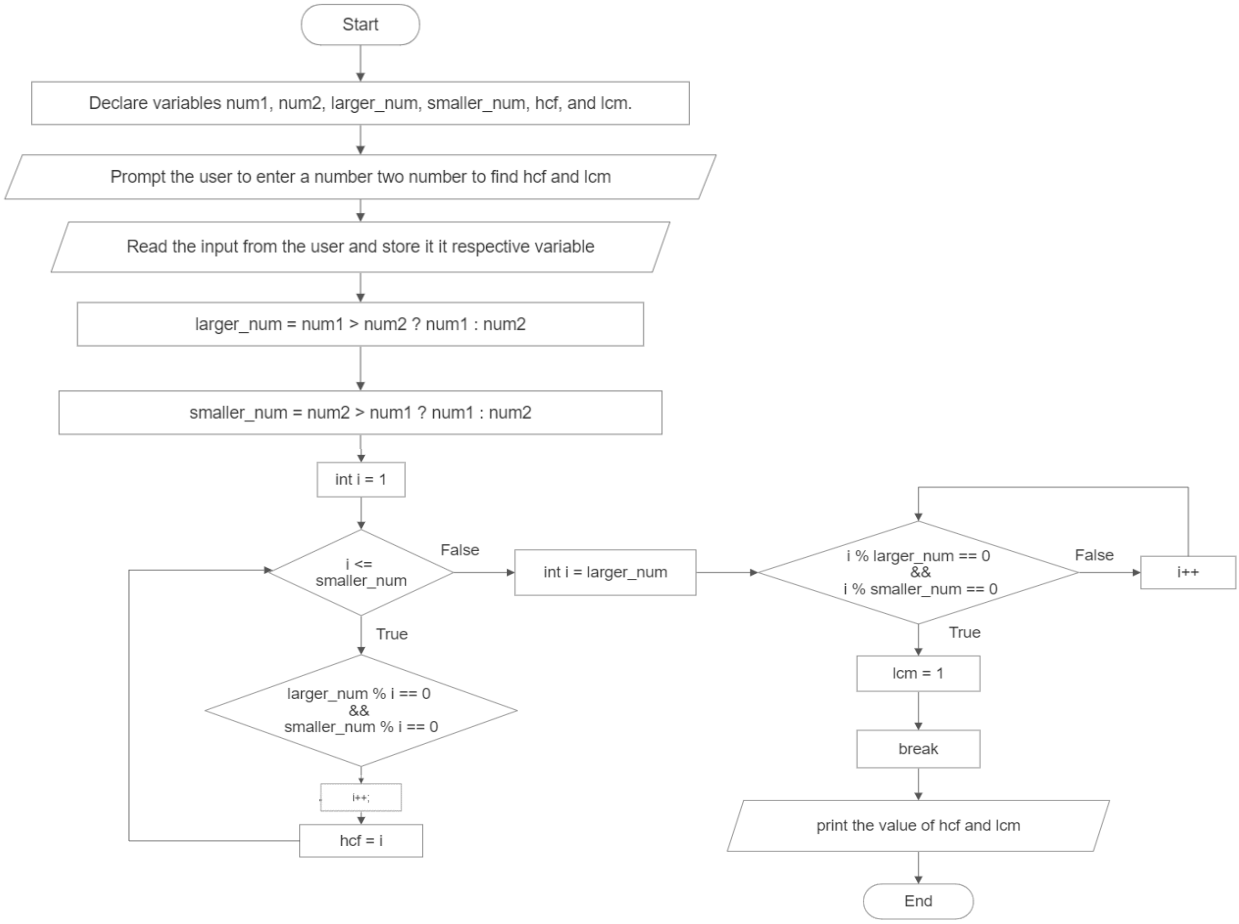
➢ Objective

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

➢ Algorithm

i. Start.
ii. Declare variables num1, num2, larger_num, smaller_num, hcf, and lcm.
iii. Prompt the user to enter two numbers.
iv. Read the values of num1 and num2 from the user.
v. Determine the larger and smaller numbers by comparing num1 and num2.
vi. Use a for loop to find the HCF.
vii. Start the loop from 1 and iterate up to the value of the smaller number.
viii. Check if both the larger and smaller numbers are divisible by the current iteration value.
ix. If they are divisible, update the value of hcf to the current iteration value.
x. Use an infinite for loop to find the LCM.
xi. Start the loop from the larger number and increment by the value of the larger number in each iteration.
xii. Check if both the current iteration value and the smaller number are divisible by the larger number.
xiii. If they are divisible, update the value of lcm to the current iteration value and break the loop.
xiv. Print the values of hcf and lcm.
xv. 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.

13. WAP to print the following patterns:

➢ Objective

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

➢ Algorithm

a) main():

i. Start.
ii. Print a newline and the string "Pattern 1: ".
iii. Call the pattern1 function.
iv. Print a newline and the string "Pattern 2: ".
v. Call the pattern2 function.
vi. Print a newline and the string "Pattern 3: ".
vii. Call the pattern3 function.
viii. Print a newline and the string "Pattern 4: ".
ix. Call the pattern4 function.
x. Print a newline and the string "Pattern 5: ".
xi. Call the pattern5 function.
xii. Print a newline and the string "Pattern 6: ".
xiii. Call the pattern6 function.
xiv. End.

b) pattern1():

i. Start.
ii. Use a loop to iterate over the rows from 1 to 5.
iii. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
iv. Print the column number for each iteration.
v. Move to the next line after printing the columns.
vi. End.

c) pattern2():

i. Start.
ii. Use a loop to iterate over the rows from 5 to 1 in a descending order.
iii. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
iv. Print the column number for each iteration.
v. Move to the next line after printing the columns.
vi. End.

d) pattern3():

i. Start.
ii. Use a loop to iterate over the rows from 1 to 5.
iii. Inside the row loop, use a loop to print spaces based on the row number (5 - current row).
iv. Use another loop to print stars based on the row number (2 * current row - 1).
v. Move to the next line after printing the spaces and stars.
vi. End.

e) pattern4():

i. Start the pattern4 function.
ii. Initialize a variable num to 1.
iii. Use a loop to iterate over the rows from 1 to 5.
iv. Inside the row loop, use another loop to iterate over the columns from 1 to the current row number.
v. Print the current value of num and increment it by 1 for each iteration.
vi. Move to the next line after printing the columns.
vii. End the function.

f) patter5():

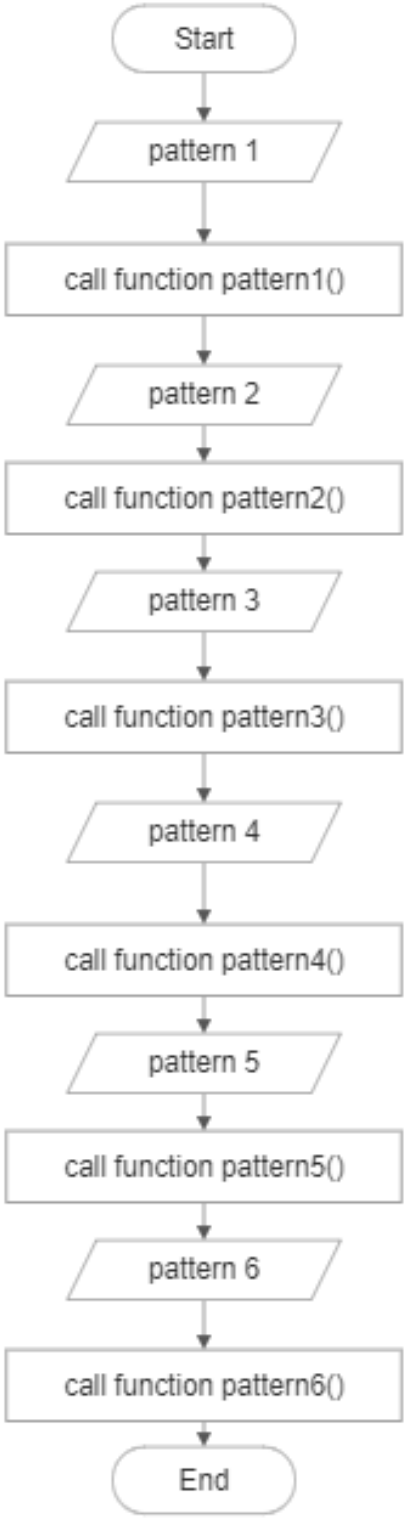
i. Start.
ii. Use a loop to iterate over the rows from 1 to 5.
iii. Inside the row loop, use a loop to print spaces based on the row number.
iv. Use another loop to print numbers in an increasing order from 1 to (6 - row).
v. Use a third loop to print numbers in a decreasing order from (5 - row) to 1.
vi. Move to the next line after printing the spaces and numbers.
vii. End.

g) pattern6():

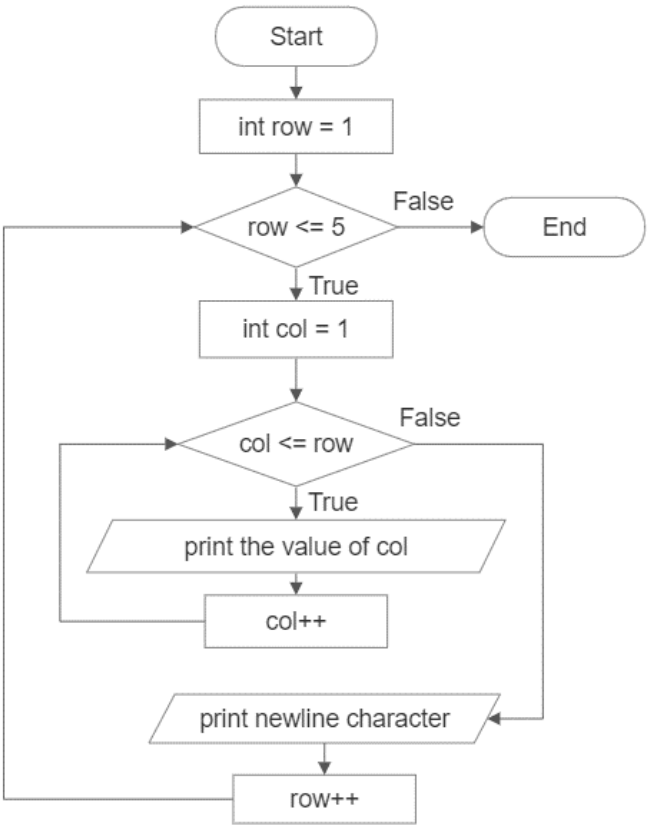
i. Start.
ii. Use a loop to iterate over the rows from 5 to 1 in a descending order.
iii. Inside the row loop, use another loop to iterate over the columns from 5 to (6 - row).
iv. Print the column number for each iteration.
v. Move to the next line after printing the columns.
vi. End.

- **Flowchart**

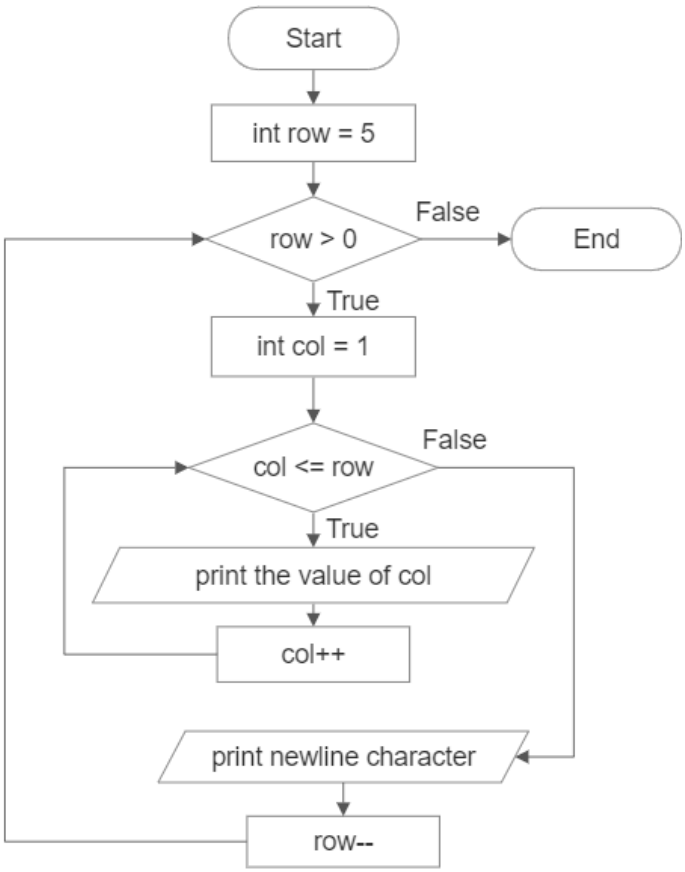
Main Function



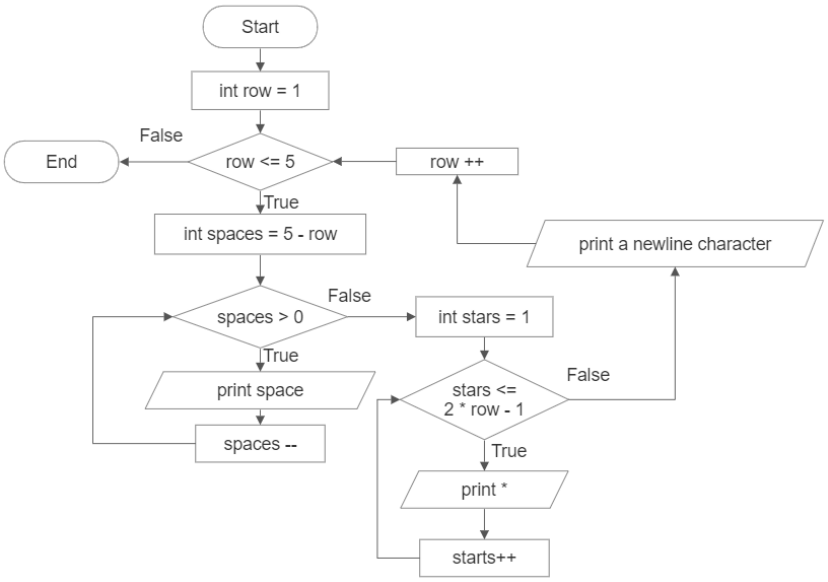
pattern1



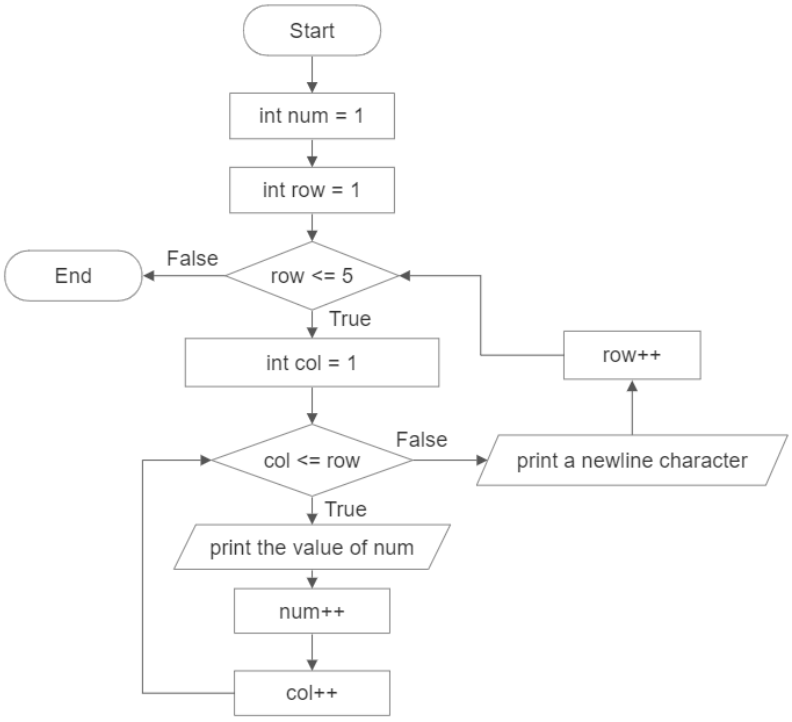
pattern2



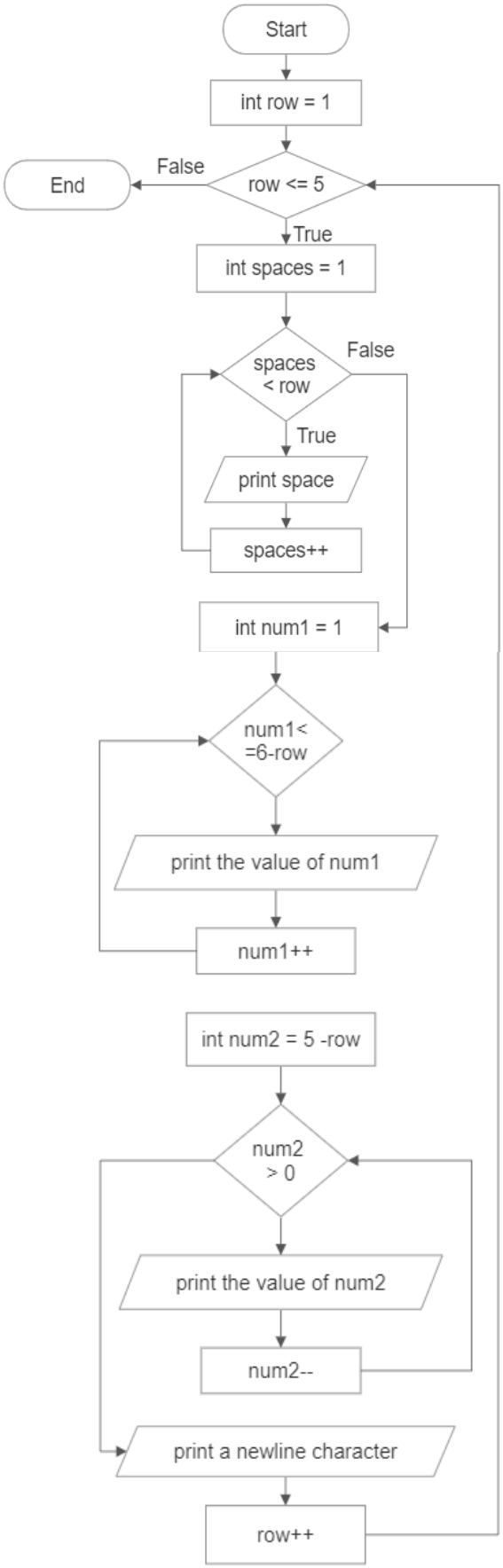
pattern3



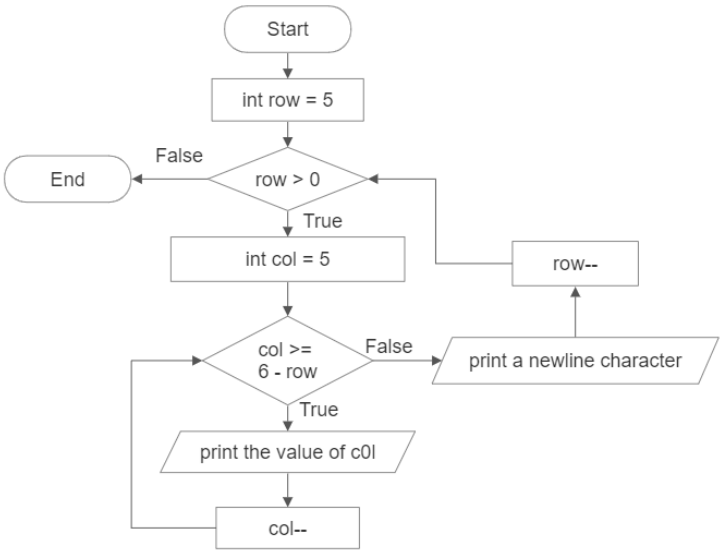
pattern4



pattern5



pattern6



➢ 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:
5 4 3 2 1
123454321
 1234321
  12321
   121
    1

Pattern 6:
5
5 4
5 4 3
5 4 3 2
```

➢ 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.