Lab Sheet 3

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

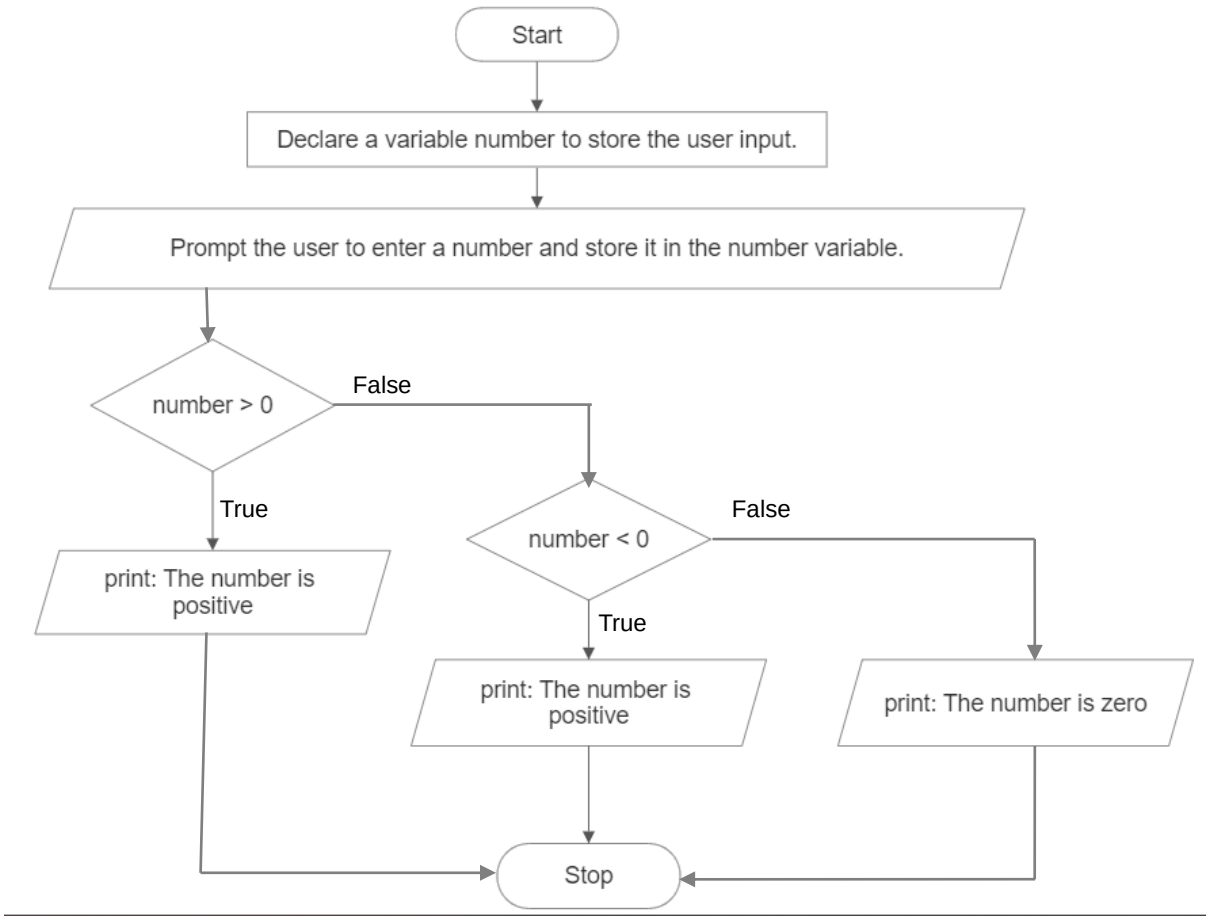
➢ Objective

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**

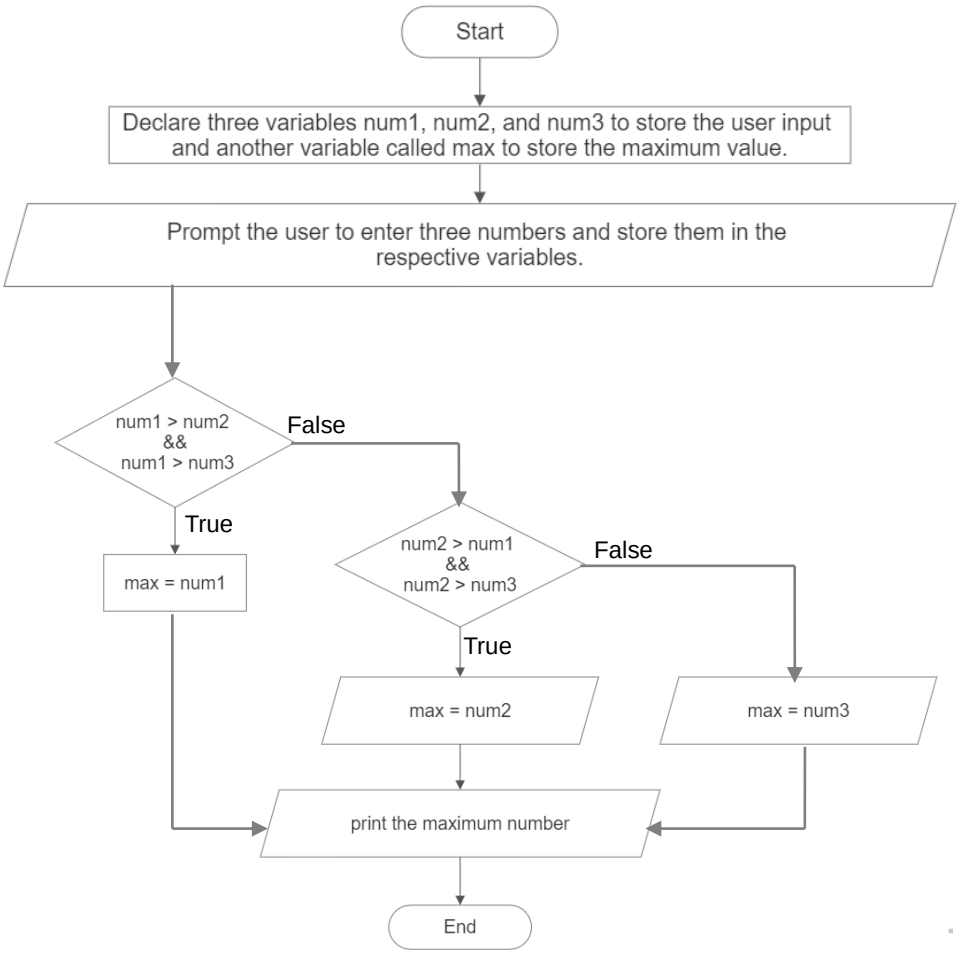
  The objective of this program is to write a code that takes three numbers as input from the user and determines the maximum among them. The program will use conditional statements to compare the values of the three numbers and display the maximum value.

- **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

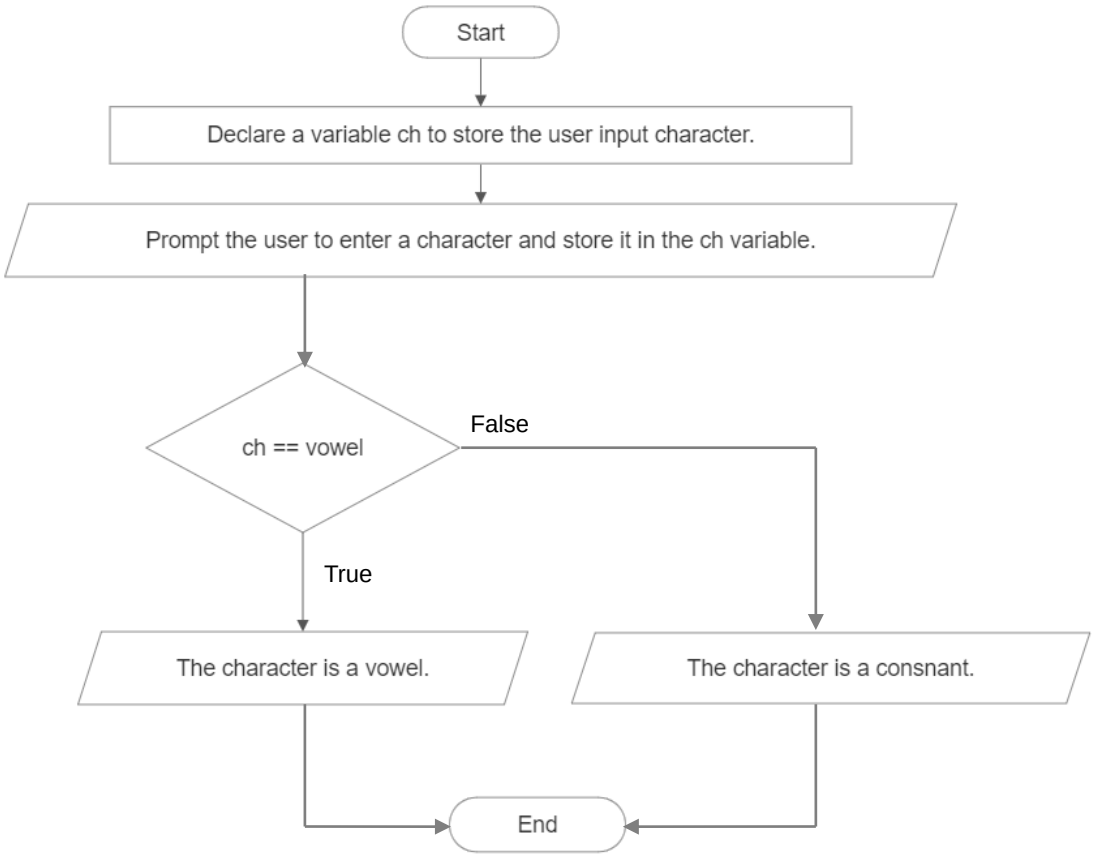
The objective of this program is to write a code that takes a character as input from the user and determines whether the character is a vowel or a consonant.

➢ 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

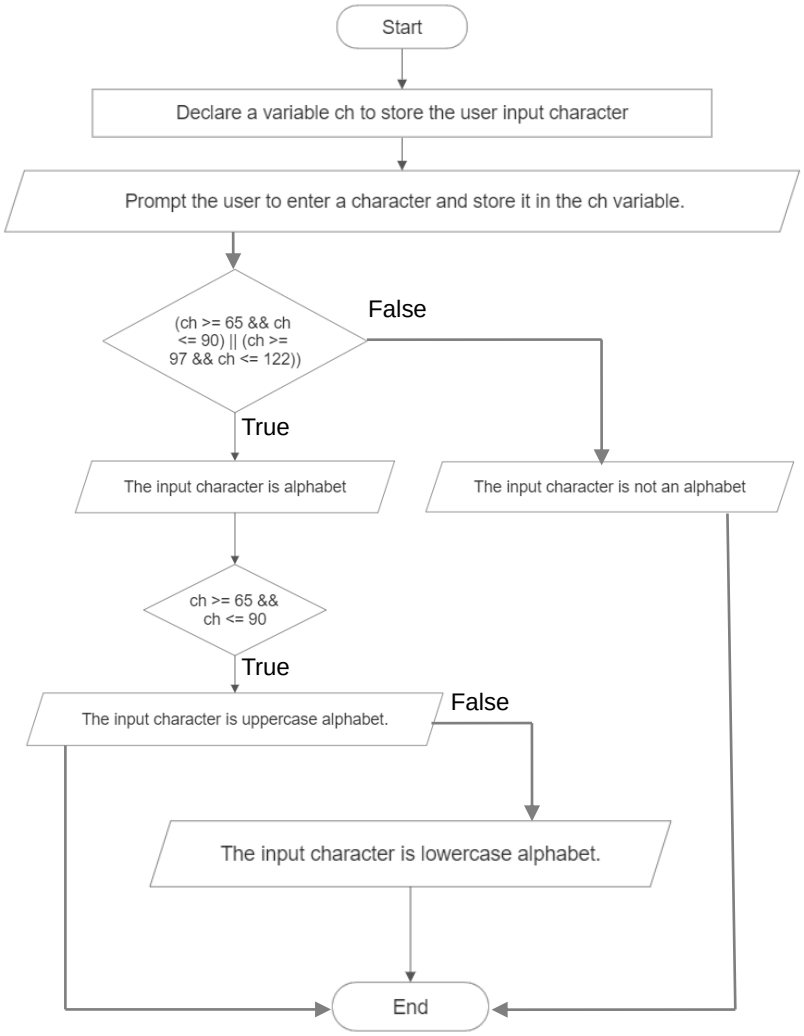
➢ **Objective**

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

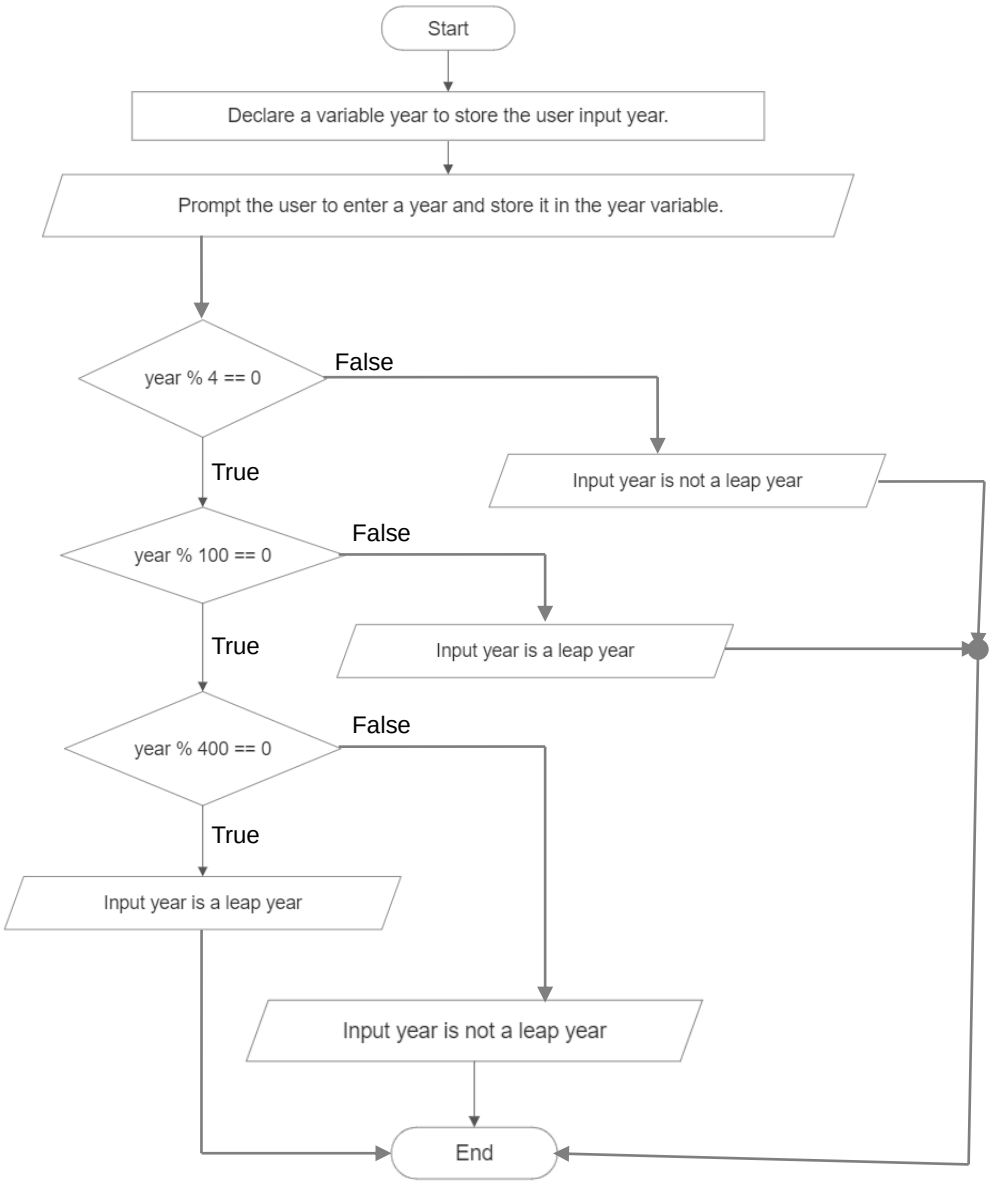
The objective of this program is to write a code that takes a year as input from the user and determines whether the year is a leap year or not.

➢ 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

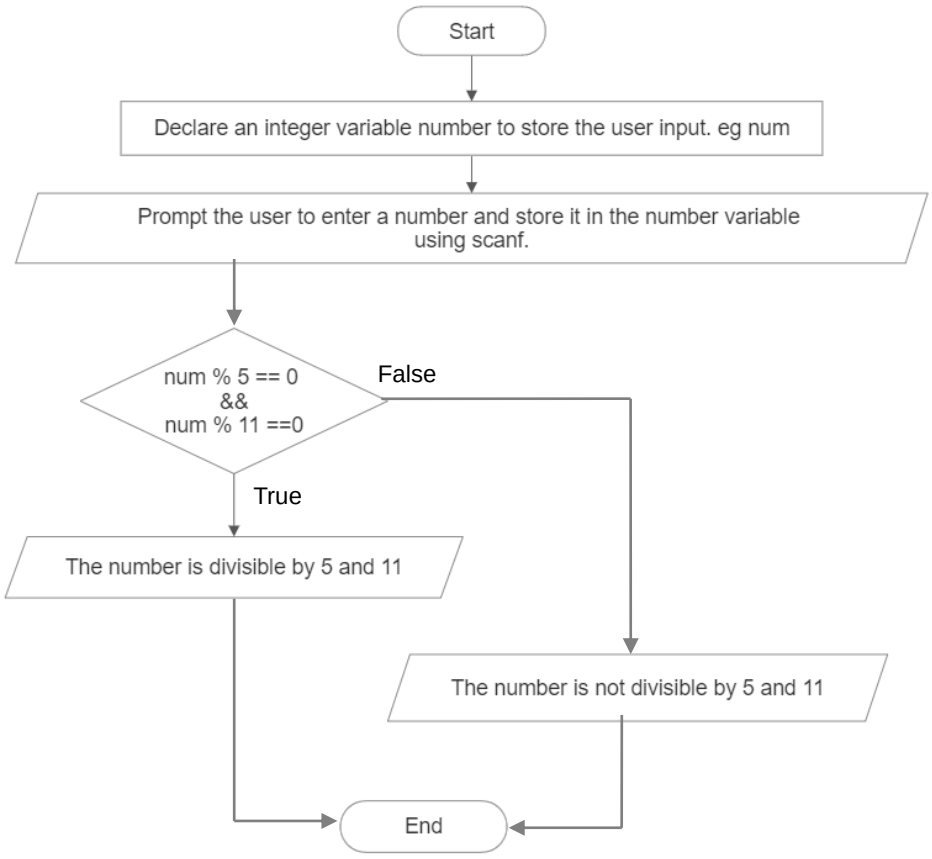
- **Objective**

  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.

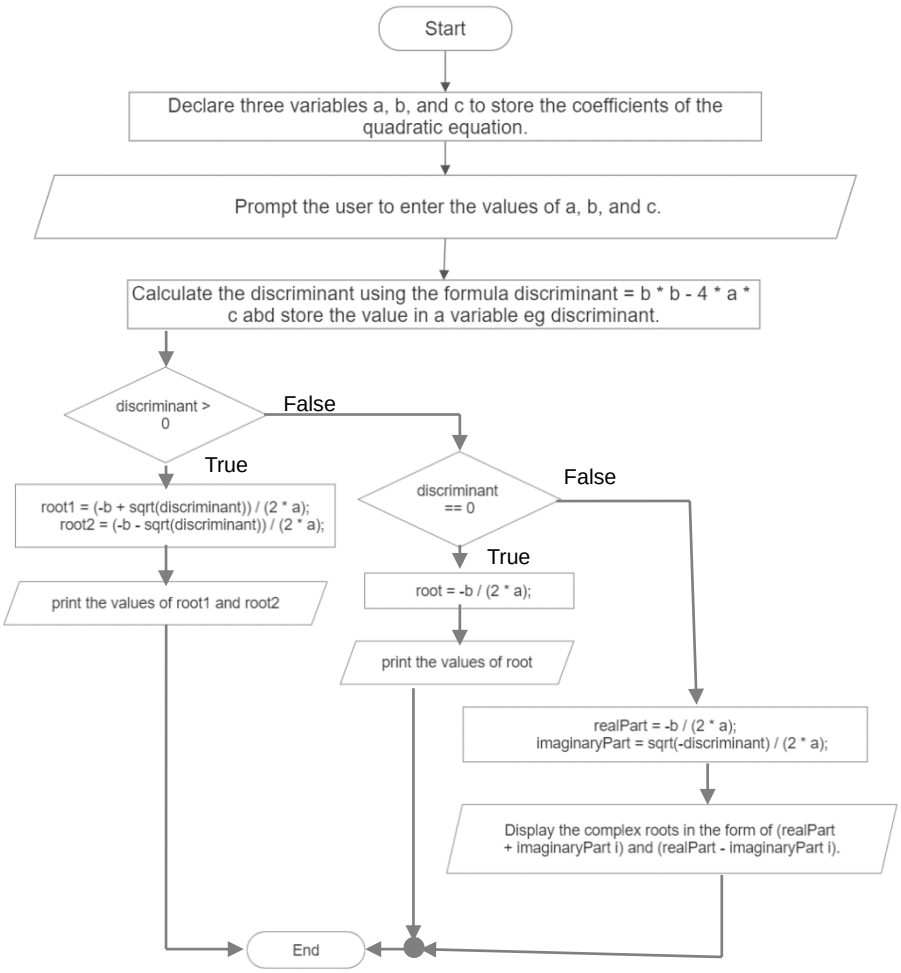
➢ Objective

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 /.

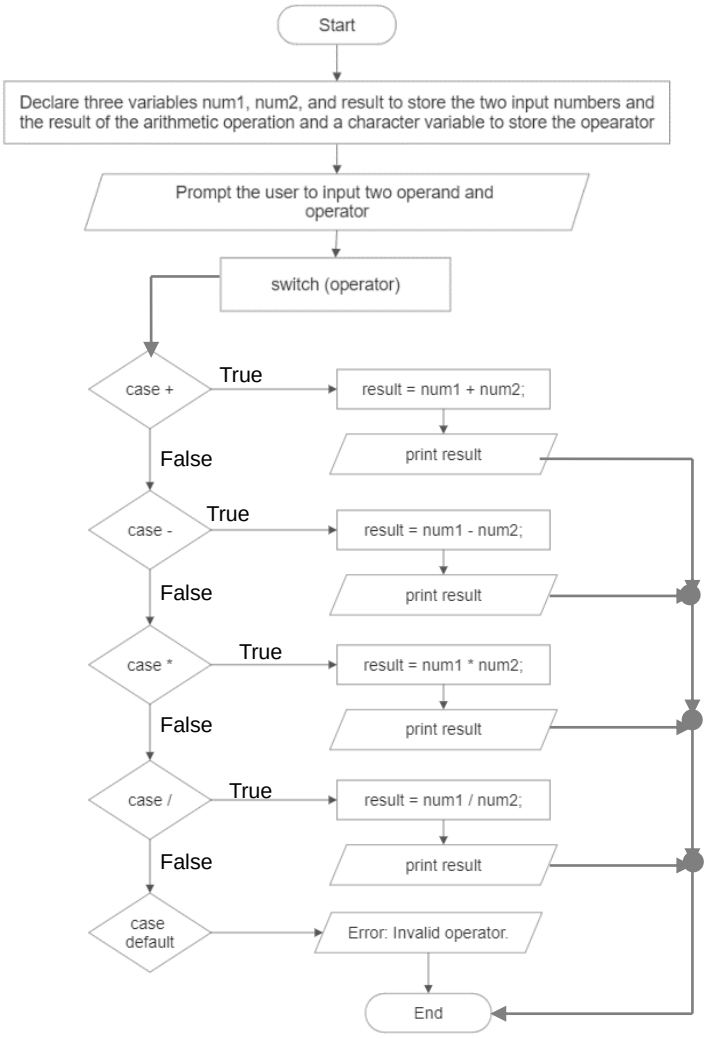
➢ Objective

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

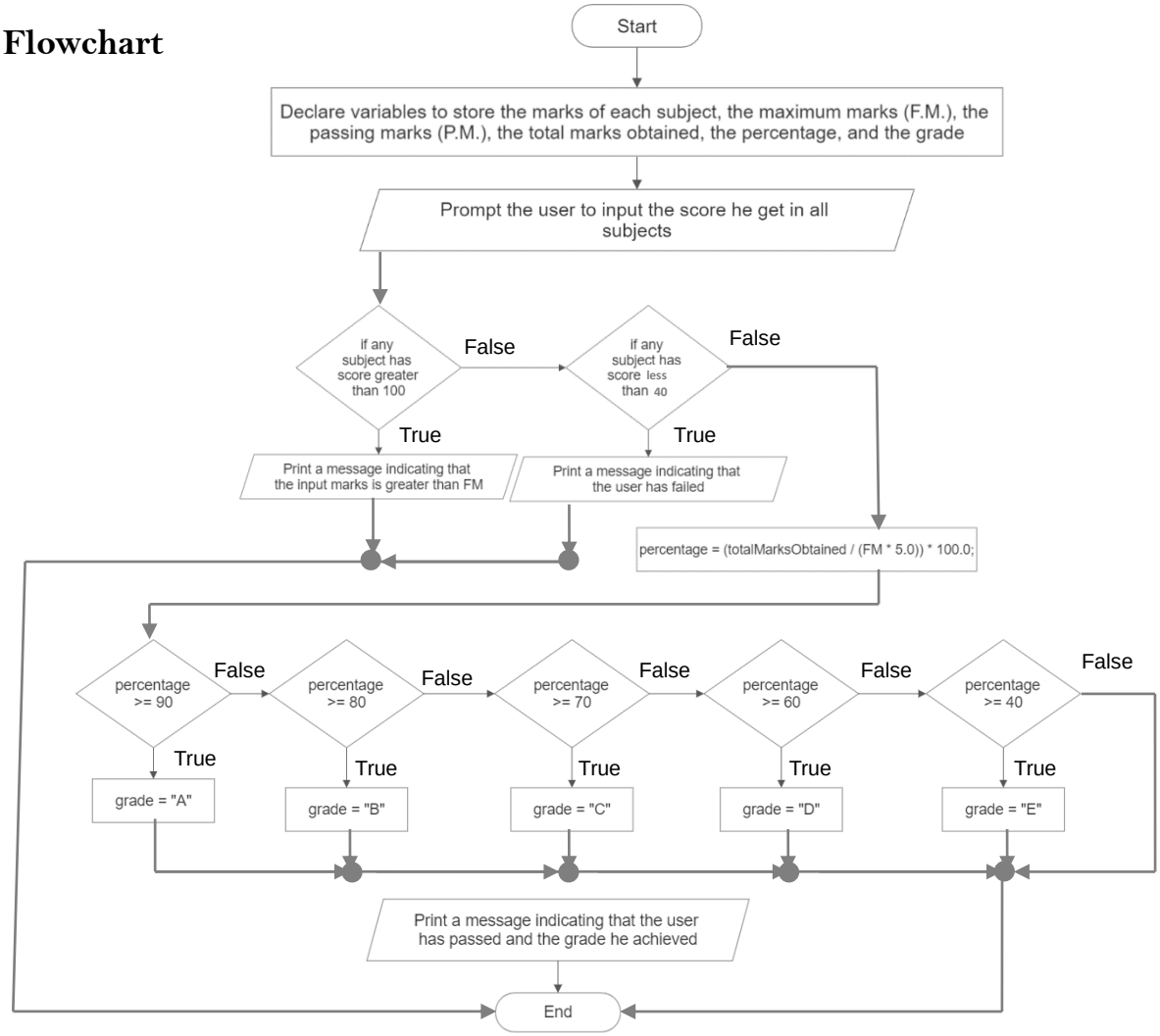
➢ Objective

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.

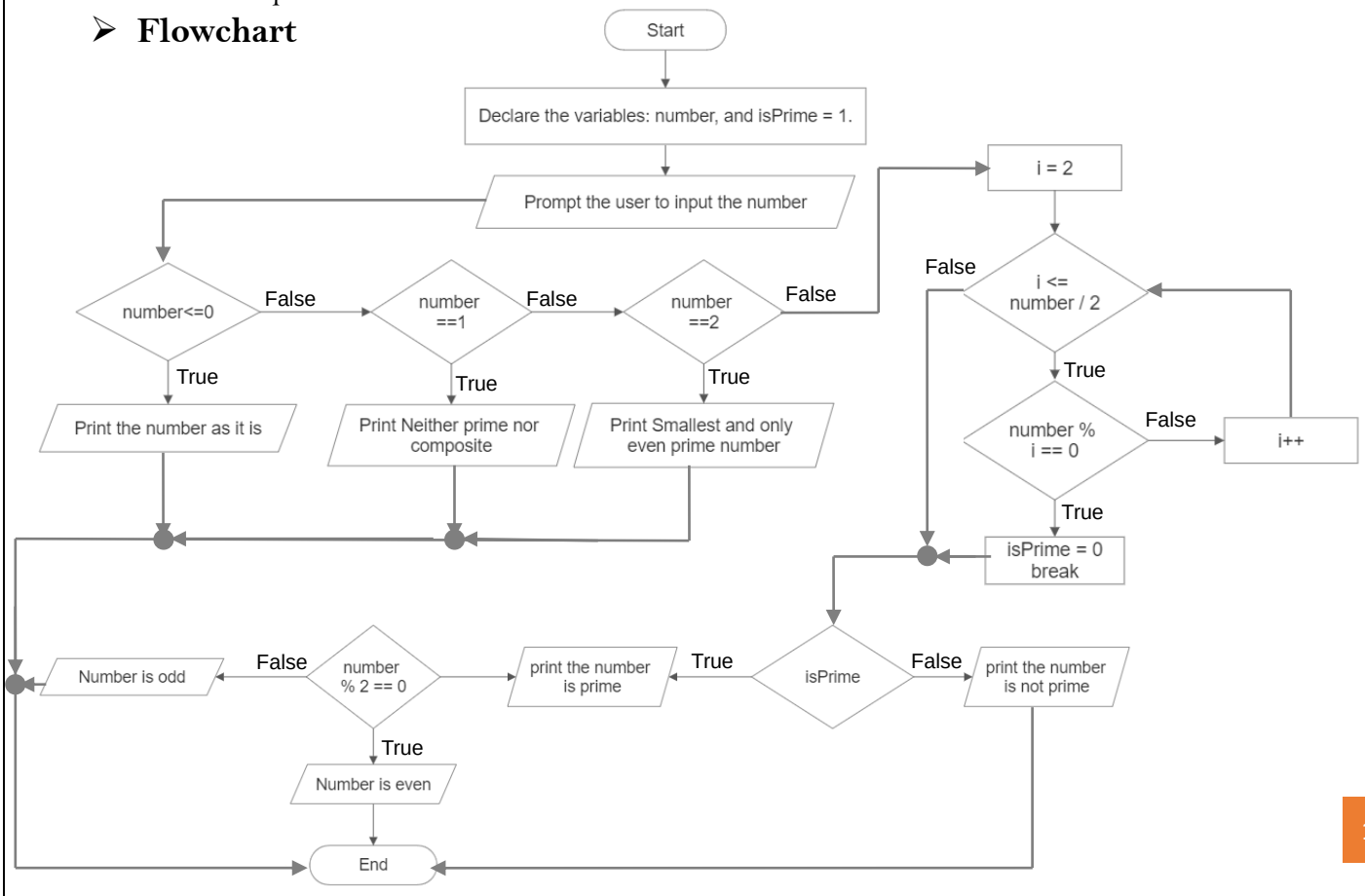
- **Objective**

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.