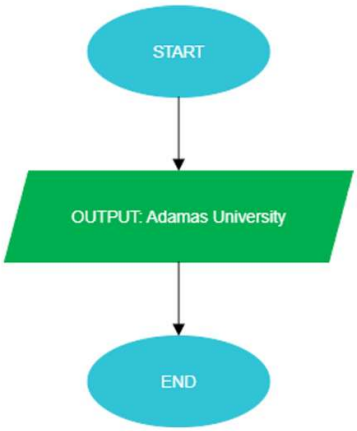
1. Print "Adamas University"

* **Goal:** Display a string on the screen.

* **Steps:**

1. Start.
2. Use the `printf` function to output the text "Adamas University".
3. Stop.

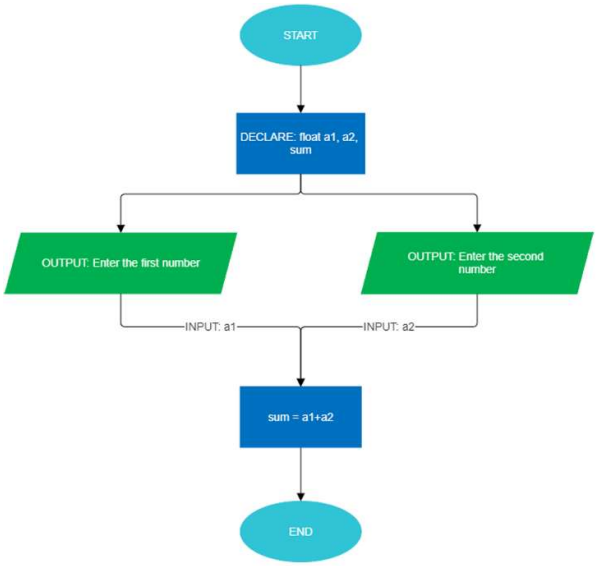


2. Add Two Float Numbers

* **Goal:** Calculate the sum of two floating-point numbers provided by the user.

* **Steps:**

1. Start.
2. Declare three float variables: `num1`, `num2`, and `sum`.
3. Prompt the user to enter the first float number and store it in `num1`.
4. Prompt the user to enter the second float number and store it in `num2`.
5. Calculate `sum = num1 + num2`.
6. Display the value of `sum`.
7. Stop.

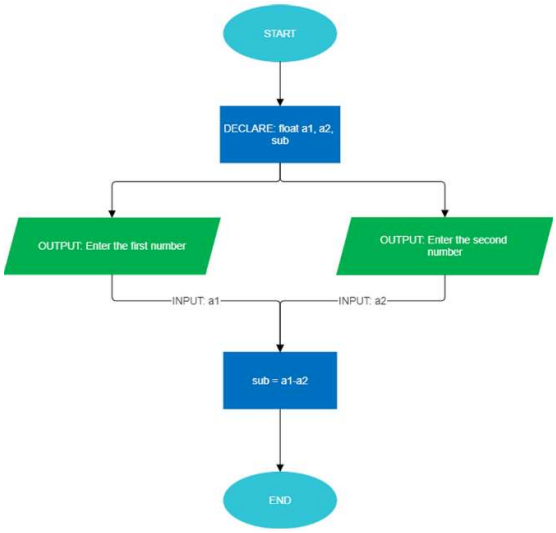


3. Subtract Two Float Numbers

* **Goal:** Calculate the difference between two floating-point numbers provided by the user.

* **Steps:**

1. Start.
2. Declare three float variables: `num1`, `num2`, and `difference`.
3. Prompt the user to enter the first and second float numbers, storing them in `num1` and `num2`.
4. Calculate `difference = num1 - num2`.
5. Display the value of `difference`.
6. Stop.

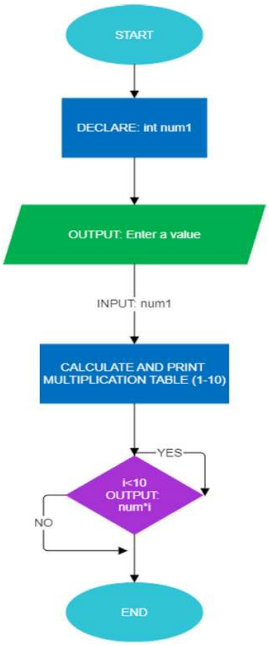


4. Print Multiplication Table Without Loop

* **Goal:** Display the multiplication table for a number without using iterative loops.

* **Steps:**

1. Start.
2. Declare an integer variable `num`.
3. Prompt the user to enter a number and store it in `num`.
4. Use a series of `printf` statements to display the multiplication results (e.g., `printf("5 x 1 = 5\n");`, `printf("5 x 2 = 10\n");`, ... up to 10).
5. Stop.

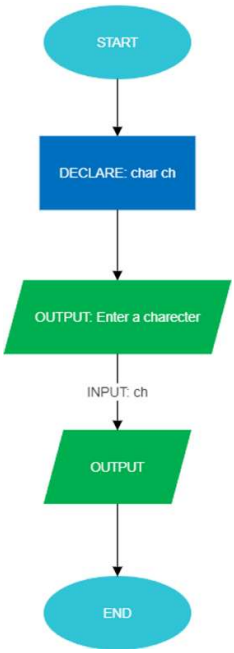


5. Find ASCII Value of a Character

* **Goal:** Find and display the ASCII value of a character input by the user.

* **Steps:**

1. Start.
2. Declare a character variable `ch`.
3. Prompt the user to enter a single character and store it in `ch`.
4. Use the format specifier `%d` with `printf` to display the integer value of the character variable `ch`.
5. Stop.

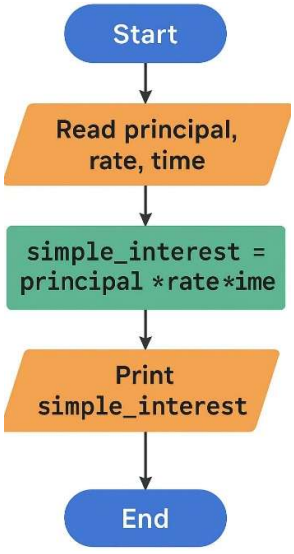


6. Calculate Simple Interest (S.I.)

* **Goal:** Compute Simple Interest based on user inputs.

* **Steps:**

1. Start.
2. Declare float variables: `principal`, `rate`, `time`, `si` (simple interest).
3. Prompt the user to enter the principal amount, rate of interest, and time period. Store them in `principal`, `rate`, and `time`.
4. Calculate `si = (principal * rate * time) / 100`.
5. Display the calculated `si`.
6. Stop.

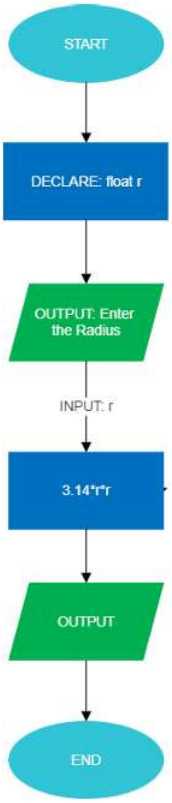


7. Calculate Area of a Circle

* **Goal:** Compute the area of a circle given its radius.

* **Steps:**

1. Start.
2. Declare float variables: `radius`, `area`.
3. Define a constant `PI` with a value of 3.14159.
4. Prompt the user to enter the radius of the circle and store it in `radius`.
5. Calculate `area = PI * radius * radius`.
6. Display the calculated `area`.
7. Stop.

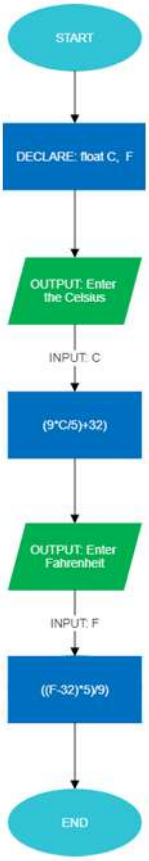


8. Convert Fahrenheit to Celsius and Vice-Versa

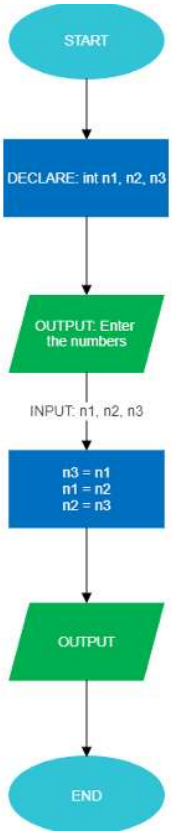
* **Goal:** Convert a temperature from Fahrenheit to Celsius and from Celsius to Fahrenheit.

* **Steps:**

1. Start.
2. Declare float variables: `celsius`, `fahrenheit`.
3. Prompt the user to enter a temperature in Celsius.
4. Calculate `fahrenheit = (celsius * 9/5) + 32` and display the result.
5. Prompt the user to enter a temperature in Fahrenheit.
6. Calculate `celsius = (fahrenheit - 32) * 5/9` and display the result.
7. Stop.



9. Swap Two Numbers Using a Third Variable

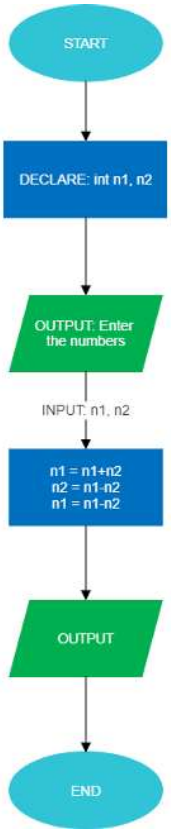


* **Goal:** Exchange the values of two variables using a temporary variable.

* **Steps:**

1. Start.
2. Declare three variables: `a`, `b`, `temp`.
3. Read values for `a` and `b` from the user.
4. Assign the value of `a` to `temp`.
5. Assign the value of `b` to `a`.
6. Assign the value of `temp` to `b`.
7. Display the new values of `a` and `b`.
8. Stop.

10. Swap Two Numbers Without Using a Third Variable

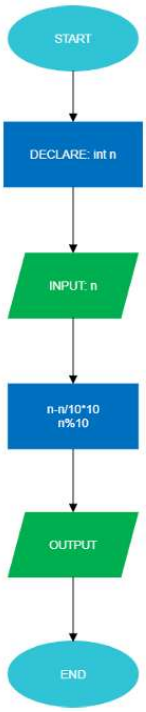


* **Goal:** Exchange the values of two variables without a temporary variable.

* **Steps:**

1. Start.
2. Declare two variables: `a` and `b`.
3. Read values for `a` and `b` from the user.
4. Calculate `a = a + b`.
5. Calculate `b = a - b`.
6. Calculate `a = a - b`.
7. Display the new values of `a` and `b`.
8. Stop.

11. Find Last Digit of an Integer

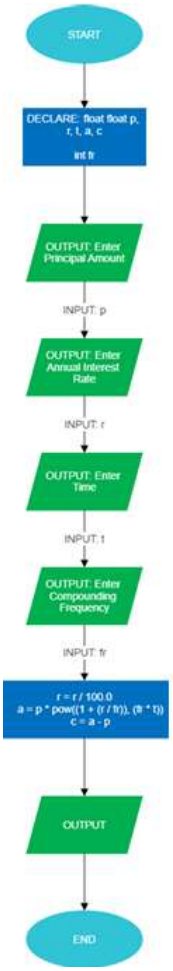


* **Goal:** Find the last digit of an integer with and without the modulus operator.

* **Steps:**

1. Start.
2. Declare an integer variable `num`.
3. Read an integer value from the user and store it in `num`.
4. **Using Modulus:** Calculate `lastDigitMod = num % 10` and display it.
5. **Without Modulus:** Calculate `lastDigitNoMod = num - (num / 10) * 10` and display it.
6. Stop.

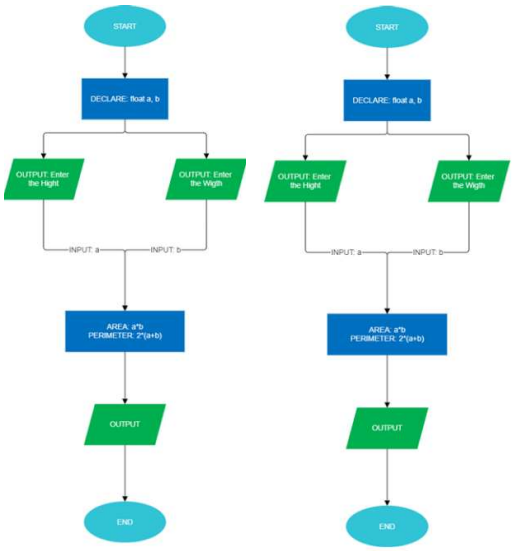
12. Calculate Compound Interest (C.I.)



* **Goal:** Compute Compound Interest.

* **Steps:**

1. Start.
2. Declare float variables: `principal`, `rate`, `time`, `ci`, `amount`.
3. Prompt the user for `principal`, `rate`, and `time`.
4. Calculate `amount = principal * pow( (1 + rate/100), time )`.
5. Calculate `ci = amount - principal`.
6. Display the calculated `ci`.
7. Stop.

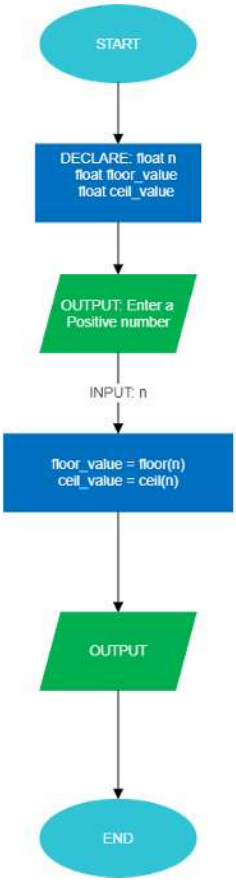


13. Find Area and Perimeter of a Rectangle

* **Goal:** Calculate the area and perimeter of a rectangle.

* **Steps:**

1. Start.
2. Declare float variables: `length`, `width`, `area`, `perimeter`.
3. Prompt the user to enter the length and width of the rectangle.
4. Calculate `area = length * width`.
5. Calculate `perimeter = 2 * (length + width)`.
6. Display both `area` and `perimeter`.
7. Stop.



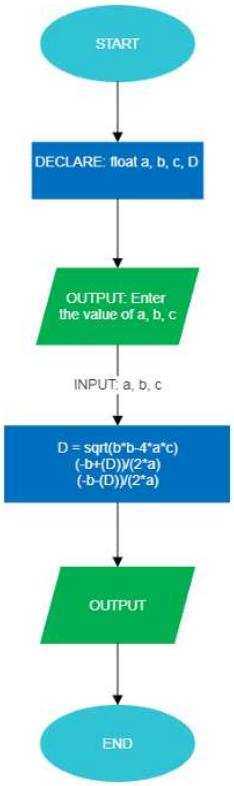
14. Print Floor and Ceiling Value

* **Goal:** Find and display the floor and ceiling of a given number.

* **Steps:**

1. Start.
2. Declare a float variable `num` and integer variables `floorVal`, `ceilVal`.
3. Read a number from the user and store it in `num`.
4. **For Positive Number:**
    * `floorVal = (int)num`
    * `ceilVal = (num > floorVal) ? floorVal + 1 : floorVal`
5. **For Negative Number:**
    * `ceilVal = (int)num` (This truncates towards zero, which is the ceiling for negatives).
    * `floorVal = (num < ceilVal) ? ceilVal - 1 : ceilVal`
6. Display `floorVal` and `ceilVal`.

7. Stop.



15. Find Roots of a Quadratic Equation

* **Goal:** Find the roots of a quadratic equation of the form $ax^2 + bx + c = 0$.

* **Steps:**

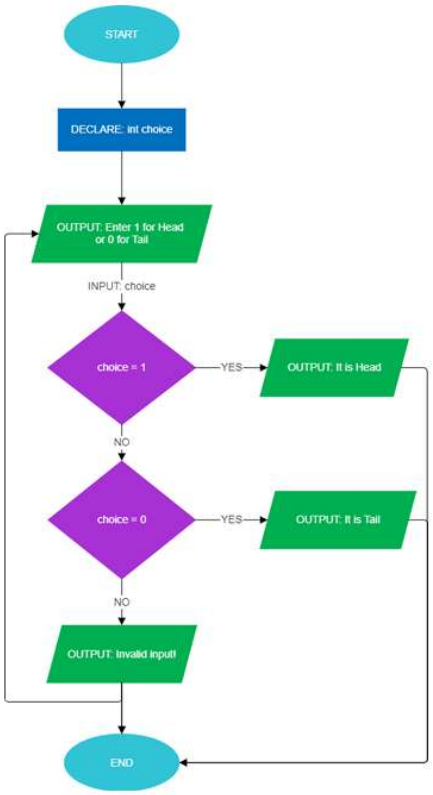
1. Start.
2. Declare float variables: `a`, `b`, `c`, `discriminant`, `root1`, `root2`, `realPart`, `imagPart`.
3. Prompt the user for coefficients `a`, `b`, and `c`.
4. Calculate `discriminant = b*b - 4*a*c`.
5. If `discriminant > 0`, calculate and print two real and distinct roots.
6. If `discriminant == 0`, calculate and print two real and equal roots.
7. If `discriminant < 0`, calculate and print two complex roots.
8. Stop.

16. Check Head or Tail (Coin Toss)

* **Goal:** Simulate a single coin toss and output the result.

* **Steps:**

1. Start.
2. Use a random number generator to produce either 0 or 1.
3. If the generated number is 0, print "Heads".
4. Else, print "Tails".
5. Stop.

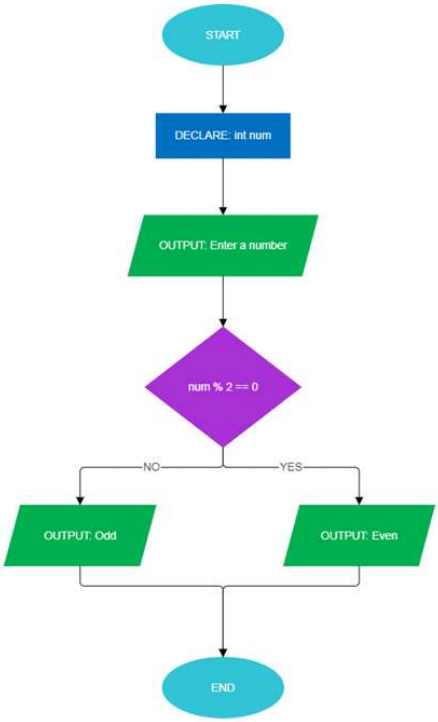


17. Check Positive or Negative (Ladder if-else)

* **Goal:** Determine if a number is positive, negative, or zero using `else if`.

* **Steps:**

1. Start.
2. Declare a variable `num`.
3. Read a number from the user.
4. If `num > 0`, print "Positive".
5. Else if `num < 0`, print "Negative".
6. Else, print "Zero".
7. Stop.

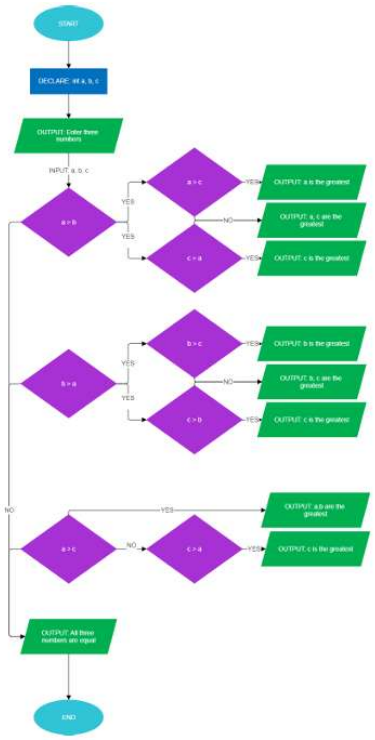


18. Find Greatest Among 3 Numbers (Ladder if-else)

* **Goal:** Find the largest of three numbers using `else if`.

* **Steps:**

1. Start.
2. Declare three variables: `a`, `b`, `c`.
3. Read values for `a`, `b`, and `c`.
4. If `a >= b` and `a >= c`, print `a` as the largest.
5. Else if `b >= a` and `b >= c`, print `b` as the largest.
6. Else, print `c` as the largest.
7. Stop.

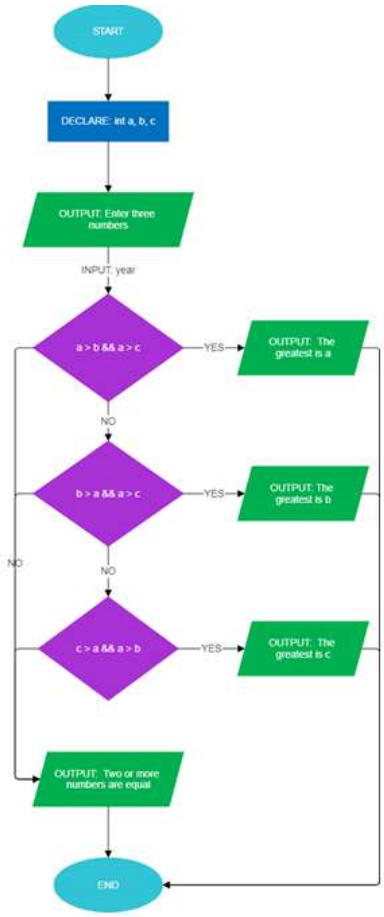


19. Check Maximum Among 3 Numbers (Nested if-else)

* **Goal:** Find the largest of three numbers using nested `if-else`.

* **Steps:**

1. Start.
2. Declare three variables: `a`, `b`, `c`.
3. Read values for `a`, `b`, and `c`.
4. If `a >= b`:
    * If `a >= c`, print `a` is largest.
    * Else, print `c` is largest.
5. Else:
    * If `b >= c`, print `b` is largest.
    * Else, print `c` is largest.
6. Stop.

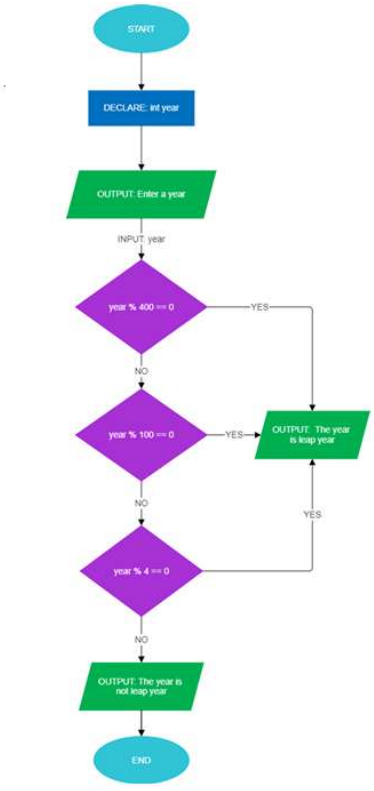


20. Check Leap Year

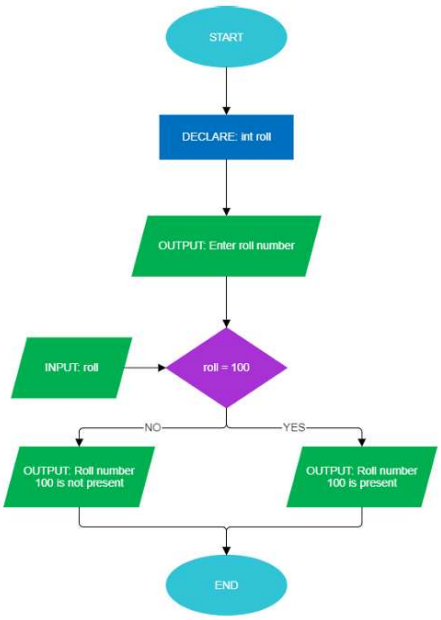
* **Goal:** Determine if a given year is a leap year.

* **Steps:**

1. Start.
2. Declare an integer variable `year`.
3. Read the year from the user.
4. If `(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)`, print "Leap Year".



5. Else, print "Not a Leap Year".
6. Stop.

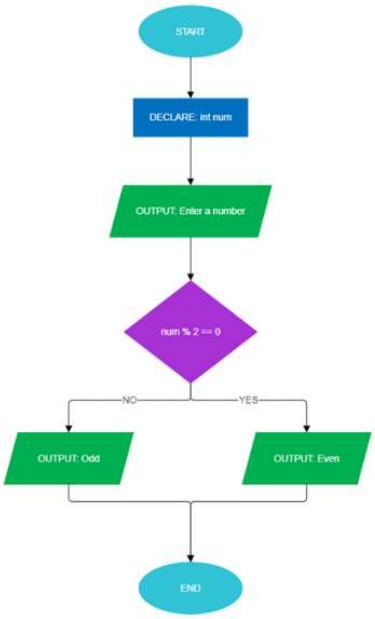


21. Check for Roll Number 100 (if statement)

* **Goal:** Check if a roll number is 100 using a simple `if` condition.

* **Steps:**

1. Start.
2. Declare an integer variable `rollNumber`.
3. Read the roll number from the user.
4. If `rollNumber == 100`, print "Roll Number 100 is present".
5. Stop.



22. Check Odd or Even

* **Goal:** Check if an integer is odd or even.

* **Steps:**

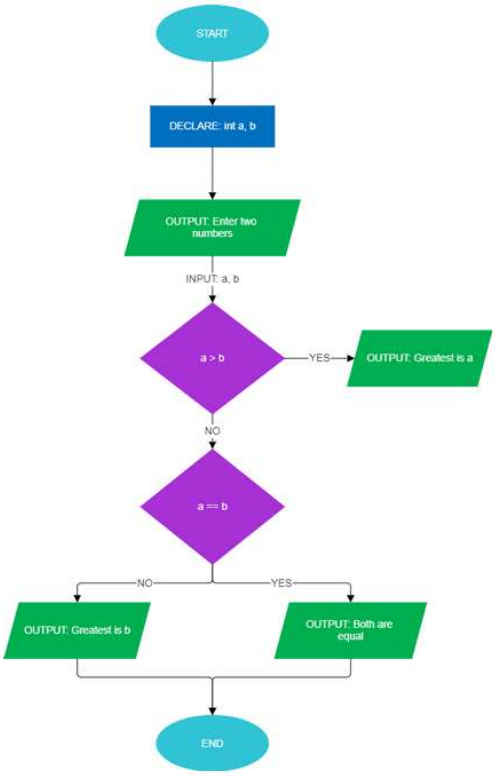
1. Start.
2. Declare an integer variable `num`.
3. Read a number from the user.
4. If `num % 2 == 0`, print "Even".
5. Else, print "Odd".
6. Stop.

23. Check Greatest Among Two Numbers

* **Goal:** Find the larger of two numbers.

* **Steps:**

1. Start.
2. Declare two variables `a` and `b`.
3. Read values for `a` and `b`.
4. If `a > b`, print `a` is greater.
5. Else if `b > a`, print `b` is greater.
6. Else, print "Numbers are equal".
7. Stop.

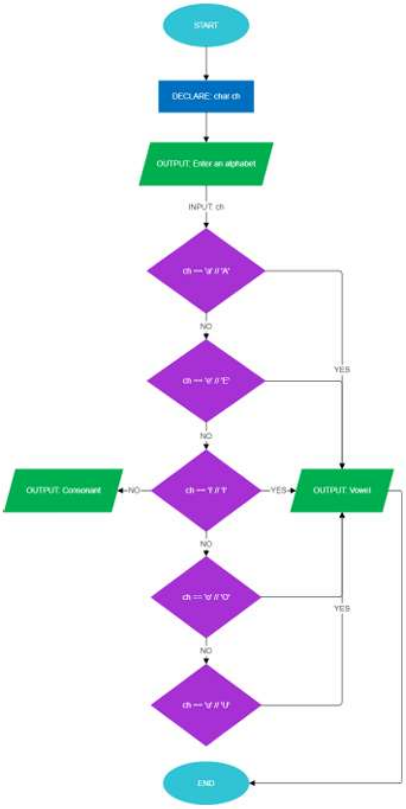


24. Check Vowel or Consonant

* **Goal:** Check if an input letter is a vowel or a consonant.

* **Steps:**

1. Start.
2. Declare a character variable `ch`.
3. Read a character from the user.
4. Convert the character to lowercase to simplify checking.
5. If `ch` is 'a', 'e', 'i', 'o', or 'u', print "Vowel".
6. Else, print "Consonant".
7. Stop.



25. Find Size of Data Types

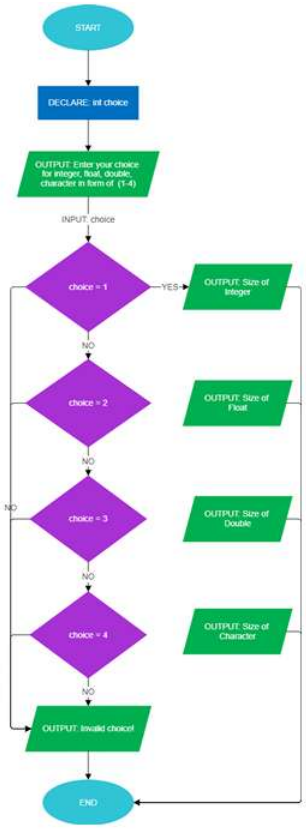
* **Goal:** Display the size (in bytes) of basic data types.

* **Steps:**

1. Start.

2. Use the `sizeof` operator inside `printf` statements to find and print the size of `int`, `float`, `double`, and `char`.

3. Stop.



26. Simple Calculator Using Switch-case

* **Goal:** Perform basic arithmetic operations based on user's choice.

* **Steps:**

1. Start.

2. Declare variables: `char operator`, `float num1`, `num2`, `result`.

3. Read the operator and two numbers from the user.

4. Use a `switch` statement on the `operator`.

5. For case '+', calculate `result = num1 + num2`.

6. For case '-', calculate `result = num1 - num2`.

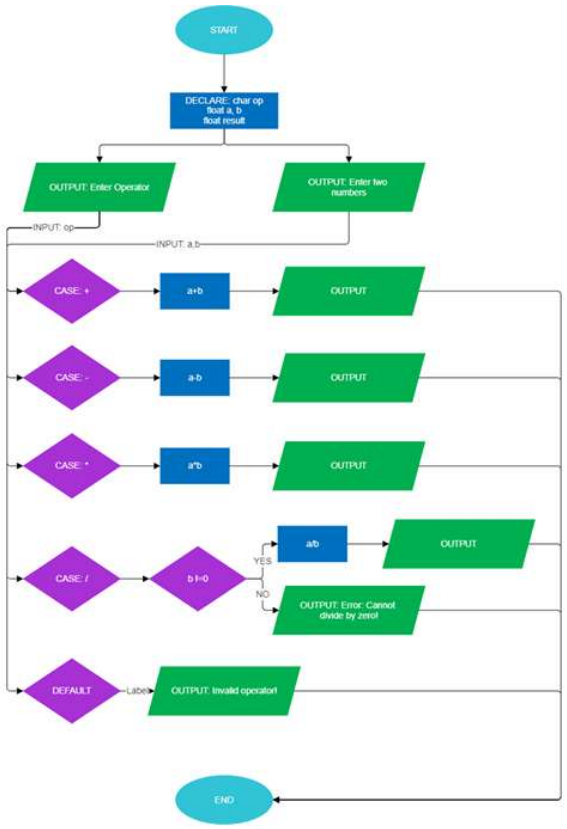
7. For case '*', calculate `result = num1 * num2`.

8. For case '/', check if `num2` is not zero, then calculate `result = num1 / num2`; else, print an error.

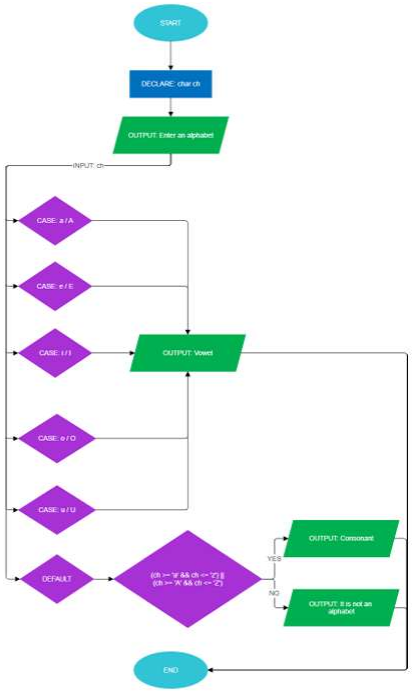
9. For any other character, print "Invalid operator".

10. Display the `result`.

11. Stop.



27. Check Vowel or Consonant Using Switch-case



* **Goal:** Determine if a character is a vowel using a `switch` statement.

* **Steps:**

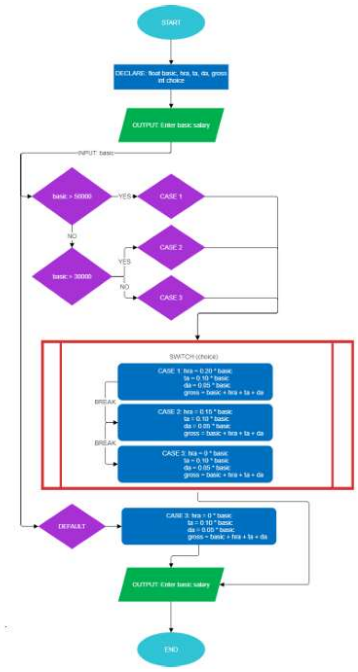
1. Start.
2. Declare a character variable `ch`.
3. Read a character from the user.
4. Convert the character to lowercase.
5. Use a `switch` statement on `ch`.
6. For cases 'a', 'e', 'i', 'o', 'u': print "Vowel".
7. For the `default` case: print "Consonant".
8. Stop.

28. Calculate Total Salary of an Employee Using Switch-case

* **Goal:** Calculate an employee's total salary based on their grade.

* **Steps:**

1. Start.
2. Declare variables: `char grade`, `float baseSalary, totalSalary, allowance`.
3. Read the employee's grade and base salary.
4. Use a `switch` on `grade` to determine the allowance percentage.
    * Case 'A': `allowance = 50% of baseSalary`
    * Case 'B': `allowance = 45% of baseSalary`
    * Case 'C': `allowance = 40% of baseSalary`
    * Default: `allowance = 30% of baseSalary`
5. Calculate `totalSalary = baseSalary + allowance`.
6. Display `totalSalary`.
7. Stop.

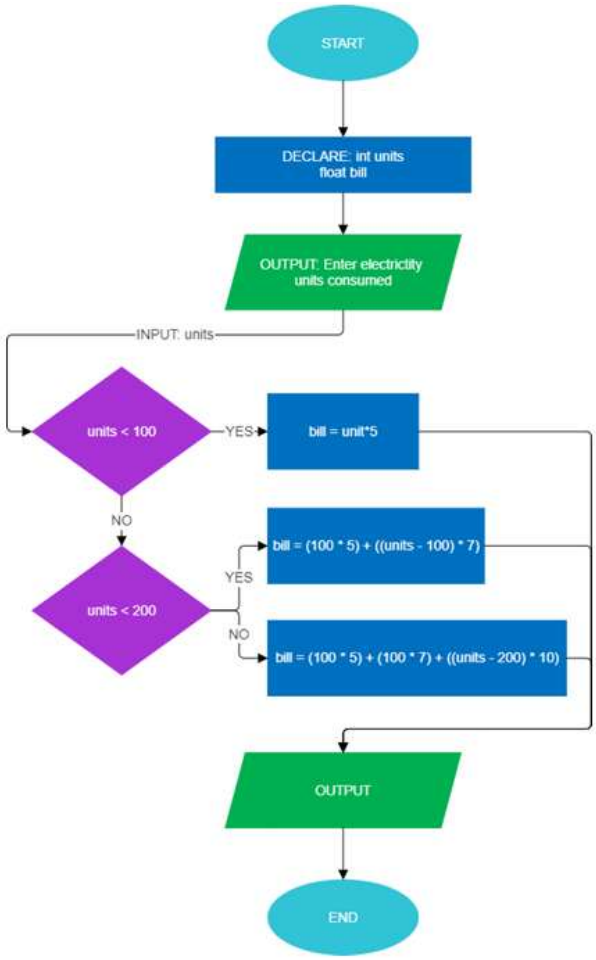


29. Calculate Electricity Bill Using Switch-case

* **Goal:** Compute electricity bill based on slabs of units consumed.

* **Steps:**

1. Start.
2. Declare `int units`, `float bill`.
3. Read the number of units consumed.
4. Use a `switch` on the slab (e.g., `units/100` or a series of `if` checks to determine a case).
    * Case 1 (0-100 units): `bill = units * 1.50`
    * Case 2 (101-200 units): `bill = 100*1.50 + (units-100)*2.50`
    * Case 3 (201-300 units): `bill = 100*1.50 + 100*2.50 + (units-200)*4.00`
    * Default (>300 units): `bill = ... + (units-300)*5.50`
5. Display the total `bill`.
6. Stop.



30. Display Grade Based on Marks Using Switch-case

* **Goal:** Assign a grade to a student based on their marks.

* **Steps:**

1. Start.

2. Declare `int marks`.

3. Read the marks from the user.

4. Calculate an integer `grade = marks / 10`. This converts marks to a case (e.g., 90-100 becomes 9 or 10).

5. Use a `switch` on `grade`:

    * Case 10:

    * Case 9: Print "Grade A"

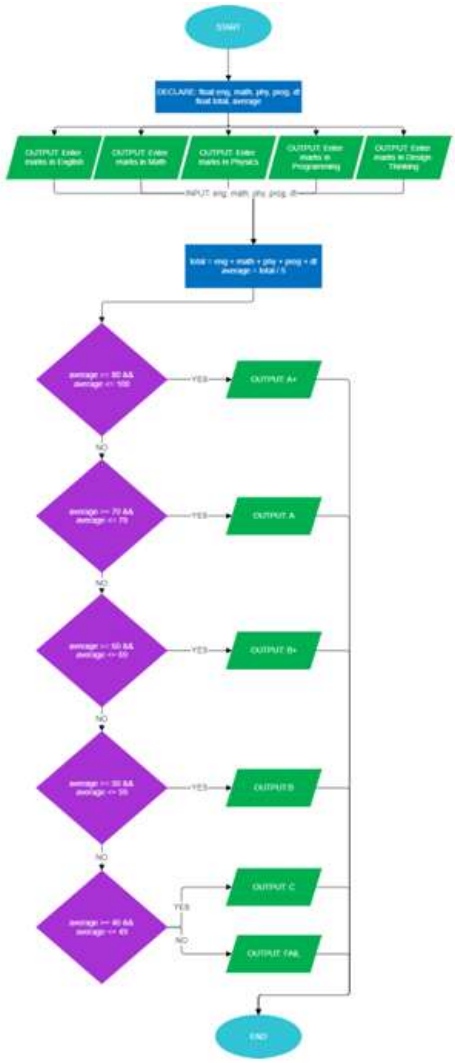
    * Case 8: Print "Grade B"

    * Case 7: Print "Grade C"

    * Case 6: Print "Grade D"

    * Default: Print "Grade F"

6. Stop.

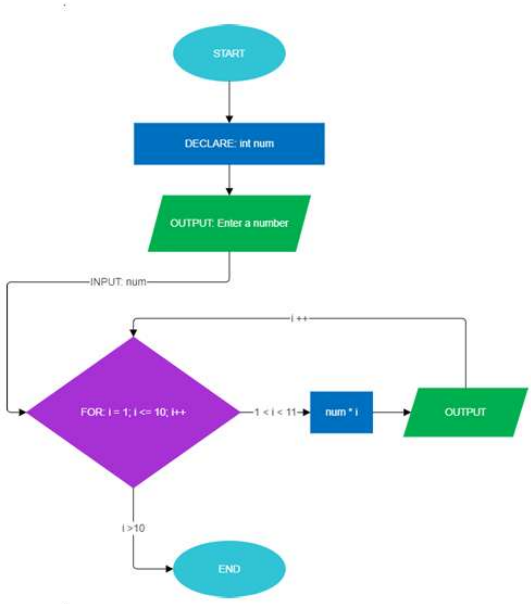


31. Print Multiplication Table Using For Loop

* **Goal:** Print the multiplication table for a given number using a `for` loop.

* **Steps:**

1. Start.
2. Declare integers `num`, `i`.
3. Read the number for which the table is to be printed.
4. Use a `for` loop with `i` from 1 to 10.
5. Inside the loop, print `num * i`.
6. Stop.

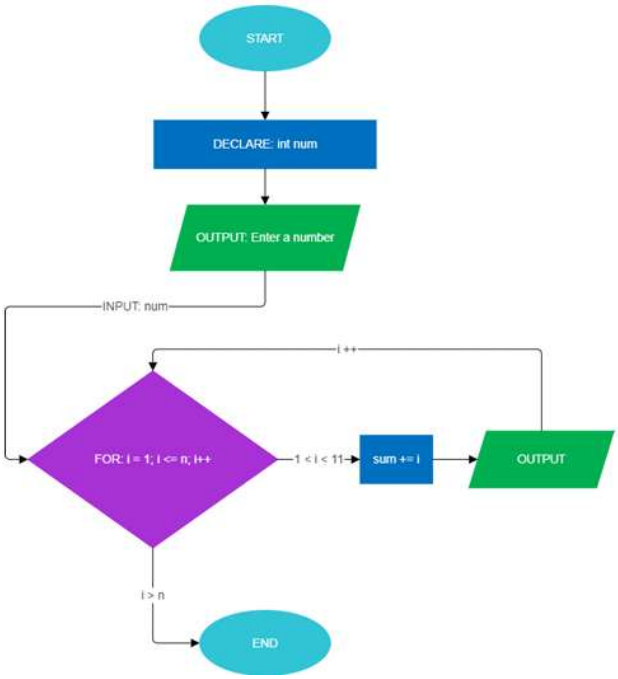


32. Find Sum of N Natural Numbers

* **Goal:** Calculate the sum of the first N natural numbers.

* **Steps:**

1. Start.
2. Declare integers `n`, `i`, `sum = 0`.
3. Read the value of `n`.
4. Use a `for` loop with `i` from 1 to `n`.
5. In each iteration, add `i` to `sum`.
6. After the loop, display the value of `sum`.
7. Stop.

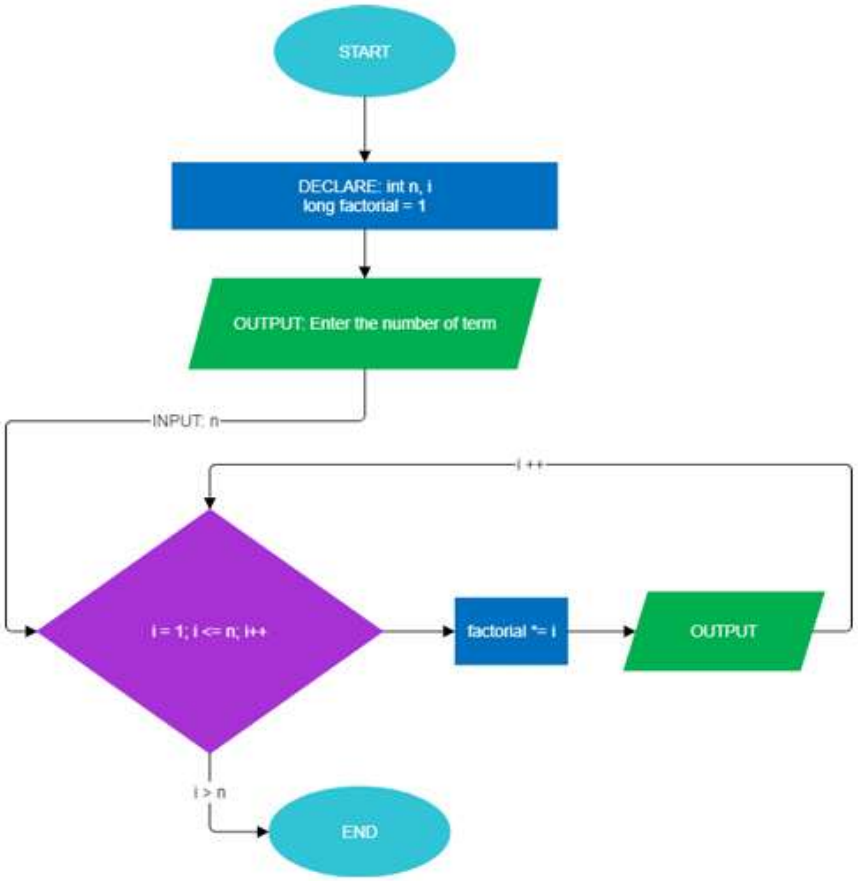


33. Print Factorial of a Number Using For Loop

* **Goal:** Compute the factorial of a non-negative integer.

* **Steps:**

1. Start.
2. Declare integers `n`, `i`, `factorial = 1`.
3. Read the value of `n`.
4. If `n < 0`, print an error. Else, proceed.
5. Use a `for` loop with `i` from 1 to `n`.
6. In each iteration, multiply `factorial` by `i`.
7. After the loop, display `factorial`.
8. Stop.

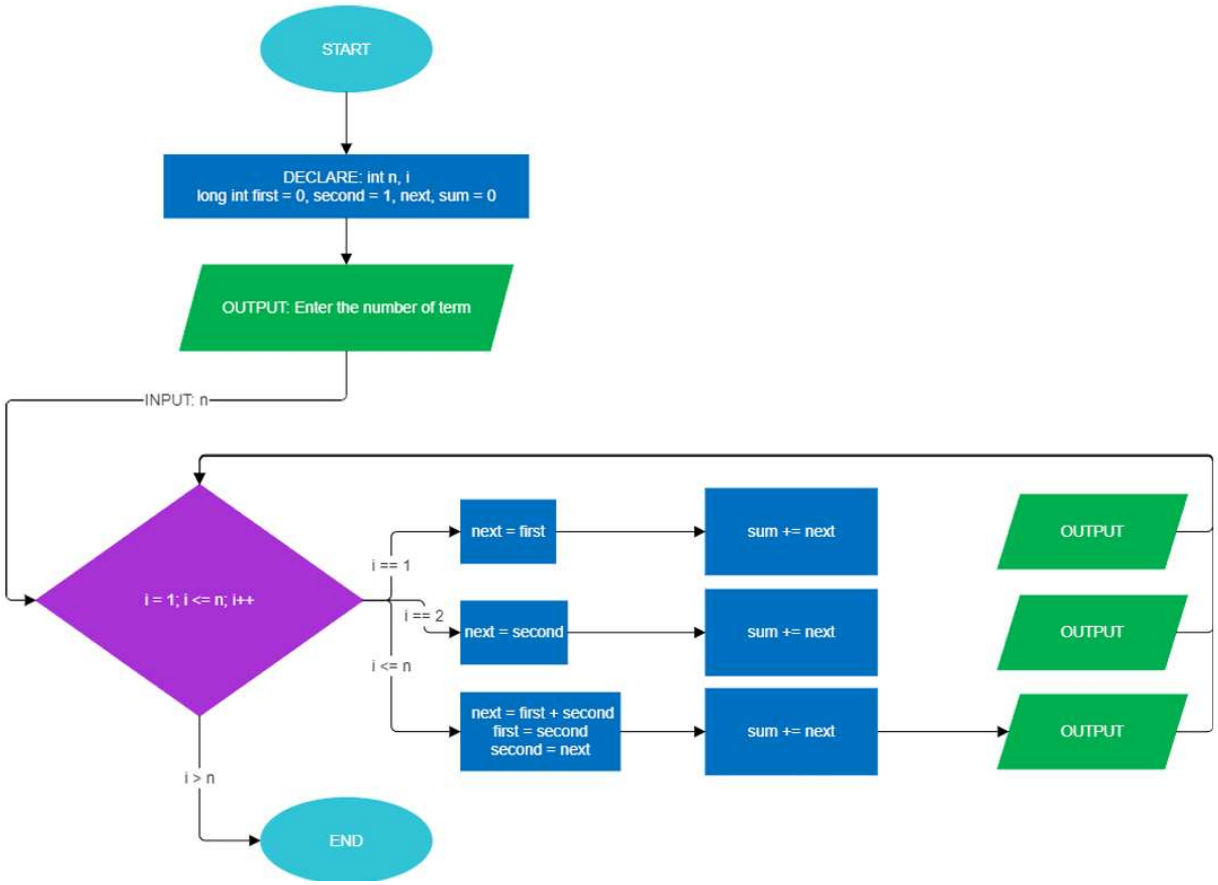


34. Print Fibonacci Series Up to N Terms

* **Goal:** Print the Fibonacci sequence up to a specified number of terms.

* **Steps:**

1. Start.
2. Declare integers `n`, `i`, `t1 = 0`, `t2 = 1`, `nextTerm`.
3. Read the number of terms `n`.
4. Print the first two terms `t1` and `t2`.
5. Use a `for` loop with `i` from 3 to `n`.
6. In each iteration:
    * Calculate `nextTerm = t1 + t2`.
    * Print `nextTerm`.
    * Update `t1 = t2` and `t2 = nextTerm`.
7. Stop.



35. Print All Prime Numbers Between a Range

* **Goal:** Find and print all prime numbers within a given range.

* **Steps:**

1. Start.
2. Declare integers `low`, `high`, `i`, `j`, `isPrime`.
3. Read the lower and upper bounds of the range (`low` and `high`).
4. Use an outer `for` loop with `i` from `low` to `high`.
5. For each `i`, if it is less than 2, skip it. Else, set `isPrime = 1`.
6. Use an inner `for` loop with `j` from 2 to `i/2`.
7. If `i` is divisible by `j`, set `isPrime = 0` and break the inner loop.
8. After the inner loop, if `isPrime` is still 1, print `i`.
9. Stop.