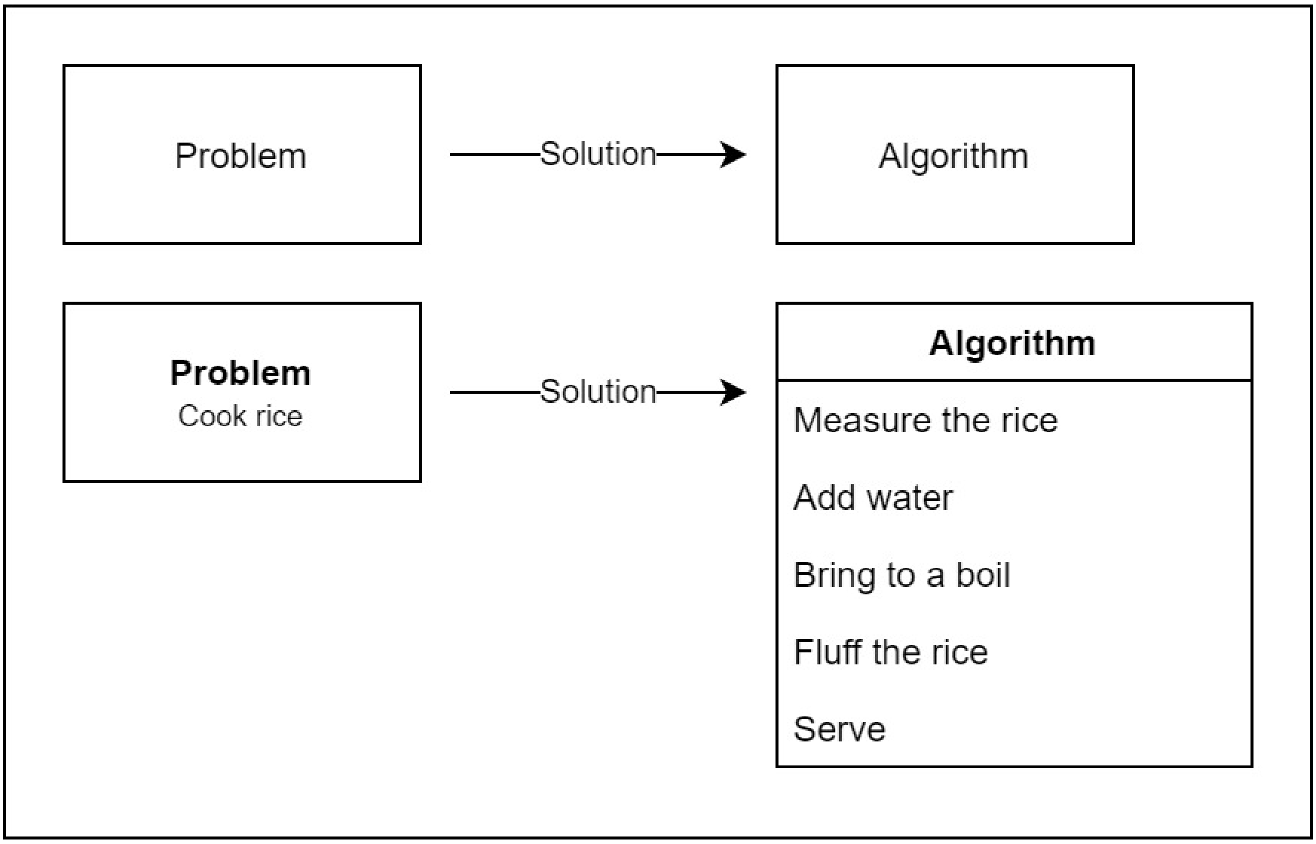
What is an Algorithm?

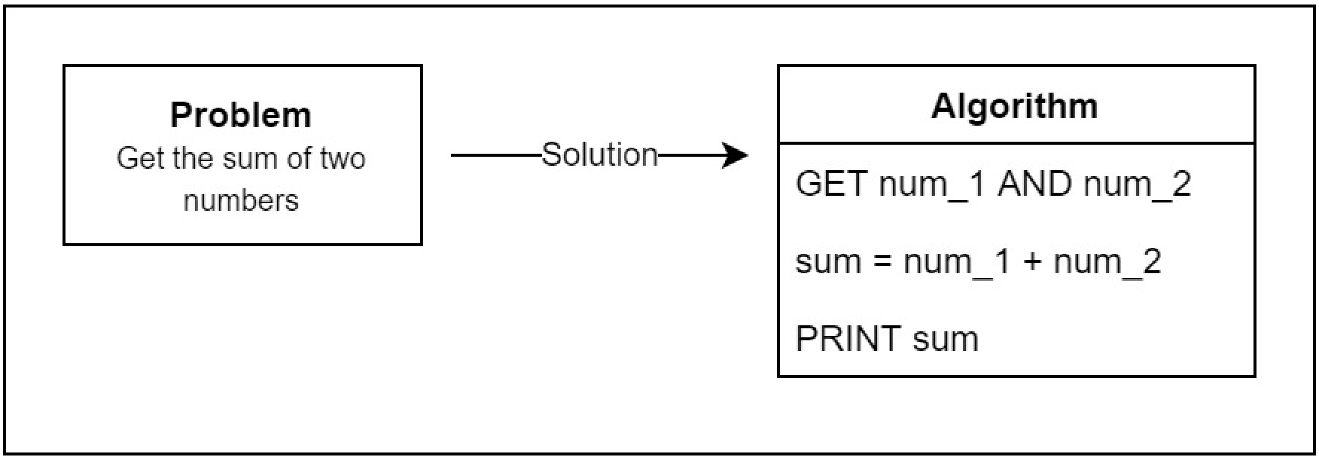
An algorithm is like a ***recipe*** for solving a specific problem. It consists of a well-defined ***set of instructions*** that guide the computer through a series of steps to produce the desired output. Just like a recipe helps you cook a delicious meal, an algorithm enables a computer to perform tasks efficiently and accurately.



Algorithm Example: Adding Two Numbers

To better grasp the concept, let's take a look at a simple algorithm: adding two numbers together.

1. First, we take two number inputs.
2. Next, we add these numbers using the "+" operator.
3. Finally, we display the result, which is the sum of the two numbers.



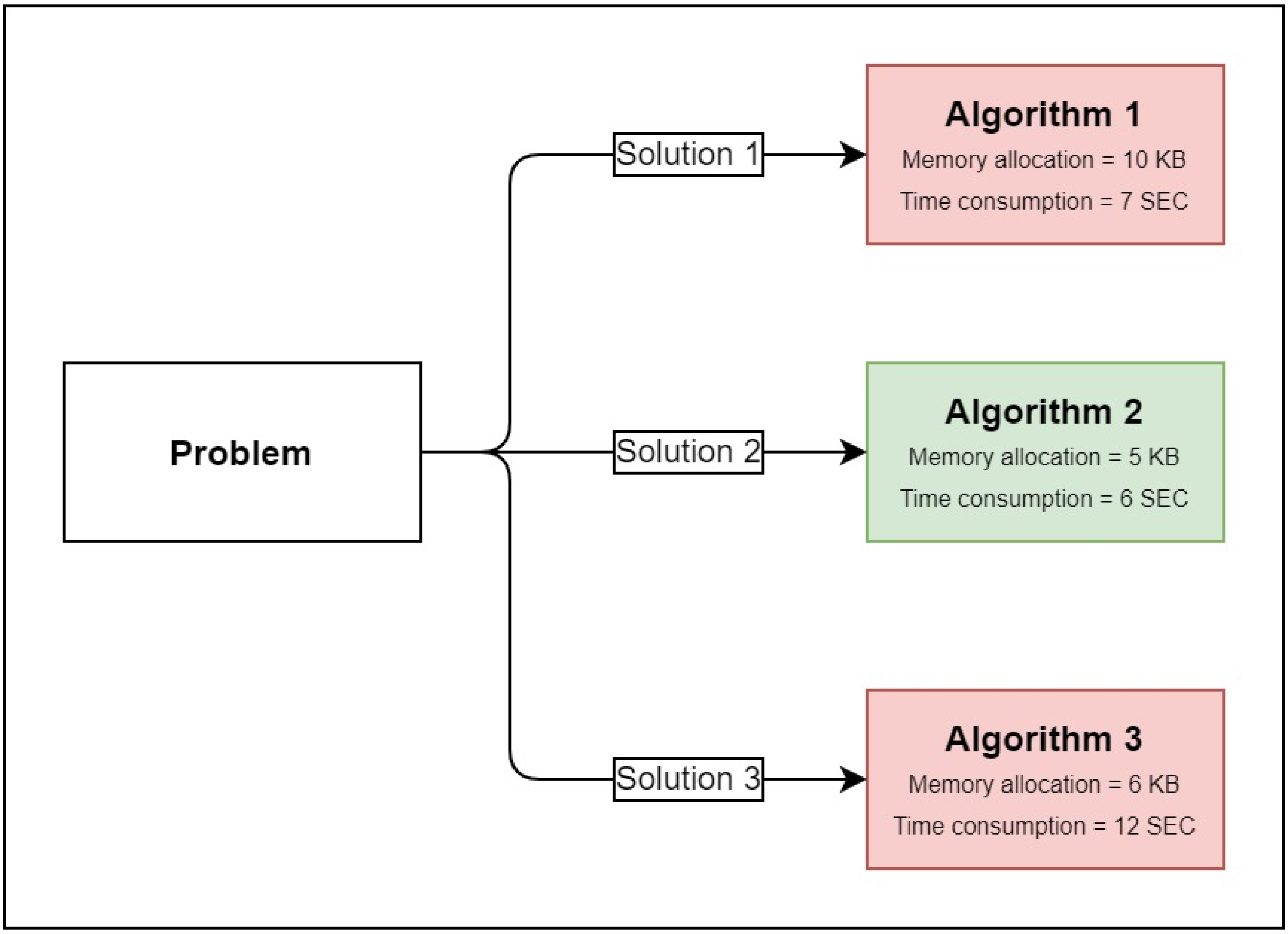
```
GET num_1 AND num_2
sum = num_1 + num_2
PRINT sum
```

That's it! This is a basic algorithm, but it demonstrates the core elements of input, process, and output, which are fundamental to any algorithm.

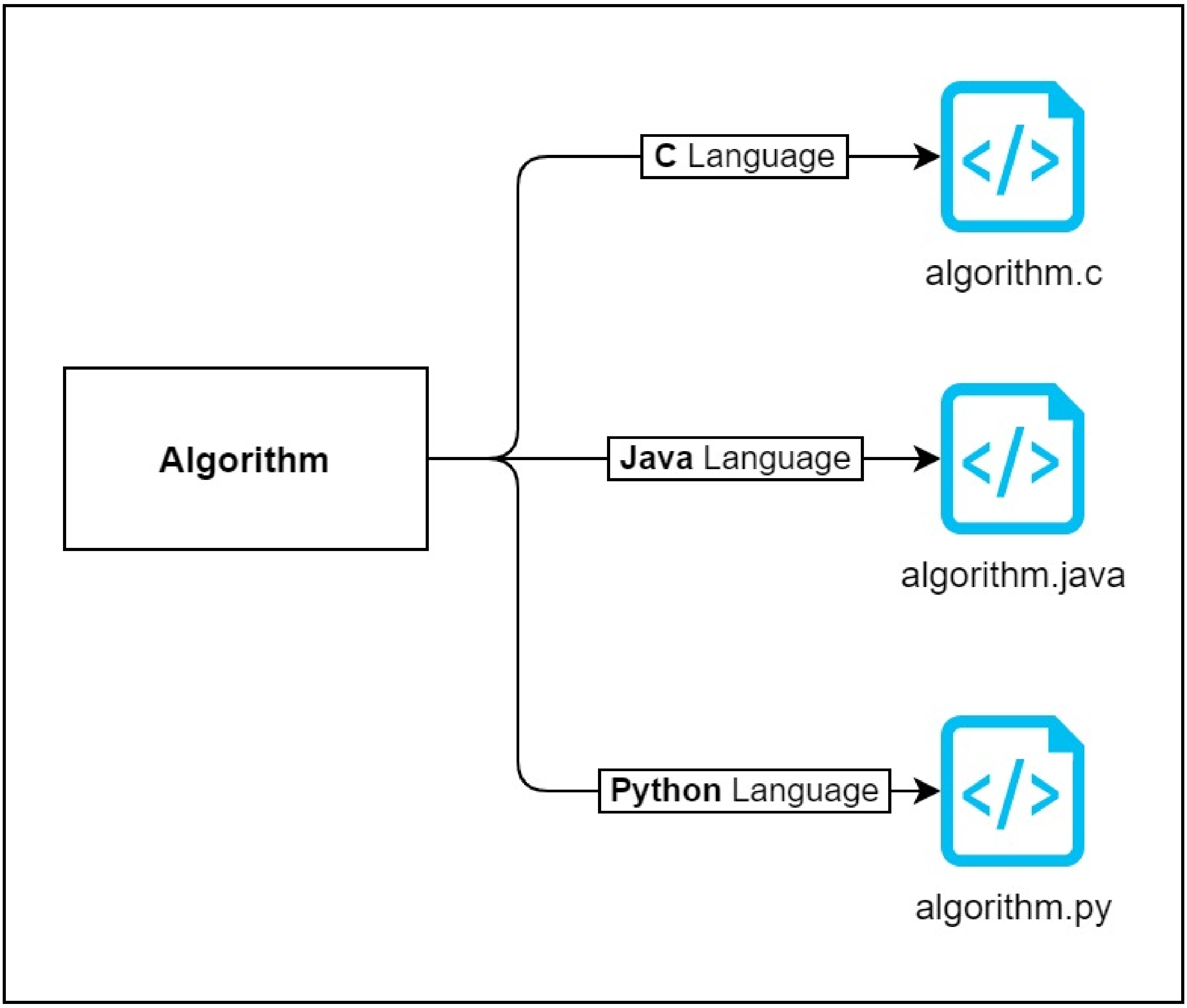
Qualities of a Good Algorithm

A good algorithm possesses specific qualities that ensure its effectiveness and usefulness. Let's explore these qualities:

1. **Precisely Defined Input and Output:** A well-defined algorithm clearly states what inputs it expects and what output it will produce. This clarity is essential for accurate problem-solving.
2. **Clear and Unambiguous Steps:** Each step in an algorithm should be crystal clear and unambiguous. Ambiguity can lead to confusion, making the algorithm difficult to follow.
3. **Effectiveness in Problem Solving:** A good algorithm is efficient and effective among various ways to tackle a problem. It should achieve the desired result in a reasonable amount of ***time*** and ***resources***.



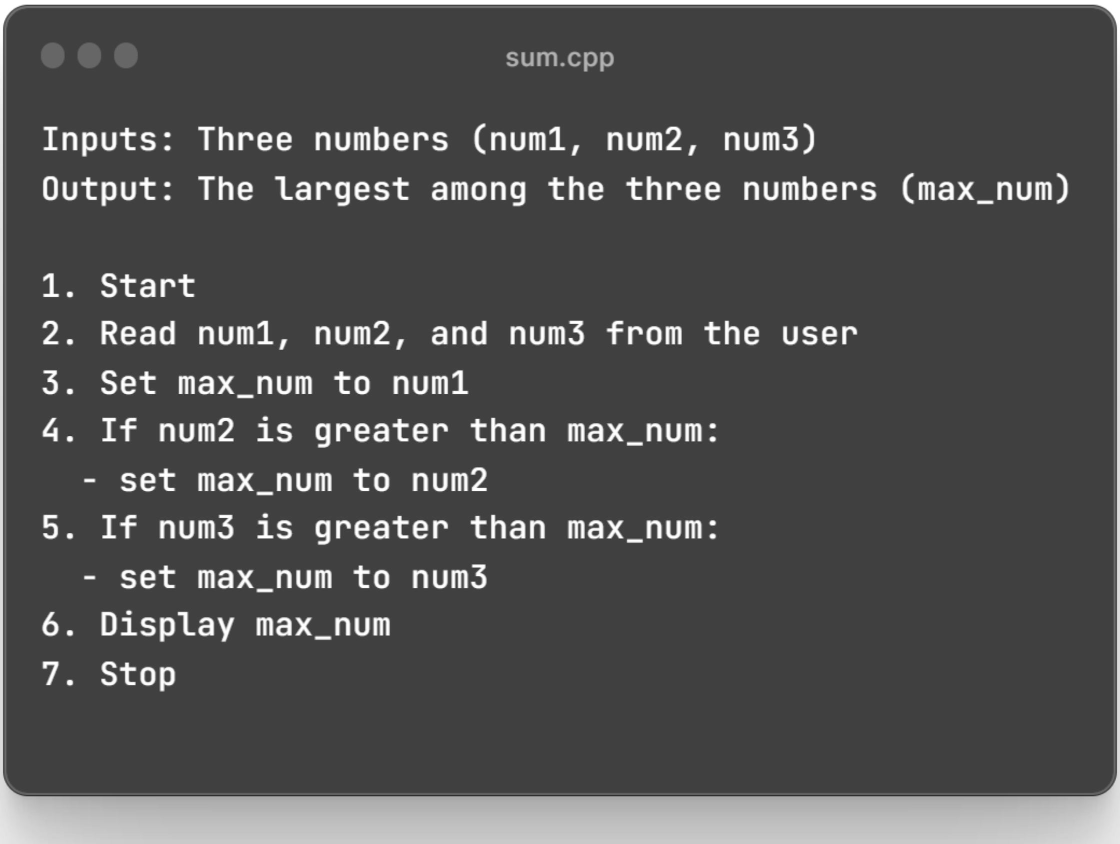
4. **Universal Applicability:** Algorithms should not be tied to a ***specific programming language***. They should be written in a way that allows them to be implemented in various programming languages, making them versatile and adaptable.



Algorithm Examples

Now that we understand the qualities of a good algorithm, let's explore some more examples:

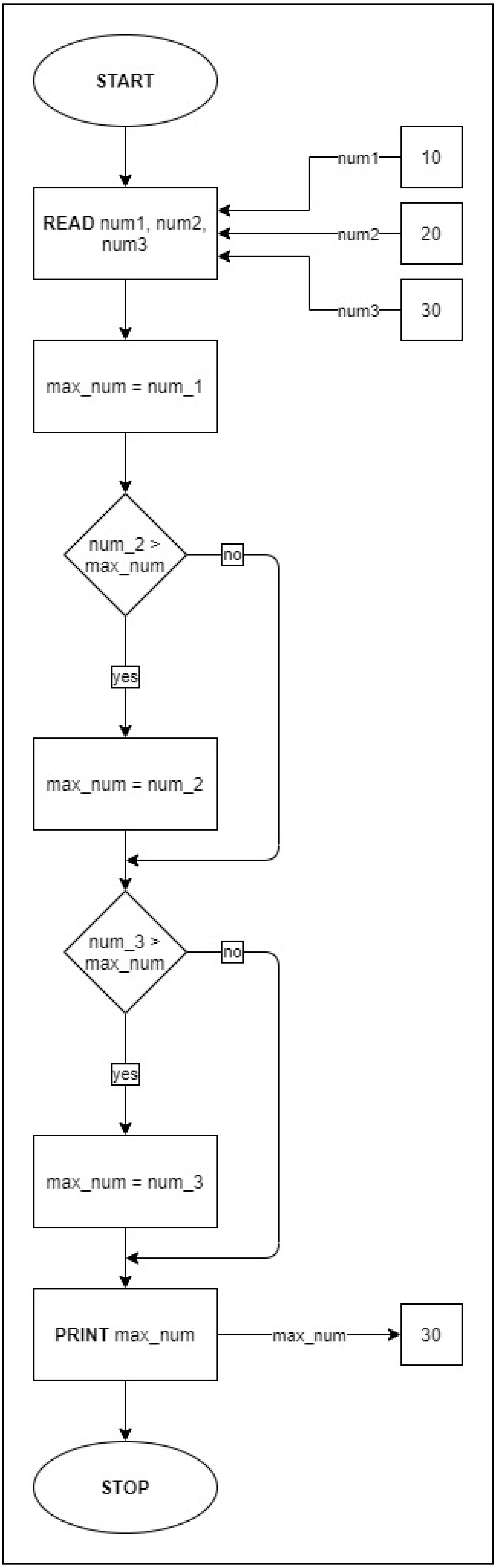
1. **Finding the Largest Among Three Numbers:** An algorithm that takes three numbers as input and determines the largest among them.



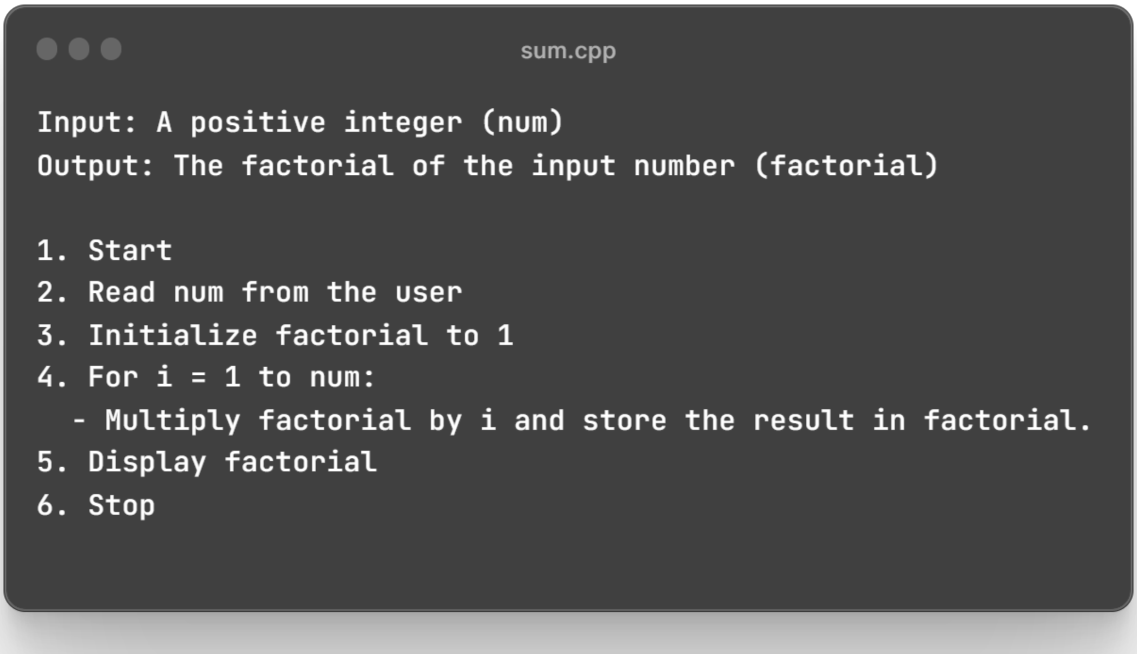
```
Inputs: Three numbers (num1, num2, num3)
Output: The largest among the three numbers (max_num)

1. Start
2. Read num1, num2, and num3 from the user
3. Set max_num to num1
4. If num2 is greater than max_num:
  - set max_num to num2
5. If num3 is greater than max_num:
  - set max_num to num3
6. Display max_num
7. Stop
```

Example:


START
READ num1, num2, num3
num1
10
num2
20
num3
30
max_num = num_1
num_2 > max_num
no
yes
max_num = num_2
num_3 > max_num
no
yes
max_num = num_3
PRINT max_num
max_num
30
STOP

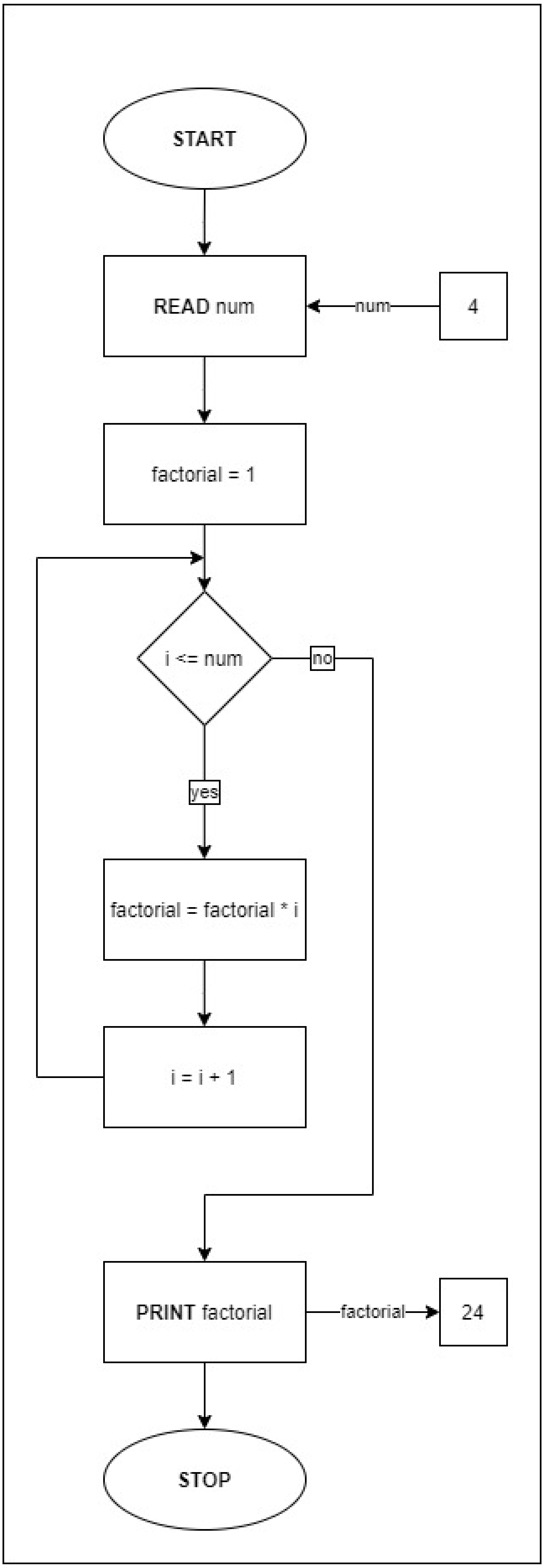
2. **Calculating the Factorial:** An algorithm to find the factorial of a given number, which is the product of all positive integers up to that number.



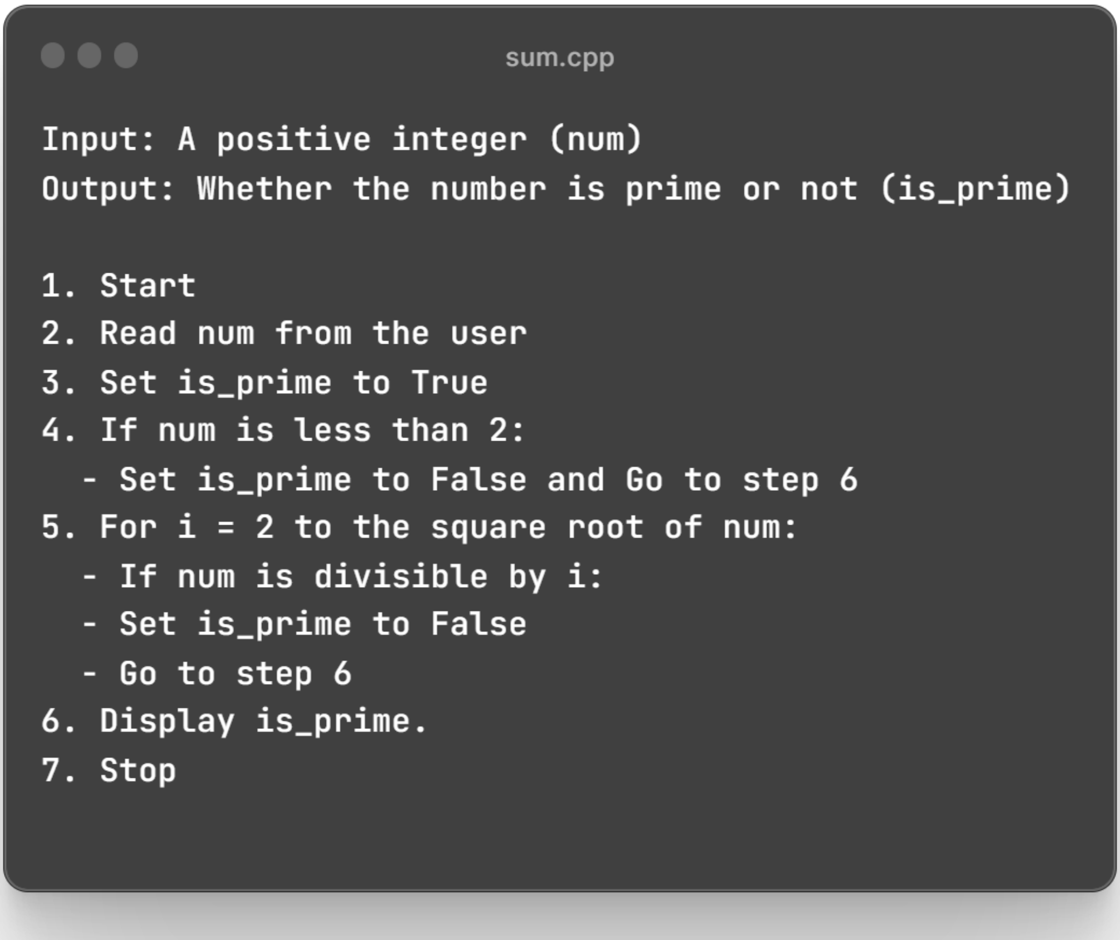
```
Input: A positive integer (num)
Output: The factorial of the input number (factorial)

1. Start
2. Read num from the user
3. Initialize factorial to 1
4. For i = 1 to num:
  - Multiply factorial by i and store the result in factorial.
5. Display factorial
6. Stop
```

Example:



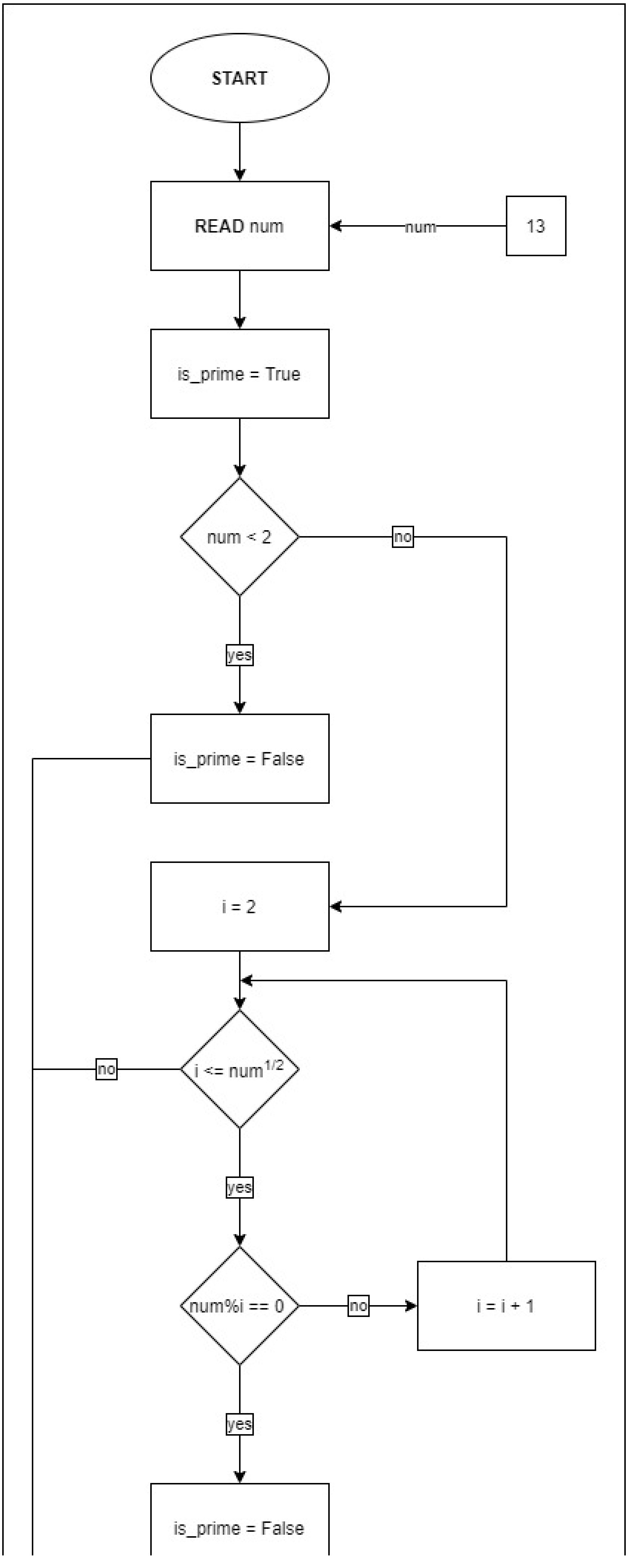
3. **Checking for Prime Numbers:** An algorithm to determine whether a given number is a prime number or not.

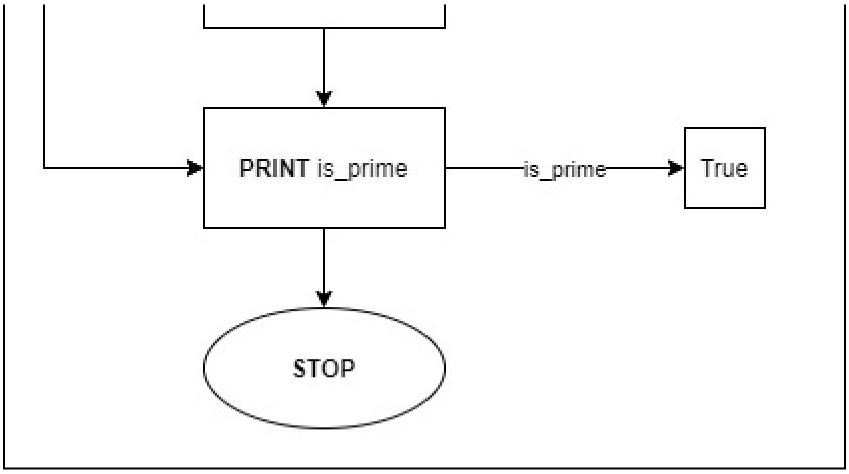


```
Input: A positive integer (num)
Output: Whether the number is prime or not (is_prime)

1. Start
2. Read num from the user
3. Set is_prime to True
4. If num is less than 2:
   - Set is_prime to False and Go to step 6
5. For i = 2 to the square root of num:
   - If num is divisible by i:
   - Set is_prime to False
   - Go to step 6
6. Display is_prime.
7. Stop
```

Example:


START
READ num
num
13
is_prime = True
num < 2
no
yes
is_prime = False
i = 2
no
i <= num$^{1/2}$
yes
num%i == 0
no
i = i + 1
yes
is_prime = False


PRINT is_prime
is_prime
True
STOP