Table of Contents

## 0.1 FunctionDef findMax(arr[], size)

# 1 Function: findMax(int arr[], int size)

## 1.1 Overview

The findMax function iterates through an array of integers to find and return the largest value.

## 1.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| arr | int[] | An array of integers from which the maximum value will be found. |
| size | int | The total number of elements in the arr array. |

## 1.3 Description

This function provides a straightforward way to determine the maximum value within an integer array.

The logic begins by initializing a local variable max with the value of the first element in the array, arr[0]. It then enters a for loop that iterates through the remaining elements of the array, starting from the second element at index 1 up to the element at index size - 1.

Inside the loop, each element arr[i] is compared to the current max value. If arr[i] is greater than max, the max variable is updated with the value of arr[i]. This process ensures that after each comparison, max holds the largest value encountered so far.

Once the loop completes, the function returns the final max value, which is the largest integer in the entire array.

## 1.4 Usage Notes

* The input array arr should not be empty. The function accesses arr[0] without checking if size is 0, which will lead to undefined behavior if the array is empty.
* The size parameter must accurately reflect the number of elements in the array to prevent out-of-bounds memory access.

**Output Example**: The function returns a single integer representing the highest value found in the array. For an input array {10, 5, 45, 12, 8}, the function would return 45.

## 1.5 Example

#include <stdio.h>  
  
// Assuming the findMax function is defined here or included  
int findMax(int arr[], int size) {  
 int max = arr[0];  
 for (int i = 1; i < size; i++) {  
 if (arr[i] > max)  
 max = arr[i];  
 }  
 return max;  
}  
  
int main() {  
 int numbers[] = {10, 5, 45, 12, 8};  
 int array\_size = 5;  
   
 // Call the findMax function  
 int result = findMax(numbers, array\_size);  
   
 printf("The maximum value is: %d\n", result);  
   
 return 0;  
}

**Output:**

The maximum value is: 45

## 1.6 FunctionDef stringLength(str[])

# 2 Function: stringLength

## 2.1 Overview

The stringLength function calculates the length of a null-terminated string by counting the number of characters before the null terminator.

## 2.2 parameters

* **str** char[]: A character array representing the C-style string whose length is to be measured.

## 2.3 Description

This function provides a manual implementation for determining the length of a C-style string. It operates based on the principle that strings in C are terminated by a special null character, \0.

The function initializes an integer counter, length, to 0. It then enters a while loop that iterates through the input character array str. The loop continues as long as the character at the current index, str[length], is not the null terminator \0.

Inside the loop, the length variable is incremented with each character processed. This variable serves a dual purpose: it acts as the index for accessing characters in the array and as a counter for the string’s length.

Once the loop encounters the null terminator, the condition str[length] != '\0' becomes false, and the loop terminates. The function then returns the final value of length, which represents the total number of characters in the string, excluding the null terminator itself.

// The core logic of the function  
int length = 0;  
while (str[length] != '\0') {  
 length++;  
}  
return length;

## 2.4 Usage Notes

* The input array str must be properly null-terminated. If the null terminator \0 is missing, the function will continue to read beyond the bounds of the array, leading to undefined behavior and a potential program crash.
* The length returned by the function does not include the final null terminator character. For example, the string "abc" has a length of 3.

**Output Example**: The function returns an integer representing the number of characters in the string. For the input "Hello", the return value would be 5.

## 2.5 Example

#include <stdio.h>  
  
// Function definition  
int stringLength(char str[]) {  
 int length = 0;  
 while (str[length] != '\0') {  
 length++;  
 }  
 return length;  
}  
  
int main() {  
 char myString[] = "Hello, World!";  
 int len = stringLength(myString);  
 printf("The length of the string \"%s\" is %d\n", myString, len);  
  
 char emptyString[] = "";  
 int emptyLen = stringLength(emptyString);  
 printf("The length of the string \"%s\" is %d\n", emptyString, emptyLen);  
   
 return 0;  
}

**Output:**

The length of the string "Hello, World!" is 13  
The length of the string "" is 0

## 2.6 FunctionDef isPrime(n)

# 3 Function: isPrime(int n)

## 3.1 Overview

The isPrime function determines whether a given integer is a prime number.

## 3.2 parameters

* n: An int representing the integer to be checked for primality.

## 3.3 Description

This function implements an efficient trial division algorithm to test if a number n is prime.

The logic proceeds as follows: 1. First, it handles the base cases. By definition, prime numbers must be greater than 1. The function checks if n <= 1. If this condition is true, it immediately returns false. 2. If n is greater than 1, the function enters a for loop that iterates from i = 2 up to the square root of n. The condition i \* i <= n is a common optimization; if n has a divisor larger than its square root, it must also have a corresponding divisor smaller than its square root. Therefore, we only need to check for divisors up to sqrt(n). 3. Inside the loop, it checks if n is perfectly divisible by the current iterator i using the modulo operator (n % i == 0). If a divisor is found, n is not a prime number, and the function immediately returns false. 4. If the loop completes without finding any divisors, it means n is not divisible by any integer between 2 and its square root. In this case, the number is prime, and the function returns true.

## 3.4 Usage Notes

* The function returns a boolean value (true or false) which requires including the <stdbool.h> header in C.
* The input number n must be an integer. The function correctly handles non-positive integers by returning false.
* This implementation is efficient for moderately sized integers but may be slow for very large numbers.

**Output Example**: A possible return value for a prime number input.

true

## 3.5 Example

The following C code demonstrates how to use the isPrime function.

#include <stdio.h>  
#include <stdbool.h>  
  
// Assuming the isPrime function is defined here or included  
bool isPrime(int n) {  
 if (n <= 1)  
 return false;  
 for (int i = 2; i \* i <= n; i++) {  
 if (n % i == 0)  
 return false;  
 }  
 return true;  
}  
  
int main() {  
 int number1 = 17;  
 int number2 = 18;  
  
 if (isPrime(number1)) {  
 printf("%d is a prime number.\n", number1);  
 } else {  
 printf("%d is not a prime number.\n", number1);  
 }  
  
 if (isPrime(number2)) {  
 printf("%d is a prime number.\n", number2);  
 } else {  
 printf("%d is not a prime number.\n", number2);  
 }  
  
 return 0;  
}

**Output:**

17 is a prime number.  
18 is not a prime number.

## 3.6 FunctionDef main

# 4 Function: main

## 4.1 Overview

The main function serves as the entry point for the program, demonstrating the usage of three distinct utility functions: findMax, stringLength, and isPrime.

## 4.2 parameters

This function does not accept any command-line arguments.

## 4.3 Description

The main function executes a sequence of operations to showcase the functionality of other helper functions within the program.

1. **Finding the Maximum Number**:
   * An integer array named nums is initialized with the values {5, 12, 3, 19, 7}.
   * The variable size is calculated to determine the number of elements in the nums array. This is achieved by dividing the total size of the array (sizeof(nums)) by the size of a single element (sizeof(nums[0])).
   * The findMax function is called with nums and size as arguments.
   * The returned maximum value is then printed to the console using printf.

* int nums[] = {5, 12, 3, 19, 7};  
  int size = sizeof(nums) / sizeof(nums[0]);  
  printf("Maximum number: %d\n", findMax(nums, size));

1. **Calculating String Length**:
   * A character array (string) named name is initialized with the value "Prateek".
   * The stringLength function is called with name as the argument.
   * The returned length of the string is printed to the console.

* char name[] = "Prateek";  
  printf("Length of '%s': %d\n", name, stringLength(name));

1. **Checking for a Prime Number**:
   * An integer variable number is initialized with the value 17.
   * The isPrime function is called with number.
   * An if-else statement checks the boolean result from isPrime. A message is printed to the console indicating whether 17 is a prime number or not.

* int number = 17;  
  if (isPrime(number))  
   printf("%d is a prime number.\n", number);  
  else  
   printf("%d is not a prime number.\n", number);

Finally, the function returns 0, signaling that the program has executed successfully.

## 4.4 Usage Notes

* This main function is designed as a driver to test and demonstrate other functions.
* For this code to compile and run, the functions findMax, stringLength, and isPrime must be defined elsewhere in the project.
* The output is sent to the standard output stream (typically the console).
* The <stdio.h> header is required for the printf function.

**Output Example**: The function will print the results of the function calls directly to the console.

## 4.5 Example

To run this code, you would compile it with a C compiler (like GCC) and then execute the resulting binary. The main function is the program’s entry point and is not called directly by user code.

**Compilation and Execution:**

# Assuming the full code is in a file named program.c  
gcc program.c -o program  
./program

**Output:**

Maximum number: 19  
Length of 'Prateek': 7  
17 is a prime number.