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 integer array to find and return its largest element.

## 1.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| arr | int[] | An array of integers in which to find the maximum value. |
| size | int | The number of elements in the arr array. |

## 1.3 Description

This function identifies the maximum value within an integer array. It begins by assuming the first element of the array, arr[0], is the maximum and assigns it to a local variable max.

The function then enters a for loop that iterates through the rest of the array elements, starting from the second element (i = 1) up to the element at index size - 1. In each iteration, it compares the current element arr[i] with the value stored in max.

If arr[i] is greater than max, the function updates max to the value of arr[i]. This process ensures that max always holds the largest value encountered so far in the traversal. After the loop completes, the function returns the final value of max.

## 1.4 Usage Notes

* The array arr should not be empty. The function accesses arr[0] without checking if size is greater than 0. Passing an empty array (with size as 0) will result in undefined behavior.
* The size parameter must accurately represent the number of elements in the array to prevent reading from memory outside the array’s bounds.
* The function works correctly with arrays containing positive, negative, and zero values.

**Output Example**: A single integer representing the largest value in the array.

## 1.5 Example

#include <stdio.h>  
  
// The function to be documented  
int findMax(int arr[], int size) {  
 if (size <= 0) {  
 // Handle empty or invalid size case, e.g., return a specific value or handle error  
 return -1; // Or some other error indicator  
 }  
 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, 100, -2, 50, 99};  
 int array\_size = sizeof(numbers) / sizeof(numbers[0]);  
   
 int max\_value = findMax(numbers, array\_size);  
   
 printf("The maximum value is: %d\n", max\_value);  
   
 return 0;  
}

**Output:**

The maximum value is: 100

## 1.6 FunctionDef stringLength(str[])

# 2 Function: stringLength(char str[])

## 2.1 Overview

The stringLength function calculates the length of a null-terminated string, excluding the final null character.

## 2.2 parameters

* **str** (char[]): A character array representing the C-style string. This string must be properly terminated with a null character (\0).

## 2.3 Description

This function provides a manual implementation for determining the length of a C-style string. It operates by iterating through the characters of the input array str until it encounters the null terminator (\0), which signifies the end of the string.

The function initializes an integer counter, length, to 0. It then enters a while loop that continues as long as the character at the current index, str[length], is not the null character. Inside the loop, the length variable is incremented with each iteration. This process effectively counts the number of characters in the string.

When the loop condition str[length] != '\0' becomes false (i.e., the null terminator is found), the loop terminates. The final value of length represents the total number of characters before the null terminator. This value is then returned as the result.

int stringLength(char str[]) {  
 int length = 0;  
 while (str[length] != '\0') {  
 length++;  
 }  
 return length;  
}

## 2.4 Usage Notes

* The input character array str **must** be null-terminated. Failure to provide a null-terminated string will cause the function to read beyond the bounds of the array, leading to undefined behavior and potential program crashes.
* The returned length does not include the null terminator character \0 in its count.
* The function does not modify the content of the input string str.

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

## 2.5 Example

The following C code demonstrates how to use the stringLength function. It requires a main function to be executed.

#include <stdio.h>  
  
// Assuming stringLength function is defined in the same file or included  
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 string is: \"%s\"\n", myString);  
 printf("The calculated length is: %d\n", len);  
  
 char emptyString[] = "";  
 int emptyLen = stringLength(emptyString);  
 printf("The string is: \"%s\"\n", emptyString);  
 printf("The calculated length is: %d\n", emptyLen);  
   
 return 0;  
}

**Output:**

The string is: "Hello, World!"  
The calculated length is: 13  
The string is: ""  
The calculated length is: 0

## 2.6 FunctionDef isPrime(n)

# 3 Function: isPrime

## 3.1 Overview

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

## 3.2 parameters

* n: int - The integer to be checked for primality.

## 3.3 Description

This function implements a primality test to verify if the input number n is prime. The logic proceeds as follows:

1. **Base Case Check**: The function first checks if n is less than or equal to 1. By definition, numbers less than or equal to 1 are not considered prime numbers, so the function immediately returns false.
2. **Iterative Division Check**: A for loop is initiated to check for potential divisors. The loop starts with i = 2 and continues as long as i \* i <= n. This condition is an optimization based on the fact that if a number 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 test for divisors up to sqrt(n).
3. **Modulus Operation**: Inside the loop, the expression n % i == 0 checks if n is perfectly divisible by the current value of i.
4. **Return Value**:
   * If a divisor is found (i.e., n % i is 0), n is not a prime number, and the function immediately returns false.
   * If the loop completes without finding any divisors, it means n is only divisible by 1 and itself, confirming it is a prime number. In this case, the function returns true.

## 3.4 Usage Notes

* The function correctly handles edge cases by returning false for any integer less than or equal to 1.
* The function uses an efficient trial division algorithm, which is suitable for moderately sized integers.
* The return type is bool. To use the bool, true, and false keywords in C, you must include the <stdbool.h> header.

**Output Example**: The function returns a boolean value. For an input of 29, the output would be:

true

## 3.5 Example

#include <stdio.h>  
#include <stdbool.h>  
  
// Definition of the isPrime function  
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 num1 = 7;  
 int num2 = 10;  
  
 // Example usage  
 bool result1 = isPrime(num1);  
 bool result2 = isPrime(num2);  
  
 printf("Is %d a prime number? %s\n", num1, result1 ? "true" : "false");  
 printf("Is %d a prime number? %s\n", num2, result2 ? "true" : "false");  
  
 return 0;  
}

**Output:**

Is 7 a prime number? true  
Is 10 a prime number? false

## 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 utility functions: findMax, stringLength, and isPrime.

## 4.2 parameters

This function does not accept any parameters.

## 4.3 Description

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

1. **Find Maximum in an Array**:
   * An integer array nums is initialized with the values {5, 12, 3, 19, 7}.
   * The variable size is calculated by dividing the total size of the nums array by the size of a single integer element, effectively determining the number of elements in the array.
   * It then calls the findMax function, passing nums and size as arguments.
   * The maximum value returned by findMax is printed to the console.
2. **Calculate String Length**:
   * A character array name is initialized with the string literal “Prateek”.
   * It calls the stringLength function with name as the argument.
   * The length of the string returned by stringLength is printed to the console.
3. **Check for Prime Number**:
   * An integer variable number is initialized with the value 17.
   * The isPrime function is called with number as the argument.
   * An if-else statement checks the boolean result from isPrime. It prints a message to the console indicating whether 17 is a prime number or not.

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

## 4.4 Usage Notes

* This function is the primary entry point for the C program.
* It depends on the external functions findMax, stringLength, and isPrime. These functions must be defined and linked for the program to compile and run correctly.
* All results are printed directly to the standard output (the console).

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

## 4.5 Example

The main function is the starting point of execution. To run it, you compile the entire C program and execute the resulting binary.

#include <stdio.h>  
  
// Assume findMax, stringLength, and isPrime are defined elsewhere  
int findMax(int arr[], int size);  
int stringLength(char str[]);  
int isPrime(int n);  
  
int main() {  
 int nums[] = {5, 12, 3, 19, 7};  
 int size = sizeof(nums) / sizeof(nums[0]);  
   
 printf("Maximum number: %d\n", findMax(nums, size));  
   
 char name[] = "Prateek";  
 printf("Length of '%s': %d\n", name, stringLength(name));  
   
 int number = 17;  
 if (isPrime(number))  
 printf("%d is a prime number.\n", number);  
 else  
 printf("%d is not a prime number.\n", number);  
  
 return 0;  
}

**Output:**

Assuming the helper functions are implemented correctly, the output will be:

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