Given a string, find its first non-repeating character

Given a string, find the first non-repeating character in it. For example, if the input string is “GeeksforGeeks”, then output should be ‘f’ and if input string is “GeeksQuiz”, then output should be ‘G’.

We can use string characters as index and build a count array. Following is the algorithm.

```
1) Scan the string from left to right and construct the count array.
2) Again, scan the string from left to right and check for count of each
 character, if you find an element who's count is 1, return it.
```

Example:

```
Input string: str = geeksforgeeks
1: Construct character count array from the input string.
   ....
  count['e'] = 4
  count['f'] = 1
  count['g'] = 2
  count['k'] = 2
  ……
2: Get the first character who's count is 1 ('f').
```

Implementation:

```
#include<stdlib.h>
#include<stdio.h>
#define NO_OF_CHARS 256
```

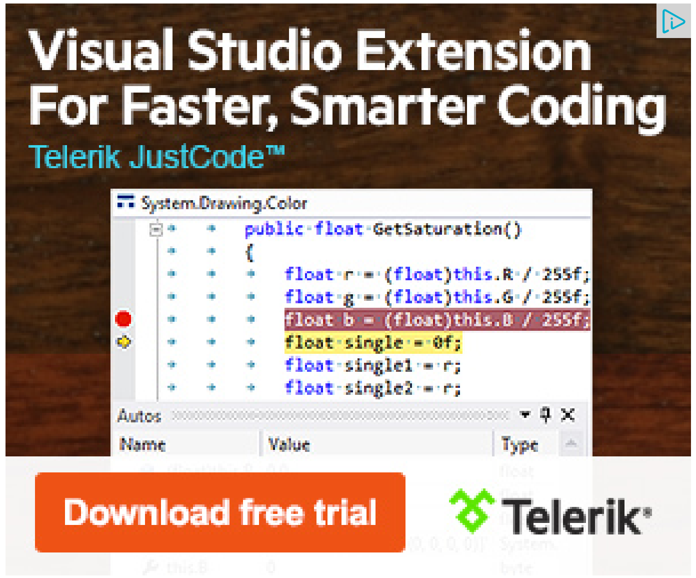
Google™ Custom Search

GeeksforGeeks
52,731 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```c
/* Returns an array of size 256 containg count
   of characters in the passed char array */
int *getCharCountArray(char *str)
{
   int *count = (int *)calloc(sizeof(int), NO_OF_CHARS);
   int i;
   for (i = 0; *(str+i);  i++)
      count[*(str+i)]++;
   return count;
}

/* The function returns index of first non-repeating
   character in a string. If all characters are repeating
   then returns -1 */
int firstNonRepeating(char *str)
{
  int *count = getCharCountArray(str);
  int index = -1, i;

  for (i = 0; *(str+i);  i++)
  {
    if (count[*(str+i)] == 1)
    {
      index = i;
      break;
    }
  }

  free(count); // To avoid memory leak
  return index;
}

/* Driver program to test above function */
int main()
{
  char str[] = "geeksforgeeks";
  int index =  firstNonRepeating(str);
  if (index == -1)
    printf("Either all characters are repeating or string is empty");
  else
   printf("First non-repeating character is %c", str[index]);
  getchar();
  return 0;
}
```

Output:



```
System.Drawing.Color
public float GetSaturation()
{
    float r = (float)this.R / 255f;
    float g = (float)this.G / 255f;
    float b = (float)this.B / 255f;
    float single = 0f;
    float single1 = r;
    float single2 = r;
Autos
Name    Value    Type
```

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
First non-repeating character is f
```

Can we do it by traversing the string only once?
The above approach takes O(n) time, but in practice it can be improved. The first part of the algorithm runs through the string to construct the count array (in O(n) time). This is reasonable. But the second part about running through the string again just to find the first non-repeater is not good in practice. In real situations, your string is expected to be much larger than your alphabet. Take DNA sequences for example: they could be millions of letters long with an alphabet of just 4 letters. What happens if the non-repeater is at the end of the string? Then we would have to scan for a long time (again).
We can augment the count array by storing not just counts but also the index of the first time you encountered the character e.g. (3, 26) for 'a' meaning that 'a' got counted 3 times and the first time it was seen is at position 26. So when it comes to finding the first non-repeater, we just have to scan the count array, instead of the string. Thanks to Ben for suggesting this approach.

Following is C implementation of the extended approach that traverses the input string only once.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#define NO_OF_CHARS 256

// Structure to store count of a character and index of the first
// occurrence in the input string
struct countIndex {
   int count;
   int index;
};

/* Returns an array of above structure type. The size of
   array is NO_OF_CHARS */
struct countIndex *getCharCountArray(char *str)
{
   struct countIndex *count =
        (struct countIndex *)calloc(sizeof(countIndex), NO_OF_CHARS);
   int i;
   for (i = 0; *(str+i);  i++)
   {
      (count[*(str+i)].count)++;

      // If it's first occurrence, then store the index
```

```
        if (count[*(str+i)].count == 1)
           count[*(str+i)].index = i;
    }
    return count;
}

/* The function returns index of the first non-repeating
    character in a string. If all characters are repeating
    then reurns INT_MAX */
int firstNonRepeating(char *str)
{
  struct countIndex *count = getCharCountArray(str);
  int result = INT_MAX, i;

  for (i = 0; i < NO_OF_CHARS;  i++)
  {
    // If this character occurs only once and appears
    // before the current result, then update the result
    if (count[i].count == 1 && result > count[i].index)
       result = count[i].index;
  }

  free(count); // To avoid memory leak
  return result;
}

/* Driver program to test above function */
int main()
{
  char str[] = "geeksforgeeks";
  int index =  firstNonRepeating(str);
  if (index == INT_MAX)
    printf("Either all characters are repeating or string is empty");
  else
   printf("First non-repeating character is %c", str[index]);
  getchar();
  return 0;
}
```

Output:

```
First non-repeating character is f
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

695 Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 20 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 40 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 40 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

Better Than Hadoop.
HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.
LexisNexis
Learn More

Related Tpoics:

- Printing Longest Common Subsequence
- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Find the first non-repeating character from a stream of characters
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string
- Dynamic Programming | Set 29 (Longest Common Substring)

Writing code in comment? Please use **ideone.com** and share the link here.