Write your own strcmp that ignores cases

Write a modified strcmp function which ignores cases and returns -1 if s1 < s2, 0 if s1 = s2, else returns 1. For example, your strcmp should consider "GeeksforGeeks" and "geeksforgeeks" as same string.

Source: Microsoft Interview Set 5

Following solution assumes that characters are represented using ASCII representation, i.e., codes for ‘a’, ‘b’, ‘c’, … ‘z’ are 97, 98, 99, … 122 respectively. And codes for ‘A’, “B”, ‘C’, … ‘Z’ are 65, 66, … 95 respectively.

Following are the detailed steps.
1) Iterate through every character of both strings and do following for each character.
…**a)** If str1[i] is same as str2[i], then continue.
…**b)** If inverting the 6th least significant bit of str1[i] makes it same as str2[i], then continue. For example, if str1[i] is 65, then inverting the 6th bit will make it 97. And if str1[i] is 97, then inverting the 6th bit will make it 65.
…**c)** If any of the above two conditions is not true, then break.
2) Compare the last (or first mismatching in case of not same) characters.

```
#include <stdio.h>

/* implementation of strcmp that ingnores cases */
int ic_strcmp(char *s1, char *s2)
{
    int i;
    for (i = 0; s1[i] && s2[i]; ++i)
    {
        /* If characters are same or inverting the 6th bit makes them
        if (s1[i] == s2[i] || (s1[i] ^ 32) == s2[i])
```


```
            continue;
        else
            break;
    }

    /* Compare the last (or first mismatching in case of not same) cha
    if (s1[i] == s2[i])
        return 0;
    if ((s1[i]|32) < (s2[i]|32)) //Set the 6th bit in both, then compa
        return -1;
    return 1;
}

// Driver program to test above function
int main(void)
{
    printf("ret: %d\n", ic_strcmp("Geeks", "apple"));
    printf("ret: %d\n", ic_strcmp("", "ABCD"));
    printf("ret: %d\n", ic_strcmp("ABCD", "z"));
    printf("ret: %d\n", ic_strcmp("ABCD", "abcdEghe"));
    printf("ret: %d\n", ic_strcmp("GeeksForGeeks", "gEEksFORGeEKs"));
    printf("ret: %d\n", ic_strcmp("GeeksForGeeks", "geeksForGeeks"));
    return 0;
}
```

Output:

```
ret: 1
ret: -1
ret: -1
ret: -1
ret: 0
ret: 0
```

This article is compiled by **Narendra Kangralkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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

Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Binary representation of a given number
- Find the element that appears once

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

25 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

algo1 • 2 months ago

```
#include<stdio.h>

#include<string.h>

using namespace std;

int main()

{

char s1[100],s2[100];

scanf("%s",s1);

scanf("%s",s2);

if(strlen(s1)!=strlen(s2)) {cout<<"Strings are not equal\n";return 0;}

char a,b;
```

see more

Reply • Share ›

Chinchore Navnath • 11 months ago

the code posted by u is useful for me

Reply • Share ›

rka143 • a year ago

How are we making sure in above code that both strings are pointing to alphat with 32. if s1 [i] is equal to the s2[i]^32 but the s1[i] is not in alphabetic range fo that case, the above mentioned code will return true but actually it should retur

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 31 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

```
     char str1[10] = {31,'B','C','D','E','F','G','H'};
    char str2[10] = {63,'b','c','d','e','f','g','h'};

Output:
       str1:BCDEFGH  str2:?bcdefgh ret: 0

which is wrong

Another problem is :
    char str1[10] = {32,'B','C','D','E','F','G','H'};
    char str2[10] = {30,'b','c','d','e','f','g','h'};

Output:
    str1: BCDEFGH  str2:bcdefgh ret: -1
```

see more

• Reply • Share ›

abhishek08aug • a year ago

Here is a recursive code to compare two alphabetical strings (A-Z a-z):
Comment out the two ERROR "else if"'s to make it general:

```
 #include<stdio.h>

char tolower(char c) {
  if(c>=65 && c<=90) {
    return c+32;
  } else {
    return c;
  }
}
```

```
int strcmp_nocase(char * a, char * b) {
  if (a==b) { printf("Both the strings are same\n"); return 0;}
  else if(!(tolower(*a)>=65 && tolower(*a)<=90) && !(tolower(*a)>=97 &
  else if(!(tolower(*b)>=65 && tolower(*b)<=90) && !(tolower(*b)>=97 &
  else if(tolower(*a) == '&#92&#48' && tolower(*b) != '&#92&#48') {ret
```

see more

1 ▲ | ▼ • Reply • Share ›

atul • a year ago

we can change the if condition given in the code :-

```
if (s1[i] == s2[i] || (s1[i] ^ 32) == s2[i])
```

to this one, it will still work :-

```
if (s1[i] == s2[i] || (s1[i] | 32) == (s2[i]|32))
```

▲ | ▼ • Reply • Share ›

neelabhsingh ➦ atul • 6 months ago

Instead of writing

if (s1[i] == s2[i] || (s1[i] | 32) == (s2[i]|32))
write only

if ((s1[i] | 32) == (s2[i]|32)) not require to use s1[i]==s2[i];

▲ | ▼ • Reply • Share ›

Rahul Samskruthyayan • a year ago

wats the required functionality?

• Reply • Share ›

Pallavi Deshpande • a year ago

What if do it this way:

```
int ic_strcmp(char *s1, char *s2) {.
int i;
int result = 0;.
for (i = 0; s1[i] && s2[i]; ++i) {.
//check if chars are same r invert 6th bit to check for lower and upper case of
if (s1[i] == s2[i] || (s1[i] ^32) == s2[i]) {.
result = 1;.
continue;
} else
break;
}

return result;

}
```

Reply • Share ›

saran • a year ago

it's very useful to me...thank u

Reply • Share ›

Mike • a year ago

The code provided thinks that this string "@~" is equal to "`^". Those symbols

Reply • Share ›

algo1 Mike • 2 months ago

Exactly ,this should be taken care of

Reply • Share ›

AMIT ➚ Mike • 11 months ago

yesss...this should be checked

• Reply • Share ›

algopiggy • a year ago

Shouldn't this -

```
if (s1[i] == s2[i])
        return 0;
```

be written as -

```
if (s1[i] == s2[i] || (s1[i] ^ 32) == s2[i] )
        return 0;
```

1 • Reply • Share ›

Kartik ➚ algopiggy • a year ago

@algopiggy: The original code is fine as we come out of the loop only \
string termination character (backslash zero)

• Reply • Share ›

algopiggy ➚ Kartik • a year ago

Thanks :)

```
/* Paste your code here (You may delete these lines if r
```

• Reply • Share ›

Vikrant • a year ago

My version of strcmp

```
#include <stdio.h>
#include <stdlib.h>

int my_strcmp(char* s1, char* s2)
{
    int i = 0;

    while( (abs(s1[i] - s2[i]) &(~('a' - 'A'))) == 0 && (s1[i] != NULL
        i++;

    if ((s1[i] == NULL) && (s2[i] == NULL))
        return 0;
    else if ((s1[i]|('a' - 'A')) < (s2[i]|('a' - 'A')))
        return -1;
    else
```

see more

• Reply • Share ›

Karthick • a year ago

But if we compare "Geeks" with "apple" it will return -1,but still we are ignoring

```
/* Paste your code here (You may delete these lines if not writing co
```

• Reply • Share ›

GeeksforGeeks ➚ Karthick • a year ago

Thanks for pointing out this case. To handle this case, we have chang following

```
//Set the 6th bit in both characters, then compare them
if ((s1[i]|32) < (s2[i]|32))
    return -1;
```

• Reply • Share ›

rog ➚ GeeksforGeeks • a year ago

works fine now

• Reply • Share ›

Pranav • a year ago

Make those arguments to the conversion function as constant.

• Reply • Share ›

Rohit Agarwal • a year ago

please explain output of below statement?

```
printf("ret: %d\n", ic_strcmp("ABCD", "z"));
```

• Reply • Share ›

ashutosh • a year ago

can anyone explain the inverting operation s[i]^32 ?

• Reply • Share ›

Ritesh ➚ ashutosh • a year ago

a=97
A=65

Difference between the two is 32 (a-A) or (A-a)

So, if you somehow produce a change of 32 in the ASCII value, you ca
case.

A+32=a
a-32=A

When you invert the 6th least significant bit, you are producing the sam
significant bit in binary representation is 32.

32 = (100000) in binary.

I hope it makes it clear!

Raj Ritesh • a year ago
Thanks.....

Venki • a year ago
Nice code.

It should be evident now why 'A' is placed at 65 and 'a' is placed at '97' in the A