

Forum:[Hack Forums](#)

Thread: [Find length of array of string](#)

SIM-2362662 - April 14, 2020, 11:48 am

I'm trying to find the length of an array of strings.

Let's say I have this :

```
Code string my_str[] = {"a", "b", "c", "d", "e", "f"};
```

In order to get the length of my_str, I can do:

```
Code int n = sizeof(my_str) / sizeof(my_str[0]);
```

And everything works fine. But if I try to do that in a function, it doesn't work. Like this:

```
Code void myfct(string strArr[]) {  
    int n = sizeof(strArr) / sizeof(strArr[0]);  
    cout << n;  
}
```

```
int main() {  
    string my_str[] = {"1", "2", "E", "E", "1", "E"};  
    myfct(my_str);  
}I get 0.
```

So I would like to know why this happens and how else I can do this.

saltycracker - April 14, 2020, 12:44 pm

Well look at the warning my compiler generates?

```
Quote:warning: 'sizeof' on array function parameter 'strArr' will return size of 'std::__cxx11::string*  
{aka std::__cxx11::basic_string<char>*}' [-Wsizeof-array-argument]  
    std::cout << sizeof(strArr) << std::endl;
```

Hint... It becomes a pointer when you pass it to the function. If memory serves me here.. You have to pass the array size to the function too.

akali - April 14, 2020, 1:20 pm

(Apr 14, 2020 11:48 AM)? SoFt ? Wrote: [\[CITE\]](#) I'm trying to find the length of an array of strings.

Let's say I have this :

```
Code string my_str[] = {"a", "b", "c", "d", "e", "f"};
```

In order to get the length of my_str, I can do:

```
Code int n = sizeof(my_str) / sizeof(my_str[0]);
```

And everything works fine. But if I try to do that in a function, it doesn't work. Like this:

```
Code void myfct(string strArr[]) {  
    int n = sizeof(strArr) / sizeof(strArr[0]);  
    cout << n;  
}
```

```
int main() {  
    string my_str[] = {"1", "2", "E", "E", "1", "E"};  
    myfct(my_str);  
}I get 0.
```

So I would like to know why this happens and how else I can do this.
sizeof(array) is done at compile time. Otherwise it would give you pointer size. Whenever you're dealing with pointers to arrays, the size needs to be passed as an argument as well. They do not know their own size.

SIM-2362662 - April 14, 2020, 1:31 pm

(Apr 14, 2020 01:20 PM)akali Wrote: [\[CITE\]](#) sizeof(array) is done at compile time. Otherwise it would give you pointer size. Whenever you're dealing with pointers to arrays, the size needs to be passed as an argument as well. They do not know their own size.

So you're saying there is no way for my function to know the length of the array without me passing it as an argument. Right?

saltycracker - April 14, 2020, 1:38 pm

(Apr 14, 2020 01:31 PM)? SoFt ? Wrote: [\[CITE\]](#) So you're saying there is no way for my function to know the length of the array without me passing it as an argument. Right?

You have to pass the array size or have some sort of sentinel value to terminate the array... Like a C string.

creki - April 14, 2020, 1:56 pm

gay but works

SIM-2362662 - April 14, 2020, 1:58 pm

(Apr 14, 2020 01:56 PM)creki Wrote: [\[CITE\]](#)

gay but works

You don't necessarily get empty if you exceed the index, do you?

I think you just get whatever is in the memory

akali - April 14, 2020, 3:24 pm

(Apr 14, 2020 01:56 PM)creki Wrote: [\[CITE\]](#)

gay but works

I'm pretty sure the code you posted results in undefined behavior. Because at some point, you're trying to call a method of an object that may not even exist in memory. The compiler might just be doing some trickery there and it's working. But no you shouldn't do that. And the only reason string functions in general don't need a length is because they check for a null terminator.

(Apr 14, 2020 01:58 PM)? SoFt ? Wrote: [\[CITE\]](#) You don't necessarily get empty if you exceed the index, do you?

I think you just get whatever is in the memory

Exactly. His code doesn't make any sense.

zNinja - April 15, 2020, 11:28 pm

PHP Code

```
<?php
template <class T,class A>
size_t array_size(A&array){
    return sizeof(array)/ sizeof(T);
}
int main(int argc,char*argv[])
{
```

```
std::string test[] = { "a", "b", "c","defg" };
std::cout << array_size<std::string>(test) << std::endl;
return 0;
}
```

dracx - April 18, 2020, 7:42 pm

Well am not sure bout those C++ string, butnin C char array, you simply have to loop over the starting address then stop counting the first time you hit a null byte. This is not the case for wide char array tho, each wide char array is composed of two bytes and is terminated by two nill bytes too. Am on mobile so cant properly supply at least some pseudo-code. Google "custom strlen".

zNinja - April 23, 2020, 11:28 am

(Apr 18, 2020 07:42 PM)dracx Wrote: [\[CITE\]](#) Well am not sure bout those C++ string, butnin C char array, you simply have to loop over the starting address then stop counting the first time you hit a null byte. This is not the case for wide char array tho, each wide char array is composed of two bytes and is terminated by two nill bytes too. Am on mobile so cant properly supply at least some pseudo-code. Google "custom strlen".

Well, C string(char array) is a little bit tricky, if you have const char array, you can get size easy. For pointer you can not use "null byte" as orientation, since array can contain binary content which can have multiple null bytes.

You must track size of array trough code, Like when you reading from a file with C api, you must retrieve size of file then allocate container (char array) and then read.

dracx - April 23, 2020, 7:06 pm

(Apr 23, 2020 11:28 AM)zNinja Wrote: [\[CITE\]](#) Well, C string(char array) is a little bit tricky, if you have const char array, you can get size easy. For pointer you can not use "null byte" as orientation, since array can contain binary content which can have multiple null bytes.

You must track size of array trough code, Like when you reading from a file with C api, you must retrieve size of file then allocate container (char array) and then read.

We we're talkin about string A.K.A. char array, not blob of data, no?