

Forum:[Hack Forums](#)

Thread: [Programming Challenge](#)

g4143 - August 15, 2015, 9:58 am

Here's a simple programming challenge that should get the rookies thinking.

Writing a program which will take a mutable c-string and remove characters from it. You are free to do this any way you want but you should be able to defend your solution and recognize any possible short-comings.

Here's some code to start with:

```
Code int main(int argc, char** argv) {  
    char s[]="The C string to work with";  
    return 0;  
}
```

Now take some time and think about a solution. Why is your solution good? Why is your solution bad?

Please post your solutions and the pros and cons and we'll comment on them.

0xDEAD10CC - August 15, 2015, 11:37 am

Code #include <stdio.h>

#include <stdlib.h>

#include <string.h>

```
int main(void)
```

```
{  
    size_t n = 4; /* index of the 'C' */  
    char s[] = "The C string to work with";  
    char *p = s;  
    memmove(p + 1, p, n); ++p; /* remove 'C' */  
    memmove(p + 1, p, n); ++p; /* remove extra space which was after 'C' */  
    puts(p);  
    exit(0);  
}
```

Pros:

- no reallocations (removing chars means the buffer always shrinks -- no need to realloc really)
- no string concatenation of lower and upper halves to "remove" a char

Cons:

- original chars not preserved, NOTE: A fix for that might look something like below

Code #include <stdio.h>

#include <stdlib.h>

#include <string.h>

```
int main(void)
```

```
{  
    char ch;  
    size_t n = 4; /* index of the 'C' */
```

```

char s[] = "The C string to work with";
char *p = s;

ch = p[n];
memmove(p + 1, p, n); ++p; /* remove 'C' */
*(p - 1) = ch;

ch = p[n];
memmove(p + 1, p, n); ++p; /* remove extra space which was after 'C' */
*(p - 1) = ch;

puts(p);

fputs("Junked chars -> ", stdout);
fwrite(s, 1, p - s, stdout);
putc("\n", stdout);
exit(0);
}

```

Blender - August 15, 2015, 1:29 pm

(Aug 15, 2015 11:37 AM)0xDEAD10CC Wrote: [\[CITE\]](#) Code #include <stdio.h>

#include <stdlib.h>

#include <string.h>

```

int main(void)
{
    size_t n = 4; /* index of the 'C' */
    char s[] = "The C string to work with";
    char *p = s;
    memmove(p + 1, p, n); ++p; /* remove 'C' */
    memmove(p + 1, p, n); ++p; /* remove extra space which was after 'C' */
    puts(p);
    exit(0);
}

```

Pros:

- no reallocations (removing chars means the buffer always shrinks -- no need to realloc really)
- no string concatenation of lower and upper halves to "remove" a char

Cons:

- original chars not preserved, NOTE: A fix for that might look something like below

Code #include <stdio.h>

#include <stdlib.h>

#include <string.h>

```

int main(void)
{
    char ch;
    size_t n = 4; /* index of the 'C' */
    char s[] = "The C string to work with";
    char *p = s;

    ch = p[n];
    memmove(p + 1, p, n); ++p; /* remove 'C' */
}

```

```
*(p - 1) = ch;
```

```
ch = p[n];
```

```
memmove(p + 1, p, n); ++p; /* remove extra space which was after 'C' */
```

```
*(p - 1) = ch;
```

```
puts(p);
```

```
fputs("Junked chars -> ", stdout);
```

```
fwrite(s, 1, p - s, stdout);
```

```
putc("\n", stdout);
```

```
exit(0);
```

```
}
```

So I'm a little confused. Isn't it easier to use Code string s = "The C string to work with"; ? Also, why did you put void inside of the main function? Leaving it blank does the same thing, doesn't it? Unless it's more of a style decision.

sdk - August 15, 2015, 2:48 pm

Actually can you help me with something, I practically made something that removes the beginning character from the string somehow...

It's kinda on topic.

Code #include "stdafx.h"

#include <iostream>

```
int _tmain(int argc, _TCHAR* argv[])
```

```
{
```

```
    for (int i = 0; i < 100; i++) {
```

```
        std::cout << "The current number " + i << std::endl;
```

```
    }
```

```
    std::cin.ignore();
```

```
    return 0;
```

```
}
```

d on topic lolz.

B1N4RY - August 15, 2015, 2:53 pm

(Aug 15, 2015 01:29 PM)Blender Wrote: [\[CITE\]](#) Also, why did you put void inside of the main function? Leaving it blank does the same thing, doesn't it? Unless it's more of a style decision.

Your main should always be declared as a void, as it "returns" nothing. On that note, you should always have a return 0 at the end of main ;)

Blender - August 15, 2015, 4:32 pm

(Aug 15, 2015 02:53 PM)_B1N4RY_ Wrote: [\[CITE\]](#) Your main should always be declared as a void, as it "returns" nothing. On that note, you should always have a return 0 at the end of main ;)

I've never had to declare it as a void. It does this automatically if you leave it empty. Only time I've ever had to declare a void was when I created a function that had no return type. But with parameters, never.

And why isn't anyone doing

using namespace std ?

It saves the time to do std::cout(unless you're using something with multiple libraries that has the

potential to conflict with the functions in either library, but I haven't seen that in the code presented on this thread anyway)

(Aug 15, 2015 02:48 PM) sdk? Wrote: [\[CITE\]](#) Actually can you help me with something, I practically made something that removes the beginning character from the string somehow...

It's kinda on topic.

Code #include "stdafx.h"

#include <iostream>

```
int _tmain(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 100; i++) {
        std::cout << "The current number " + i << std::endl;
    }
    std::cin.ignore();
    return 0;
}
```

Code #include "stdafx.h"

#include <iostream>

using namespace std;

```
int _tmain(int argc, _TCHAR* argv[])
{
    string s = "The current number";

    for(int i = 0; i < 100; i++)
    {
        cout << s << ": " << i << endl;
    }
    cin.ignore();
    return 0;
}
```

That will fix it.

Masm4ever - August 15, 2015, 4:42 pm

(Aug 15, 2015 02:53 PM) _B1N4RY_ Wrote: [\[CITE\]](#) Your main should always be declared as a void, as it "returns" nothing. On that note, you should always have a return 0 at the end of main ;)

You are confused:

1. int main(void) means that main is declared as integer without arguments passed. And that is exactly right if you don't plan to pass options via the command line.

2. main() always returns something. The return 0 is optional, because by default main() - and only main() - returns zero. You may return a non-zero value to flag an error, which you can test (under Windows at least) with echo %errorlevel%.

Blender - August 15, 2015, 4:44 pm

Your main really doesn't need arguments so long as you're not planning to make it do that which in my opinion would be pretty messy and makes things too complicated, because you're most likely not going to be doing console applications for the rest of your life.

sdk - August 15, 2015, 5:24 pm

(Aug 15, 2015 02:53 PM)_B1N4RY_ Wrote: [\[CITE\]](#) Your main should always be declared as a void, as it "returns" nothing. On that note, you should always have a return 0 at the end of main ;)
int _tmain is the way Visual Studio makes the default main method. Also I know I as told to return 0 or something to avoid memory leaks.

0xDEAD10CC - August 15, 2015, 7:01 pm

(Aug 15, 2015 01:29 PM)Blender Wrote: [\[CITE\]](#) So I'm a little confused. Isn't it easier to use Code string s = "The C string to work with"; ? Also, why did you put void inside of the main function? Leaving it blank does the same thing, doesn't it? Unless it's more of a style decision.

You are clearly confused yes. The whole point of the challenge was C strings, not C++ strings. And void means that no arguments can be passed to the function, it's not about style.

(Aug 15, 2015 02:53 PM)_B1N4RY_ Wrote: [\[CITE\]](#) Your main should always be declared as a void, as it "returns" nothing. On that note, you should always have a return 0 at the end of main ;)

That's not what he was asking, and main does return an int, which is why it's declared to return int.
:S

(Aug 15, 2015 02:48 PM)sdk? Wrote: [\[CITE\]](#) Actually can you help me with something, I practically made something that removes the beginning character from the string somehow...

It's kinda on topic.

```
Code #include "stdafx.h"
#include <iostream>
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 100; i++) {
        std::cout << "The current number " + i << std::endl;
    }
    std::cin.ignore();
    return 0;
}
```

d on topic lolz.

If it's not a solution to the challenge, it's off topic, get out of this thread.

(Aug 15, 2015 04:32 PM)Blender Wrote: [\[CITE\]](#) I've never had to declare it as a void. It does this automatically if you leave it empty. Only time I've ever had to declare a void was when I created a function that had no return type. But with parameters, never.

And why isn't anyone doing

using namespace std ?

It saves the time to do std::cout(unless you're using something with multiple libraries that has the potential to conflict with the functions in either library, but I haven't seen that in the code presented on this thread anyway)

```
Code #include "stdafx.h"
#include <iostream>
```

```
using namespace std;
```

```

int _tmain(int argc, _TCHAR* argv[])
{
    string s = "The current number";

    for(int i = 0; i < 100; i++)
    {
        cout << s << ": " << i << endl;
    }
    cin.ignore();
    return 0;
}

```

That will fix it.

Because 'using namespace std;' or 'using namespace' in general defeats the whole point of namespaces, and it's bad to be doing that. Lastly, this thread is not even about C++.

Can we lose the skids in this thread please?? ?

(Aug 15, 2015 05:24 PM) sdk? Wrote: [\[CITE\]](#) int _tmain is the way Visual Studio makes the default main method. Also I know I as told to return 0 or something to avoid memory leaks.

Honestly, what are you talking about? return 0 at the end of main()'s scope can be omitted as per the C++ standard, and reaching the terminating } of main() will have the same effect as 0. There's nothing even remotely related to memory leaks with the return of main(). Additionally, you should never omit the return type to any function, including main(). void main() is old, and now also, very poor practice, and so is main() without a return type.

(Aug 15, 2015 04:44 PM) Blender Wrote: [\[CITE\]](#) Your main really doesn't need arguments so long as you're not planning to make it do that which in my opinion would be pretty messy and makes things too complicated, because you're most likely not going to be doing console applications for the rest of your life.

And you don't think GUI applications take command line arguments? I can tell you that you're wrong.

Internet Explorer, Google Chrome, Firefox, Notepad, and pretty much 90% of all other programs I know, allow command line arguments for various things.

Proof: Open cmd.exe and invoke this ->
Code notepad "file.txt"

Kantos - August 15, 2015, 11:57 pm
@dead

I would have done something like yours using memmove, however I don't really understand what your code is doing (im high as fuck so thats probably why)

Like I would have thought to do
Code memmove(p + n - 1, p + n, strlen(p+n) + 1);
basically shifting the rest of the string past the character to be removed to the left but idk

0xDEAD10CC - August 16, 2015, 9:39 am
(Aug 15, 2015 11:57 PM) Kantos Wrote: [\[CITE\]](#) @dead

I would have done something like yours using memmove, however I don't really understand what

your code is doing (im high as fuck so thats probably why)

Like I would have thought to do

```
Code memmove(p + n - 1, p + n, strlen(p+n) + 1);
```

basically shifting the rest of the string past the character to be removed to the left but idk

I shift everything before the character to be moved to the right by one space, and keep a pointer to the first character of the string. Every time I move the left side to the right to "overwrite" the one character, I increment a pointer which starts at the beginning of the string so that we end up with "JUNK HERE|String we want here|0"

The pointer would be set to the 'S' after everything is said and done.

This way I don't need to call strlen() every time either, I didn't feel the need to--we know that removing one character makes the string length less by one each time. I think what you are meaning to do is just the reverse of what I did, and having junk past the null terminator instead.

Blender - August 17, 2015, 9:23 am

(Aug 15, 2015 07:01 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You are clearly confused yes. The whole point of the challenge was C strings, not C++ strings. And void means that no arguments can be passed to the function, it's not about style.

That's not what he was asking, and main does return an int, which is why it's declared to return int.
:S

If it's not a solution to the challenge, it's off topic, get out of this thread.

Because 'using namespace std;' or 'using namespace' in general defeats the whole point of namespaces, and it's bad to be doing that. Lastly, this thread is not even about C++.

Can we lose the skids in this thread please?? ?

Honestly, what are you talking about? return 0 at the end of main()'s scope can be omitted as per the C++ standard, and reaching the terminating } of main() will have the same effect as 0. There's nothing even remotely related to memory leaks with the return of main(). Additionally, you should never omit the return type to any function, including main(). void main() is old, and now also, very poor practice, and so is main() without a return type.

And you don't think GUI applications take command line arguments? I can tell you that you're wrong.

Internet Explorer, Google Chrome, Firefox, Notepad, and pretty much 90% of all other programs I know, allow command line arguments for various things.

Proof: Open cmd.exe and invoke this ->

Code notepad "file.txt"

Listen m8 calm down I can see your pantaloons are ruffled. No need to call people names.

In C++ I never had to put void inside of a parentheses to indicate that there were no arguments because leaving it blank did that for me.

I've never had to put characters inside of the int main() parentheses because in C++, that isn't needed anymore to pass characters to the console. That's seriously some old-style C shit, m8. If

you don't believe me, fire up a new cpp application in Visual Studio and create an app without using those characters.

Damn bruh chill out :(

0xDEAD10CC - August 17, 2015, 4:14 pm

Quote:In C++ I never had to put void inside of a parentheses to indicate that there were no arguments

Also, my code is C not C++. And there is a difference between a the void being there and being omitted if you do some research.

Quote:That's seriously some old-style C shit, m8

Maybe because my code IS C?...

Other than that, I don't know what you're talking about here:

Quote:I've never had to put characters inside of the int main() parentheses because in C++, that isn't needed anymore to pass characters to the console.

It doesn't make sense to me.

Blender - August 17, 2015, 5:52 pm

(Aug 17, 2015 04:14 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Also, my code is C not C++. And there is a difference between a the void being there and being omitted if you do some research.

Maybe because my code IS C?...

Other than that, I don't know what you're talking about here:

It doesn't make sense to me.

Sometimes in old C++ to input into the console people would need to do like int main(char(mander)*something) but it isnt required any longer because C++ is now upgraded into the cyberdalek form of happy. So, hopefully that clears everyting up sirs and sirs.

0xDEAD10CC - August 18, 2015, 6:45 pm

(Aug 17, 2015 05:52 PM)Blender Wrote: [\[CITE\]](#) Sometimes in old C++ to input into the console people would need to do like int main(char(mander)*something) but it isnt required any longer because C++ is now upgraded into the cyberdalek form of happy. So, hopefully that clears everyting up sirs and sirs.

Old C++? There's nothing old about the common and valid signatures for the main() function, and absolutely nothing new about grabbing arguments via Win32 functions either... "C++ is now upgraded" -- You don't know what you're talking about at all.

Blender - August 19, 2015, 10:20 am

(Aug 18, 2015 06:45 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Old C++? There's nothing old about the common and valid signatures for the main() function, and absolutely nothing new about grabbing arguments via Win32 functions either... "C++ is now upgraded" -- You don't know what you're talking about at all.

Or maybe you don't know how to interpret my words into something sensible which is quite alright since most normal people don't. What I am basically saying is that C++ has more or less "evolved" since the early 2000's and having anything within the goddamn main function is unnecessary unless you need to take parameters for something other than the console.

How do I know this?

GOOD QUESTION

I've coded applications inside the console before without the need of anything inside of the `int main()` function. I said "upgrade" because long ago when Yoda was still cool, you needed something inside of the `int main()` function if you wanted the console to take input.

I LIKE PIZZA

0xDEAD10CC - August 19, 2015, 3:12 pm

Any interpretation of your words can't help anyone comprehend something sensible from what you're trying to say because there is no such sensible thing in regards to what you are trying to say. Anyone with real experience with the language wouldn't explain the concepts like that. C++ has evolved but any of the points of evolution of the language that you have mentioned have nothing to do with the language of C++ itself, nor have they been part of the "evolution" of the language because they have been there for as long as I've dealt with the C++ language. This valid signature:
`Code int main()`

Has been around for a long time within the series of language standards for C++.

This makes no sense:

Quote:having anything within the goddamn main function is unnecessary unless you need to take parameters for something other than the console

Something other than the console? Whatever it is you're trying to say -- either you don't know what it is you're trying to say exactly, or you make it sound like it's some kind of black magic when you change the signature of the `main()` function.

Quote:I've coded applications inside the console before without the need of anything inside of the `int main()` function.

You still don't know what you're talking about as far as I can tell, and hello world programs don't count. I never said it was invalid to have no parameters, I simply said that it was not "old C++" to do so as you were calling it. You don't even explain things with proper terminology...

Masm4ever - August 19, 2015, 3:18 pm

(Aug 17, 2015 05:52 PM)Blender Wrote: [\[CITE\]](#) Sometimes in old C++ to input into the console people would need to do like `int main(char(mander)*something)` but it isn't required any longer because C++ is now upgraded into the cyberdalek form of happy. So, hopefully that clears everything up sirs and sirs.

Really, you should stop smoking that kraut, the side effects are too strong.

Blender - August 19, 2015, 8:23 pm

(Aug 19, 2015 03:12 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Any interpretation of your words can't help anyone comprehend something sensible from what you're trying to say because there is no such sensible thing in regards to what you are trying to say. Anyone with real experience with the language wouldn't explain the concepts like that. C++ has evolved but any of the points of evolution of the language that you have mentioned have nothing to do with the language of C++ itself, nor have they been part of the "evolution" of the language because they have been there for as long as I've dealt with the C++ language. This valid signature:

`Code int main()`

Has been around for a long time within the series of language standards for C++.

This makes no sense:

Something other than the console? Whatever it is you're trying to say -- either you don't know what it is you're trying to say exactly, or you make it sound like it's some kind of black magic when you change the signature of the `main()` function.

You still don't know what you're talking about as far as I can tell, and hello world programs don't count. I never said it was invalid to have no parameters, I simply said that it was not "old C++" to do so as you were calling it. You don't even explain things with proper terminology...

So why on earth would you use characters in the `int main()` ? I just don't really see why it's necessary.

0xDEAD10CC - August 20, 2015, 4:59 pm

(Aug 19, 2015 08:23 PM)Blender Wrote: [\[CITE\]](#) So why on earth would you use characters in the `int main()` ? I just don't really see why it's necessary.

characters?... As I said, you don't know what you're talking about. It's an argument count and an array of c-strings...

Blender - August 20, 2015, 7:42 pm

(Aug 20, 2015 04:59 PM)0xDEAD10CC Wrote: [\[CITE\]](#) characters?... As I said, you don't know what you're talking about. It's an argument count and an array of c-strings...

An argument count isn't needed in the main function of C++.

Masm4ever - August 21, 2015, 8:53 am

(Aug 20, 2015 07:42 PM)Blender Wrote: [\[CITE\]](#) An argument count isn't needed in the main function of C++.

@noobs reading this: Please ignore the above, it's nonsense - and the compilers know it. Example:

Code `#include <stdio.h>`

// wrong arg list:

```
int main(char *argv[]) {
    int argc=2;
    for (int i=0;i<argc;i++) printf("%i\t%s\n", i, argv[i]);
    return 0;
}
```

G++ warnings:

Code Tmp.cpp:3:5: warning: first argument of 'int main(char**)' should be 'int' [-Wmain]

```
int main(char *argv[]) {
    ^
```

Tmp.cpp:3:5: warning: 'int main(char**)' takes only zero or two arguments [-Wmain]

It gets compiled, though, but the exe crashes.

The correct way is like this:

Code `#include <stdio.h>`

```
int main(int argc, char *argv[]) {
    for (int i=0;i<argc;i++) printf("%i\t%s\n", i, argv[i]);
    return 0;
}
```

Note that `argv[0]` is the name of the executable itself. Launched e.g. with `Tmp.exe` These are the arguments, you get this:

Code 0 Tmp.exe

```
1    These
2    are
3    the
4    arguments
```

If you don't need arguments, you can choose

```
int main() {
```

or

```
int main(void) {
```

Hacktivator - August 21, 2015, 3:02 pm

I seriously didnt get what I should do, what do you mean by taking out characters?

is it for example my string is "hello world"

you choose to take "llo" off so I should print "he world"?

please dont hate me :(I would love to do this challenge, please explain :)

Masm4ever - August 21, 2015, 5:37 pm

(Aug 21, 2015 03:02 PM)Hacktivator Wrote: [\[CITE\]](#) I seriously didnt get what I should do, what do you mean by taking out characters?

is it for example my string is "hello world"

you choose to take "llo" off so I should print "he world"?

please dont hate me :(I would love to do this challenge, please explain :)

Yes, exactly. Something like this:

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

```
int main(void) {
    char s[]="hello world";
    strcpy(&s[2], &s[5]);
    printf("[%s]", &s[0]);
}
```

There are various ways to achieve this, strcpy() being the simplest one. Find your own, and explain why it works.

Or start by explaing why strcpy() works ;-)

0xDEAD10CC - August 21, 2015, 8:19 pm

(Aug 20, 2015 07:42 PM)Blender Wrote: [\[CITE\]](#) An argument count isn't needed in the main function of C++.

You have fun with buffer overruns then and determining somehow on how many pointers argv contains ?

Clearly as proven yet again you have no clue. A main() function that has only 1 parameter is not specified by any valid signature for main() in any of the language standards...

sunrix - August 22, 2015, 2:57 am

Guys I really tried this challenge. This is what I came up with.(Mind the extreme "noobness"). I'm still an extreme beginner.

PHP Code

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

```

#include <string.h>

int main()
{
    char str[] = "This is a C string to work with";
    char sub[20], new[50];
    char *ptr;
    int num;

    printf("Enter in a substring to delete ?:");
    fgets(sub, 20, stdin);

    printf("Enter in the position of substring in the string (starting from 0):");
    scanf("%d", &num);

    sub[strcspn(sub, "\n")] = '\0';

    strcpy(new, str);

    ptr = strstr(new, sub);

    if (ptr == NULL)
    {
        printf("Substring does not exist...\n");
        exit(0);
    }
    memmove((new + num), (ptr + strlen(sub)), (strlen(ptr + strlen(sub)) + 1));

    printf("%s\n", new);

    return 0;
}

```

I just want to know is there a way I can get the index number without having to count till the starting of the substring. ? Thanks for the challenge :)

Eight - August 22, 2015, 5:24 am

My solution based on the explanation suggested by Hacktivor :

> is it for example my string is "hello world"
 >you choose to take "llo" off so I should print "he world"?

PHP Code

```

<?php
#include <stdio.h>
#include <string.h>

void remove_chars(char *str, char *substr);
int main(int argc, char *argv[])
{
    char original[] = "foobarfoobarfoobar";
    remove_chars(original, "foo");
    printf("[%s]\n", original);
}

```

```

    return 0;
}

void remove_chars(char *str, char *substr)
{
    while(1)
    {
        char *occurrence = strstr(str, substr);
        if(occurrence == NULL)
            return;
        size_t substr_len = strlen(substr), str_len = strlen(str);
        size_t i;
        for(i = occurrence - str; i < str_len - substr_len; i++)
            str[i] = str[i + substr_len];
        str[i] = '\0';
    }
}

```

Pros :

- No dynamic allocation

- Removes all the occurrences of the substring in the original string ((foobarfoobar, foo) becomes barbar)

Cons :

- I may have done something stupid without realizing it

- The for loop could probably be replaced by a clever function call