

Forum:[Hack Forums](#)

Thread: [Text adventure help](#)

LaliLinux - January 26, 2016, 4:48 pm

So i'm creating a basic text adventure to help with my skills.

I'm running into a problem with an if-else statement. It keeps wanting to run the

```
Code #include <iostream>
#include <string>
using namespace std;

int main() {

    int UserInpInt;
    string UserInpChar;

    cout << "Welcome to my first shitty game!\n" << "Please specify what difficulty you wish to play
in. Type 'easy' or 'hard' to begin\n"
    << "I choose: ";
    cin >> UserInpChar;

    if (UserInpChar == "easy" || "hard") {
        cout << endl << "Congrats! You are ready to play!\n";
        cin.get();
    }
    else {
        cout << "That's not a valid input!\n";
    }

    cin.get();
    return 0;
}
```

Excuse the cin.get()'s. Im using visual studios so it's the only way to keep the console up?...

Anywho, it keeps giving me the Congrats line no matter what I type in. Why is my else statement not working?

Also, is there a way I can keep the console up without having to use cin.get() in every single statement/function? It's really annoying imo

Goodies - January 26, 2016, 5:03 pm

This is the issue.

PHP Code

```
<?php
if (UserInpChar == "easy" || "hard") {
    //...
}
```

```
try:
PHP Code
<?php
if(UserInpChar == "easy" || UserInpChar == "hard"){
    //...
}
```

Also, I just use `system("pause")` but whatever works.

6577657534735 - January 26, 2016, 5:05 pm

Set breakpoints to keep the console open at whatever point without moving forward

I always have the final "return 0" in main breakpointed.

Goodies - January 26, 2016, 5:26 pm

Also, you might want to use `transform` to make sure the input will be caught. For example, if I type "Easy", it will not work because only "easy" is correct. Instead of doing something stupid like:

```
if (... "easy" || ... "Easy" || "EAsy" ... etc)
```

You can do something like:

```
PHP Code
<?php
#include <set>
#include <string>
#include <algorithm>
#include <iostream>
#include <cctype>

int main() {
    std::set<std::string> levels = { "easy", "medium", "hard" };
    std::string uInput;
    std::cout << "Enter easy, medium, or hard: ";
    std::cin >> uInput;
    std::transform(uInput.begin(), uInput.end(), uInput.begin(), std::tolower);
    if (std::find(levels.begin(), levels.end(), uInput) != levels.end()) {
        std::cout << "ready!" << std::endl;
    }
    else {
        std::cout << "not ready" << std::endl;
    }
    system("pause");
}
```

LaliKlinux - January 26, 2016, 7:44 pm

Thanks! Really helped a lot mate

0xDEAD10CC - January 26, 2016, 7:49 pm

(Jan 26, 2016 06:26 PM)Goodies Wrote: [\[CITE\]](#) Also, you might want to use `transform` to make sure the input will be caught. For example, if I type "Easy", it will not work because only "easy" is correct. Instead of doing something stupid like:

```
if (... "easy" || ... "Easy" || "EAsy" ... etc)
```

You can do something like:

PHP Code

```
<?php
#include <set>
#include <string>
#include <algorithm>
#include <iostream>
#include <cctype>

int main() {
    std::set<std::string> levels = { "easy", "medium", "hard" };
    std::string uInput;
    std::cout << "Enter easy, medium, or hard: ";
    std::cin >> uInput;
    std::transform(uInput.begin(), uInput.end(), uInput.begin(), std::tolower);
    if (std::find(levels.begin(), levels.end(), uInput) != levels.end()) {
        std::cout << "ready!" << std::endl;
    }
    else {
        std::cout << "not ready" << std::endl;
    }
    system("pause");
}
```

Why would you use `std::find()` when `std::set::find()` already exists?

<http://www.cplusplus.com/reference/set/set/find/>

I didn't like the usage of `system("pause")` thrown into the example either, not really necessary to use it in snippets for learning material, but you haven't included the proper header to use it either (`<cstdlib>`)

Goodies - January 26, 2016, 10:05 pm

(Jan 26, 2016 08:49 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Why would you use `std::find()` when `std::set::find()` already exists? <http://www.cplusplus.com/reference/set/set/find/>

I didn't like the usage of `system("pause")` thrown into the example either, not really necessary to use it in snippets for learning material, but you haven't included the proper header to use it either (`<cstdlib>`)

1) I'm brand new to this. How would you use that? just like:

```
std::set<std::string> myset = {"a", "b", "c"};
myset.find("b")
?
```

2) How do you do it? `std::cin.getline()`?

0xDEAD10CC - January 26, 2016, 10:10 pm

If I ever need to `std::cin.get()` works completely fine for me. I hardly ever have a reason to do so in a production application though, and you really don't see it very often anyways. That's batch stuff...

Goodies - January 26, 2016, 10:28 pm

(Jan 26, 2016 11:10 PM)0xDEAD10CC Wrote: [\[CITE\]](#) If I ever need to `std::cin.get()` works completely fine for me. I hardly ever have a reason to do so in a production application though, and you really don't see it very often anyways. That's batch stuff...

Right. Do you not do a lot of console applications? or do you test from command line? Or are you just focused on debugging etc so CLI output is not really relevant? I would assume so.

0xDEAD10CC - January 26, 2016, 10:46 pm

(Jan 26, 2016 11:28 PM)Goodies Wrote: [CITE](#) Right. Do you not do a lot of console applications? or do you test from command line? Or are you just focused on debugging etc so CLI output is not really relevant? I would assume so.

There are other ways to debug without writing output to the console buffers. Visual Studio comes pre-packaged with a fully equipped debugger that saves more time if you actually know how to use it properly. There's really no reason to keep a line of code like that in a production program though, console or not. If you open up a command prompt in Windows, type the executable name, cmd.exe remains running and you can see output even after the program has terminated. Chances are unless the program takes no command line arguments, or if you create a shortcut that invokes it already with specific arguments, you're going to invoke the program that obvious way anyways.