

Forum:[Hack Forums](#)

Thread: [Trying to make my calculator take Integers & decimal numbers](#)

D_MuLa - September 14, 2016, 2:22 pm

I want it to take user input with integers & decimal numbers but I'm getting an error for line 14 ?

```
Code #include <iostream>
#include <string>
```

```
using namespace std;
```

```
int main()
{
    int wholenum1;
    int wholenum2;
    double decnum1;
    double decnum2;

    cout << ": ";
    getline(cin, wholenum1, wholenum2, decnum1, decnum2, 'endl');
    cout << wholenum1 << " / " << wholenum2 << " = " << wholenum1 / wholenum2 << endl;
    cout << ": ";
    getline(cin, wholenum1, wholenum2, decnum1, decnum2, 'endl');
    cout << decnum1 << " / " << decnum2 << " = " << decnum1 / decnum2 << endl;
}btw i was just toying around with getline to see what i can do with it, thanks
```

zNinja - September 14, 2016, 3:48 pm

You need to change single quotes to double from 'endl' to "endl"

Also you passed wrong args to getline.

It takes stream, string and optional char

<http://www.cplusplus.com/reference/string/string/getline/>

D_MuLa - September 14, 2016, 4:07 pm

(Sep 14, 2016 03:48 PM)zNinja Wrote: [\[CITE\]](#) You need to change single quotes to double from 'endl' to "endl"

Also you passed wrong args to getline.

It takes stream, string and optional char

<http://www.cplusplus.com/reference/string/string/getline/> stream? you lost me there. The book im studying with says to pass in a source of input, in my case it was cin (well at least i thought it was)?

Oscura - September 15, 2016, 11:45 am

(Sep 14, 2016 04:07 PM)D_MuLa Wrote: [\[CITE\]](#) stream? you lost me there. The book im studying with says to pass in a source of input, in my case it was cin (well at least i thought it was)?

cin = the standard input stream. (cout is the standard output stream).

D_MuLa - September 15, 2016, 5:26 pm

(Sep 15, 2016 11:45 AM)Oscura Wrote: [\[CITE\]](#) cin = the standard input stream. (cout is the standard output stream).so how were my args incorrect?

and my program crashes when i input a double

<https://gyazo.com/57d1d6c1aedb600db42cf71fc542ddc2>

Kantos - September 16, 2016, 12:36 pm

(Sep 15, 2016 05:26 PM)D_MuLa Wrote: [\[CITE\]](#) so how were my args incorrect?

and my program crashes when i input a double

<https://gyazo.com/57d1d6c1aedb600db42cf71fc542ddc2>

What kind of book are you reading?

The args for getline

<http://www.cplusplus.com/reference/string/string/getline/>

(1)

```
istream& getline (istream& is, string& str, char delim);
```

(2)

```
istream& getline (istream& is, string& str);
```

I'm astonished that your code even compiled, when I compile it using g++ on Linux I get all the expected errors and warnings since you're shoving in arguments to getline that it doesn't accept and you're putting multiple characters into single quotes when it should only be one character.

Getline will simply read from the stream is until it hits the newline character '\n' or the optional character delim and puts the read characters it into the string str. Getline doesn't do anything else, it doesn't parse out integers or floats for you. It just does what it's called: gets a line.

Use the stream extract operator >> or parse the string getline returns or something else.

D_MuLa - September 16, 2016, 5:26 pm

(Sep 16, 2016 12:36 PM)Kantos Wrote: [\[CITE\]](#) What kind of book are you reading?

The args for getline

<http://www.cplusplus.com/reference/string/string/getline/>

(1)

```
istream& getline (istream& is, string& str, char delim);
```

(2)

```
istream& getline (istream& is, string& str);
```

I'm astonished that your code even compiled, when I compile it using g++ on Linux I get all the expected errors and warnings since you're shoving in arguments to getline that it doesn't accept and you're putting multiple characters into single quotes when it should only be one character.

Getline will simply read from the stream is until it hits the newline character '\n' or the optional character delim and puts the read characters it into the string str. Getline doesn't do anything else, it doesn't parse out integers or floats for you. It just does what it's called: gets a line.

Use the stream extract operator >> or parse the string getline returns or something else. oh well that specific code compiled because i wasnt using getline instead i just used cin lol but i'm going to practice with getline more. Args are a bit confusing right now lol but some more studying and i should get the hang of it, thanks

0xDEAD10CC - September 16, 2016, 6:53 pm

getline() reads data into a string, not an int or a double or any numeric type. You can't do that with getline(). You can read data from the stream object using getline(), then parse it from there, but not directly.

D_MuLa - September 17, 2016, 8:44 am

(Sep 16, 2016 06:53 PM)0xDEAD10CC Wrote: [\[CITE\]](#) getline() reads data into a string, not an int or a double or any numeric type. You can't do that with getline(). You can read data from the stream object using getline(), then parse it from there, but not directly.what do you mean by parse it ?

0xDEAD10CC - September 17, 2016, 11:19 am

(Sep 17, 2016 08:44 AM)D_MuLa Wrote: [\[CITE\]](#) what do you mean by parse it ?
I mean parsing it from the string object from the data that populates it with getline()...

Dostoevsky - September 17, 2016, 3:48 pm

(Sep 17, 2016 08:44 AM)D_MuLa Wrote: [\[CITE\]](#) what do you mean by parse it ?
Computers don't represent decimals in our text notation (e.g. 10.5). So, when you input that into a std::cin stream, it literally just reads in those characters and treats it as a string. You'd have to parse the decimal out of the string and convert it into the primitive type's representation (sign, mantissa, exponent).

Here's an example of reading decimal numbers into doubles from strings (parsing from strings):

PHP Code

```
<?php
#include <iostream>
#include <string>
```

```
int main()
{
    std::string strDecimal;
    std::cout << "Enter a decimal, bro: ";
    getline (std::cin, strDecimal);
```

// by parsing the string and producing a primitive, we can now perform arithmetic because the computer can treat it as a number now rather than a string.

```
    double decimal = std::stod(strDecimal);
    std::cout << "Performing some arithmetic: " << (decimal * 3.5f) << std::endl;
}
```

The arithmetic is evidence of the computer's new representation of the decimal number (into a double; there is also a float type that is less precise - float is 4 bytes, double is double that so 8 bytes) since it can now treat it as a number in the form of 8 bytes rather than a string in the form of how many characters make up your decimal string notation.

D_MuLa - September 17, 2016, 5:08 pm

(Sep 17, 2016 03:48 PM)Dostoevsky Wrote: [\[CITE\]](#) Computers don't represent decimals in our text notation (e.g. 10.5). So, when you input that into a std::cin stream, it literally just reads in those characters and treats it as a string. You'd have to parse the decimal out of the string and convert it into the primitive type's representation (sign, mantissa, exponent).

Here's an example of reading decimal numbers into doubles from strings (parsing from strings):

PHP Code

```
<?php
#include <iostream>
#include <string>
```

```
int main()
{
    std::string strDecimal;
    std::cout << "Enter a decimal, bro: ";
    getline (std::cin, strDecimal);
```

// by parsing the string and producing a primitive, we can now perform arithmetic because the computer can treat it as a number now rather than a string.

```
    double decimal = std::stod(strDecimal);
    std::cout << "Performing some arithmetic: " << (decimal * 3.5f) << std::endl;
}
```

The arithmetic is evidence of the computer's new representation of the decimal number (into a double; there is also a float type that is less precise - float is 4 bytes, double is double that so 8 bytes) since it can now treat it as a number in the form of 8 bytes rather than a string in the form of how many characters make up your decimal string notation. hey bro that code isn't compiling, what did you mean to put when for "std::stod"?

Dostoevsky - September 17, 2016, 5:13 pm

(Sep 17, 2016 05:08 PM)D_MuLa Wrote: [\[CITE\]](#) hey bro that code isn't compiling, what did you mean to put when for "std::stod"?

The std::stod function is part of C++11. Add the -std=c++11 compile flag, or use atof() from the C standard library like this:

PHP Code

```
<?php
#include <iostream>
#include <string>
#include <cstdlib>
```

```
int main()
{
    std::string strDecimal;
    std::cout << "Enter decimal, bro: ";
    getline (std::cin, strDecimal);

    double decimal = atof(strDecimal.c_str());
    std::cout << (decimal * 3.5) << std::endl;
}
```

D_MuLa - September 17, 2016, 5:27 pm

(Sep 17, 2016 05:13 PM)Dostoevsky Wrote: [\[CITE\]](#) The std::stod function is part of C++11. Add the -std=c++11 compile flag, or use atof() from the C standard library like this:

PHP Code

```
<?php
#include <iostream>
```

```
#include <string>
#include <cstdlib>
```

```
int main()
{
    std::string strDecimal;
    std::cout << "Enter decimal, bro: ";
    getline (std::cin, strDecimal);

    double decimal = atof(strDecimal.c_str());
    std::cout << (decimal * 3.5) << std::endl;
}
```

atof() worked, thanks alot. dont know wth it is but it worked lol

Kantos - September 18, 2016, 3:35 pm

(Sep 17, 2016 03:48 PM)Dostoevsky Wrote: [\[CITE\]](#) Computers don't represent decimals in our text notation (e.g. 10.5). So, when you input that into a std::cin stream, it literally just reads in those characters and treats it as a string. You'd have to parse the decimal out of the string and convert it into the primitive type's representation (sign, mantissa, exponent).

Here's an example of reading decimal numbers into doubles from strings (parsing from strings):

PHP Code

```
<?php
#include <iostream>
#include <string>
```

```
int main()
{
    std::string strDecimal;
    std::cout << "Enter a decimal, bro: ";
    getline (std::cin, strDecimal);
```

// by parsing the string and producing a primitive, we can now perform arithmetic because the computer can treat it as a number now rather than a string.

```
    double decimal = std::stod(strDecimal);
    std::cout << "Performing some arithmetic: " << (decimal * 3.5f) << std::endl;
}
```

The arithmetic is evidence of the computer's new representation of the decimal number (into a double; there is also a float type that is less precise - float is 4 bytes, double is double that so 8 bytes) since it can now treat it as a number in the form of 8 bytes rather than a string in the form of how many characters make up your decimal string notation. (Sep 17, 2016 05:13 PM)Dostoevsky Wrote: [\[CITE\]](#) The std::stod function is part of C++11. Add the -std=c++11 compile flag, or use atof() from the C standard library like this:

PHP Code

```
<?php
#include <iostream>
#include <string>
#include <cstdlib>
```

```
int main()
```

```

{
    std::string strDecimal;
    std::cout << "Enter decimal, bro: ";
    getline (std::cin, strDecimal);

    double decimal = atof(strDecimal.c_str());
    std::cout << (decimal * 3.5) << std::endl;
}

```

This is kind of bad....

First of, you're doing no error checking here. `atof()` will throw an exception if you enter gibberish instead of a decimal or a number unrepresentable by a double, which if unhandled like in your code will terminate the program. You should be catching these errors with try-catch.

`atof()` will simply return 0.0 if you enter gibberish. It causes undefined behavior if it reads a number that isn't representable by a double, and doesn't set any `errno` values in either case.

The proper way to convert string to floating point in C is to use `strtod`, which gives you more robust error checking, although it's not intuitive for a newbie.

Secondly, if you're trying to parse a more complex string (for example, if you're making a calculator program and want to be able to parse a line `X <op> Y`) then what you posted is useless.

Honestly for OP's use, just use `cin >>` and then do the proper error checking.

Example:

```

Code #include <iostream>
#include <string>
#include <limits>

```

```

using namespace std;

```

```

int main()
{
    int wholenum1;
    int wholenum2;
    double decnum1;
    double decnum2;

    cout << "Enter two integers to be divided:\n";
    if(!(cin >> wholenum1 >> wholenum2)){
        cerr << "You didn't enter integers you idiot\n";
        //clear error
        cin.clear();
        //flush bad characters from buffer/line
        cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
    else{
        cout << wholenum1 << " / " << wholenum2 << " = " << wholenum1 / wholenum2 << endl;
    }

    cout << "Enter two integers or decimals to be divided:\n";

```

```

if(!(cin >> decnum1 >> decnum2)){
    cerr << "You didn't enter integers or decimal numbers you idiot\n";
    //clear error
    cin.clear();
    //flush bad characters from buffer/line
    cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
else{
    cout << decnum1 << " / " << decnum2 << " = " << decnum1 / decnum2 << endl;
}

//do other stuff
}

```

Dostoevsky - September 18, 2016, 4:38 pm

(Sep 18, 2016 03:35 PM)Kantos Wrote: [\[CITE\]](#) This is kind of bad....

First of, you're doing no error checking here. `atod()` will throw an exception if you enter gibberish instead of a decimal or a number unrepresentable by a double, which if unhandled like in your code will terminate the program. You should be catching these errors with try-catch.

`atof()` will simply return 0.0 if you enter gibberish. It causes undefined behavior if it reads a number that isn't representable by a double, and doesn't set any `errno` values in either case.

The proper way to convert string to floating point in C is to use `strtod`, which gives you more robust error checking, although it's not intuitive for a newbie.

Secondly, if you're trying to parse a more complex string (for example, if you're making a calculator program and want to be able to parse a line `X <op> Y`) then what you posted is useless.

Honestly for OP's use, just use `cin >>` and then do the proper error checking.

Example:

```

Code #include <iostream>
#include <string>
#include <limits>

```

```

using namespace std;

```

```

int main()
{
    int wholenum1;
    int wholenum2;
    double decnum1;
    double decnum2;

    cout << "Enter two integers to be divided:\n";
    if(!(cin >> wholenum1 >> wholenum2)){
        cerr << "You didn't enter integers you idiot\n";
        //clear error
        cin.clear();
        //flush bad characters from buffer/line
        cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
}

```

```

    }
    else{
        cout << wholenum1 << " / " << wholenum2 << " = " << wholenum1 / wholenum2 << endl;
    }

    cout << "Enter two integers or decimals to be divided:\n";
    if(!(cin >> decnum1 >> decnum2)){
        cerr << "You didn't enter integers or decimal numbers you idiot\n";
        //clear error
        cin.clear();
        //flush bad characters from buffer/line
        cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
    else{
        cout << decnum1 << " / " << decnum2 << " = " << decnum1 / decnum2 << endl;
    }

    //do other stuff
}
atod()? I think you mean atof(). It also throws nothing.
The function I already suggested OP used, std::stod(), does throw exceptions though. EDIT: After
re-reading your post, did you make some typo or something?

```

I never wrote the code to demonstrate input validation or exception handling, I wrote it to show the function being used. The assignment OP is writing this for wouldn't care for your exception handling. Also, I resigned to using std::getline because OP used it in his question. Error handling isn't really required for assignments such as OPs.

Please don't assume the code I post here is the epitome of my programming ability (you seemed rather patronising in your post - your comments also must assume I'm some kind of lesser mortal), it's merely a quick answer to an assignment (an assignment whose grader won't care for any error checking).

Also, if I was writing a real calculator that accepted arithmetic expressions, I'd use Flex & Bison (or, more recently, ANTLR). As much as I think your post will help OP, please don't direct it at me. Thanks.