Abusing cin in while loops for int assignment

simply trying to compare two user defined vectors to see if they are equal, current code:

```
vector<int> ivec1, ivec2; //vectors, uninitialized

int temp1;

cout << "Enter integers to be stored in ivec1." << endl;

while(cin >> temp1) //takes input from user and creates new element in the vector to store
it
{
    ivec1.push_back(temp1);
}

int temp2;

cout << "Enter integers to be stored in ivec2." << endl;

while(cin >> temp2) //same as above with different vector
{
    ivec2.push_back(temp2);
}

if(ivec1 == ivec2)
    cout << "ivec1 and ivec2 are equal!" << endl;
else
    cout << "ivec1 and ivec2 are NOT equal!" << endl;
```

So far it lets me assign values to ivec1 just fine, but as I exit the while loop by entering a letter to make cin fail, it skips the second while block. Out of curiosity I tried putting in other cin statements after the first while loop, and it ignores them all as well.

Does forcing cin to fail cause the program to ignore all other calls for it or something, or is there another problem? If so, how can I get this program to do what I want?

screenshot for your viewing pleasure: http://img695.imageshack.us/img695/2677/cinfailure.png

*PS. having temp1 and temp2 was just me trying to figure out if using the same int for both assignment loops was causing the problem, anyway I just figured I'd leave it there

c++ loops while-loop cin

edited Oct 25 '13 at 2:57
Prashant Kumar
6,072 10 27 46

asked Jul 28 '11 at 14:55
user863492
32 4

4 Answers

You would have to do `cin.clear()` to reset the stream state. Then you will have to make sure that the offending character is read from the stream (using one of the techniques described here), so that the next input operation does not fail as well.

answered Jul 28 '11 at 14:58
Björn Pollex
45.1k 9 102 191

You mean that you a doing a ctrl-D to give end-of-file for the first loop.

The problem with that is that once EOF is achived it will persist and the second loop will also see the EOF and never read anything.

Instead use a terminating charater such as a blank line or a '.' and specifically test for that in toy while loop instead of `while (cin >> tmp1)`

answered Jul 28 '11 at 14:58

Soren
7,548 1 13 38

Use `cin.clear()` between the loops. This command resets the state of the stream back to a usable one.

Might be helpful to know that you don't always have to enter an invalid character to exit a loop, you can also use (on windows) a ctrl-z (ctrl-d on other systems) on the console, which stimulates an EOF. You'd still have to `cin.clear()` (because an EOF still invalidates the stream) - but it's not as dangerous

answered Jul 28 '11 at 14:59

Schnommus
2,021 7 18

Thanks for the tip on ctr-z, I did not know that! – user863492 Jul 28 '11 at 15:54

@Martin: That's true; but in this case it would be appropriate to read inputs from 2 separate files anyway. – Schnommus Jul 28 '11 at 16:37

Depending on two EOFs creates a program that can't be fed from a single input in a pipeline... uglier than a sentinel in many ways. Sadly, he cleanest sentinel here's probably an empty line, and to detect it requires using getline to e.g. a string first then creating a istringstream from that then attempting the int parsing, and explicitly checking for trailing garbage on the line. iostreams can be messy! – Tony D Apr 18 '14 at 13:01

When the *first* `while` loop exits because of failure of `std::cin` , it also sets the failure flag internally. All you need to clear that flag by writing the following *after* the first `while` loop:

```
scin.clear();
```

It clears all the failure flag, so that `cin` can be used to read further inputs.

answered Jul 28 '11 at 14:59

Nawaz
179k 45 383 608
