"break;" out of "if" statement?

Asked 10 years, 5 months ago Modified 2 years, 7 months ago Viewed 276k times

56

Can you break out of an if statement or is it going to cause crashes? I'm starting to acquaint myself with C, but this seems controversial. The first image is from a book on C ("Head First C") and the snippet shows code written by Harvard's CS classes staff. What is actually going on and has it something to do with C standards?

> **breaks don't break if statements.**
>
> On January 15, 1990, AT&T's long-distance telephone system crashed, and 60,000 people lost their phone service. The cause? A developer working on the C code used in the exchanges tried to use a `break` to break out of an `if` statement. But `break`s don't break out of `if`s. Instead, the program skipped an entire section of code and introduced a bug that interrupted 70 million phone calls over nine hours.

```
for (size = 0; size < HAY_MAX; size++)
{
    // wait for hay until EOF
    printf("\nhaystack[%d] = ", size);
    int straw = GetInt();
    if (straw == INT_MAX)
        break;

    // add hay to stack
    haystack[size] = straw;
}
printf("\n");
```

c

Share Improve this question Follow

asked Jul 12, 2014 at 13:50
Nick_Tsaftaridis
689 ●1 ●5 ●6

4 You can't `break` out of `if` statement until the `if` is inside a loop. – Ankur Jul 12, 2014 at 15:05

1 that is how the language works.. a break is only generally useful if it is conditional, and to be conditional, it pretty much has to be in an if statement, right – Grady Player Jul 12, 2014 at 15:19

7 The behaviour of the `break` statement is well specified and, generally, well understood. An inexperienced coder may cause crashes though a lack of understanding in many ways. Misuse of the `break` statement isn't special. – CB Bailey Jul 12, 2014 at 15:22

1 I upvoted cause I couldn't find a duplicate... Even if is an obvious question I think it is a valid question. – Alberto Bonsanto Jul 12, 2014 at 15:31

2 This does not break out of the `if`, it breaks out of the `while` loop. C does not allow `break` when not inside a `while`, `for`, or `switch`. – Jongware Jul 12, 2014 at 15:32

5 Answers

Sorted by: Highest score (default)

57

`break` interacts solely with the closest enclosing loop or switch, whether it be a `for`, `while` or `do .. while` type. It is frequently referred to as a goto in disguise, as all loops in C can in fact be transformed into a set of conditional gotos:

```
for (A; B; C) D;
// translates to
A;
goto test;
loop: D;
iter: C;
test: if (B) goto loop;
end:

while (B) D;          // Simply doesn't have A or C
do { D; } while (B);  // Omits initial goto test
continue;             // goto iter;
break;                // goto end;
```

The difference is, `continue` and `break` interact with virtual labels automatically placed by the compiler. This is similar to what `return` does as you know it will always jump ahead in the program flow. Switches are slightly more complicated, generating arrays of labels and computed gotos, but the way break works with them is similar.

The programming error the notice refers to is misunderstanding `break` as interacting with an enclosing block rather than an enclosing loop. Consider:

```
for (A; B; C) {
   D;
   if (E) {
       F;
       if (G) break;   // Incorrectly assumed to break if(E), breaks for()
       H;
   }
   I;
}
J;
```

Someone thought, given such a piece of code, that `G` would cause a jump to `I`, but it jumps to `J`. The intended function would use `if (!G) H;` instead.

Share Improve this answer Follow

answered Jul 12, 2014 at 15:38
Yann Vernier
15.9k ● 2 ● 30 ● 27

Your answer seems great. Could you clarify "for (A; B; C) D; // translates to" in which way specifically? Is it assembly or pseudocode you wrote? Seems pseudocode. I'm pretty sure I understand it and I agree with it thought, I'm just asking because I'm very interested.
– Santropedro Jul 4, 2019 at 20:18

1 It's actually not pseudocode aside from the capital letters for some arbitrary expressions or statements; that's how labels and `goto` are written in C. The compiler essentially does these substitutions except that it will pick unique names for the labels belonging to each loop or switch. `default:` and `case X:` are similarly special labels, addressable by `switch` but not `goto`. – Yann Vernier Jul 4, 2019 at 20:29

3

This is actually the conventional use of the `break` statement. If the `break` statement wasn't nested in an `if` block the `for` loop could only ever execute one time.

MSDN lists this as their example for the `break` statement.

Share Improve this answer Follow

answered Jul 12, 2014 at 15:24
Jonathan Mee
38.9k ● 24 ● 144 ● 308

3

As already mentioned that, *break-statement* works **only** with *switches* and *loops*. Here is another way to achieve what is being asked. I am reproducing https://stackoverflow.com/a/257421/1188057 as nobody else mentioned it. It's just a trick involving the do-while loop.

```
do {
  // do something
  if (error) {
    break;
  }
  // do something else
  if (error) {
    break;
  }
  // etc..
} while (0);
```

Though I would prefer the use of *goto-statement*.

Share

Improve this answer

Follow

edited May 23, 2017 at 12:18
Community Bot
1 ● 1

answered Apr 21, 2017 at 5:58
Abhinav
83 ● 8

Like Duff's device, it's a bit questionable whether this is elegant or horrid. The `do { } while (0)` combination is also often used to make macros resemble function calls, such that adding a `;` after them doesn't create an extra null statement. – Yann Vernier Oct 29, 2017 at 17:50

I would recommend using this when handling roots of a function, since you dont really know how many times you would need to loop a function to be close to 0 and encounter the root, and this loop can guarantee an optimal loop time to find the nearest 0.000... approach. – Gastón Saillén Aug 7, 2019 at 15:19

0

I think the question is a little bit fuzzy - for example, it can be interpreted as a question about best practices in programming loops with `if` inside. So, I'll try to answer this question with this particular interpretation.

If you have `if` inside a loop, then in most cases you'd like to know how the loop has ended - was it "broken" by the `if` or was it ended "naturally"? So, your sample code can be modified in this way:

```
bool intMaxFound = false;
for (size = 0; size < HAY_MAX; size++)
{
  // wait for hay until EOF
  printf("\nhaystack[%d] = ", size);
  int straw = GetInt();
  if (straw == INT_MAX)
     {intMaxFound = true; break;}

  // add hay to stack
  haystack[size] = straw;
}
if (intMaxFound)
{
  // ... broken
}
else
{
  // ... ended naturally
}
```

The problem with this code is that the `if` statement is buried inside the loop body, and it takes some effort to locate it and understand what it does. A more clear (even *without* the `break` statement) variant will be:

```
bool intMaxFound = false;
for (size = 0; size < HAY_MAX && !intMaxFound; size++)
```

```
{
  // wait for hay until EOF
  printf("\nhaystack[%d] = ", size);
  int straw = GetInt();
  if (straw == INT_MAX)
     {intMaxFound = true; continue;}

  // add hay to stack
  haystack[size] = straw;
}
if (intMaxFound)
{
  // ... broken
}
else
{
  // ... ended naturally
}
```

In this case you can clearly see (just looking at the loop "header") that this loop *can* end prematurely. If the loop body is a multi-page text, written by somebody else, then you'd thank its author for saving your time.

UPDATE:

Thanks to SO - it has just suggested the already answered question about crash of the AT&T phone network in 1990. It's about a risky decision of C creators to use a single reserved word `break` to exit from both loops and `switch`.

Anyway this interpretation doesn't follow from the sample code in the original question, so I'm leaving my answer as it is.

Share

Improve this answer

Follow

edited May 23, 2017 at 12:34

Community Bot

1 ● 1

answered Jul 13, 2014 at 0:30

HEKTO

4,171 ● 2 ● 29 ● 50

0

You could possibly put the if into a foreach a for, a while or a switch like this

Then break and continue statements will be available

```
foreach ([1] as $i) if ($condition) { // Breakable if
    //some code
    $a = "b";
    // Le break
    break;
    // code below will not be executed
}
for ($i=0; $i < 1 ; $i++) if ($condition) {
    //some code
    $a = "b";
    // Le break
```

```
     break;
     // code below will not be executed
}

switch(0){ case 0: if($condition){

   //some code
   $a = "b";
   // Le break
   break;
   // code below will not be executed

}}



while(!$a&&$a=1) if ($condition) {

   //some code
   $a = "b";
   // Le break
   break;
   // code below will not be executed

}
```

Share Improve this answer Follow

answered May 20, 2022 at 17:03

ThomazPom
150 ● 1 ● 13

The question is about the C language. This answer does not work in C. – DimP Apr 16 at 20:10