WIKIPEDIA
The Free Encyclopedia

# Off-by-one error

An **off-by-one error** or **off-by-one bug** (known by acronyms **OBOE**, **OBO**, **OB1** and **OBOB**) is a logic error that involves a number that differs from its intended value by +1 or −1. An off-by-one error can sometimes appear in a mathematical context. It often occurs in computer programming when a loop iterates one time too many or too few, usually caused by the use of non-strict inequality (≤) as the terminating condition where strict inequality (<) should have been used, or vice versa. Off-by-one errors also stem from confusion over zero-based numbering.

## Cases

### Looping over arrays

Consider an array of items, and items $m$ through $n$ (inclusive) are to be processed. How many items are there? An intuitive answer may be $n - m$, but that is off by one, exhibiting a fencepost error; the correct answer is $n - m + 1$.

For this reason, ranges in computing are often represented by half-open intervals; the range from $m$ to $n$ (inclusive) is represented by the range from $m$ (inclusive) to $n + 1$ (exclusive) to avoid fencepost errors. For example, a loop that iterates five times (from 0 to 4 inclusive) can be written as a half-open interval from 0 to 5:

```
for (index = 0; index < 5; index++)
{
    /* Body of the loop */
}
```

The loop body is executed first of all with `index` equal to 0; `index` then becomes 1, 2, 3, and finally 4 on successive iterations. At that point, `index` becomes 5, so `index < 5` is false and the loop ends. However, if the comparison used were <= (less than or equal to), the loop would be carried out six times: `index` takes the values 0, 1, 2, 3, 4, and 5. Likewise, if `index` were initialized to 1 rather than 0, there would only be four iterations: `index` takes the values 1, 2, 3, and 4. Both of these alternatives can cause off-by-one errors.

Another such error can occur if a do-while loop is used in place of a while loop (or vice versa.) A do-while loop is guaranteed to run at least once.

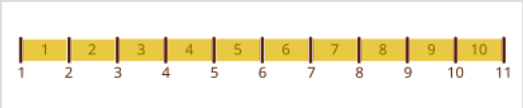
Array-related confusion may also result from differences in programming languages. Numbering from 0 is most common, but some languages start array numbering with 1. Pascal has arrays with user-defined indices. This makes it possible to model the array indices after the problem domain.

### Fencepost error

A **fencepost error** (occasionally called a **telegraph pole, lamp-post,** or **picket fence error**) is a specific type of off-by-one error. An early description of this error appears in the works of Vitruvius.[1] The following problem illustrates the error:

> If you build a straight fence 30 feet long with posts spaced 3 feet apart, how many posts do you need?

The common answer of 10 posts is wrong. This response comes from dividing the length of the fence by the spacing apart from each post, with the quotient being erroneously classified as the number of posts. In actuality, the fence has 10 sections and 11 posts.



A straight fence with 10 sections requires 11 posts. More generally, *n* sections would require *n* + 1 posts.

In this scenario, a fence with $n$ sections will have $n + 1$ posts. Conversely, if the fence contains $n$ posts, it will contain $n - 1$ sections. This relationship is important to consider when dealing with the reverse error. The reverse error occurs when the number of posts is known and the number of sections is assumed to be the same. Depending on the design of the fence, this assumption can be correct or incorrect.

The following problem demonstrates the reverse error:

> If you have $n$ posts, how many sections are there between them?

The interpretation for the fence's design changes the answer to this problem. The correct number of sections for a fence is $n - 1$ if the fence is a free-standing line segment bounded by a post at each of its ends (e.g., a fence between two passageway gaps), $n$ if the fence forms one complete, free-standing loop (e.g., enclosure accessible by surmounting, such as a boxing ring), or $n + 1$ if posts do not occur at the ends of a line-segment-like fence (e.g., a fence between and wall-anchored to two buildings). The precise problem definition must be carefully considered, as the setup for one situation may give the wrong answer for other situations. Fencepost errors come from counting things rather than the spaces between them, or vice versa, or by neglecting to consider whether one should count one or both ends of a row.

Fencepost errors can also occur in units other than length. For example, the Time Pyramid, consisting of 120 blocks placed at 10-year intervals between blocks, is scheduled to take 1,190 years to build (not 1,200), from the installation of the first block to the last block. One of the earliest fencepost errors involved time, where the Julian calendar originally calculated leap years incorrectly, due to counting inclusively rather than exclusively, yielding a leap year every three years rather than every four.

"Fencepost error" can, in rare occasions, refer to an error induced by unexpected regularities in input values,[2] which can (for instance) completely thwart a theoretically efficient binary tree or hash function implementation. This error involves the difference between expected and worst case behaviours of an algorithm.

In larger numbers, being off by one is often not a major issue. In smaller numbers, however, and specific cases where accuracy is paramount, committing an off-by-one error can be disastrous. Sometimes such an issue will also be repeated and, therefore, worsened, by someone passing on an incorrect calculation, if the following person makes the same kind of mistake again (of course, the error might also be reversed).

An example of this error can occur in the computational language MATLAB with the `linspace()` linear interpolation function, whose parameters are (*`lower value, upper value, number of values`*) and not (*`lower value, upper value, number of increments`*). A programmer who misunderstands the third parameter to be the number of increments might hope that `linspace(0,10,5)` would achieve a sequence `[0, 2, 4, 6, 8, 10]` but instead would get `[0, 2.5, 5, 7.5, 10]`.

### Security implications

A common off-by-one error which results in a security-related bug is caused by misuse of the C standard library `strncat` routine. A common misconception with `strncat` is that the guaranteed null termination will not write beyond the maximum length. In reality it will write a terminating null character one byte beyond the maximum length specified. The following code contains such a bug:

```
void foo (char *s)
{
    char buf[15];
    memset(buf, 0, sizeof(buf));
    strncat(buf, s, sizeof(buf)); // Final parameter should be: sizeof(buf)-1
}
```

Off-by-one errors are common in using the C library because it is not consistent with respect to whether one needs to subtract 1 byte – functions like `fgets()` and `strncpy` will never write past the length given them (`fgets()` subtracts 1 itself, and only retrieves (length – 1) bytes), whereas others, like `strncat` will write past the length given them. So the programmer has to remember for which functions they need to subtract 1.

On some systems (little endian architectures in particular) this can result in the overwriting of the least significant byte of the frame pointer. This can cause an exploitable condition where an attacker can hijack the local variables for the calling routine.

One approach that often helps avoid such problems is to use variants of these functions that calculate how much to write based on the total length of the buffer, rather than the maximum number of characters to write. Such functions include `strlcat` and `strlcpy`, and are often considered "safer" because they make it easier to avoid accidentally writing past the end of a buffer. (In the code example above, calling `strlcat(buf, s, sizeof(buf))` instead would remove the bug.)

## See also

- Boundary-value analysis
- Pigeonhole principle

- Rounding error
- Zero-based numbering

# References

## Citations


1. Moniot, Robert K., *Who first described the "fence-post error?"* (https://web.archive.org/web/20160305221341/http://www.dsm.fordham.edu/~moniot/Opinions/fencepost-error-history.shtml), Fordham University, archived from the original (http://www.dsm.fordham.edu/~moniot/Opinions/fencepost-error-history.shtml) on 2016-03-05, retrieved 2016-07-07.
2. Raymond, Eric. "The Jargon File" (http://www.catb.org/~esr/jargon/html/F/fencepost-error.html). Retrieved 17 May 2021.


## Sources


- *An earlier version of this article was based on fencepost error (http://foldoc.org/fencepost%20error) at FOLDOC (http://foldoc.org), used with permission.*
- Dijkstra, Edsger Wybe (May 2, 2008). "Why numbering should start at zero (EWD 831)" (http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html). *E. W. Dijkstra Archive*. University of Texas at Austin. Retrieved 2011-03-16.
- In the Common Weakness Enumeration system this issue is listed as CWE-193: Off-by-one Error (https://cwe.mitre.org/data/definitions/193.html)


# Further reading


- Parker, Matt (2021). *Humble Pi: When Math Goes Wrong in the Real World*. Riverhead Books. ISBN 978-0593084694.