What is an ideal variable naming convention for loop variables? [closed]

Asked 16 years, 3 months ago Modified 5 years ago Viewed 42k times

52

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 11 years ago.

If you are writing a *simple* little loop, what *should* you name the counter?

Provide example loops!

language-agnostic naming-conventions

Share

Improve this question

Follow

edited Feb 25, 2013 at 16:20

Mog

103k ● 38 ● 197 ● 231

asked Sep 19, 2008 at 10:55

just mike
1,172 ● 3 ● 12 ● 22

4 It's a shame that this was closed. Some of the answers provide really useful insights. Please open this! – Roland Aug 15, 2016 at 7:52

"struct_" + context. This is the purpose of a struct in C, so I use the prefix struct whenever I develop source code data structures that aren't seen outside the source code. en.wikipedia.org/wiki/Struct_(C_programming_language) – JustBeingHelpful May 22 at 17:42

28 Answers

Sorted by: Highest score (default)

47

I always use a **meaningful name** unless it's a single-level loop and the variable has no meaning other than "the number of times I've been through this loop", in which case I use `i`.

When using meaningful names:

- the code is more understandable to colleagues reading your code,
- it's easier to find bugs in the loop logic, and
- text searches for the variable name to return relevant pieces of code operating on the same data are more reliable.

Example - spot the bug

It can be tricky to find the bug in this nested loop using single letters:

```
int values[MAX_ROWS][MAX_COLS];

int sum_of_all_values()
{
    int i, j, total;

    total = 0;
    for (i = 0; i < MAX_COLS; i++)
        for (j = 0; j < MAX_ROWS; j++)
             total += values[i][j];
    return total;
}
```

whereas it is easier when using meaningful names:

```
int values[MAX_ROWS][MAX_COLS];

int sum_of_all_values()
{
    int row_num, col_num, total;

    total = 0;
    for (row_num = 0; row_num < MAX_COLS;
row_num++)
        for (col_num = 0; col_num < MAX_ROWS;
col_num++)
             total += values[row_num][col_num];
    return total;
}
```

Why `row_num`? - rejected alternatives

In response to some other answers and comments, these are some alternative suggestions to using `row_num` and

`col_num` and why I choose not to use them:

- **`r`** and **`c`**: This is slightly better than `i` and `j`. I would only consider using them if my organisation's standard were for single-letter variables to be integers, and also always to be the first letter of the equivalent descriptive name. The system would fall down if I had two variables in the function whose name began with "r", and readability would suffer even if other objects beginning with "r" appeared anywhere in the code.
- **`rr`** and **`cc`**: This looks weird to me, but I'm not used to a double-letter loop variable style. If it were the standard in my organisation then I imagine it would be slightly better than `r` and `c`.
- **`row`** and **`col`**: At first glance this seems more succinct than `row_num` and `col_num`, and just as descriptive. However, I would expect bare nouns like "row" and "column" to refer to structures, objects or pointers to these. If `row` could mean *either* the row structure itself, *or* a row number, then confusion will result.
- **`iRow`** and **`iCol`**: This conveys extra information, since `i` can mean it's a loop counter while `Row` and `Col` tell you what it's counting. However, I prefer to be able to read the code almost in English:
    - `row_num < MAX_COLS` reads as "the **row num**ber is **less than** the **max**imum (number of) **col**umn**s**";

- `iRow < MAX_COLS` at best reads as "the **integer loop counter** for the **row** is **less than** the **max**imum (number of) **col**umn**s**".
- It may be a personal thing but I prefer the first reading.

An alternative to `row_num` I would accept is `row_idx`: the word "index" uniquely refers to an array position, unless the application's domain is in database engine design, financial markets or similar.

My example above is as small as I could make it, and as such some people might not see the point in naming the variables descriptively since they can hold the whole function in their head in one go. In real code, however, the functions would be larger, and the logic more complex, so decent names become more important to aid readability and to avoid bugs.

In summary, my aim with all variable naming (not just loops) is to be **completely unambiguous**. If *anybody* reads any portion of my code and can't work out what a variable is for immediately, then I have failed.

edited Jan 20, 2009 at 11:15

answered Sep 19, 2008 at 11:38

Paul Stephenson
69.3k ● 9 ● 52 ● 51

1 I find the first to be more readable. The longer names (especially the "_num" everywhere) just adds to the visual clutter. – Derek Park Sep 19, 2008 at 23:45

3 The problem is not with the index variables, but I think a new code reader is less likely to spot that there *is* a bug in the first example. In the second, the expression "row_num < MAX_COLS" should definitely set alarm bells ringing, even with a casual browse through the code. – Paul Stephenson Sep 20, 2008 at 8:30

1 Why not abbreviate row_num to r and col_num to c? The meaning of the variable is just as clear to a casual reader, and you don't have so much visual clutter to deal with. – Pitarou Sep 20, 2008 at 21:40

1 I think Petr K and Derek haven't maintained other people's code very much. The second example is a lot easier to read and, more importantly, understand. Do some maintenance programming for a while and you'll long for meaningful variable names. – Onorio Catenacci Sep 24, 2008 at 13:20

1 Paul: you got my vote, i totally agree with your convention.using i & j you having to examine the way its used in the loop to find the bug (context). with *_num you can see the potential bug without the context. i & j can also be easily confused, as sometimes you do row major and other column major – mattlant Sep 27, 2008 at 2:12

37

1) For normal old style small loops - i, j, k - If you need more than 3 level nested loops, this means that either the algorithm is very specific and complex, or you should consider refactoring the code.

Java Example:

```
for(int i = 0; i < ElementsList.size(); i++) {
  Element element = ElementsList.get(i);
  someProcessing(element);
  ....
}
```

2) For the new style java loops like `for(Element element: ElementsList)` it is better to use normal meanigful name

Java Example:

```
for(Element element: ElementsList) {
  someProcessing(element);
  ....
}
```

3) If it is possible with the language you use, convert the loop to use iterator

Java Iterator Example: click here

Share Improve this answer

Follow

edited Nov 26, 2019 at 8:19

extremepayne

49 ● 1 ● 7

answered Sep 19, 2008 at 11:06

m_pGladiator

8,620 ● 8 ● 45 ● 61

1 i certainly agree with @Ande Turner that "i" *is* self descriptive for 99.9% of programmers. however, you've made my point for me in your Java example... if you "refactor" your loop to not use "i" -- you can't do a global replace on the loop

because of "someProcessing". you could if it was "ii" ;-) – just mike Sep 19, 2008 at 18:56

@just mike: You can do a global replace: \bi\b – jfs Sep 25, 2008 at 1:10

1 An interesting alternative I learned some years ago is to use "idx" (short for "index") and then, if you have a second or nested loop, use "jdx", "kdx" etc. So it continues the i,j,k tradition but is sort of readable. – mtruesdell Oct 29, 2008 at 15:09

Never try to use 'i' in for loop for the interviews. just for Competitive coding is OK! – Vaibhav Pallod Mar 26, 2022 at 13:34

14

Examples: *. . . In Java*

Non-Iterative Loops:

Non-Nested Loops: . . . The Index is a value.

> *. . . using* `i`*, as you would in Algebra, is the most common practise . . .*

```
for (int i = 0; i < LOOP_LENGTH; i++) {

    // LOOP_BODY
}
```

Nested Loops: . . . Differentiating Indices lends to comprehension.

> . . . *using a descriptive suffix* . . .

```
for (int iRow = 0; iRow < ROWS; iRow++) {

    for (int iColumn = 0; iColumn < COLUMNS;
iColumn++) {

        // LOOP_BODY
    }
}
```

`foreach` **Loops:** . . . An `Object` needs a name.

> *. . . using a descriptive name . . .*

```
for (Object something : somethings) {

    // LOOP_BODY
}
```

Iterative Loops:

`for` **Loops:** . . . Iterators reference Objects. An Iterator it

is neither; an Index, nor an Indice.

> . . . `iter` *abreviates an Iterators purpose* . . .

```
for (Iterator iter = collection.iterator();
iter.hasNext(); /* N/A */) {

    Object object = iter.next();

    // LOOP_BODY
}
```

`while` Loops: . . . Limit the scope of the Iterator.

> . . . *commenting on the loops purpose* . . .

```
/* LOOP_DESCRIPTION */ {

    Iterator iter = collection.iterator();

    while (iter.hasNext()) {

        // LOOP_BODY
    }
}
```

> *This last example reads badly without comments, thereby encouraging them. It's verbose perhaps, but useful in scope limiting loops in C.*

Share Improve this answer Follow

edited Nov 23, 2012 at 2:04

community wiki
12 revs
Ande

14

My experience is that most people use single letters, e.g.: `i`, `j`, `k`, ... or `x`, `y`, or `r`, `c` (for row/column) or `w`, `h` (for width/height) , etc.

But I learned a great alternative a long time ago, and have used it ever since: double letter variables.

```
// recommended style                    •    //
"typical" single-letter style
                                        •
for (ii=0; ii<10; ++ii) {               •    for (i=0;
i<10; ++i) {
    for (jj=0; jj<10; ++jj) {           •        for
(j=0; j<10; ++j) {
        mm[ii][jj] = ii * jj;           •
m[i][j] = i * j;
    }                                   •        }
}                                       •    }
```

In case the benefit isn't immediately obvious: searching through code for any single letter will find many things that *aren't* what you're looking for. The letter `i` occurs quite often in code where it isn't the variable you're looking for.

Share Improve this answer

edited Dec 18, 2015 at 6:10

Follow

James Dunn
8,254 ● 14 ● 54 ● 87

answered Sep 19, 2008 at 10:55

just mike
1,172 ● 3 ● 12 ● 22

3 They look quite unreadable to me, especially parallel-line letters (e.g., ii, ll, jj, etc.) – Jon Limjap Sep 19, 2008 at 10:58

4 Besides, why do you need to search for loop variables when they shouldn't reside anywhere beyond the loop? Not unless you have tons of code in between -- which isn't a good idea. – Jon Limjap Sep 19, 2008 at 11:02

7 If you need to search for your iteration-variables, then it's refactoring-time! – Magnar Sep 19, 2008 at 11:14

1 Switch to an editor that lets you include word boundaries in search patterns, e.g. '\<i\>' or has a "whole word only" checkbox. – finnw Sep 19, 2008 at 11:17

11 Maybe a bit personal thing, but I really do not like this naming convention. I find it ugly, confusing and higly unreadable. -1 – petr k. Dec 20, 2008 at 3:43

9

Always try to name the variable something meaningful and in context.

If you cannot decide, then use "index", if only so that someone else (maybe you!) can more easily click on it for refactoring later.

Paul Stephenson See this answer for an example.

Share Improve this answer

Follow

edited May 23, 2017 at 12:18

Community Bot

1 ● 1

answered Sep 19, 2008 at 10:59

Nescio

28.4k ● 10 ● 55 ● 76

People may argue that naming index variables is pointless, but I find it keep you in the habit of giving conscious thought to variable naming. Besides, if you're using C, you need to make sure you avoid variable name clashes with any other loops in the same method. – Mal Ross Sep 19, 2008 at 11:57

1 Mal, use C99 and declare your loop variables in the loop heading (assuming you can choose to use C99, of course). – Derek Park Sep 19, 2008 at 23:47

Ooops. Thanks for the heads-up. It's clearly been a long time since I used a C compiler. – Mal Ross Sep 24, 2008 at 13:45

1 Thing is, `i` is meaningful to pretty much anyone who programs. Why bother with typing `index` when `i` is so much shorter, and, in my opinion, *more* readable? – rydwolf Oct 3, 2019 at 23:55

7

`i`

if I have a nested loop then also `j`.

This convention is so common that if you manage to come across a variable `i` in a block of code that you can't see the start of you still instantly recognise it for what it is.

Share Improve this answer

Follow

answered Sep 19, 2008 at 11:05

AnthonyWJones

189k 35 235 307

7

I use **i**, **j**, **k** (or **r** & **c** for row-column looping). If you need **more than three loop variables in a method**, the the method is probably too long and complex and your code would likely **benefit from splitting the method up** into more methods and **naming** them **properly**.

Share Improve this answer

Follow

answered Sep 20, 2008 at 17:22

petr k.

8,100 7 43 52

5

I use i, ii, iii, iv, v ... Never got higher than iii, though.

Share Improve this answer

Follow

answered Sep 19, 2008 at 23:19

Oddmund

1,344 13 21

4

I use single letters only when the loop counter is an index. I like the thinking behind the double letter, but it makes the code quite unreadable.

answered Sep 19, 2008 at 11:00

Vaibhav
11.4k ● 11 ● 53 ● 71

3

If the counter is to be used as an index to a container, I use `i`, `j`, `k`.

If it is to be used to iterate over a range (or perform a set number of iterations), I often use `n`. Though, if nesting is required I'll usually revert to `i`, `j`, `k`.

In languages which provide a `foreach`-style construct, I usually write like this:

```
foreach widget in widgets do
  foo(widget)
end
```

I think some people will tell me off for naming `widget` so similarly to `widgets`, but I find it quite readable.

answered Sep 19, 2008 at 11:34

Jez
31 ● 2

I often use k to iterate over range and n or n_something to mark the size of the range. Otherwise I am complete agreement with you. – Antti Rasinen Sep 19, 2008 at 11:48

1 I like that you used widget for each element in widgets – epochwolf Sep 27, 2008 at 0:43

I have long used the i/j/k naming scheme. But recently I've started to adapt a more consequent naming method.

I allready named all my variables by its meaning, so why not name the loop variable in the same deterministic way.

As requested a few examples:

If you need to loop trough a item collection.

```
for (int currentItemIndex = 0; currentItemIndex <
list.Length; currentItemIndex++)
{
    ...
}
```

But i try to avoid the normal for loops, because I tend to want the real item in the list and use that, not the actual position in the list. so instead of beginning the for block with a:

```
Item currentItem = list[currentItemIndex];
```

I try to use the foreach construct of the language. which transforms the.

```
for (int currentItemIndex = 0; currentItemIndex <
list.Length; currentItemIndex++)
{
    Item currentItem = list[currentItemIndex];
    ...
}
```

into

```
foreach (Item currentItem in list)
{
   ...
}
```

Which makes it easier to read because only the real meaning of the code is expressed (process the items in the list) and not the way we want to process the items (keep an index of the current item en increase it until it reaches the length of the list and thereby meaning the end of the item collection).

The only time I still use one letter variables is when I'm looping trough dimensions. But then I will use x, y and sometimes z.

Share Improve this answer Follow

edited Oct 1, 2008 at 8:25

answered Sep 19, 2008 at 13:00

Davy Landman

15.4k • 6 • 53 • 76

2

Like a previous poster, I also use ii, jj,.. mainly because in many fonts a single i looks very similar to 1.

Share Improve this answer

Follow

answered Sep 19, 2008 at 11:25

Emile

2,230 ● 28 ● 36

1 why not use a different letter then? or a whole word? – Jon Limjap Sep 19, 2008 at 11:36

1 It's also much easier to search for / highlight "ii" or "jj" in an editor. – John Millikin Sep 19, 2008 at 19:18

3 Once again, I would have to question why you need to search for loop variables. Unless you are coding complex (complicated as oppose to Complex!) mathematical algorithms, loop vars should have a very limited scope. – Mitch Wheat Sep 27, 2008 at 2:02

1 Code is written once and read many many times. So it should be immediately clear what you read. For very small scope, a short "word" should be short but still distinguishable from a 1. I think ii en jj are good balance. For larger scopes I agree with the previous commenter and use a word. – Emile Nov 13, 2008 at 14:25

1

I use "counter" or "loop" as the variable name. Modern IDEs usually do the word completion , so longer variable names are not as tedious to use. Besides , to name the variable to its functionality makes it clear to the programmer who is going to maintain your code as to what your intentions were.

Share Improve this answer

Follow

answered Sep 19, 2008 at 11:37

Learning

8,175 ● 3 ● 38 ● 47

Perl standard

1

In Perl, the standard variable name for an inner loop is **$_**. The **for**, **foreach**, and **while** statements default to this variable, so you don't need to declare it. Usually, **$_** may be read like the neuter generic pronoun "it". So a fairly standard loop might look like:

```
foreach (@item){
    $item_count{$_}++;
}
```

In English, that translates to:

> For each item, increment it's item_count.

Even more common, however, is to not use a variable at all. Many Perl functions and operators default to **$_**:

```
for (@item){
    print;
}
```

In English:

> For [each] item, print [it].

This also is the standard for counters. (But counters are used far less often in Perl than in other languages such

as C). So to print the squares of integers from 1 to 100:

```
for (1..100){
    print "$_*$_\n";
}
```

Since only one loop can use the **$_** variable, usually it's used in the inner-most loop. This usage matches the way English usually works:

> For each car, look at each tire and check it's pressure.

In Perl:

```
foreach $car (@cars){
    for (@{$car->{tires}}){
        check_pressure($_);
    }
}
```

As above, it's best to use longer, descriptive names in outer loops, since it can be hard to remember in a long block of code what a generic loop variable name really means.

Occasionally, it makes sense to use shorter, non-descriptive, generic names such as **$i**, **$j**, and **$k**, rather than **$_** or a descriptive name. For instance, it's useful to match the variables use in a published algorithm, such as cross product.

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Sep 19, 2008 at 19:13

Jon Ericson

21.5k ● 12 ● 102 ● 151

1

The first rule is that the length of the variable name should match the scope of the variable. The second rule is that meaningful names make bugs more shallow. The third rule is that if you feel like adding comment to a variable name, you chose the wrong variable name. The final rule is do as your teammates do, so long as it does not counteract the prior rules.

Share Improve this answer

Follow

answered Sep 20, 2008 at 0:21

Tim Ottinger

1,452 ● 9 ● 5

> i agree with everything you say (except i've come to learn that it's better to say "recommendation" than "rule") -- and most people think my variable names are too long. "... do as your teammates do..." is one reason i used the "consistency" tag. another reason is of course "do as you yourself do".
> – just mike Sep 20, 2008 at 0:33

1

@JustMike . . . **A FEW C EXAMPLES:** . . . to accompany the Java ones.



NON-NESTED loop: . . . limiting scope where possible

```
/*LOOP_DESCRIPTION*/ {

    int i;

    for (i = 0; i < LOOP_LENGTH; i++) {

        // loop body
    }
}
```

NESTED loop: . . . ditto

```
/*LOOP_DESCRIPTION*/ {

    int row, column;

    for (row = 0; row < ROWS; row++) {

        for (column = 0; column < COLUMNS;
column++) {

            // loop body
        }
    }
}
```

One good thing about this layout is it reads badly without comments, thereby encouraging them.
It's verbose perhaps, but personally this is how I do loops in C.

Also: *I did use "index" and "idx" when I started, but this usually got changed to "i" by my peers.*

Share Improve this answer Follow

edited Sep 20, 2008 at 1:47

answered Sep 19, 2008 at 22:14

Ande Turner

7,202 ● 19 ● 82 ● 107

1

for numerical computations, matlab, and the likes of it, dont use i, j

these are reserved constants, but matlab wont complain.

My personal favs are

index first,second counter count

Share Improve this answer Follow

answered Sep 20, 2008 at 18:14

Midhat

17.8k ● 22 ● 90 ● 116

1

Whatever you choose, use the same index consistently in your code wherever it has the same meaning. For example, to walk through an array, you can use `i`, `jj`, `kappa`, whatever, but always do it the same way everywhere:

```
for (i = 0; i < count; i++) ...
```

The best practice is to make this part of the loop look the same throughout your code (including consistently using

`count` as the limit), so that it becomes an idiom that you can skip over mentally in order to focus on the meat of the code, the body of the loop.

Similarly, if you're walking through an 2d array of pixels, for example, you might write

```
for (y = 0; y < height; y++)
  for (x = 0; x < width; x++)
    ...
```

Just do it the same way in every place that you write this type of loop.

You want your readers to be able to ignore the boring setup and see the brilliance of what you're doing in the actual loop.

Share Improve this answer

Follow

answered Sep 22, 2008 at 14:22

Derek Clegg

202 ● 1 ● 4

1

Steve McConnell's *Code Complete* has, as usual, some excellent advice in this regard. The relevant pages (in the first edition anyway) are 340 and 341. Definitely advise anyone who's interested in improving their loop coding to give this a look. McConnell recommends meaningful loop counter names but people should read what he's got to say themselves rather than relying on my weak summary.

Share Improve this answer

answered Sep 24, 2008 at 13:26

Follow

Onorio Catenacci
15.3k ● 16 ● 84 ● 134

1

i also use the double-letter convention. ii, jj, kk. you can grep those and not come up with a bunch of unwanted matches.

i think using those letters, even though they're doubled, is the best way to go. it's a familiar convention, even with the doubling.

there's a lot to say for sticking with conventions. it makes things a lot more readable.

Share Improve this answer

Follow

edited Feb 13, 2013 at 1:11

answered Sep 27, 2008 at 1:55

Randy L
14.7k ● 14 ● 49 ● 75

0

I've started using perlisms in php.

if its a singular iteration, `$_` is a good name for those who know its use.

Share Improve this answer

Follow

answered Sep 19, 2008 at 11:00

Kent Fredric
57.3k ● 14 ● 111 ● 151

My habit is to use 't' - close to 'r' so it follows easily aftewr typing 'for'

0

Share Improve this answer

Follow

answered Sep 19, 2008 at 11:04

Simon Munro

5,419 ● 7 ● 34 ● 40

If it is a simple counter, I stick to using 'i' otherwise, have name that denotes the context. I tend to keep the variable length to 4. This is mainly from code reading point of view, writing is doesn't count as we have auto complete feature.

0

Share Improve this answer

Follow

answered Sep 19, 2008 at 12:33

Karthi

33 ● 1 ● 7

I've started to use context-relevant loop variable names mixed with hungarian.

When looping through rows, I'll use `iRow`. When looping through columns I'll use `iCol`. When looping through cars I'll use `iCar`. You get the idea.

0

Share Improve this answer

Follow

edited Sep 20, 2008 at 18:12

answered Sep 20, 2008 at 17:59

Tim Gradwell

2,402 ● 5 ● 25 ● 25

0

My favorite convention for looping over a matrix-like set is to use x+y as they are used in cartesian coordinates:

```
for x in width:
    for y in height:
        do_something_interesting(x,y)
```

Share Improve this answer

Follow

answered Sep 20, 2008 at 21:12

Ed L

2,017 ● 2 ● 18 ● 30

1 In which case they are actually meaningful names – Nat Sep 25, 2008 at 0:53

0

I usually use:

```
for(lcObject = 0; lcObject < Collection.length();
lcObject++)
{
   //do stuff
}
```

Share Improve this answer

Follow

answered Sep 25, 2008 at 2:55

Riddari

1,773 ● 3 ● 28 ● 57

...assuming of course that nothing adds or removes anything from the Collection! – Mitch Wheat Sep 27, 2008 at 2:03

0

For integers I use int index, unless it's nested then I use an Index suffix over what's being iterated like int groupIndex and int userIndex.

Share Improve this answer

Follow

answered Sep 27, 2008 at 2:08

loudej

1,887 ● 11 ● 17

0

In Python, I use i, j, and k if I'm only counting times through. I use x, y, and z if the iteration count is being used as an index. If I'm actually generating a series of arguments, however, I'll use a meaningful name.

Share Improve this answer

Follow

answered Nov 26, 2008 at 14:04

J.T. Hurley

521 ● 9 ● 12