Do people use the Hungarian Naming Conventions in the real world? [closed]

Asked 16 years, 4 months ago Modified 10 years, 7 months ago

Viewed 13k times

34

Closed. This question is off-topic. It is not currently accepting answers.

Want to improve this question? Update the question so it's on-topic for Stack Overflow.

Closed 12 years ago.

Improve this question

Is it worth learning the convention or is it a bane to readability and maintainability?

conventions hungarian-notation self-documenting-code

Share

Improve this question

Follow

edited Sep 3, 2009 at 1:59

Phil Miller

38k • 13 • 70 • 91

asked Aug 7, 2008 at 22:31

Brock D
551 ● 1 ● 7 ● 10

20 Answers

Sorted by: Highest score (default)

59

Considering that most people that use *Hungarian Notation* is following the misunderstood version of it, I'd say it's pretty pointless.

If you want to use the original definition of it, it might make more sense, but other than that it is mostly syntactic sugar.

If you read the Wikipedia article on the subject, you'll find two conflicting notations, *Systems Hungarian Notation* and *Apps Hungarian Notation*.

The original, good, definition is the *Apps Hungarian Notation*, but most people use the *Systems Hungarian Notation*.

As an example of the two, consider prefixing variables with l for length, a for area and v for volume.

With such notation, the following expression makes sense:

```
int vBox = aBottom * lVerticalSide;
```

but this doesn't:

```
int aBottom = lSide1;
```

If you're mixing the prefixes, they're to be considered part of the equation, and volume = area * length is fine for a box, but copying a length value into an area variable should raise some red flags.

Unfortunately, the other notation is less useful, where people prefix the variable names with the type of the value, like this:

```
int iLength;
int iVolume;
int iArea;
```

some people use n for number, or i for integer, f for float, s for string etc.

The original prefix was meant to be used to spot problems in equations, but has somehow devolved into making the code slightly easier to read since you don't have to go look for the variable declaration. With todays smart editors where you can simply hover over any variable to find the full type, and not just an abbreviation for it, this type of hungarian notation has lost a lot of its meaning.

But, you should make up your own mind. All I can say is that I don't use either.

Edit Just to add a short notice, while I don't use *Hungarian Notation*, I do use a prefix, and it's the underscore. I prefix all private fields of classes with a _ and otherwise spell their names as I would a property, titlecase with the first letter uppercase.

Share Improve this answer Follow

edited Apr 30, 2014 at 8:07

answered Aug 7, 2008 at 22:39

Lasse V. Karlsen
391k ● 106 ● 646 ● 844

I recommend reading blogs.msdn.com/larryosterman/archive/2004/06/22/162629.aspx – Anzurio May 7, 2009 at 4:18

1 "With todays smart editors where you can simply hover over any variable to find the full type, and not just an abbreviation for it, this type of hungarian notation has lost a lot of its meaning." What if you consider sites like pastebin.com, stackoverflow, any board with code tags or even web based repository browsers? In this case i find those prefixes just as useful as in a crappy editor. In fact it's the main reason im still using it. – mibollma May 14, 2011 at 0:38

2 I still think that prefixing a variable i for integer is pointless, the name of the variable should be far more expressive than the single i. – Lasse V. Karlsen May 14, 2011 at 11:33

4 "I don't use neither" means you use both. – Ally Feb 13, 2014 at 11:48

`typedef volume_t int; typedef area_t int; typedef length_t int;` – vasilescur Nov 19, 2020 at 6:30

The Hungarian Naming Convention can be useful when used correctly, unfortunately it tends to be misused more often than not.

Read Joel Spolsky's article Making Wrong Code Look Wrong for appropriate perspective and justification.

Essentially, type based Hungarian notation, where variables are prefixed with information about their type (e.g. whether an object is a string, a handle, an int, etc.) is mostly useless and generally just adds overhead with very little benefit. This, sadly, is the Hungarian notation most people are familiar with. However, the intent of Hungarian notation as envisioned is to add information on the "kind" of data the variable contains. This allows you to partition kinds of data from other kinds of data which shouldn't be allowed to be mixed together except, possibly, through some conversion process. For example, pixel based coordinates vs. coordinates in other units, or unsafe user input versus data from safe sources, etc.

Look at it this way, if you find yourself spelunking through code to find out information on a variable then you probably need to adjust your naming scheme to contain that information, this is the essence of the Hungarian convention.

Note that an alternative to Hungarian notation is to use more classes to show the intent of variable usage rather than relying on primitive types everywhere. For example,

instead of having variable prefixes for unsafe user input, you can have simple string wrapper class for unsafe user input, and a separate wrapper class for safe data. This has the advantage, in strongly typed languages, of having partitioning enforced by the compiler (even in less strongly typed languages you can usually add your own tripwire code) but adds a not insignificant amount of overhead.

Share Improve this answer

Follow

answered Aug 7, 2008 at 22:56

Wedge

19.8k ● 7 ● 50 ● 71

14

I still use Hungarian Notation when it comes to UI elements, where several UI elements are related to a particular object/value, e.g.,

lblFirstName for the label object, txtFirstName for the text box. I definitely can't name them both "FirstName" even if *that* is the concern/responsibility of both objects.

How do others approach naming UI elements?

Share Improve this answer

Follow

answered Aug 8, 2008 at 1:44

Jon Limjap

95.3k ● 15 ● 103 ● 153

5 I use firstNameLabel and firstNameTextBox – Mark Cidade Sep 10, 2008 at 0:29

11

I think hungarian notation is an interesting footnote along the 'path' to more readable code, and if done properly, is preferable to not-doing it.

In saying that though, I'd rather do away with it, and instead of this:

```
int vBox = aBottom * lVerticalSide;
```

write this:

```
int boxVolume = bottomArea * verticalHeight;
```

It's 2008. We don't have 80 character fixed width screens anymore!

Also, if you're writing variable names which are much longer than that you should be looking at refactoring into objects or functions anyway.

Share Improve this answer

Follow

answered Aug 7, 2008 at 22:52

Orion Edwards

123k ● 66 ● 245 ● 339

8

It is pointless (and distracting) but is in relatively heavy use at my company, at least for types like ints, strings, booleans, and doubles.

Things like `sValue`, `iCount`, `dAmount` or `fAmount`, and `bFlag` are everywhere.

Once upon a time there was a good reason for this convention. Now, it is a cancer.

Share Improve this answer

Follow

answered Aug 7, 2008 at 22:45

Justin Standard

21.5k ● 22 ● 82 ● 90

1 what happen ? and how i start to become a cancer? – justjoe Aug 9, 2010 at 14:49

6

I see Hungarian Notation as a way to circumvent the capacity of our short term memories. According to psychologists, we can store approximately **7 plus-or-minus 2 chunks** of information. The extra information added by including a prefix helps us by providing more details about the meaning of an identifier even with no other context. In other words, we can guess what a variable is for without seeing how it is used or declared. This can be avoided by applying oo techniques such as encapsulation and the single responsibility principle.

I'm unaware of whether or not this has been studied empirically. I would hypothesize that the amount of effort increases dramatically when we try to understand classes with more than nine instance variables or methods with more than 9 local variables.

Share Improve this answer

Follow

answered Sep 17, 2008 at 3:24

James Wulkan

808 ● 7 ● 12

5

Sorry to follow up with a question, but does prefixing interfaces with "I" qualify as hungarian notation? If that is the case, then yes, a lot of people are using it in the real world. If not, ignore this.

Share Improve this answer

Follow

answered Aug 13, 2008 at 16:31

Theo

133k ● 22 ● 167 ● 213

4

When I see Hungarian discussion, I'm glad to see people thinking hard about how to make their code clearer, and how to mistakes more visible. That's exactly what we should all be doing!

But don't forget that you have some powerful tools at your disposal besides naming.

Extract Method If your methods are getting so long that your variable declarations have scrolled off the top of the screen, consider making your methods smaller. (If you have too many methods, consider a new class.)

Strong typing If you find that you are taking *zip codes* stored in an integer variable and assigning them to a *shoe size* integer variable, consider making a class for zip codes and a class for shoe size. Then your bug will be caught at compile time, instead of requiring careful inspection by a human. When I do this, I usually find a bunch of zip code- and shoe size-specific logic that I've

peppered around my code, which I can then move in to my new classes. Suddenly all my code gets clearer, simpler, and protected from certain classes of bugs. Wow.

To sum up: yes, think hard about how you use names in code to express your ideas clearly, but also look to the other powerful OO tools you can call on.

Share Improve this answer

Follow

answered Sep 9, 2008 at 23:05

Jay Bazuzi

46.4k ● 16 ● 115 ● 170

4

Isn't scope more important than type these days, e.g.

- l for local
- a for argument
- m for member
- g for global
- etc

With modern techniques of refactoring old code, search and replace of a symbol because you changed its type is tedious, the compiler will catch type changes, but often will not catch incorrect use of scope, sensible naming conventions help here.

Share Improve this answer

Follow

answered Sep 20, 2008 at 8:44

titanae

2,289 ● 2 ● 21 ● 31

2

I don't use a very strict sense of hungarian notation, but I do find myself using it sparing for some common custom objects to help identify them, and also I tend to prefix gui control objects with the type of control that they are. For example, labelFirstName, textFirstName, and buttonSubmit.

Share Improve this answer

Follow

answered Oct 24, 2008 at 13:46

AgentThirteen

261 ● 1 ● 4 ● 8

1

I use Hungarian Naming for UI elements like buttons, textboxes and lables. The main benefit is grouping in the Visual Studio Intellisense Popup. If I want to access my lables, I simply start typing lbl.... and Visual Studio will suggest all my lables, nicley grouped together.

However, after doing more and more Silverlight and WPF stuff, leveraging data binding, I don't even name all my controls anymore, since I don't have to reference them from code-behind (since there really isn't any codebehind anymore ;)

Share Improve this answer

Follow

answered Aug 8, 2008 at 6:09

Jonas Follesø

6,531 ● 7 ● 42 ● 54

What's wrong is mixing standards.

1

What's right is making sure that everyone does the same thing.

```
int Box = iBottom * nVerticleSide
```

Share Improve this answer

Follow

answered Aug 13, 2008 at 17:17

ColinYounger

6,865 ● 5 ● 33 ● 33

1

> The original prefix was meant to be used to spot problems in equations, but has somehow devolved into making the code slightly easier to read since you don't have to go look for the variable declaration. With todays smart editors where you can simply hover over any variable to find the full type, and not just an abbreviation for it, this type of hungarian notation has lost a lot of its meaning.

I'm breaking the habit a little bit but prefixing with the type can be useful in JavaScript that doesn't have strong variable typing.

Share Improve this answer

Follow

answered Aug 27, 2008 at 13:55

Will

3,241 ● 1 ● 20 ● 17

1 It's a huge benefit whend debuging as well, treat it as the 'intent' of the expected type stored. – Daniel Sokolowski Apr

7, 2016 at 13:55

1

When using a dynamically typed language, I occasionally use Apps Hungarian. For statically typed languages I don't. See my explanation in the other thread.

Share Improve this answer

Follow

edited May 23, 2017 at 12:17

Community Bot

1 • 1

answered Aug 27, 2008 at 14:21

Konrad Rudolph

545k • 139 • 956 • 1.2k

1

Hungarian notation is pointless in type-safe languages. e.g. A common prefix you will see in old Microsoft code is "lpsz" which means "long pointer to a zero-terminated string". Since the early 1700's we haven't used segmented architectures where short and long pointers exist, the normal string representation in C++ is always zero-terminated, and the compiler is type-safe so won't let us apply non-string operations to the string. Therefore none of this information is of any real use to a programmer - it's just more typing.

However, I use a similar idea: prefixes that clarify the **usage** of a variable. The main ones are:

- m = member
- c = const

- s = static
- v = volatile
- p = pointer (and pp=pointer to pointer, etc)
- i = index or iterator

These can be combined, so a static member variable which is a pointer would be "mspName".

Where are these useful?

- Where the usage is important, it is a good idea to constantly remind the programmer that a variable is (e.g.) a volatile or a pointer
- Pointer dereferencing used to do my head in until I used the p prefix. Now it's really easy to know when you have an object (Orange) a pointer to an object (pOrange) or a pointer to a pointer to an object (ppOrange). To dereference an object, just put an asterisk in front of it for each p in its name. Case solved, no more deref bugs!
- In constructors I usually find that a parameter name is identical to a member variable's name (e.g. size). I prefer to use "mSize = size;" than "size = theSize" or "this.size = size". It is also much safer: I don't accidentally use "size = 1" (setting the parameter) when I meant to say "mSize = 1" (setting the member)
- In loops, my iterator variables are all meaningful names. Most programmers use "i" or "index" and

then have to make up new meaningless names ("j", "index2") when they want an inner loop. I use a meaningful name with an i prefix (iHospital, iWard, iPatient) so I always know what an iterator is iterating.

- In loops, you can mix several related variables by using the same base name with different prefixes: Orange orange = pOrange[iOrange]; This also means you don't make array indexing errors (pApple[i] looks ok, but write it as pApple[iOrange] and the error is immediately obvious).
- Many programmers will use my system without knowing it: by add a lengthy suffix like "Index" or "Ptr" - there isn't any good reason to use a longer form than a single character IMHO, so I use "i" and "p". Less typing, more consistent, easier to read.

This is a simple system which adds meaningful and useful information to code, and eliminates the possibility of many simple but common programming mistakes.

Share Improve this answer

Follow

answered May 25, 2009 at 15:48

Jason Williams
57.9k ● 11 ● 109 ● 140

0

I've been working for IBM for the past 6 months and I haven't seen it anywhere (thank god because I hate it.) I see either camelCase or c_style.

```
thisMethodIsPrettyCool()
this_method_is_pretty_cool()
```

Share Improve this answer

Follow

answered Aug 7, 2008 at 22:39

Steve Willard

16.7k • 4 • 29 • 26

0

It depends on your language and environment. As a rule I wouldn't use it, unless the development environment you're in makes it hard to find the type of the variable.

There's also two different types of Hungarian notation. See Joel's article. I can't find it (his names don't exactly make them easy to find), anyone have a link to the one I mean?

Edit: Wedge has the article I mean in his post.

Share Improve this answer

Follow

edited Aug 7, 2008 at 23:19

answered Aug 7, 2008 at 22:43

Ray

46.5k • 29 • 127 • 170

0

Original form (The Right Hungarian Notation :)) where prefix means type (i.e. length, quantity) of value stored by variable is OK, but not necessary in all type of applications.

The popular form (The Wrong Hungarian Notation) where prefix means type (String, int) is useless in most of modern programming languages.

Especially with meaningless names like strA. I can't understand we people use meaningless names with long prefixes which gives nothing.

Share Improve this answer

Follow

answered Aug 27, 2008 at 14:14

Grzegorz Gierlik

11.2k ● 4 ● 48 ● 57

0

I use type based (Systems HN) for components (eg editFirstName, lblStatus etc) as it makes autocomplete work better.

I sometimes use App HN for variables where the type infomation is isufficient. Ie fpX indicates a fixed pointed variable (int type, but can't be mixed and matched with an int), rawInput for user strings that haven't been validated etc

Share Improve this answer

Follow

answered Sep 9, 2008 at 23:41

Sean

0

Being a PHP programmer where it's very loosely typed, I don't make a point to use it. However I will occasionally identify something as an array or as an object depending on the size of the system and the scope of the variable.

Share Improve this answer

Follow

answered Sep 3, 2009 at 2:19

Joshua K

507 ● 3 ● 13