Difference between strongly and weakly typed languages?

Asked 11 years, 6 months ago Modified 7 years, 9 months ago

Viewed 32k times

14

I have read several pages, including the wiki page http://en.wikipedia.org/wiki/Strong_and_weak_typing dealing with strongly and weakly typed languages. For the most part, I think I understand the difference. However, I would like a straight to the point answer differentiating the two.

From my understanding, in weakly typed languages, data types do not have to be explicitly called. This would be a language like Matlab where you can add 4 and 2.3 without having to typecast. Strongly typed languages require the programmer to declare a data type for each variable and/or value. For instance in C, you would need to do something like 4 + (int) 2.3 or (float)4 + 2.3 (can't remember if that is valid C type-casting).

Any information expanding or correcting my understanding of these concepts would be greatly appreciated.

theory

Share

Improve this question

Follow

edited Dec 8, 2013 at 19:09

Michael0x2a

63.7k • 32 • 187 • 238

asked Jun 12, 2013 at 17:59

Josh

1,052 • 2 • 12 • 24

I wonder if this question might be a better fit for somewhere like Theoretical Computer Science or Programmers...?
– summea Jun 12, 2013 at 18:01

Yea, I wasn't sure where to put it. I will add that. – Josh Jun 12, 2013 at 18:03

In C it is perfectly legal to add an `int` to a `float`.
– Eric Lippert Jun 13, 2013 at 15:04

possible duplicate of Can someone tell me what Strong typing and weak typing means and which one is better?
– nawfal Jul 24, 2014 at 9:54

" in C, you would need to do something like 4 + (int) 2.3 or (float)4 + 2.3" -- nope; ints are automatically coerced to float.
– Jim Balter Sep 13, 2019 at 3:52

3 Answers

Sorted by: Highest score (default)

25

The difference is not about declaring types on variables. It's a bit more subtle than that (and *pace* Eric Lippert, i think the term is reasonably well-defined). The distinction is that in a strongly-typed language, **every expression has a type which can be determined at compile time**,

and **only operations appropriate to that type are allowed**.

In an untyped ("weakly typed" to critics, "dynamically typed" to fans) language, that is not the case. The language allows any operation to be performed on any type, with the rather substantial proviso that the operation may fail. That is, while the *language* may allow the operation, the *runtime* may not.

Note that it's possible to have a strongly-typed language without requiring type declarations everywhere. Indeed, no strongly-typed language does. Consider this bit of Java:

```
String s = "hellO";
int l = s.getBytes().length;
```

How does the compiler decide that `.length` is legal there? It's legal because it's being used on a `byte[]`. But there is no declaration of anything as being a `byte[]` here. Rather, the compiler knows that `s` is a `String`, and that when you call `getBytes()` on a `String`, you get a `byte[]`. It infers from those facts that the type of `s.getBytes()` is a `byte[]`, and so that it is legal to ask for its `length`.

Some languages whose type systems are more sophisticated than Java's allow the compiler to infer more than this. For example, in Scala, you can say:

```
val s = "hello"
val l = s.getBytes().length
```

And the compiler will infer the types of `s` and `l`, as well as of the intermediate expressions.

Languages which have strong typing but artificial limits on type inference which require redundant type declarations (like Java) are described as having *manifest typing*, because the types must be made manifest, which is a fancy way of saying explicitly brought into existence, which is a fancy way of saying written down.

Share Improve this answer Follow

edited Mar 5, 2014 at 12:02

answered Jun 12, 2013 at 18:13

Tom Anderson

47.1k ● 17 ● 95 ● 137

3 Strong/weak typing is orthogonal to static/dynamic typing. For example, the Ruby type system is strong and dynamic. – ComDubh Oct 14, 2015 at 11:05

"'weakly typed" to critics, "dynamically typed" to fans' -- these are totally different things. "pace Eric Lippert, i think the term is reasonably well-defined" -- that's rather ironic, since hardly anyone agrees with your definition, which is decidedly non-standard. – Jim Balter Sep 13, 2019 at 3:57

10

Check Eric Lippert's blog out. There's an entry about just what you're looking for here.

From the looks of his blog, those terms are subjective, so "speak more precisely about type system features."

Share Improve this answer

Follow

answered Jun 12, 2013 at 18:03

masotann

911 ● 10 ● 30

2

As you said... ...in weakly typed languages, data types do not have to be explicitly called.

Strongly typed languages require the programmer to declare a data type for each variable and/or value.

This is correct...

There is also a sort of paradigm in so called "strongly" typed languages like c# in which types can be declared if necessary or wanted by the programmer... e.g. C# has the "var" type, but also has strong types (Int32, String, Boolean, etc) which many programmers that use this language prefer.

In this way a language can be both "strongly" and "weakly" typed.

I hope this helps further your understanding of this concept...

Share Improve this answer

Follow

answered Jun 12, 2013 at 18:07

Jeff

111 • 2

1 Note that var in C# maintains the "string" typing of variables. The type name isn – dlev Jun 13, 2013 at 15:33

1 You misunderstand `var`, which is not a type. C# is a strongly typed language with type inference. – Jim Balter Sep 13, 2019 at 3:59