Statically typing and dynamically typing

Ziming Ren

Abstract

Every programming language has a data type system. Without a type system, computers wouldn't know how to represent the data in our programs. The type system of a programming language can be categorized into dynamically typing and statically typing. This article will introduce what dynamic typing and static typing are, analyze the strengths and weaknesses of both types, and examine under which application requirements one should choose a particular type of language to fully leverage the advantages of the language.

Keywords: statically typing, dynamically typing

ACM Reference Format:
Ziming Ren. 2023. Statically typing and dynamically typing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction: What Are Data Types?

In programming, a data type is what we tell the computer the type of data it's dealing with, e.g. a string, number, or object. When defining a variable, a computer needs both the name and the type of data before it can store and process it. This way it knows much much memory to set aside, how to access it and how to change it. Each programming language has a different way of handling how it assigns data types.

2 Definition

2.1 statically typing

In statically typed programming languages, type checking occurs at compile time. At compile time, source code in a specific programming language is converted to a machine-readable format. This means that before source code is compiled, the type associated with each and every single variable must be known.For most of these languages, the programmer is required to explicitly state the data type of each variable when the variable is being declared. Another notable characteristic of this category of programming languages is that the detection of errors in variable-data type associations will halt the compilation process until the error has been rectified. For example, Java, C, C++ are statically typed programming languages.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 dynamically typing

In dynamically typed languages, type checking takes place at runtime or execution time. For this category of programming languages, there's no requirement to explicitly state the data type during variable declarations. The language system is able to detect the type of a variable at runtime. In addition, altering the data type of previously declared variables is allowed. For example, JavaScript, PHP, Python, Ruby are dynamically typed programming languages.

3 The advantages and disadvantages

3.1 statically typing

Advantages of statically typed languages include the early detection of type mismatches during compilation, enabling editors to help prevent potential errors that might occur during runtime. Additionally, explicit type declarations in the program allow the compiler to perform optimizations, enhancing the program's execution speed.

```
int x = 1;
if (x > 0)
        int b = 2;
else
        int c = "123";
```

In the above code, even though statements with type errors will never be executed, the compiler is unaware of this fact. It still detects the type errors during compilation. This code wouldn't compile, preventing errors at their root. So for this type of language, errors related to data types are essentially non-existent at runtime.

```
int s = 0;
for (int i = 1; i <= 100000000; i++)
        s += i;
```

For code with a substantial amount of runtime execution like this, statically typed languages can deliver results more quickly compared to dynamically typed languages, which tend to be slower.

The drawbacks of statically typed languages lie in the requirement for programmers to adhere to strict contracts by specifying data types for each variable. Furthermore, type

declarations introduce additional code, potentially distracting programmers from concentrating on business logic during the coding process.

```
int a = 1
double pi = 3.14159
string s = "statically"

int add (int a, int b) {
        return a + b
}
```

In C++ code, it is necessary to include the types of variables and functions before their declarations, leading to a significant increase in content and code volume, especially when working on large-scale projects.

3.2 dynamically typing

Advantages of dynamically typed languages include reduced code volume and a more concise appearance, enabling programmers to focus more on business logic. Fewer lines of code contribute to improved readability during program inspection. The flexibility introduced by the leniency toward variable types in dynamically typed languages provides significant coding flexibility. Without the need for type checking, programmers can attempt to invoke methods on any object without considering its original design with respect to that method.

```
a = 1
b = a + 2
a = "PL"
a = max
```

In the above code snippet, the flexibility of dynamic typing is evident. The type of variable a can change arbitrarily, including integers, strings, and functions, without triggering errors. This provides programmers with greater versatility and possibilities in their code.

The main drawback of dynamically typed languages is the inability to guarantee variable types, potentially leading to type-related errors during program execution. And the lack of type differentiation may make the program harder to understand in certain situations. Data types in these dynamic languages are typically determined at runtime. This makes it hard to catch many errors until they reach a production environment. It may work fine on your local development machine, but the production runtime environment could be slightly different, yielding some different guesswork by the interpreter.

```
def f(x):
        if x >0:
                return x + 3
        else:
                return x + "abc"
```

In this code snippet, if it were in a statically typed language, it would not pass compilation. However, in a dynamically typed language, the program would run successfully, and calling $f(1)$ would complete without errors, potentially confusing the programmer. It's only when running $f(-1)$ that an error would occur, allowing the programmer to catch the mistake.

4 Application

Deciding which language to use is usually dependent on the programmer and the purpose of the program. It's difficult to conclude that one is better than the other, as they both have their own perks and drawbacks.

Here are some suggestions:

- Large-scale Projects: For large and complex projects, static typing languages are often more suitable as they provide stronger type safety and performance optimizations.
- Rapid Prototyping: If you need rapid prototyping or are working on a small-scale project, dynamic typing languages may be more appropriate as they can offer faster development speed and greater programming flexibility.
- Team Collaboration: In team collaboration, static typing languages are generally more advantageous for maintenance and collaboration since type information can provide more documentation and readability.
- Personal Projects: For personal projects, you can choose between static or dynamic typing languages based on your preference to meet your own needs. For instance, if a programmer wants to write and execute code easier, then dynamically typed languages are a good match. However, it's the responsibility of the programmer to practice good type management. If more rigid code is preferred, then a better option would be a statically typed language.

5 Conclusion

This article aims to provide a comprehensive overview of the definitions and fundamental differences between dynamic and static typing languages, comparing their respective strengths and weaknesses. It discusses how to make informed decisions when it comes to selecting the appropriate language for our projects, especially concerning type systems.