A Survey on Abstract Data Types

Anthony Hunt

February 8, 2025

1 Introduction

From the categorization of species to the periodic table of elements, the organization and manipulation of data is integral to all aspects of science and technology. Although the collection of data is not a new field of research by any means, the advent of computerized systems, with its ability to transfer and use massive amounts of data in mere seconds, necessitates understandable and sensible organization structures.

Over the past 60 years, as computational resource restrictions have eased dramatically, programming languages and software creation tools have evolved with increasingly better methods of dealing with data. By viewing pieces of data through specific lenses, programmers can then write algorithms that operate on any data matching the shape of those lenses, rather than the data itself. These lenses are more commonly referred to as data types, which assign a type to a piece of data. As we will see later in this paper, further abstractions over data types themselves enable extremely concise and readable code, both for the programmer and computer.

The rest of this article will explore data type concepts from ¡X¿ languages, structures to handle errors that might appear, and a small review of ergonomics and usability of those data types.

2 Data Types in ¡X¿+ Languages

Abstract Data Types (ADTs) can look quite different in many languages, but often share the same principles and rules that make it understandable for programmers. In its essential form, an ADT is a collection of data together with operations that can be carried out on that data [17]. The key differences between ADTs from different languages are the conventions used to collect data and the power operations have over collected data.

In popular object-oriented programming (OOP) languages, ADTs are approached with a "class" mindset, where programmers specify a blueprint for objects that will eventually be used in a program. For better code reuse, classes can be built from pre-existing classes through the use of inheritance, a term used to express that a new class inherits all the properties of the old class. Objects

created by classes are referred to as instances of the class and often hold some amount of mutable data, with methods that have full access to the data. The methods can choose to make use of "side effects", referring to the modification of data not explicitly passed through regular function arguments or return values. Side effects in methods have the consequence that the output of a method may change independent of the values passed through its arguments. In other words, the object a method has access to may contain some state that influences the outcome of the method.

At the opposite end of the programming world lies purely functional languages. ADTs in these languages are more often referred to as data types rather than classes and are only used for collecting and operating on data. "Objects" created from these data types are immutable and methods are instead replaced with pure functions that explicitly take in the relevant data type as input. The operators attached to a functionally pure ADT are functionally pure themselves, meaning they cannot make use of any hidden state and therefore cannot have side effects.

References

[1] Jean-Raymond Abrial. *Modeling in Event-B : system and software engineering*. eng. Cambridge: Cambridge University Press, 2010. ISBN: 9781139195881.

[2] *C++ reference*. 2024. URL: https://en.cppreference.com/w/ (visited on 02/07/2025).

[3] Wikipedia contributors. *Type class — Wikipedia*. 2025. URL: https://en.wikipedia.org/wiki/Type_class (visited on 02/07/2025).

[4] *Event-B and Rodin Documentation Wiki*. 2024. URL: https://wiki.event-b.org/index.php/Main_Page (visited on 02/07/2025).

[5] A. Hsu. *Is APL Dead?* 2020. URL: https://www.sacrideo.us/is-apl-dead/ (visited on 02/07/2025).

[6] Kenneth E. Iverson. "Notation as a tool of thought". In: *Commun. ACM* 23.8 (Aug. 1980), pp. 444–465. ISSN: 0001-0782. DOI: 10.1145/358896.358899. URL: https://doi.org/10.1145/358896.358899.

[7] S. Klabnik and C. Nichols. *The Rust Programming Language, 2nd Edition*. No Starch Press, 2023. ISBN: 9781718503106. URL: https://books.google.ca/books?id=SE2GEAAAQBAJ.

[8] *LearningAPL*. 2024. URL: https://xpqz.github.io/learnapl/intro.html (visited on 02/07/2025).

[9] M. Lipovaca. *Learn You a Haskell for Great Good!: A Beginner's Guide*. No Starch Press Series. No Starch Press, 2011. ISBN: 9781593272838. URL: https://books.google.ca/books?id=R2RbBAAAQBAJ.

[10] J.J. Martin. *Data Types and Data Structures*. Prentice-Hall International series in personal computing. Prentice-Hall International, 1986. ISBN: 9780131959835. URL: https://books.google.ca/books?id=554ZAQAAIAAJ.

[11] *OMG Unified Modeling Language*. Version 2.5.1. Object Management Group. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (visited on 02/07/2025).

[12] A. Rauschmayer. *JavaScript metaprogramming with the 2022-03 decorators API*. 2024. URL: https://2ality.com/2022/10/javascript-decorators.html (visited on 02/07/2025).

[13] D. Rosenwasser. *Announcing Typescript 5.0*. 2024. URL: https://devblogs.microsoft.com/typescript/announcing-typescript-5-0/#decorators (visited on 02/07/2025).

[14] *The Dafny Programming and Verification Language*. 2024. URL: https://dafny.org/ (visited on 02/07/2025).

[15] *The Python Language Reference (3.13.2)*. Python Software Foundation. 2025. URL: https://docs.python.org/3/reference/ (visited on 02/07/2025).

[16] *The Rust Reference*. 2024. URL: https://doc.rust-lang.org/reference/ (visited on 02/07/2025).

[17] P.G. Thomas, H. Robinson, and J. Emms. *Abstract Data Types: Their Specification, Representation, and Use*. Oxford applied mathematics and computing science series. Clarendon Press, 1988. ISBN: 9780198596639. URL: https://books.google.ca/books?id=u61QAAAAMAAJ.

[18] *Type — HaskellWiki*. HaskellWiki. 2024. URL: https://wiki.haskell.org/index.php?title=Type (visited on 02/07/2025).

[19] *Typeclassopedia — HaskellWiki*. HaskellWiki. 2022. URL: https://wiki.haskell.org/index.php?title=Typeclassopedia (visited on 02/07/2025).

[20] *Typescript Documentation*. 2024. URL: https://www.typescriptlang.org/docs/ (visited on 02/07/2025).

[21] *Welcome to Agda's documentation!* Version 2.7.0.1. 2025. URL: https://agda.readthedocs.io/en/v2.7.0.1/# (visited on 02/07/2025).