WIKIPEDIA

Declarative programming

In computer science, **declarative programming** is a programming paradigm—a style of building the structure and elements of computer programs—that expresses the logic of a computation without describing its control flow.[1]

Many languages that apply this style attempt to minimize or eliminate side effects by describing *what* the program must accomplish in terms of the problem domain, rather than describe *how* to accomplish it as a sequence of the programming language primitives[2] (the *how* being left up to the language's implementation). This is in contrast with imperative programming, which implements algorithms in explicit steps.

Declarative programming often considers programs as theories of a formal logic, and computations as deductions in that logic space. Declarative programming may greatly simplify writing parallel programs.[3]

Common declarative languages include those of database query languages (e.g., SQL, XQuery), regular expressions, logic programming, functional programming, and configuration management systems.

Contents

Definition

Declarative programming is often defined as any style of programming that is not imperative. A number of other common definitions exist that attempt to give the term a definition other than simply contrasting it with imperative programming. For example:

- A high-level program that describes what a computation should perform.
- Any programming language that lacks side effects (or more specifically, is referentially transparent)
- A language with a clear correspondence to mathematical logic.[4]

These definitions overlap substantially.

Declarative programming contrasts with imperative and procedural programming. Declarative programming is a non-imperative style of programming in which programs describe their desired results without explicitly listing commands or steps that must be performed. Functional and logical programming languages are characterized by a declarative programming style. In logical programming languages, programs consist of logical statements, and the program executes by searching for proofs of the statements.

In a pure functional language, such as Haskell, all functions are without side effects, and state changes are only represented as functions that transform the state, which is explicitly represented as a first class object in the program. Although pure functional languages are non-imperative, they often provide a facility for describing the effect of a function as a series of steps. Other functional languages, such as Lisp, OCaml and Erlang, support a mixture of procedural and functional programming.

Some logical programming languages, such as Prolog, and database query languages, such as SQL, while declarative in principle, also support a procedural style of programming.

Subparadigms

Declarative programming is an umbrella term that includes a number of better-known programming paradigms.

Constraint programming

Constraint programming states relations between variables in the form of constraints that specify the properties of the target solution. The set of constraints is solved by giving a value to each variable so that the solution is consistent with the maximum number of constraints. Constraint programming often complements other paradigms: functional, logical, or even imperative programming.

Domain-specific languages

Well-known examples of declarative domain-specific languages (DSLs) include the yacc parser generator input language, QML, the Make build specification language, Puppet's configuration management language, regular expressions, and a subset of SQL (SELECT queries, for example). DSLs have the advantage of being useful while not necessarily needing to be Turing-complete, which makes it easier for a language to be purely declarative.

Many markup languages such as HTML, MXML, XAML, XSLT or other user-interface markup languages are often declarative. HTML, for example, only describes what should appear on a webpage - it specifies neither control flow for rendering a page nor the page's possible interactions with a user.

As of 2013 some software systems combine traditional user-interface markup languages (such as HTML) with declarative markup that defines what (but not how) the back-end server systems should do to support the declared interface. Such systems, typically using a domain-specific XML namespace, may include abstractions of SQL database syntax or parameterised calls to web services using representational state transfer (REST) and SOAP.

Hybrid languages

Makefiles, for example, specify dependencies in a declarative fashion,[5] but include an imperative list of actions to take as well. Similarly, yacc specifies a context free grammar declaratively, but includes code snippets from a host language, which is usually imperative (such as C).

Logic programming

Logic programming languages such as Prolog state and query relations. The specifics of *how* these queries are answered is up to the implementation and its theorem prover, but typically take the form of some sort of unification. Like functional programming, many logic programming languages permit side effects, and as a result are not strictly declarative.

Modeling

Models, or mathematical representations, of physical systems may be implemented in computer code that is declarative. The code contains a number of equations, not imperative assignments, that describe ("declare") the behavioral relationships. When a model is expressed in this formalism, a computer is able to perform algebraic manipulations to best formulate the solution algorithm. The mathematical causality is typically imposed at the boundaries of the physical system, while the behavioral description of the system itself is declarative or acausal. Declarative modeling languages and environments include Analytica, Modelica and Simile.[6]

See also

- List of declarative programming languages
- Comparison of programming paradigms
- Inductive programming

References

1. Lloyd, J.W., *Practical Advantages of Declarative Programming*
2. Declarative language (http://foldoc.org/declarative%20language) in The Free On-line Dictionary of Computing, Editor Denis Howe.
3. "DAMP 2009: Workshop on Declarative Aspects of Multicore Programming" (http://www.cse.unsw.edu.au/~pls/damp09/). Cse.unsw.edu.au. 2009-01-20. Retrieved 2013-08-15.
4. Chakravarty, Manuel M. T. (14 February 1997). *On the Massively Parallel Execution of Declarative Programs* (http://www.cse.unsw.edu.au/~chak/papers/diss.ps.gz) (Doctoral dissertation). Technische Universität Berlin. Retrieved 26 February 2015. "In this context, the criterion for calling a programming language declarative is the existence of a clear, mathematically established correspondence between the language and mathematical logic such that a declarative semantics for the language can be based on the model or the proof theory (or both) of the logic."
5. [1] (http://phoenix.labri.fr/wiki/doku.php?id=an_overview_on_dsls) Archived (https://web.archive.org/web/20071023021126/http://phoenix.labri.fr/wiki/doku.php?id=an_overview_on_dsls) October 23, 2007, at the Wayback Machine.
6. "Declarative modelling" (http://www.simulistics.com/tour/declarative.htm). Simulistics. Retrieved 2013-08-15.

External links

- Frans Coenen. Characteristics of declarative programming languages (http://www.csc.liv.ac.uk/~frans/OldLectures/2CS24/declarative.html#detail). 1999.
- Robert Harper.
  - What, If Anything, Is A Declarative Language? (https://existentialtype.wordpress.com/2013/07/18/what-if-anything-is-a-declarative-language/). 2013.
  - There Is Such A Thing As A Declarative Language, and It's The World's Best DSL (https://existentialtype.wordpress.com/2013/07/22/there-is-such-a-thing-as-a-declarative-language/). 2013.
- Olof Torgersson. A Note on Declarative Programming Paradigms and the Future of Definitional Programming (http://www.cs.chalmers.se/~oloft/Papers/wm96/wm96.html). 1996.