Declaration (computer programming)

In computer programming, a **declaration** is a language construct that specifies properties of an identifier: it declares what a word (identifier) "means:.[1] *Declarations are most commonly used for functions, variables, constants, and classes, but can also be used for other entities such as enumerations and type definitions.*[1] *Beyond the name (the identifier itself) and the kind of entity (function, variable, etc.), declarations typically specify the data type (for variables and constants), or the type signature (for functions); types may also include dimensions, such as for arrays. A declaration is used to announce the existence of the entity to the compiler; this is important in those strongly typed languages that require functions, variables, and constants, and their types to be specified with a declaration before use, and is used in forward declaration.*[2] *The term "declaration" is frequently contrasted with the term "definition",*[1] *but meaning and usage varies significantly between languages; see below.*

Declarations are particularly prominent in languages in the ALGOL tradition, including the BCPL family, most prominently C and C++, and also Pascal. Java uses the term "declaration", though Java does not have separate declarations and definitions.

Contents

1 Declaration vs. definition
2 Declarations and Definitions
3 Undefined variables
4 See also
5 Notes
6 References
7 External links

Declaration vs. definition

A basic dichotomy is whether a declaration contains a *definition* or not: for example, whether a declaration of a constant or variable specifies the value of the constant (respectively, initial value of a variable), or only its type; and similarly whether a declaration of a function specifies the body (implementation) of the function, or only its type signature.[1] Not all languages make this distinction: in many languages, declarations always include a definition, and may be referred to as either "declarations" or "definitions", depending on the language.[a] However, these concepts are distinguished in languages that require declaration before use (for which forward declarations are used), and in languages where interface and implementation are separated: the interface contains declarations, the implementation contains definitions.[b]

In informal usage, a "declaration" refers only to a pure declaration (types only, no value or body), while a "definition" refers to a declaration that includes a value or body. However, in formal usage (in language specifications), "declaration" includes *both* of these senses, with finer distinctions by language: in C and C++, a declaration of a function that does not include a body is called a function prototype, while a declaration of a function that does include a body is called a "function definition". By contrast in Java declarations always include the body, and the word "definition" has no technical meaning in Java.

Declarations and Definitions

In the C-family of programming languages, declarations are often collected into header files, which are included in other source files that reference and use these declarations, but don't have access to the definition. The information in the header file provides the interface between code that uses the declaration and that which defines it, a form of information hiding. A declaration is often used in order to access functions or variables defined in different source files, or in a library. A mismatch between the definition type and the declaration type generates a compiler error.

For variables, definitions assign values to an area of memory that was reserved during the declaration phase. For functions, definitions supply the function body. While a variable or function may be declared many times, it is typically defined once (in C++, this is known as the One Definition Rule or ODR).

Dynamic languages such as JavaScript or Python generally allow functions to be redefined, that is, re-bound; a function is a variable much like any other, with a name and a value (the definition).

Here are some examples of declarations that are not definitions, in C:

```c
extern char example1;
extern int example2;
void example3(void);
```

Here are some examples of declarations that are definitions, again in C:

```c
char example1; /* Outside of a function definition it will be initialized to zero.  */
int example2 = 5;
void example3(void) { /* definition between braces */ }
```

Undefined variables

In some programming languages, an implicit declaration is provided the first time such a variable is encountered at compile time. In other languages, such a usage is considered to be an error, which may resulting in a diagnostic message. Some languages have started out with the implicit declaration behavior, but as they matured they provided an option to disable it (e.g. Perl's "`use strict`" or Visual Basic's "`Option Explicit`").

See also

- Function prototype
- Scope (programming)

Notes

a. For example, Java uses "declaration" (class declaration, method declaration), while Python uses "definition" (class definition, function definition).[3]

b. This distinction is observed in Pascal "units" (modules), and in conventional C and C++ code organization, which has header files consisting largely of pure declarations, and source files consisting of definitions, though this is not always strictly observed, nor enforced by the language.

References

1. "A declaration specifies the interpretation and attributes of a set of identifiers. A *definition* of an identifier is a declaration for that identifier that:
   - for an object [variable or constant], causes storage to be reserved for that object;
   - for a function, includes the function body;
   - for an enumeration constant, is the (only) declaration of the identifier;
   - for a typedef name, is the first (or only) declaration of the identifier."

   C11 specification, 6.7: Declarations, paragraph 5.
2. Mike Banahan. "2.5. Declaration of variables" (http://publications.gbdirect.co.uk/c_book/chapter2/variable_declaration.html). http://publications.gbdirect.co.uk/c_book/: GBdirect. Retrieved 2011-06-08. "[A] declaration [...] introduces just the name and type of something but allocates no storage[...]."
3. 7. Compound statements (https://docs.python.org/2/reference/compound_stmts.html), *The Python Language Reference*

External links

- Declare vs Define in C and C++ (http://www.cprogramming.com/declare_vs_define.html), Alex Allain
- 8.2. Declarations, Definitions and Accessibility (http://publications.gbdirect.co.uk/c_book/chapter8/declarations_and_definitions.html), *The C Book,* GBdirect
- Declarations and Definitions (C++) (http://msdn.microsoft.com/en-us/library/0e5kx78b.aspx), MSDN

  "Declarations tell the compiler that a program element or name exists. Definitions specify what code or data the name describes."

Retrieved from "https://en.wikipedia.org/w/index.php?title=Declaration_(computer_programming)&oldid=797007570"