WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Čeština
Deutsch
Eesti
Ελληνικά
Español
فارسی
Français
한국어
Italiano
Bahasa Melayu
Nederlands
日本語
Norsk bokmål
Олык марий
Polski
Português
Română
Русский
Svenska
ไทย
Türkçe
Українська
中文
Edit links

Boolean data type

From Wikipedia, the free encyclopedia

In computer science, the **Boolean data type** is a data type, having two values (usually denoted **true** and **false**), intended to represent the truth values of logic and Boolean algebra. It is named after George Boole, who first defined an algebraic system of logic in the mid 19th century. The Boolean data type is primarily associated with conditional statements, which allow different actions and change control flow depending on whether a programmer-specified Boolean *condition* evaluates to true or false. It is a special case of a more general **logical data type**; logic does not always have to be Boolean.

Contents [hide]

Generalities [edit]

In programming languages that have a built-in Boolean data type, such as Pascal and Java, the comparison operators such as `>` and `≠` are usually defined to return a Boolean value. Conditional and iterative commands may be defined to test Boolean-valued expressions.

Languages without an explicit Boolean data type, like C90 and Lisp, may still represent truth values by some other data type. Common Lisp uses an empty list for false, and any other value for true. C uses an integer type, where relational expressions like `i > j` and logical expressions connected by `&&` and `||` are defined to have value 1 if true and 0 if false, whereas the test parts of `if`, `while`, `for`, etc., treat any non-zero value as true.[1][2] Indeed, a Boolean variable may be regarded (and be implemented) as a numerical variable with a single binary digit (bit), which can store only two values. It is worth noting that the implementation of Booleans in computers are most likely represented as a full word, rather than a bit; this is usually due to the ways computers transfer blocks of information.

Most programming languages, even those that do not have an explicit Boolean type, have support for Boolean algebraic operations such as conjunction (`AND`, `&`, `*`), disjunction (`OR`, `|`, `+`), equivalence (`EQV`, `=`, `==`), exclusive or/non-equivalence (`XOR`, `NEQV`, `^`, `!=`), and negation (`NOT`, `~`, `!`).

In some languages, like Ruby, Smalltalk, and Alice the "true" and "false" values belong to separate classes—e.g. `True` and `False`, resp.—so there is no single Boolean "type."

In SQL, which uses a three-valued logic for explicit comparisons because of its special treatment of Nulls, the Boolean data type (introduced in SQL:1999) is also defined to include more than two truth values, so that SQL "Booleans" can store all logical values resulting from the evaluation of predicates in SQL. A column of Boolean type can also be restricted to just `TRUE` and `FALSE` though.

In the lambda calculus model of computing, Boolean values can be represented as Church Booleans.

ALGOL, Java, and C# [edit]

One of the earliest programming languages to provide an explicit **Boolean** data type was ALGOL 60 (1960) with values **true** and **false** and logical operators denoted by symbols '$\wedge$' (and), '$\vee$' (or), '$\supset$' (implies), '$\equiv$' (equivalence), and '$\neg$' (not). Due to input device limitations of the time, however, most compilers used alternative representations for the latter, such as `AND` or `'AND'`.

This approach ("Boolean is a separate built-in primitive data type") was adopted by many later programming languages, such as ALGOL 68 (1970),[3] Java, and C#.

Fortran [edit]

The first version of FORTRAN (1957) and its successor FORTRAN II (1958) did not have logical values or operations; even the conditional `IF` statement took an arithmetic expression and branched to one of three locations according to its sign; see arithmetic IF. FORTRAN IV (1962), however, followed the ALGOL 60 example by providing a Boolean data type (`LOGICAL`), truth literals (`.TRUE.` and `.FALSE.`), Boolean-valued numeric comparison operators (`.EQ.`, `.GT.`, etc.), and logical operators (`.NOT.`, `.AND.`, `.OR.`). In `FORMAT` statements, a specific control character ('`L`') was provided for the parsing or formatting of logical values.[4]

Lisp and Scheme [edit]

The Lisp language (1958) never had a built-in Boolean data type. Instead, conditional constructs like `cond` assume that the logical value "false" is represented by the empty list `()`, which is defined to be the same as the special atom `nil` or `NIL`; whereas any other s-expression is interpreted as "true". For convenience, most modern dialects of Lisp predefine the atom `t` to have value `t`, so that one can use `t` as a mnemonic notation for "true".

This approach ("any value can be used as a Boolean value") was retained in most Lisp dialects (Common Lisp, Scheme, Emacs Lisp), and similar models were adopted by many scripting languages, even ones that do have a distinct Boolean type or Boolean values; although which values are interpreted as "false" and which are "true" vary from language to language. In Scheme, for example, the "false" value is an atom distinct from the empty list, so the latter is interpreted as "true".

Pascal, Ada, and Haskell [edit]

The Pascal language (1978) introduced the concept of programmer-defined enumerated types. A built-in `Boolean` data type was then provided as a predefined enumerated type with values `FALSE` and `TRUE`. By definition, all comparisons, logical operations, and conditional statements applied to and/or yielded `Boolean` values. Otherwise, the `Boolean` type had all the facilities which were available for enumerated types in general — such as ordering and use as indices. On the other hand, the conversion between `Boolean`s and integers (or any other types) still required explicit tests or function calls, as in ALGOL 60. This approach ("Boolean is an enumerated type") was adopted by most later languages which had enumerated types, such as Modula, Ada and Haskell.

C, C++, Objective-C, AWK, Perl [edit]

The initial implementations of the C language (1972) provided no Boolean type, and to this day Boolean values are commonly represented by integers (`int`s) in C programs. The comparison operators ('`>`', '`==`', etc.) are defined to return a signed integer (`int`) result, either 0 (for false) or 1 (for true). Logical operators ('`&&`', '`||`', '`!`', etc.) and condition-testing statements ('`if`', '`while`') assume that zero is false and all other values are true. One problem with this approach is that the tests `if(t==TRUE){...}` and `if(t)` are not equivalent.

After enumerated types (`enum`s) were added to the ANSI version of C (1989), many C programmers got used to defining their own Boolean types as such, for readability reasons. However, enumerated types are equivalent to integers according to the language standards; so the effective identity between Booleans and integers is still valid for C programs.

Standard C (since C99) and several dialects of C such as Objective-C provide definitions of a Boolean type as an integer type and macros for "false" and "true" as 0 and 1, respectively. Thus logical values can be stored in integer variables, and used anywhere integers would be valid, including in indexing, arithmetic, parsing, and formatting. This approach ("Boolean values are just integers") has been retained in all later versions of C.

C++ has a separate Boolean data type (`'bool'`), but with automatic conversions from scalar and pointer values that are very similar to those of C. This approach was adopted also by many later languages, especially by some scripting ones such as AWK.

Objective-C also has a separate Boolean data type (`'BOOL'`), with possible values being `YES` or `NO`, equivalents of true and false respectively.[5]

Perl has many values of `false`: the number zero, the strings "0" and "", the empty list "()", and the special value `undef`.[6] Everything else evaluates to `true`.

Python, Ruby, and JavaScript [edit]

In Python from version 2.3 forward, there is a `bool` type which is a subclass of `int`, the standard integer type.[7] It has two possible values: `True` and `False`, which are "special versions" of 1 and 0 respectively and behave as such in arithmetic contexts. In addition, a numeric value of zero (integer or fractional), the null value (`None`), the empty string, and empty containers (i.e. lists, sets, etc.) are considered Boolean false; all other values are considered Boolean true by default.[8] Classes can define how their instances are treated in a Boolean context through the special method `__nonzero__` (Python 2) or `__bool__` (Python 3). For containers, `__len__` (the special method for determining the length of containers) is used if the explicit Boolean conversion method is not defined.

In Ruby, on the other hand, only `nil` (Ruby's null value) and a special `false` object are "false", everything else (including the integer 0 and empty arrays) is "true".

In JavaScript, the empty string (`""`), `null`, `undefined`, `NaN`, +0, −0 and `false`[9] are sometimes called "falsy", and their complement, "truthy", to distinguish between strictly type-checked and coerced Booleans.[10] Languages such as PHP also use this approach.

SQL [edit]

The SQL:1999 standard introduced a BOOLEAN data type as an optional feature (T031). When restricted by a `NOT NULL` constraint, a SQL BOOLEAN behaves like Booleans in other languages. In SQL however, the BOOLEAN type is nullable by default like all other SQL data types, meaning it can have the special null value as well. Although the SQL standard defines three literals for the BOOLEAN type—TRUE, FALSE and UNKNOWN—, it also says that the NULL BOOLEAN and UNKNOWN "may be used interchangeably to mean exactly the same thing".[11][12] This has caused some controversy because the identification subjects UNKNOWN to the equality comparison rules for NULL. More precisely UNKNOWN = UNKNOWN is not TRUE but UNKNOWN/NULL.[13] As of 2012 few major SQL systems implement the T031 feature.[14] PostgreSQL is a notable exception, although it does not implement the UNKNOWN literal; NULL can be used instead.[15] (PostgreSQL does implement the IS UNKNOWN operator, which is part of an orthogonal feature, F571.) In other SQL implementations various ad hoc solutions are used, like bit, byte, and character to simulate Boolean values.

See also [edit]

- true and false commands for shell scripting
- Shannon's expansion
- stdbool.h — C99 definitions for boolean

References [edit]

1. ^ Kernighan, Brian W; Ritchie, Dennis M (1978). *The C Programming Language* (1st ed.). Englewood Cliffs, NJ: Prentice Hall. p. 41. ISBN 0-13-110163-3.
2. ^ Plauger, PJ; Brodie, Jim (1992) [1989]. *ANSI and ISO Standard C Programmer's reference*. Microsoft Press. pp. 86–93. ISBN 1-55615-359-7.
3. ^ "Report on the Algorithmic Language ALGOL 68, Section 10.2.2." (PDF). August 1968. Retrieved April 2007.
4. ^ Digital Equipment Corporation, *DECSystem10 FORTRAN IV Programmers Reference Manual*. Reprinted in *Mathematical Languages Handbook*. Online version accessed 2011-11-16.
5. ^ http://developer.apple.com/library/ios/#documentation/cocoa/conceptual/ProgrammingWithObjectiveC/FoundationTypesandCollections/FoundationTypesandCollections.html
6. ^ "perlsyn - Perl Syntax / Truth and Falsehood". Retrieved 10 September 2013.
7. ^ Van Rossum, Guido (3 April 2002). "PEP 285 – Adding a bool type". Retrieved 15 May 2013.
8. ^ "Expressions". *Python v3.3.2 documentation*. Retrieved 15 May 2013.
9. ^ "ECMAScript Language Specification" (PDF). p. 43.
10. ^ "The Elements of JavaScript Style". Douglas Crockford. Retrieved 5 March 2011.
11. ^ C. Date (2011). *SQL and Relational Theory: How to Write Accurate SQL Code*. O'Reilly Media, Inc. p. 83. ISBN 978-1-4493-1640-2.
12. ^ ISO/IEC 9075-2:2011 §4.5
13. ^ Martyn Prigmore (2007). *Introduction to Databases With Web Applications*. Pearson Education Canada. p. 197. ISBN 978-0-321-26359-9.
14. ^ Troels Arvin, Survey of BOOLEAN data type implementation

15. ^ http://www.postgresql.org/docs/current/static/datatype-boolean.html

v · t · e	Data types [hide]
Uninterpreted	Bit · Byte · Trit · Tryte · Word
Numeric	Bignum · Complex · Decimal · Fixed point · Floating point (Double precision · Extended precision · Half precision · Minifloat · Octuple precision · Quadruple precision · Single precision) · Integer (signedness) · Interval · Rational
Text	Character · String (null-terminated)
Pointer	Address (physical · virtual) · Reference
Composite	Algebraic data type (generalized) · Array · Associative array · Class · Dependent · Equality · Inductive · List · Object (metaobject) · Option type · Product · Record · Set · Union (tagged)
Other	**Boolean** · Bottom type · Collection · Enumerated type · Exception · Function type · Opaque data type · Recursive data type · Semaphore · Stream · Top type · Type class · Unit type · Void
Related topics	Abstract data type · Data structure · Generic · Kind (metaclass) · Parametric polymorphism · Primitive data type · Protocol (interface) · Subtyping · Type constructor · Type conversion · Type system

Categories: Boolean algebra | Data types | Primitive types