= Decltype =

In the C + + programming language , decltype is a keyword used to query the type of an expression . Introduced in C + + 11 , its primary intended use is in generic programming , where it is often difficult , or even impossible , to express types that depend on template parameters .

As generic programming techniques became increasingly popular throughout the 1990s , the need for a type @-@ deduction mechanism was recognized . Many compiler vendors implemented their own versions of the operator , typically called typeof , and some portable implementations with limited functionality , based on existing language features were developed . In 2002 , Bjarne Stroustrup proposed that a standardized version of the operator be added to the C + + language , and suggested the name " decltype " , to reflect that the operator would yield the " declared type " of an expression .

decltype 's semantics were designed to cater to both generic library writers and novice programmers . In general , the deduced type matches the type of the object or function exactly as declared in the source code . Like the sizeof operator , decltype 's operand is not evaluated .

= = Motivation = =

With the introduction of templates into the C + + programming language , and the advent of generic programming techniques pioneered by the Standard Template Library , the need for a mechanism for obtaining the type of an expression , commonly referred to as typeof , was recognized . In generic programming , it is often difficult or impossible to express types that depend on template parameters , in particular the return type of function template instantiations .

Many vendors provide the typeof operator as a compiler extension . As early as 1997 , before C + + was fully standardized , Brian Parker proposed a portable solution based on the sizeof operator . His work was expanded on by Bill Gibbons , who concluded that the technique had several limitations and was generally less powerful than an actual typeof mechanism . In an October 2000 article of Dr. Dobb 's Journal , Andrei Alexandrescu remarked that " [h] aving a typeof would make much template code easier to write and understand . " He also noted that " typeof and sizeof share the same backend , because sizeof has to compute the type anyway . " Andrew Koenig and Barbara E. Moo also recognized the usefulness of a built @-@ in typeof facility , with the caveat that " using it often invites subtle programming errors , and there are some problems that it cannot solve . " They characterized the use of type conventions , like the typedefs provided by the Standard Template Library , as a more powerful and general technique . However , Steve Dewhurst argued that such conventions are " costly to design and promulgate " , and that it would be " much easier to ... simply extract the type of the expression . " In a 2011 article on C + + 0x , Koenig and Moo predicted that " decltype will be widely used to make everyday programs easier to write . "

In 2002 , Bjarne Stroustrup suggested extending the C + + language with mechanisms for querying the type of an expression , and initializing objects without specifying the type . Stroustrup observed that the reference @-@ dropping semantics offered by the typeof operator provided by the GCC and EDG compilers could be problematic . Conversely , an operator returning a reference type based on the lvalue @-@ ness of the expression was deemed too confusing . The initial proposal to the C + + standards committee outlined a combination of the two variants ; the operator would return a reference type only if the declared type of the expression included a reference . To emphasize that the deduced type would reflect the " declared type " of the expression , the operator was proposed to be named decltype .

One of the cited main motivations for the decltype proposal was the ability to write perfect forwarding function templates . It is sometimes desirable to write a generic forwarding function that returns the same type as the wrapped function , regardless of the type it is instantiated with . Without decltype , it is not generally possible to accomplish this . An example , which also utilizes the trailing @-@ return @-@ type :

decltype is essential here because it preserves the information about whether the wrapped function returns a reference type .

= = Semantics = =

Similarly to the sizeof operator , the operand of decltype is unevaluated . Informally , the type returned by decltype (e) is deduced as follows :
If the expression e refers to a variable in local or namespace scope , a static member variable or a function parameter , then the result is that variable 's or parameter 's declared type
Otherwise , if e is an lvalue , decltype (e) is T & , where T is the type of e ; if e is an xvalue , the result is T & & ; otherwise , e is a prvalue and the result is T.
These semantics were designed to fulfill the needs of generic library writers , while at the same time being intuitive for novice programmers , because the return type of decltype always matches the type of the object or function exactly as declared in the source code . More formally , Rule 1 applies to unparenthesized id @-@ expressions and class member access expressions . Example :
The reason for the difference between the latter two invocations of decltype is that the parenthesized expression (a- > x) is neither an id @-@ expression nor a member access expression , and therefore does not denote a named object . Because the expression is an lvalue , its deduced type is " reference to the type of the expression " , or const double & .
In December 2008 , a concern was raised to the committee by Jaakko Järvi over the inability to use decltype to form a qualified @-@ id , which is inconsistent with the intent that decltype (e) should be treated " as if it were a typedef @-@ name " . While commenting on the formal Committee Draft for C + + 0x , the Japanese ISO member body noted that " a scope operator (: :) cannot be applied to decltype , but it should be . It would be useful in the case to obtain member type (nested @-@ type) from an instance as follows " :
This , and similar issues pertaining to the wording inhibiting the use of decltype in the declaration of a derived class and in a destructor call , were addressed by David Vandevoorde , and voted into the working paper in March 2010 .

= = Availability = =

decltype is included in the C + + Language Standard since C + + 11 . It is provided by a number of compilers as an extension . Microsoft 's Visual C + + 2010 and later compilers provide a decltype type specifier that closely mimics the semantics as described in the standards committee proposal . It can be used with both managed and native code . The documentation states that it is " useful primarily to developers who write template libraries . " decltype was added to the mainline of the GCC C + + compiler in version 4 @.@ 3 , released on March 5 , 2008 @.@ decltype is also present in Codegear 's C + + Builder 2009 , the Intel C + + Compiler , and Clang .