Generic selection (since C11)

Provides a way to choose one of several expressions at compile time, based on a type of a controlling expression

Syntax

`_Generic (` *controlling-expression* **`,`** *association-list* **`)`**	(since C11)

where *association-list* is a comma-separated list of associations, each of which has the syntax

type-name **`:`** *expression*

`default :` *expression*

where

type-name -	any complete object type that isn't variably-modified (that is, not VLA or pointer to VLA).
controlling-expression -	any expression (except for the comma operator) whose type must be compatible with one of the *type-name*s if the `default` association is not used
expression -	any expression (except for the comma operator) of any type and value category

No two *type-name*s in the *association-list* may specify compatible types. There may be only one association that uses the keyword `default`. If `default` is not used and none of the *type-name*s are compatible with the type of the controlling expression, the program will not compile.

Explanation

First, the type of *controlling-expression* undergoes lvalue conversions. The conversion is performed in type domain only: it discards the top-level cvr-qualifiers and atomicity and applies array-to-pointer/function-to-pointer transformations to the type of the controlling expression, without initiating any side-effects or calculating any values.

The type after conversion is compared with *type-name*s from the list of associations.

If the type is compatible with the *type-name* of one of the associations, then the type, value, and value category of the generic selection are the type, value, and value category of the *expression* that appears after the colon for that *type-name*.

If none of the *type-name*s are compatible with the type of the *controlling-expression*, and the `default` association is provided, then the type, value, and value category of the generic selection are the type, value, and value category of the expression after the `default :` label.

Notes

The *controlling-expression* and the *expression*s of the selections that are not chosen are never evaluated.

Because of the lvalue conversions, `"abc"` matches `char*` and not `char[4]` and `(int const){0}` matches `int`, and not `const int`.

All value categories, including function designators and void expressions, are allowed as *expression*s in a generic selection, and if selected, the generic selection itself has the same value category.

The type-generic math macros from `<tgmath.h>`, introduced in C99, were implemented in compiler-specific manner. Generic selections, introduced in C11, gave the programmers the ability to write similar type-dependent code.

Generic selection is similar to overloading in C++ (where one of several functions is chosen at compile time based on the types of the arguments), except that it makes the selection between arbitrary expressions.

Keywords

`_Generic`, `default`

Example

Run this code

```c
#include <math.h>
#include <stdio.h>

// Possible implementation of the tgmath.h macro cbrt
#define cbrt(X) _Generic((X),     \
              long double: cbrtl, \
                  default: cbrt,  \
                    float: cbrtf  \
              )(X)

int main(void)
{
    double x = 8.0;
    const float y = 3.375;
    printf("cbrt(8.0) = %f\n", cbrt(x));    // selects the default cbrt
    printf("cbrtf(3.375) = %f\n", cbrt(y)); // converts const float to float,
                                            // then selects cbrtf
}
```

Output:

```
cbrt(8.0) = 2.000000
cbrtf(3.375) = 1.500000
```

Defect reports

The following behavior-changing defect reports were applied retroactively to previously published C standards.

DR	Applied to	Behavior as publish
DR 481 (https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2396.htm#dr_481)	C11	it was underspecified if the controlling expression

References

- C23 standard (ISO/IEC 9899:2024):
  - 6.5.1.1 Generic selection (p: TBD)
- C17 standard (ISO/IEC 9899:2018):
  - 6.5.1.1 Generic selection (p: 56-57)
- C11 standard (ISO/IEC 9899:2011):
  - 6.5.1.1 Generic selection (p: 78-79)

See also

C++ documentation for **Templates**

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=c/language/generic&oldid=178619"