O

object A region of storage, the contents of which can represent values corresponding to a given type.

obsolescent A term applied to a practice or approach declaring it to be "out dated." Labelling something as obsolescent in a standard paves the way for it to be dropped from future versions of that standard. Some people use the term "deprecated" instead. *See also* future language directions; future library directions.

octal constant *See* constant, integer.

octal escape sequence A sequence of the form `\`*ddd* that represents the character having a bit-pattern with value *ddd* octal. (*ddd* is a sequence of 1–3 octal digits.) The escape sequence `\0` is commonly used to represent the null character because it is required to have an internal representation of zero binary. The maximum number of octal digits is fixed at three. However, the maximum number for a hexadecimal escape sequence is not fixed.

`offsetof`[C89] A macro, defined in stddef.h, that allows you to find the offset (in bytes) of a member from the start of its parent structure. `offsetof` expands into an integer constant expression that has type `size_t`. If the member is a bit-field, the behavior is undefined because you cannot take the address of a bit-field. An example of its use is as follows:

```
#include <stddef.h>

struct tag {
        int i;
        double d;
};

size_t value = offsetof(struct tag, d);
```

Open Group Successor to X/Open.

operand The expression on which an operator acts. Unary operators have one operand, binary have two operands, and ternary have three operands. *See also* order of evaluation.

`operator` A C++ keyword that is not part of Standard C. If you plan to move C code to a C++ environment in the future, you should refrain from using `operator` as an identifier in new C code you write.

operator One or more tokens that, when taken together, specify an operation to be performed on one or more operands producing either a

value, an object or function designator, a side effect, or a combination of these. (While most operators comprise a single token, a few do not. For example, `[ ]`, function call `()`, and `?:`.) The Standard C operators are

Symbol	*Meaning*
`!`	logical negation
`!=`	inequality
`%`	remainder
`&`	AND (bitwise) and address-of
`&&`	AND (logical)
`()`	cast and function call
`*`	multiplication and indirection
`+`	addition and unary plus
`++`	increment, prefix or postfix
`,`	comma
`-`	subtraction and unary minus
`--`	decrement, prefix or postfix
`->`	structure/union pointer (arrow)
`.`	structure/union member (dot)
`/`	division
`<`	less-than
`<<`	left-shift
`<=`	less-than-or-equal-to
`=`	assignment
`==`	equality
`>`	greater-than
`>=`	greater-than-or-equal-to
`>>`	right shift
`?:`	conditional
`[ ]`	subscript
`^`	OR (bitwise exclusive)
`|`	OR (bitwise inclusive)
`||`	OR (logical)
`~`	complement
op`=`	assignment, compound
`sizeof`	compute object size at translation-time

The unary plus operator was an invention of C89.

operator, binary An operator having two operands. Examples are `/`, `<<`, and `&&`.

operator precedence The order in which terms are grouped, as determined by a set of rules, written in the form of a precedence table. An operator in a higher row of the precedence table has higher precedence. The

grouping of terms across operators having the same precedence is resolved by their associativity. Precedence and associativity are shown in the following table:

Operator	*Associativity*
`() [] -> . ++ --`	Left to Right
`! ~ ++ -- + - * & (`*type*`) sizeof`	Right to Left
`* / %`	Left to Right
`+ -`	Left to Right
`<< >>`	Left to Right
`< <= > >=`	Left to Right
`== !=`	Left to Right
`&`	Left to Right
`^`	Left to Right
`|`	Left to Right
`&&`	Left to Right
`||`	Left to Right
`?:`	Right to Left
`= += -= *= /= %= >>= <<= &= ^= |=`	Right to Left
`,`	Left to Right

Note that C89 promoted postfix versions of `++` and `--` to have higher precedence than their prefix counterparts. This breaks no old code; however, it did make some new expressions possible (such as `p++->m`).

Operator precedence is not related to the order of evaluation of individual terms. For example, in the expression `f() + g()` the precedence is clear—both functions are called before their return values are added together—but the order in which the functions are called is unspecified.

operator, ternary An operator having three operands. C has one of these, the conditional operator `?:`.

operator, unary An operator having only one operand. Examples are `++`, `!`, and `~`.

or[C95] A macro, defined in iso646.h, that expands to the token `||`. It allows programmers using source character sets (such as ISO 646) that are missing certain characters necessary for writing C programs, to enter those characters using identifiers instead. Note that in C++, this name is a keyword.

OR **assignment operator, bitwise exclusive** A binary operator, `^=`, that permits exclusive OR and assignment to be combined such that *exp1* `^=` *exp2* is equivalent to *exp1* `=` *exp1* `^` *exp2* except that in the former, *exp1* is only evaluated once. Both operands must have integer type. The

left operand must be a modifiable lvalue. The order of evaluation of the operands is unspecified. The type of the result is the type of *exp1*. This operator associates right to left. *See also* assignment operator, compound.

OR **assignment operator, bitwise inclusive** A binary operator, `|=`, that permits OR and assignment to be combined such that *exp1* `|=` *exp2* is equivalent to *exp1* `=` *exp1* `|` *exp2* except that in the former, *exp1* is only evaluated once. Both operands must have integer type. The left operand must be a modifiable lvalue. The order of evaluation of the operands is unspecified. The type of the result is the type of *exp1*. This operator associates right to left. *See also* assignment operator, compound.

OR **operator, bitwise exclusive** A binary operator, `^`, that performs a bitwise exclusive-OR of its operands. Both operands must have integer type. The order of evaluation of the operands is unspecified. The usual arithmetic conversions are performed on the operands. This operator associates left to right.

OR **operator, bitwise inclusive** A binary operator, `|`, that performs a bitwise inclusive-OR of its operands. Both operands must have integer type. The order of evaluation of the operands is unspecified. The usual arithmetic conversions are performed on the operands. This operator associates left to right.

OR **operator, logical** A binary operator, `||`, that performs a logical OR of its operands. Both operands must have scalar type. The result has type `int` and value 0 (if false) or 1 (if true). There is a sequence point after the evaluation of the left operand. The left operand is evaluated first, and if it tests true, the right operand is not evaluated. This operator associates left to right.

order of evaluation The order in which terms in expressions are evaluated. Very few C operators specify the order of evaluation of their operands. Those that do are logical OR, logical AND, comma, and conditional. These four operators also contain sequence points. *See also* evaluation.

ordinary identifier name space The name space used for variables, functions, `typedef` names, and enumeration constants.

`or_eq`[C95] A macro, defined in iso646.h, that expands to the token `|=`. It allows programmers using source character sets (such as ISO 646) that are missing certain characters necessary for writing C programs, to enter those characters using identifiers instead. Note that in C++, this name is a keyword.

`overload` An archaic C++ keyword.

♢ ♢ ♢