CSD101: Introduction to computing and programming (ICP)

Relational operators

Relational operator expressions return true (1) if condition holds, or false (0) if it does not hold. **C** unfortunately, implements booleans as just integers with 1 standing for true and 0 standing for false. This is not the case for many later languages.

Operator	**Meaning**	**Example**
$==$	Equal to	$a == b$
$!=$	Not equal to	$a != b$
$>$	Greater than	$a > b$
$a < b$	Less than	$a < b$
$>=$	Greater than equal to	$a >= b$
$<=$	Less than equal to	$a <= b$

Logical operators

The value returned is true (1) or false (0). $e1$, $e2$ are logical expressions that evaluate to true or false.

Operator	**Meaning**	**Example**
$\|\|$	Logical or	$e1 \|\| e2$
&&	Logical and	$e1 \&\& e2$
!	Logical not	$!e1$

Assignment operators

v is a variable and e is an expression that evaluate to a value compatible with the type of v.

Operator	**Meaning**	**Example**
$=$	Assignment	$v = e1$, v is assigned value of expn $e1$, returns value of $e1$.
$+=$	Abbr. assignment	$v+ = e$ same as $v = v + e$.
$-=$	Abbr. assignment	$v- = e$ same as $v = v - e$.
$*=$	Abbr. assignment	$v* = e$ same as $v = v * e$
$/=$	Abbr. assignment	$v/ = e$ same as $v = v/e$
$\%=$	Abbr. assignment	$v\% = e$ same as $v = v\%e$. v is an int, e evaluates to an int value.

Similar abbreviated assignment operators exist for bitwise operators. Bitwise operators are discussed later.

Miscellaneous operators

e is a boolean expression evaluating to true or false. $e1$, $e2$ are expressions. v is a variable. arr is an array variable.

Operator	**Meaning**	**Example**
$e?e1 : e2$	Conditional evaln	$e?e1 : e2$ - return value of $e1$ if e true else value of $e2$
$\& < var >$	Address of	$\&v$ - address of variable v
$*$	Contents of	$*v$. where v is a pointer
$< var > []$	Array access	$arr[10]$ - Eleventh element in the array arr
,	Comma operator	$e1, e2$ - evaluates $e1$ then $e2$ returns value of $e2$
$f(< arguments >)$	Function call	$f(a1, a2)$ - f a function, $a1$, $a2$ are arguments that evaluate to compatible values
$sizeof()$	Size of operator	$sizeof(v)$, $sizeof(< type >)$ - gives storage required in bytes
$(< type >) < expn >$	Type conversion	$(float)\, i$ - converts integer i to `float` .

Associativity and precedence

- What is the result of $10 + 3 * 2$? 26 or 16.

Associativity and precedence

- What is the result of $10 + 3 * 2$? 26 or 16.
  Answer: 16 because $*$ has higher precedence than $+$. A higher precedence operator always executes first. $*$, $/$ have higher precedence than $+$, $-$.
- What is the result of $10/3 * 2$? 6 or 1.

Associativity and precedence

- What is the result of $10 + 3 * 2$? 26 or 16.
  Answer: 16 because $*$ has higher precedence than $+$. A higher precedence operator always executes first. $*$, $/$ have higher precedence than $+$, $-$.
- What is the result of $10/3 * 2$? 6 or 1.
  Answer: 6 because $/$, $*$ have the same precedence but they are left-to-right associative.
- Brackets '(' and ')' can be used to change execution order. A bracketed expression has the highest precedence and is always executed first. So, $10/(3 * 2) = 1$.

Precedence and associativity of operators

Operators	Associativity
`( ) [ ] . ++ (postfix) -- (postfix)`	L to R
`++ (prefix) -- (prefix) !~ sizeof(type) + (unary) - (unary) & (address) * (indirection)`	R to L
`* / %`	L to R
`+ -`	L to R
`<< >>`	L to R
`< <= > >=`	L to R
`== !=`	L to R
`&`	L to R
`^`	L to R
`|`	L to R
`&&`	L to R
`||`	L to R
`?:`	R to L
`= += -= *= /= %= >>= <<= &= ^= |=`	R to L
`, (comma operator)`	L to R