

Relational and Logical Operators

Topics

- Relational Operators and Expressions
- The if Statement
- The if-else Statement
- Nesting of if-else Statements
- Logical Operators and Expressions
- Truth Tables

Reading

• Sections 2.6, 4.10, 4.11

Relational Operators < less than > greater than <= less than or equal to greater than or equal to >= == is equal to != is not equal to Relational expressions evaluate to the integer values 1 (true) or 0 (false). All of these operators are called **binary operators** because they take two expressions as operands.

	Practice with Relational Expressions				
ir	nt a = 1, b =	= 2, c = 3	•		
E	xpression	Value	Expression	Value	_
a	< C		a + b >= c		
b	<= C		a + b == c		
с	<= a		a != b		
a	> b		a + b != c		
b	>= C				4
b c a	<= c <= a > b		a + b == c a != b		4





• Arithmetic expressions evaluate to numeric values.

5

- An arithmetic expression that has a value of zero is false.
- An arithmetic expression that has a value other than zero is true.

	with Arithmo pressions	etic	
int $a = 1, b$ float $x = 3.33$	9 = 2, c = 3 ; 3, y = 6.66 ;		
Expression	Numeric Value	True/Fal	se
a + b b - 2 * a c - b - a c - a y - x y - 2 * x			



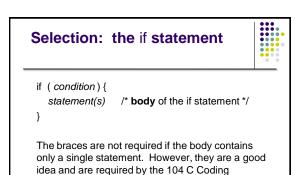
Review: Structured Programming

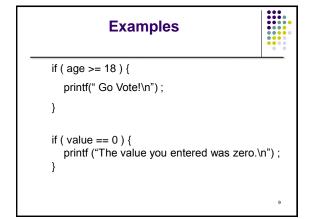
- All programs can be written in terms of only three control structures
 - The sequence structure
 - Unless otherwise directed, the statements are executed in the order in which they are written.
 - The selection structure
 - Used to choose among alternative courses of action.

• The repetition structure

Standards.

 Allows an action to be repeated while some condition remains true.



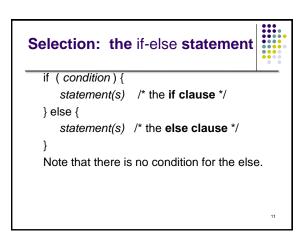


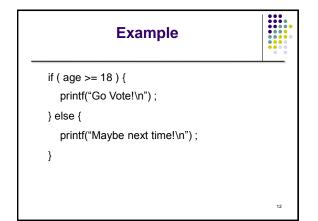
Good Programming Practice

• Always place braces around the body of an if statement.

10

- Advantages:
 - Easier to read
 - Will not forget to add the braces if you go back and add a second statement to the body
 - Less likely to make a semantic error
- Indent the body of the if statement 3 to 4 spaces -- be consistent!





Another Example



```
if ( value == 0 ) {
    printf ("The value you entered was zero.\n") ;
} else {
    printf ("Value = %d.\n", value) ;
```

Good Programming Practice



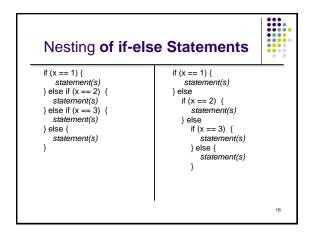
14

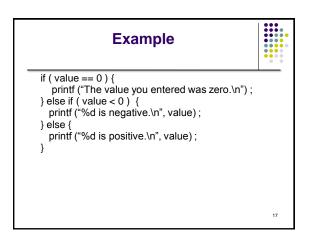
13

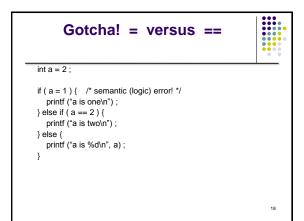
- Always place braces around the bodies of the if and else clauses of an if-else statement.
- Advantages:

}

- Easier to read
- Will not forget to add the braces if you go back and add a second statement to the clause
- Less likely to make a semantic error
- Indent the bodies of the if and else clauses 3 to 4 spaces -- be consistent!



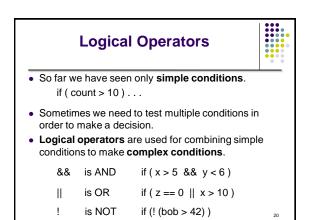


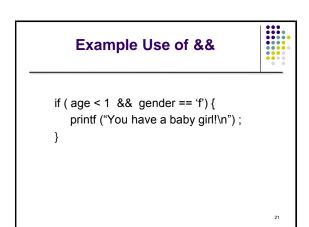




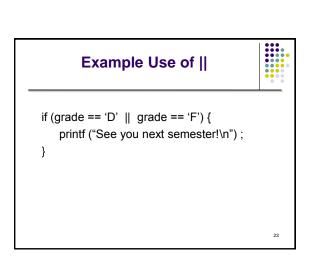


- The statement if (a = 1) is syntactically correct, so no error message will be produced. (Some compilers will produce a warning.) However, a semantic (logic) error will occur.
- An assignment expression has a value -- the value being assigned. In this case the value being assigned is 1, which is true.
- If the value being assigned was 0, then the expression would evaluate to 0, which is false.
- This is a VERY common error. So, if your if-else structure always executes the same, look for this typographical error.

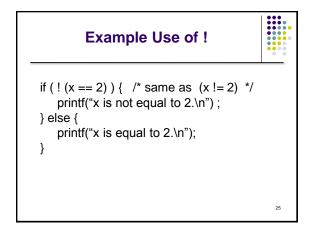




-	Truth Tab	le for &&	
Expression ₁	Expression ₂	Expression ₁ && Express	sion ₂
0	0	0	
0	nonzero	0	
nonzero	0	0	
nonzero	nonzero	1	
	p ₂ && … && ا subcondition	Exp _n will evaluate to 1 s are true.	(true)



-	Truth Table for					
Expression ₁	Expression ₂	Expression ₁ Expressi	on ₂			
0	0	0				
0	nonzero	1				
nonzero	0	1				
nonzero	nonzero	1				
	Exp ₁ && Exp ₂ && && Exp _n will evaluate to 1 (true) if only ONE subcondition is true.					
			24			

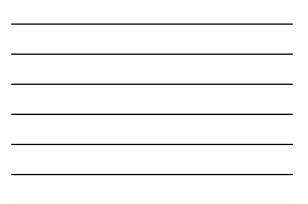


Truth ⁻	Truth Table for !		
Expression	! Expression		
0	1		
nonzero	0		
		26	

Operator Precedence and Associativity		
Precedence	Associativity	
()	left to right/inside-out	
* / %	left to right	
+ (addition) - (subtraction)	left to right	
< <= > >=	left to right	
== !=	left to right	
&&	left to right	
11	left to right	
=	right to left	



;	Some Pr	actice E	xpressions	
	int a = 1, b = 0,	c = 7;		
	Expression a b c a + b a && b a & b l c !!c !!c !!c !!c !!c a && !b a < b && b < c	<u>Numeric Value</u>	<u>True/False</u>	
	a > b && b < c a >= b b > c			28



More Practice	
Given int a = 5, b = 7, c = 17 ;	
evaluate each expression as True or False.	
1. c / b == 2 2. c % b <= a % b 3. b + c / a != c - a 4. (b < c) && (c == 7) 5. (c + 1 - b == 0) (b = 5)	
	29