Relational and Logical Operators

CMSC 104, Fall 2012
John Y. Park

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

int a = 1, b = 2, c = 3 ;

Expression	Value	Expression	Value
a < c		a + b >= c	
b <= c		a + b == c	
c <= a		a != b	
a > b		a + b != c	
b >= c			

Arithmetic Expressions: True or False

- **Arithmetic expressions** evaluate to numeric values.
- An arithmetic expression that has a value of zero is false.
- An arithmetic expression that has a value other than zero is true.

Practice with Arithmetic Expressions

```
int     a = 1, b = 2, c = 3 ;
float   x = 3.33, y = 6.66 ;
```

Expression	Numeric Value	True/False
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
    - Allows an action to be repeated while some condition remains true.

Selection: the if statement

```
if  ( condition ) {
    statement(s)      /* body of the if statement */
}
```

The braces are not required if the body contains only a single statement. However, they are a good idea and are required by the 104 C Coding Standards.

Examples

```
if ( age >= 18 ) {
    printf(" Go Vote!\n") ;
}

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

Good Programming Practice

- Always place braces around the body of an if statement.
- 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!

Selection: the if-else statement

```
if  ( condition ) {
    statement(s)    /* the if clause */
} else {
    statement(s)   /* the else clause */
}
```

Note that there is no condition for the else.

Example

```
if ( age >= 18 ) {
   printf("Go Vote!\n") ;
} else {
   printf("Maybe next time!\n") ;
}
```

Another Example

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

13

Good Programming Practice

- 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!

14

Nesting of if-else Statements

```
if ( condition₁ ) {
    statement(s)
} else if ( condition₂ ) {
   statement(s)
}
    . . .                /* more else if clauses may be here */
} else {
   statement(s)     /* the default case */
}
```

15

Nesting of if-else Statements

```
if (x == 1) {
    statement(s)
} else if (x == 2)  {
   statement(s)
} else if (x == 3)  {
   statement(s)
} else {
   statement(s)
}
```

```
if (x == 1) {
    statement(s)
} else
   if (x == 2)  {
      statement(s)
   } else
      if (x == 3)  {
         statement(s)
      } else {
         statement(s)
      }
```

Example

```
if ( value == 0 ) {
    printf (“The value you entered was zero.\n”) ;
} else if ( value < 0 )  {
   printf (“%d is negative.\n”, value) ;
} else {
   printf (“%d is positive.\n”, value) ;
}
```

Gotcha! = versus ==

```
int a = 2 ;

if ( a = 1 ) {    /* semantic (logic) error! */
   printf (“a is one\n”) ;
} else if ( a == 2 ) {
   printf (“a is two\n”) ;
} else {
   printf (“a is %d\n”, a) ;
}
```

Gotcha (con't)

- 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.

19

Logical Operators

- So far we have seen only **simple conditions**.

  if (count > 10) . . .

- Sometimes we need to test multiple conditions in order to make a decision.
- **Logical operators** are used for combining simple conditions to make **complex conditions**.

&&	is AND	if (x > 5 && y < 6)
||	is OR	if (z == 0 || x > 10)
!	is NOT	if (! (bob > 42))

20

Example Use of &&

```
if ( age < 1  &&  gender == 'f') {
    printf ("You have a baby girl!\n") ;
}
```

21

Truth Table for &&

Expression$_1$	Expression$_2$	Expression$_1$ && Expression$_2$
0	0	0
0	nonzero	0
nonzero	0	0
nonzero	nonzero	1

Exp$_1$ && Exp$_2$ && … && Exp$_n$ will evaluate to 1 (true) only if ALL **subconditions** are true.

22

Example Use of ||

```
if (grade == ‘D’  ||  grade == ‘F’) {
     printf (“See you next semester!\n”) ;
}
```

23

Truth Table for ||

Expression$_1$	Expression$_2$	Expression$_1$ || Expression$_2$
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

Example Use of !

```
if ( ! (x == 2) ) {   /* same as  (x != 2)  */
     printf("x is not equal to 2.\n") ;
} else {
     printf("x is equal to 2.\n");
}
```

25

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
||	left to right
=	right to left

27

Some Practice Expressions

int a = 1, b = 0, c = 7;

Expression	Numeric Value	True/False
a		
b		
c		
a + b		
a && b		
a || b		
!c		
!!c		
a && !b		
a < b && b < c		
a > b && b < c		
a >= b || b > c		

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)