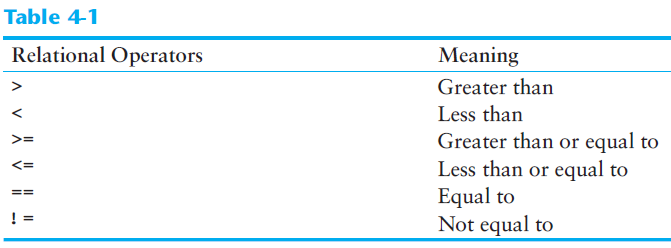
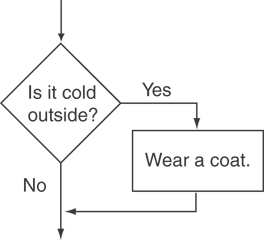
**CH4 Relational Operators**





To evaluate: if (*expression*) *statement*;

* If the *expression* is true, then *statement* is executed.
* If the *expression* is false, then *statement* is skipped.

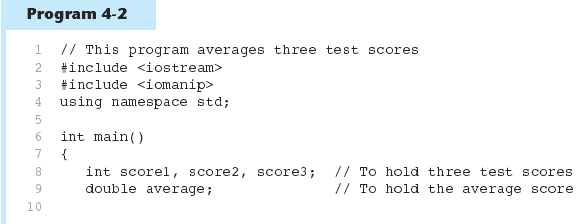
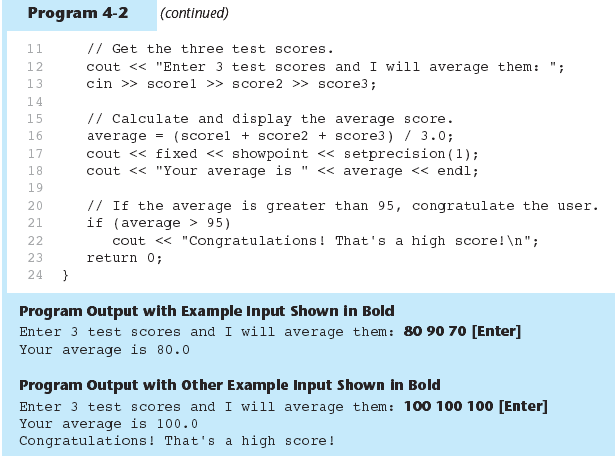
**if** Statement Notes:

* Do not place ; after (*expression*)
* Place *statement*; on a separate line after (*expression*), indented:

if (score > 90)

grade = 'A';

* Be careful testing floats and doubles for equality
* 0 is false; any other value is true

* To execute more than one statement as part of an if statement, enclose them in { }:

if (score > 90)

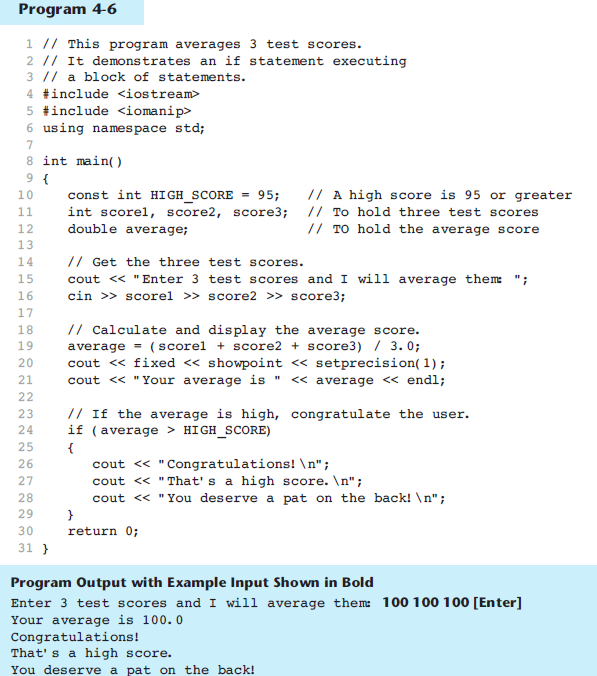
{

grade = 'A';

cout << "Good Job!\n";

}

* { } creates a block of code

 C:\Documents and Settings\Jianhua Jiang\Desktop\1.png

To evaluate:

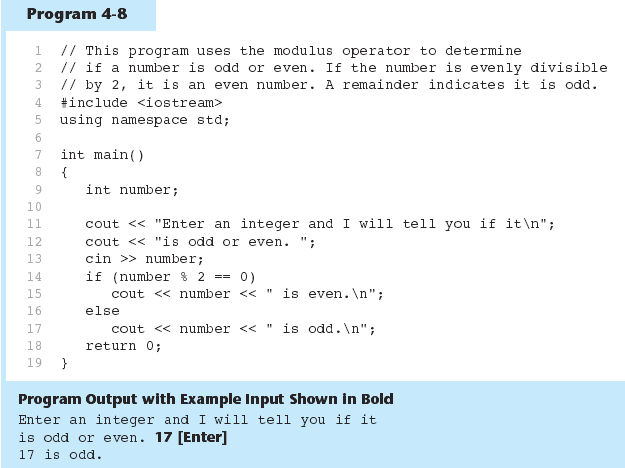
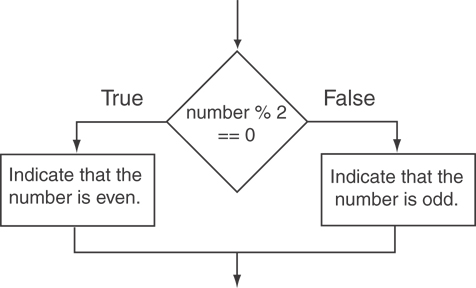
if (*expression*)

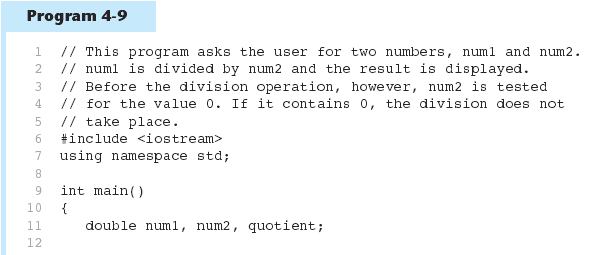
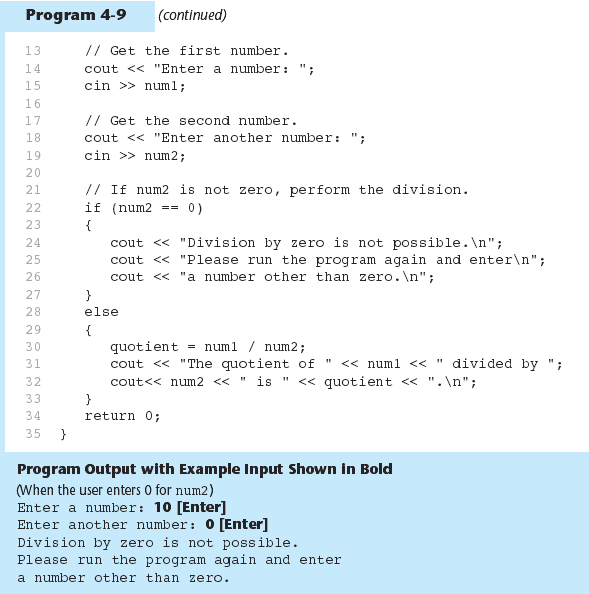
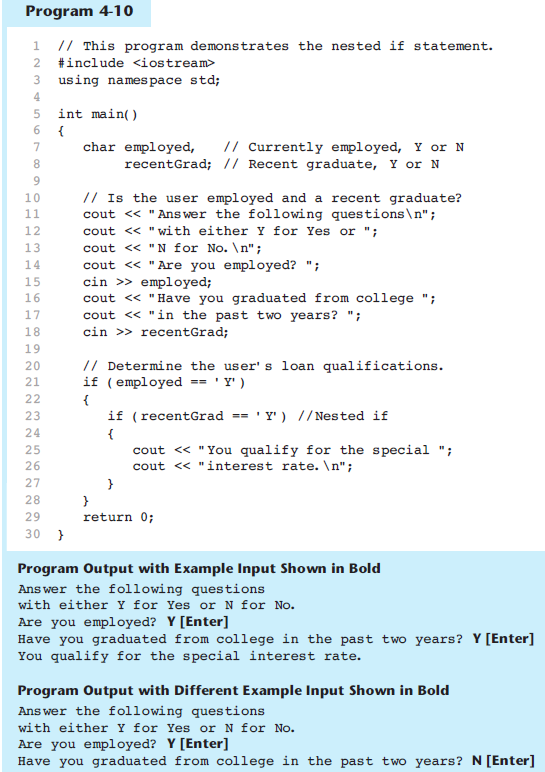
*statement1*;

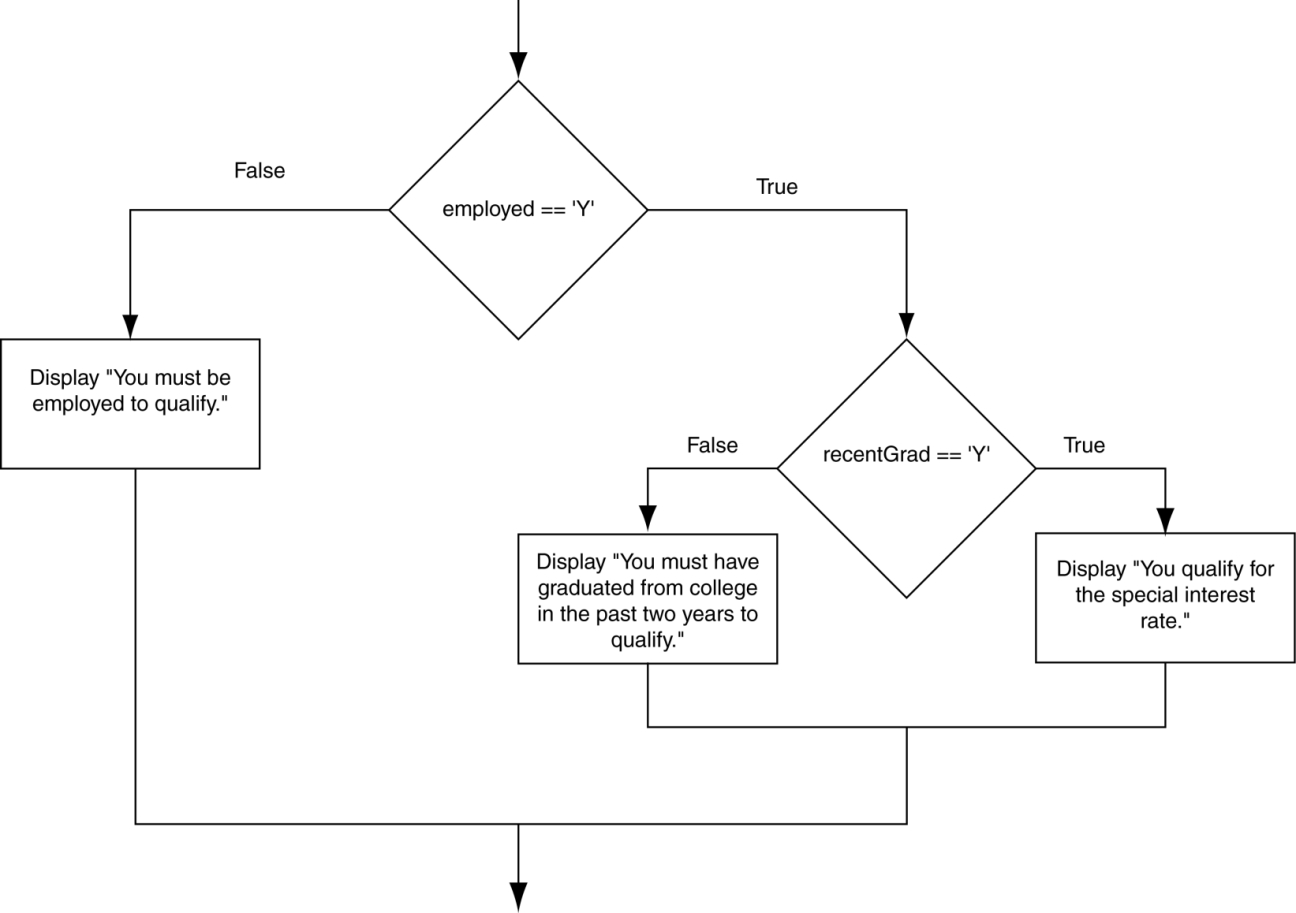
else

*statement2*;

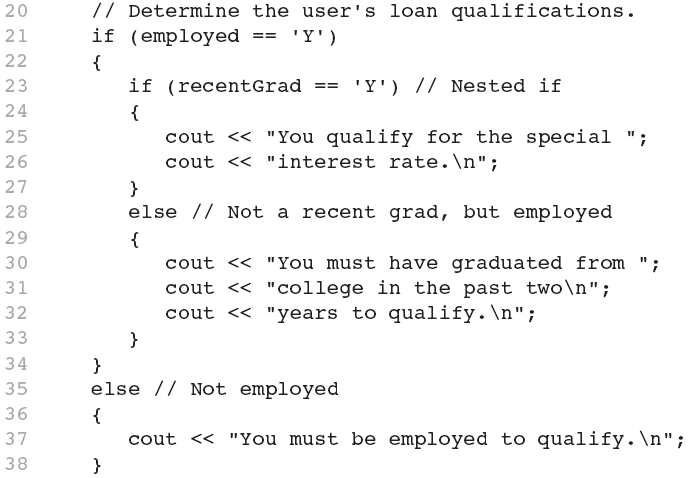
* If the *expression* is true, then *statement1* is executed and *statement2* is skipped.
* If the *expression* is *false*, then *statement1* is skipped and *statement2* is executed.

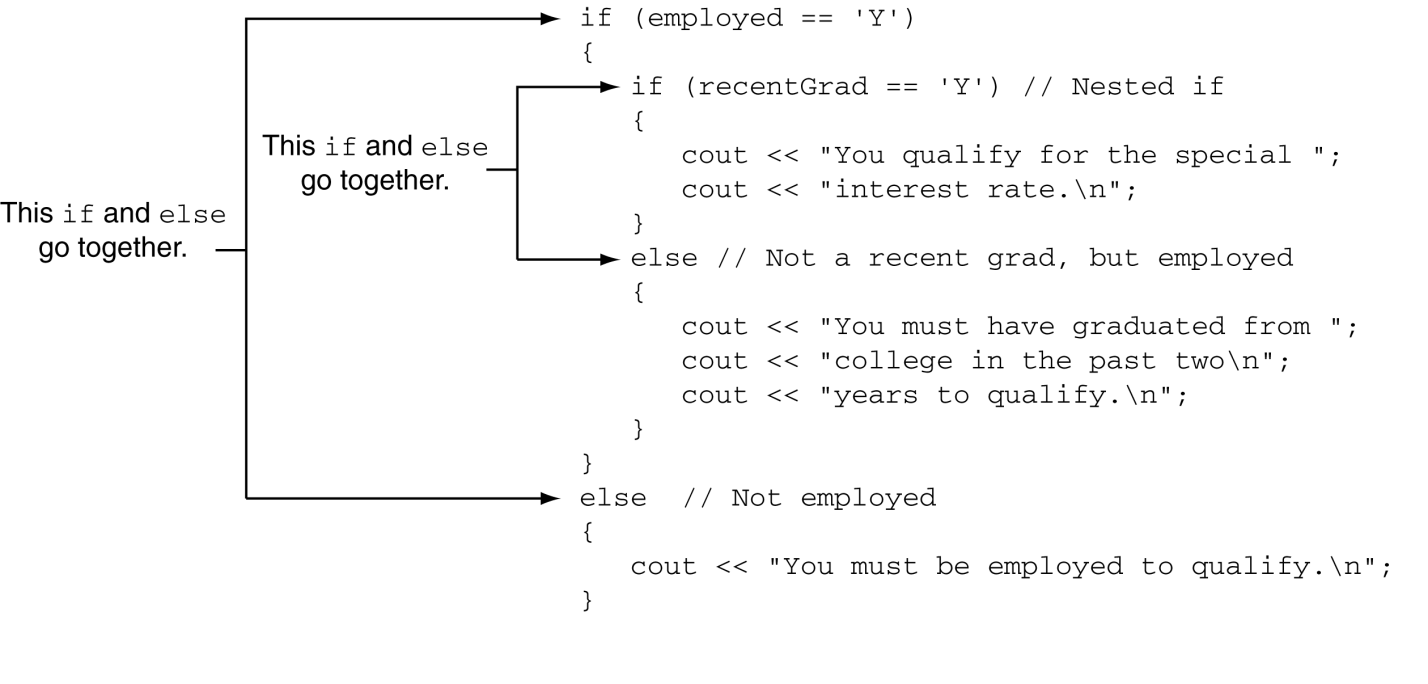
 

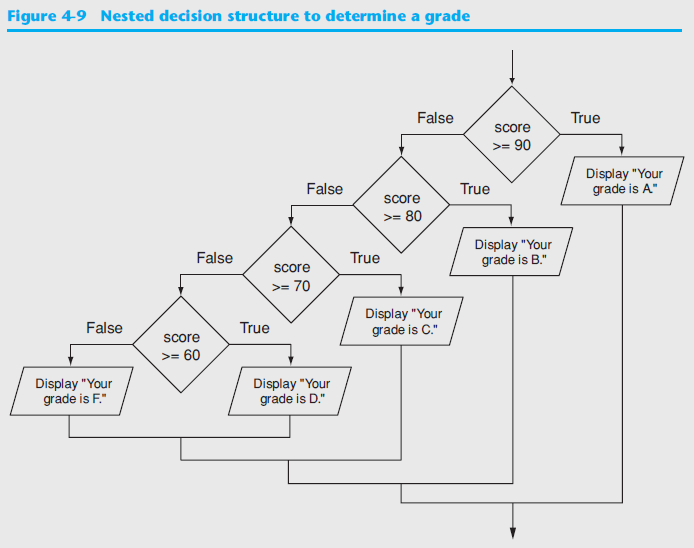
  

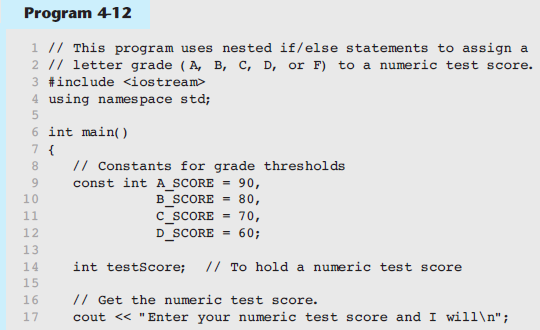
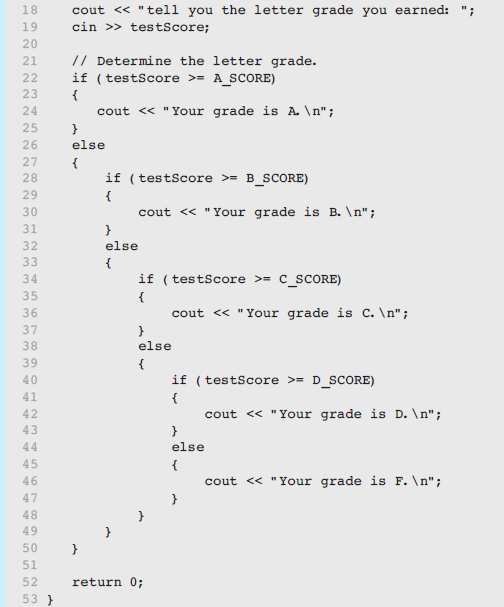
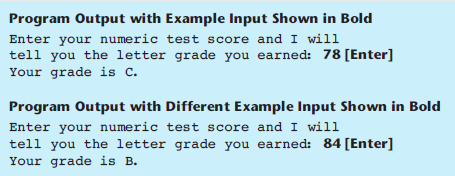


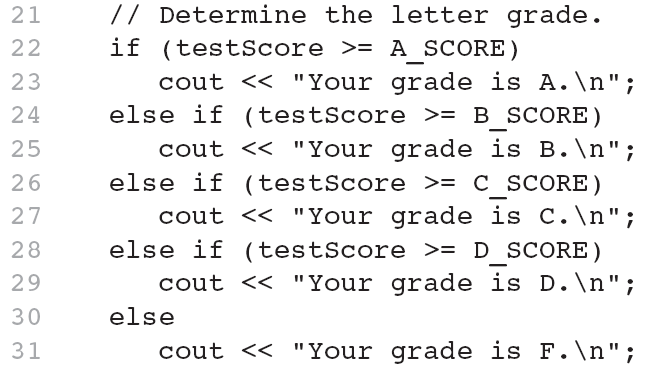
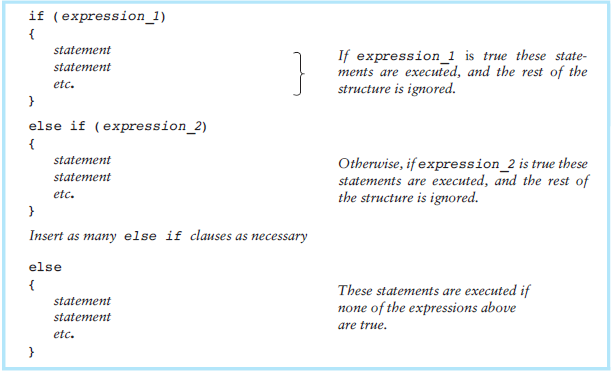
* Another example, from Program 4-1



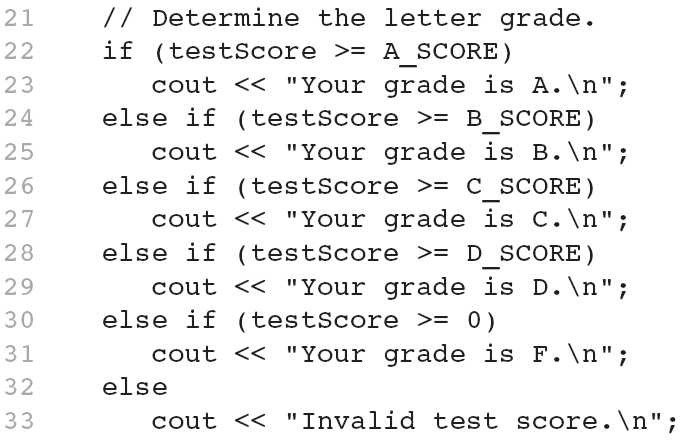


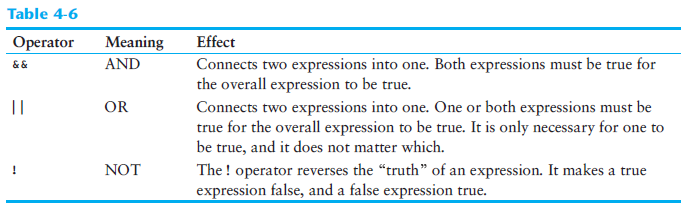


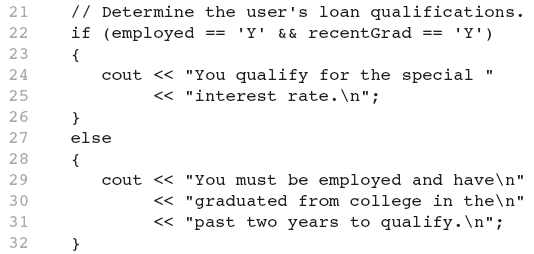


* The trailing else clause is optional, but it is best used to catch errors.

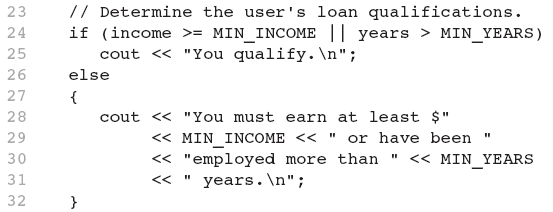




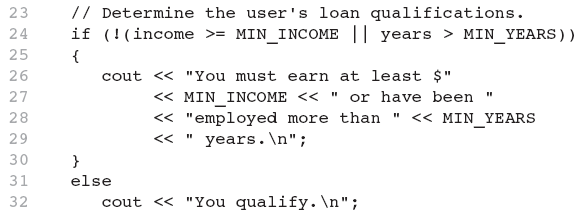
The logical **&&** operator in Program 4-15



The logical **||** Operator in Program 4-16



The logical **!** Operator in Program 4-17



Logical Operator-Notes

* ! has highest precedence, followed by &&, then ||
* If the value of an expression can be determined by evaluating just the sub-expression on left side of a logical operator, then the sub-expression on the right side will not be evaluated (*short circuit evaluation*)

Checking Numeric Ranges with Logical Operators

* Used to test to see if a value falls **inside** a range:

if (grade >= 0 && grade <= 100)

cout << "Valid grade";

* Can also test to see if value falls **outside** of range:

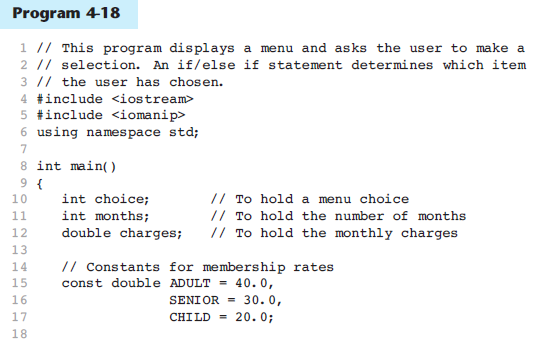
if (grade <= 0 || grade >= 100)

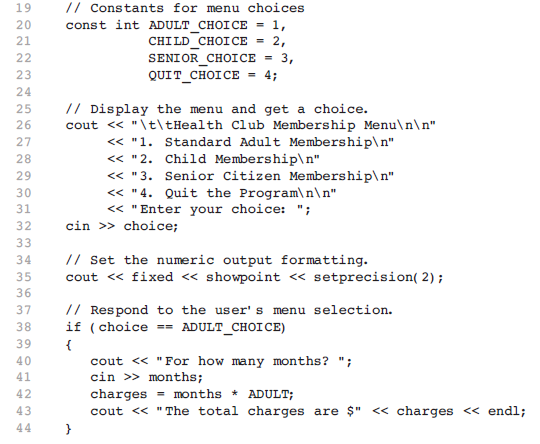
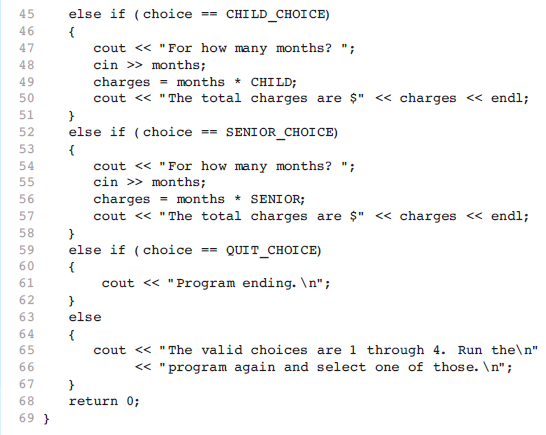
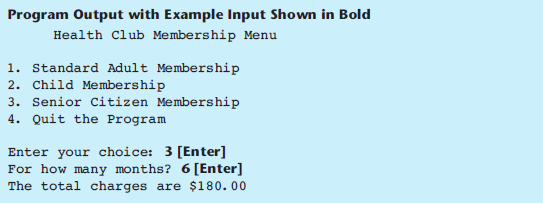
cout << "Invalid grade";

* Cannot use mathematical notation:

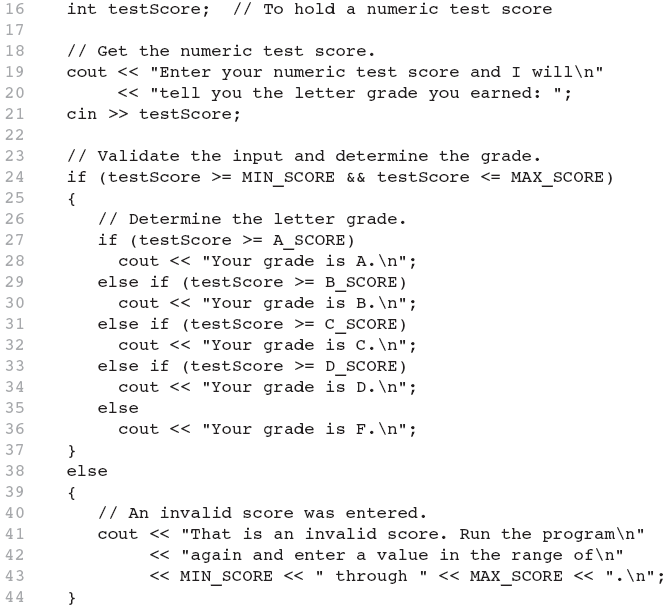
if (0 <= grade <= 100) //doesn’t work!

* Menu-driven program: program execution controlled by user selecting from a list of actions
* Menu: list of choices on the screen
* Menus can be implemented using if/else if statements
* Display list of numbered or lettered choices for actions
* Prompt user to make selection
* Test user selection in *expression*
  + if a match, then execute code for action
  + if not, then go on to next *expression*



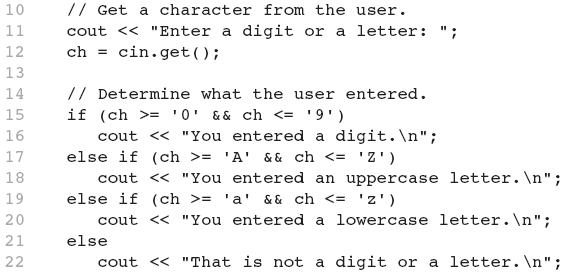
Input Validation in Program 4-19:

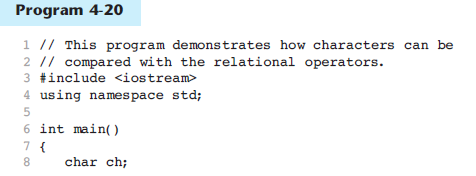
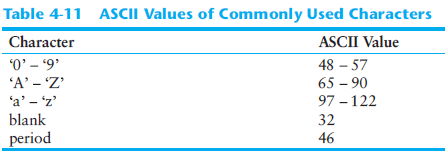


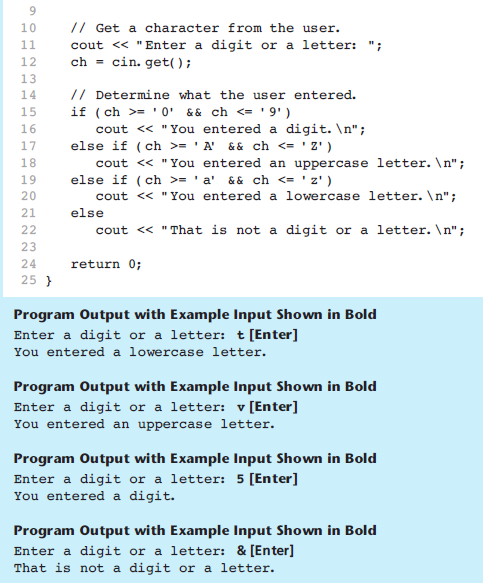
**Comparing Characters:**

* Characters are compared using their ASCII values
* 'A' < 'B'
  + The ASCII value of 'A' (65) is less than the ASCII value of 'B'(66)
* '1' < '2'
  + The ASCII value of '1' (49) is less than the ASCI value of '2' (50)
* Lowercase letters have higher ASCII codes than uppercase letters, so 'a' > 'Z'

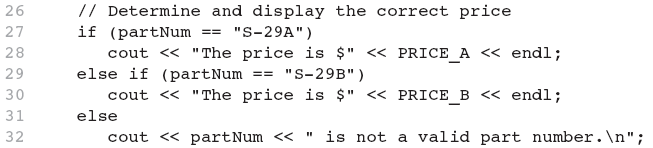
**Relational Operators Compare Characters in Program 4-20:**





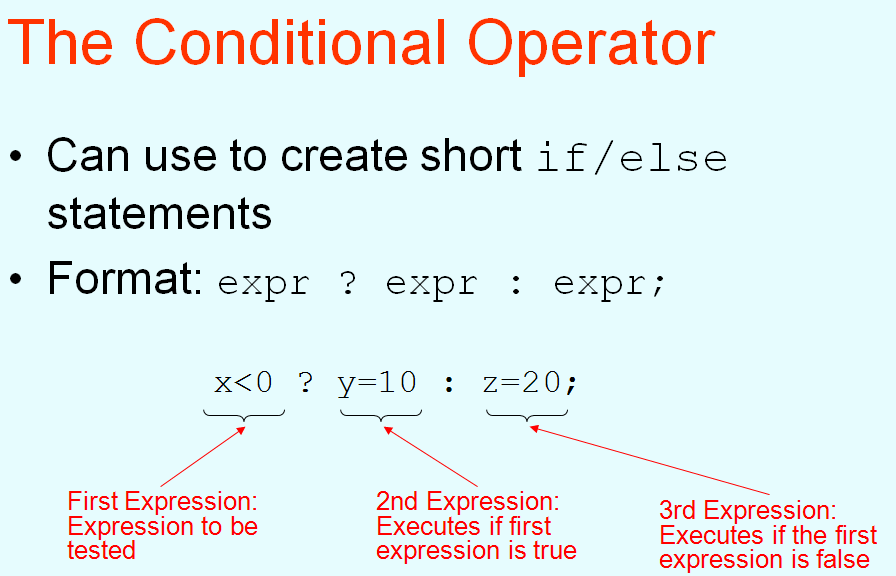


Relational Operators Compare Strings in Program 4-21:

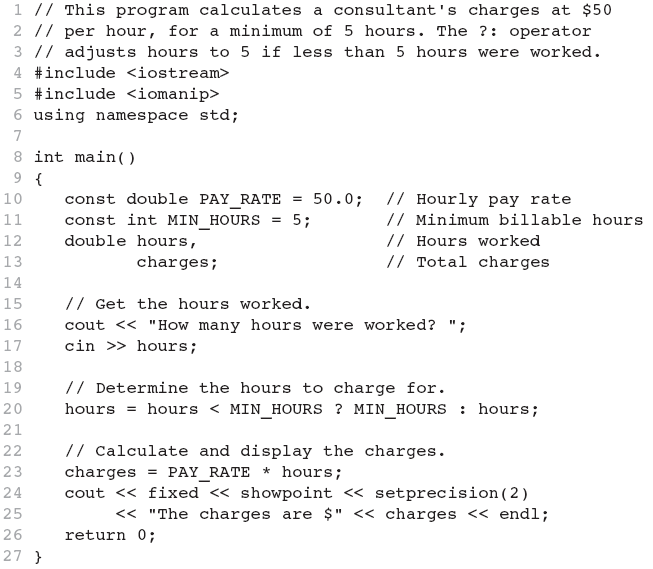


The Conditional Operator:

* The value of a conditional expression is
  + The value of the second expression if the first expression is true
  + The value of the third expression if the first expression is false
* Parentheses () may be needed in an expression due to precedence of conditional operator

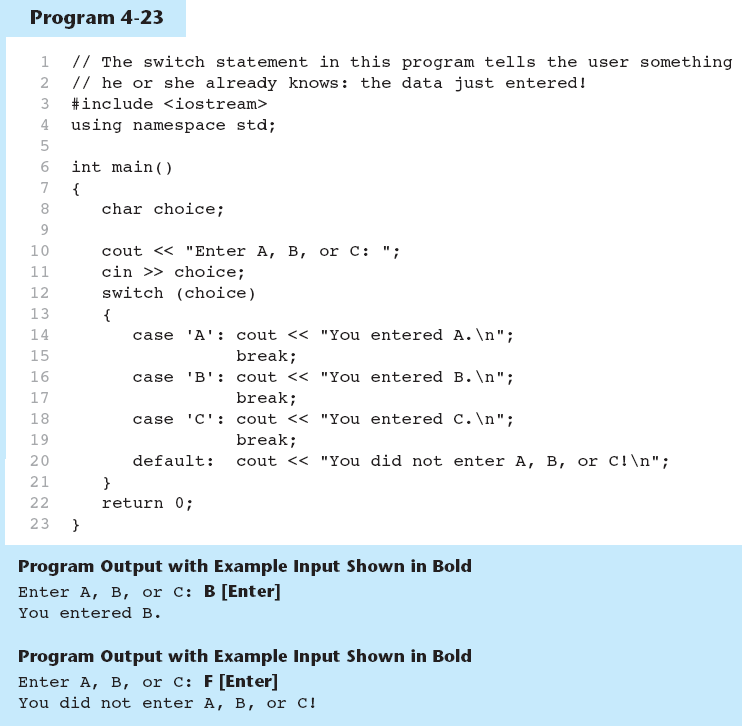


The Conditional Operator in Program 4-22:



**The switch Statement:**

* Used to select among statements from several alternatives
* In some cases, can be used instead of if/else if statements



**switch** Statement Requirements:

*expression* must be an integer variable or an expression that evaluates to an integer value

1. *exp****1*** through *exp****n*** must be constant integer expressions or literals, and must be unique in the switch statement
2. default is optional but recommended

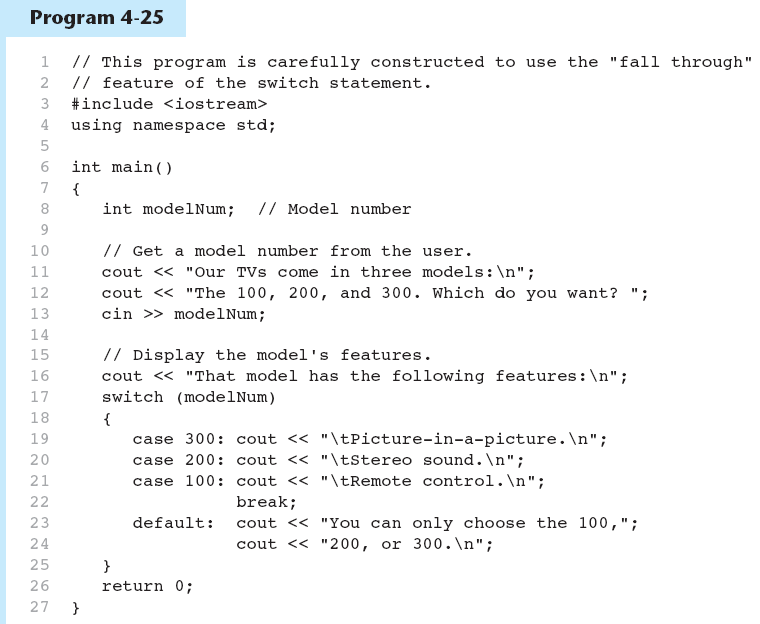
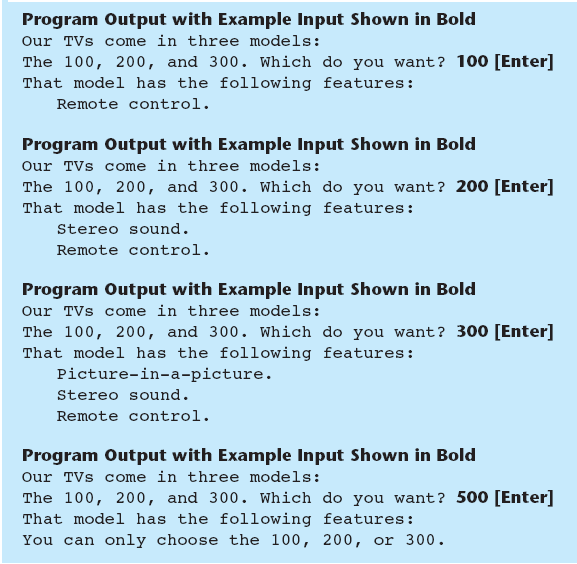
**switch** Statement-How it Works

1) *expression* is evaluated

1. The value of *expression* is compared against *exp****1*** through *exp****n***.
2. If *expression* matches value *exp****i***, the program branches to the statement following *exp****i*** and continues to the end of the switch
3. If no matching value is found, the program branches to the statement after default:

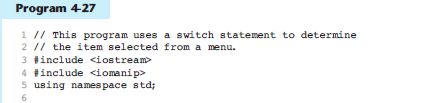
**break** Statement

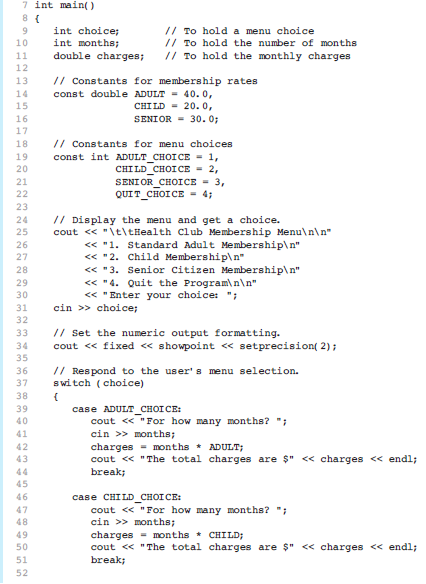
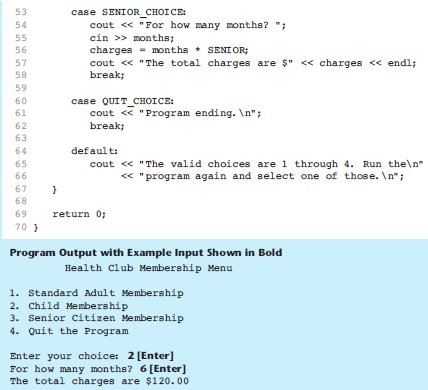
* Used to exit a switch statement
* If it is left out, the program "falls through" the remaining statements in the switch statement

**break** and **default** statements in Program 4-25:  

Using **switch** in Menu Systems:

* switch statement is a natural choice for menu-driven program:
  + display the menu
  + then, get the user's menu selection
  + use user input as expression in switch statement
  + use menu choices as *expr* in case statements

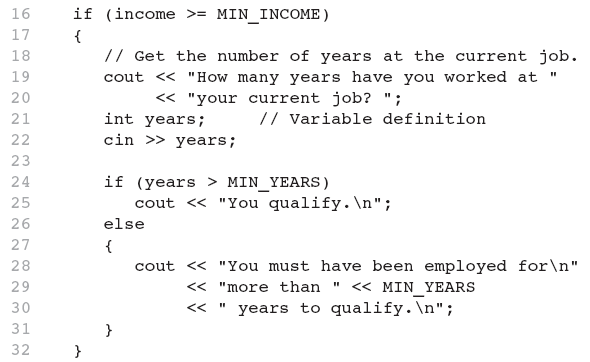


**More About Blocks and Scope:**

* Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
* Usually defined at beginning of function
* May be defined close to first use

Inner Block Variable Definition in Program 4-29:



**Variables with the Same Name**

* Variables defined inside { } have local or block scope
* When inside a block within another block, can define variables with the same name as in the outer block.
  + When in inner block, outer definition is not available
  + Not a good idea

Two Variables with the Same Name in Program 4-30:

