3.4 Equality and relational operators

Equality operators

An ***equality operator*** checks whether two operands' values are the same (==) or different (!=). Note that equality is ==, not just =.

An expression involving an equality operator evaluates to a Boolean value. A ***Boolean*** is a type that has just two values: true or false.

Table 3.4.1: Equality operators.

Equality operators	Description	Example (assume x is 3)
==	a **==** b means a is equal to b	x == 3 is true x == 4 is false
!=	a **!=** b means a is not equal to b	x != 3 is false x != 4 is true

Feedback?

PARTICIPATION ACTIVITY 3.4.1: Evaluating expressions that have equality operators. ✓

Indicate whether the expression evaluates to true or false.
x is 5, y is 7.

1) x == 5

[✓ True] False

Correct

x is 5, so 5 == 5 is true. ✓

2) x == y

True [✓ False]

Correct

x is 5 and y is 7, so 5 == 7 is false. ✓

3) y != 7

Correct ✓

True [✓ False]

y is 7, so 7 != 7 is false: "7 is not equal to 7" is false, because 7 is indeed equal to 7.

4) y != 99

[✓ True] False

✓

Correct

y is 7, so 7 != 99 is true: "7 is not equal to 99" is true, because 7 is indeed not equal to 99.

5) x != y

[✓ True] False

✓

Correct

x is 5 and y is 7, so 5 != 7 is true.

6) Is x = = y a valid expression?

Yes [✓ No]

✓

Correct

For operators that use multiple characters, spaces are not allowed between the characters.

Feedback?

PARTICIPATION ACTIVITY 3.4.2: Creating expressions with equality operators. ✓

Type the operator to complete the desired expression.

1) numDogs is 0

```
numDogs  ==   0
```

Check Show answer

✓

Correct

`==`

Saying numDogs is 0 is the same as saying numDogs equals 0, so the equality operator == is used. Note: Using = instead of ==, as in numDogs = 0, is a common error. The = operator is for assignment (thus assigning numDogs with 0), not for checking for equality.

2) numDogs and numCats are the same

```
numDogs  ==   numCats
```

Check Show answer

✓

Correct

`==`

Many English phrases can mean "equal", such as "same", "identical", "don't differ", etc.

3) numDogs and numCats differ

```
numDogs  !=   numCats
```

✓

Correct

Check Show answer

!=

Differ means not-equal, or !=. <> is *not* a valid operator.

4) numDogs is either less than or greater than numCats

`numDogs` != `numCats`

Check Show answer

Correct

!=

If numDogs is either less than or greater than numCats, then numDogs is not-equal to numCats. <> is not a valid operator, but that's no problem because != has the same meaning.

5) userChar is the character 'x'.

`userChar` == `'x'`

Check Show answer

Correct

==

Chars are commonly compared for equality. Note the single quotes around the x; omitting those is a common error.

Feedback?

Relational operators

A ***relational operator*** checks how one operand's value relates to another, like being greater than.

Some operators like >= involve two characters. A programmer cannot arbitrarily combine the >, =, and < symbols; only the shown two-character sequences represent valid operators.

Table 3.4.2: Relational operators.

Relational operators	Description	Example (assume x is 3)
<	a **<** b means a is less than b	x < 4 is true x < 3 is false
>	a **>** b means a is greater than b	x > 2 is true x > 3 is false
<=	a **<=** b means a is less than or equal to b	x <= 4 is true x <= 3 is true x <= 2 is false

>=	a **>=** b means a is greater than or equal to b	x >= 2 is true x >= 3 is true x >= 4 is false

Feedback?

PARTICIPATION ACTIVITY 3.4.3: Evaluating equations having relational operators. ✓

Indicate whether the expression evaluates to true or false.
x is 5, y is 7.

1) x <= 7

[✓ True] False

Correct

5 <= 7 is true, because 5 is less than 7. ✓

2) y >= 7

[✓ True] False

Correct

y is 7, so 7 >= 7 is true. The operator means greater than or equal, and in this case the two operands are equal. ✓

3) Is x <> y a valid expression?

Yes [✓ No]

Correct

Although intuitive, <> is not a built-in operator in the language. Fortunately, != has the same meaning. ✓

4) Is x =< y a valid expression?

Yes [✓ No]

Correct

<= is the operator for less than or equal, not =<. ✓

Feedback?

PARTICIPATION ACTIVITY 3.4.4: Creating expressions with relational operators. ✓

Type the operator to complete the desired expression.

1) numDogs is greater than 10

```
numDogs  >  10
```

Correct ✓

Check Show answer

>

Straightforward.

2) numCars is greater than or equal to 5

numCars >= 5

Check Show answer

Correct

>=

>= means greater than or equal to. => is *not* a valid operator.

3) numCars is 5 or greater

numCars >= 5

Check Show answer

Correct

>=

Each operator captures various English descriptions.

4) centsLost is a negative number

centsLost < 0

Check Show answer

Correct

<

< 0 is the common way to detect a negative number.

Feedback?

Example: Golf scores

In golf and miniature golf, each hole has a "par" indicating the normal number of strokes to get the ball in the hole. The following program outputs special names for numbers below par. The program uses one relational operator, and several equality operators, in a multi-branch if-else statement.

zyDE 3.4.1: Golf scores.

This program outputs special names for different numbers of strokes for a par 4 h
Run to see the name for the input strokes of 2. Try changing the input strokes to o
values, and press Run again to see the name.

If desired, try extending the program to print "Bogey" for 5, and "Double bogey" for
Remember that equals is ==, not =.

Try typing "= 1" instead of "== 1" and see what happens. Try typing an unsupporte
like =>, and see what happens.

Load default template...

```
   cin >> numStrokes;

   // Assumes "par 4"
   if (numStrokes <= 0) {
      cout << "Invalid entry." << endl;
   }
   else if (numStrokes == 1) {
      cout << "Hole in 1!!!" << endl;
   }
   else if (numStrokes == 2) {
      cout << "Eagle!" << endl;
   }
   else if (numStrokes == 3) {
      cout << "Birdie." << endl;
   }
   else if (numStrokes == 4) {
      cout << "Par." << endl;
   }
   else {
      cout << "Better luck next time." <<
   }
```

```
2
```

Run

```
Eagle!
```

Feedback?

CHALLENGE ACTIVITY 3.4.1: Enter the output for the branches with relational and equality operators.

Jump to level 1

Type the program's output.

```
#include <iostream>
using namespace std;

int main() {
   int numApples;

   numApples = 4;

   if (numApples != 6) {
      cout << "b" << endl;
   }
   else {
      cout << "f" << endl;
   }

   cout << "g" << endl;

   return 0;
}
```

```
b
g
```

1 2 3

Check Next **Done**. Click any level to practice more. Completion is preserv

✓ If numApples is not equal to 6, then the program outputs "b", else the program outputs "f". "

Yours
```
b
g
```

Expected
```
b
g
```

Feedback?

CHALLENGE ACTIVITY 3.4.2: Equality and relational expressions.

Jump to level 1

Complete the expression so that userPoints is assigned with 0 if userItems is less than 20 (second branch). Otherwise, userPoints is assigned with 10 (first branch).

```
#include <iostream>
using namespace std;

int main() {
   int userItems = 0;
   int userPoints = 0;

   cin >> userItems; // Program will be tested with values: 15, 20, 25, 30, 35.

   if (userItems >=20/* Your code goes here */) {
      userPoints = 10;
   }
   else {
      userPoints = 0;
   }

   cout << userPoints;

   return 0;
}
```

1 2 3

Check Next **Done**. Click any level to practice more. Completion is preserved.

✓ The >= operator can be used to evaluate if userItems is less than 20. The expression should evaluate to false so that the second branch executes and assigns userPoints with 0.

✓ 1: Compare output

Input
```
15
```
Your output
```
0
```

✓ 2: Compare output

Input
```
20
```
Your output
```
10
```

✓ 3: Compare output

Input
```
25
```
Your output
```
10
```

✓ 4: Compare output

Input
```
30
```
Your output
```
10
```

✓ 5: Compare output

Input
```
35
```
Your output
```
10
```

Feedback?

Comparing characters, strings, and floating-point types

The relational and equality operators work for integer, character, and floating-point built-in types.

Comparing characters compares their ASCII numerical encoding.

Floating-point types should not be compared using the equality operators, due to the imprecise representation of floating-point numbers, as discussed in a later section.

The operators can also be used for the string type. Strings are equal if they have the same number of characters and corresponding characters are identical. If string myStr = "Tuesday", then (myStr == "Tuesday") is true, while (myStr == "tuesday") is false because T differs from t.

PARTICIPATION ACTIVITY 3.4.5: Comparing various types.

Which comparison will compile AND consistently yield expected results? Variables have types denoted by their names.

1) myInt == 42
- OK
- Not OK

2) myChar == 'q'
- OK
- Not OK

3) myDouble == 3.25
- OK
- Not OK

4) myString == "Hello"
- OK
- Not OK

Feedback?

Common errors

Perhaps the most common error in C and C++ is to use = rather than == in an if-else expression, as in: if (numDogs = 9) { ... }. That code is not a syntax error. The statement assigns numDogs with 9, and then because that value is non-zero, the expression is considered true. C's designers allowed assignment in expressions to allow compact code, and use = for assignment rather than := or similar to save typing. Many people believe those language design decisions were mistakes, leading to many bugs. Some modern compilers provide a warning when = appears in an if-else expression.

Another common error is to use invalid character sequences like =>, !<, or <>, which are *not* valid operators.

PARTICIPATION ACTIVITY 3.4.6: Watch out for assignment in an if-else expression.

What is the final value of numItems?

1)

```
numItems = 3;
if (numItems == 3) {
   numItems = numItems + 1;
}
```

Check Show answer

2)

```
numItems = 3;
if (numItems = 10) {
   numItems = numItems + 1;
}
```

Check Show answer

Feedback?

CHALLENGE ACTIVITY

3.4.3: If-else statement: Fix errors.

Jump to level 1

Find and fix the error in the if-else statement.

```
#include <iostream>
using namespace std;

int main() {
   int userNum;

   cin >> userNum;    // Program will be tested with values: 1, 2, 3, 4.

   if (userNum != 2) {
      cout << "Num is not two" << endl;
   }
   else {
      cout << "Num is equal to two" << endl;
   }

   return 0;
}
```

1

Check Try again

Done. Click any level to practice more. Completion is preserved.

✓ The if statement has to check if userNum is not 2.

✓ 1: Compare output

Input
```
1
```
Your output
```
Num is not two
```

✓ 2: Compare output

Input
```
2
```
Your output
```
Num is equal to two
```

✓ 3: Compare output

Input
```
3
```
Your output
```
Num is not two
```

✓ 4: Compare output

Input
```
4
```
Your output
```
Num is not two
```

Feedback?

CHALLENGE ACTIVITY 3.4.4: If-else statement: Print senior citizen. ✓

Write an if-else statement that checks patronAge. If 55 or greater, print "Senior citizen", otherwise print "Not senior citizen" (without quotes). End with newline.

```
#include <iostream>
using namespace std;

int main() {
   int patronAge;

   cin >> patronAge;

   /* Your solution goes here  */
   if(patronAge>=55){
      cout << "Senior citizen"<<endl;
   }else{
      cout << "Not senior citizen"<< endl;
   }

   return 0;
}

```

Run ✓ All tests passed

✓ Testing with patronAge = 54

Your output
```
Not senior citizen
```

✓ Testing with patronAge = 55

Your output
```
Senior citizen
```

✓ Testing with patronAge = 56

Your output
```
Senior citizen
```

Feedback?