COMPARISON & CONDITIONALS.

Poggie

02.08.2024

// TODO.

Quick Recap

Logical Operators

Comparison

Conditionals

Lab 1

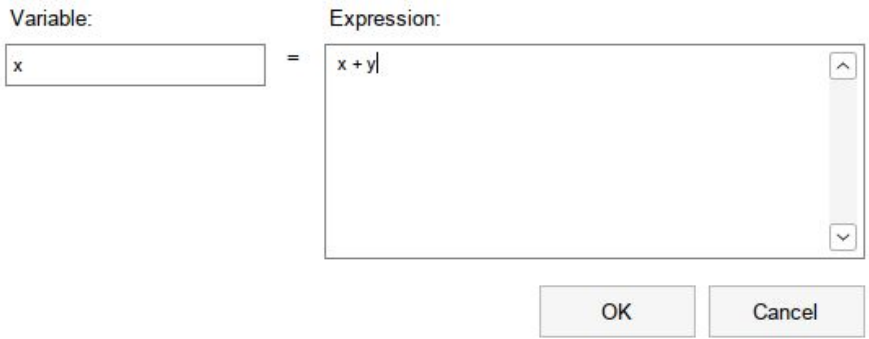
Quick Recap.

Variables

Assignments in C#

Arithmetic Operators in C#

```
int x, y = 7;
7 = x;          // not legal
3 = y + x;      // not legal
y = x + 3;
```



Quick recap cont…

Assignments in Flowgorithm

Arithmetic Operators in Flowgorithm

Lab 1

RELATIONAL OPERATORS.

Intro.

Used to check conditions and make comparisons.

The result is either true or false, that is the result is a boolean value.

Expressions containing relational operators are called “boolean expressions”.

We use a boolean variable to store the result of a “boolean expression”.

```
// Is chicken cheaper than beef?
// Is it raining outside?
// Do I have enough money for the train?

// "yes" or "no" // "true" or "false"
```

Relational operators.

Determines whether specific relationships exists between values.

```
// Is x greater than y?
x > y
// Is x less than y?
x < y
// Is x greater than or equal to y?
x >= y
// Is x less than or equal to y?
x <= y
```

Testing more than one relationship.

<= and >= test more than one relationship. If one relationship is true, the expression would be true.

>= checks if the operand on the left side is greater than or equal to the operand on the right side.

<= checks if the operand on the left side is less than or equal to the operand on the right.

```
4 >= 4;

2 <= 2;
```

“4 >= 2” demo.

Equality operator.

Determines whether the two operands are equal to one another.

Let b be a variable of type boolean, b evaluates to the same value as b

```
// Is x equal to y?
x == y
// Is x not equal to y?
x != y
```

“4 == 2” demo.

Relational Operator: Order of Operations.

1. Relational: <, >, <=, >=
2. Equality: ==

You can always use parenthesis () to raise priority

```
4 >= 2 == true;
```

Common Mistakes.

use a single = instead of a double ==

using ≠, ≥, =<, or =>

two sides of the relational operator need to be comparable (both sides of the operator need the same type)

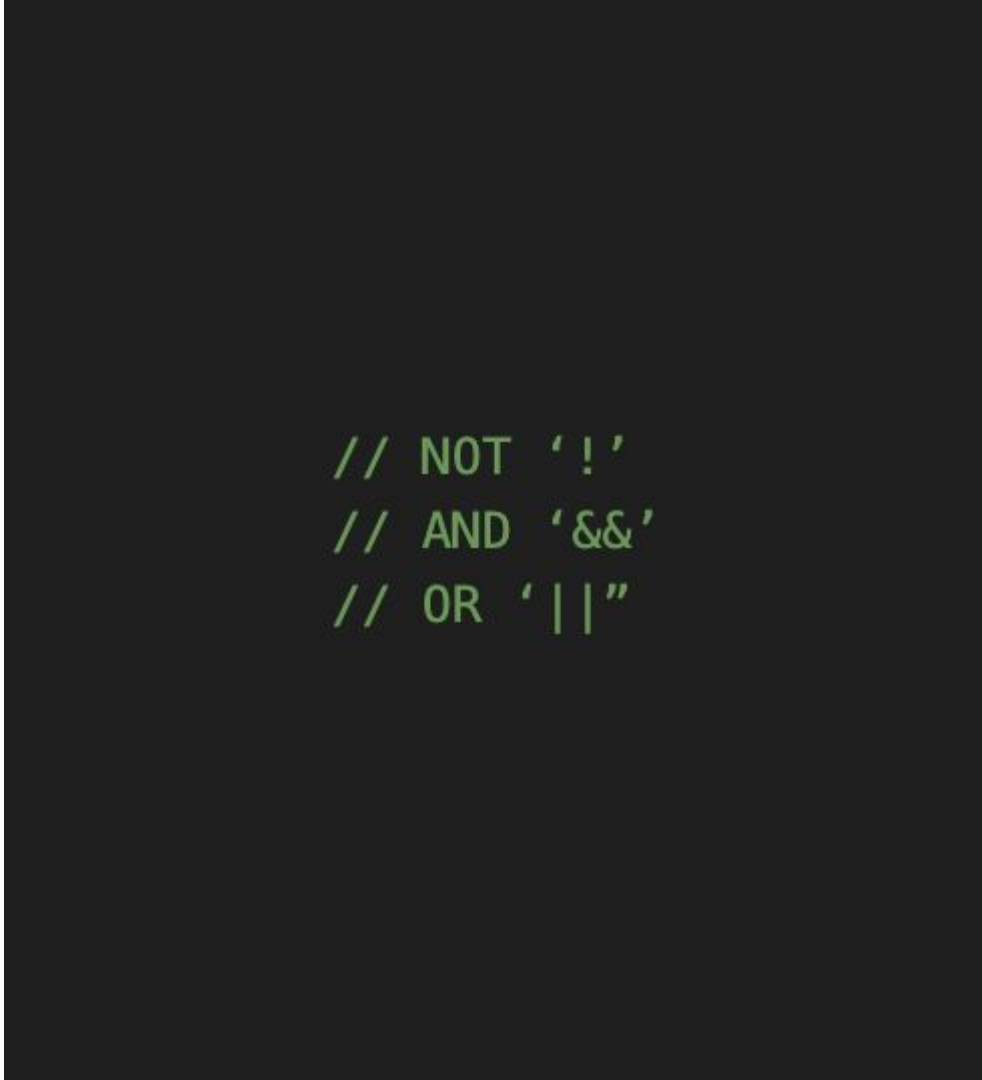
```
4 = 2;     // does not compile
4 == 2;    // is false

2 == 2.0; // is true.
2 == “2”; // does not compile.
```

LOGICAL OPERATORS.

Intro.

Logical operators take boolean expressions (i.e. expressions that evaluate to a boolean value) as inputs and produce a result of type boolean



NOT operator.

NOT is a unary operator: it takes 1 input

Let b be a variable of type boolean, !b evaluates to the opposite value of b

b	!b
true	false
false	true

“!(2 < 4)” demo.

AND operator.

AND is a binary operator: it take 2 inputs

Let a and b be two variables of type boolean, a && b evaluates to true if and only if both a and b are true.

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

“(2 < 4) && true” demo.

OR operator.

OR is a binary operator: it takes 2 inputs

Let a and b be two variables of type boolean, a || b evaluates to false if and only if both a and b are false.

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

“(2 == 1) || ! (1<2)” demo.

Logical Operator: Order of Operations.

From left to right

1. ‘!’
2. ‘&&’
3. ‘||’

You can always use parenthesis () to raise priority

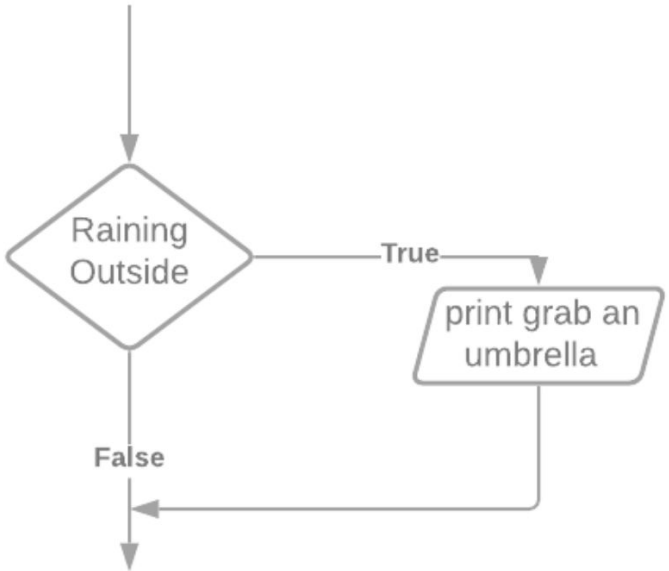
```
// a = false and b = true
b && !a || b;
```

Order of Operations.

1. Parenthesis
2. !
3. Typecasting
4. Arithmetic
   - *, /, %
   - +, -
5. Comparison
   - Relational: <, >, <=, >=
   - Equality: ==, !=
6. Boolean: &&, ||

```
false || 1 / (int) 2.0 < 3.5;
```

CONDITIONAL STATEMENTS.



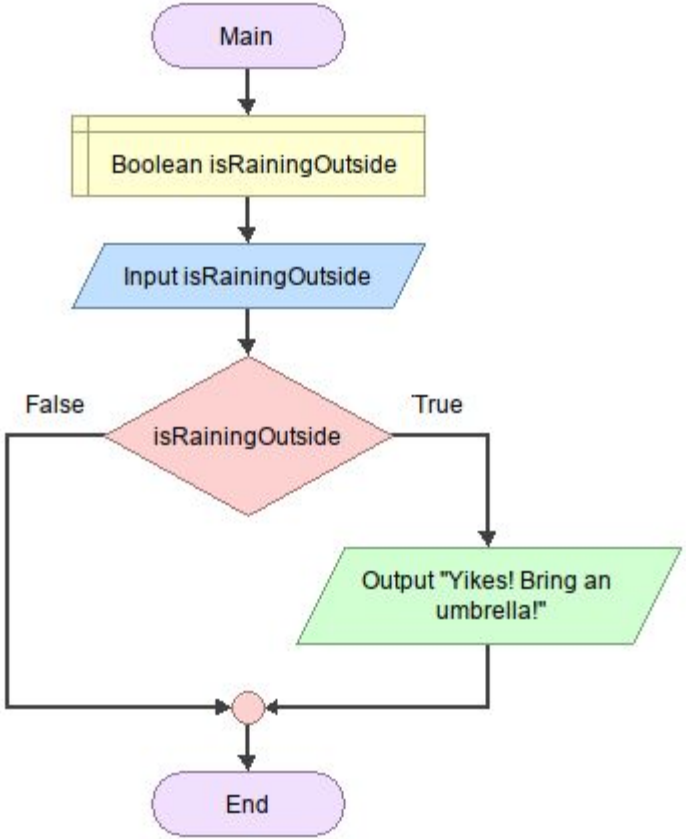
Why?

So far all the programs we wrote had a sequential structure. i.e. the statements were executed in the order they appear in the code.

A code that has a decision structure is a code that performs specific actions only if a condition exists.

We use conditional statements when we need to make a decision in our code.

These are often called Decision or Control Flow structures.

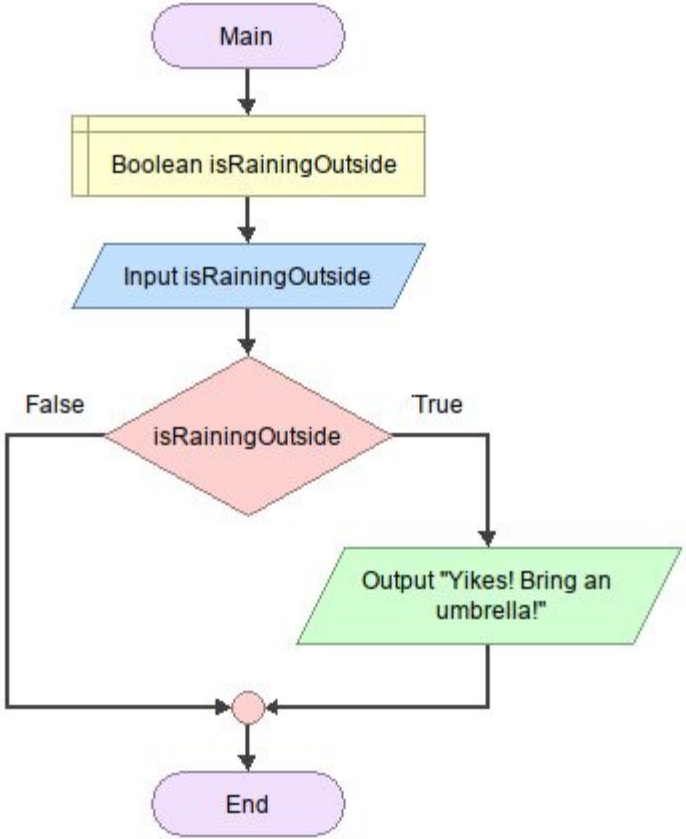


How can we use Booleans?

To write useful programs, we almost always need to check conditions.

- We might want to execute certain statements only if something is true.
- Conditional statement give us this ability

The simplest conditional statement is the if statement.

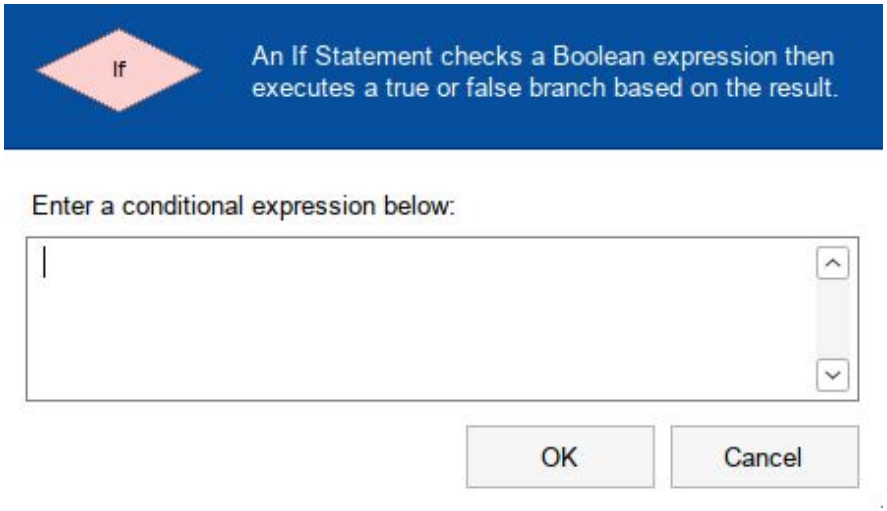


What's going on?

The diamond represents a true/false condition. If the condition is true, we follow one path, which leads to an action (printing grab an umbrella) being performed.

If the condition is false, we follow another path, which skips the action.

The simplest conditional statement is an if statement.

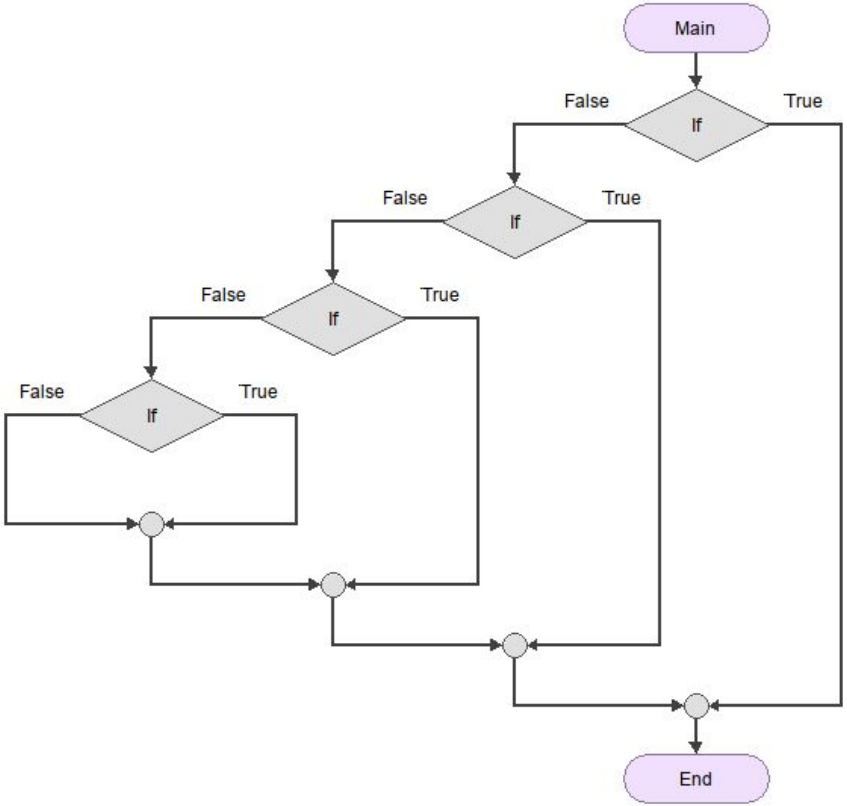


What is an ‘if’ statement?

The expression in parentheses is called the condition. It must be a boolean expression.

When an if statement is executed, the condition is tested. If the condition is true, the block statements are executed. Otherwise, block statements are skipped.

“num > 0” demo.

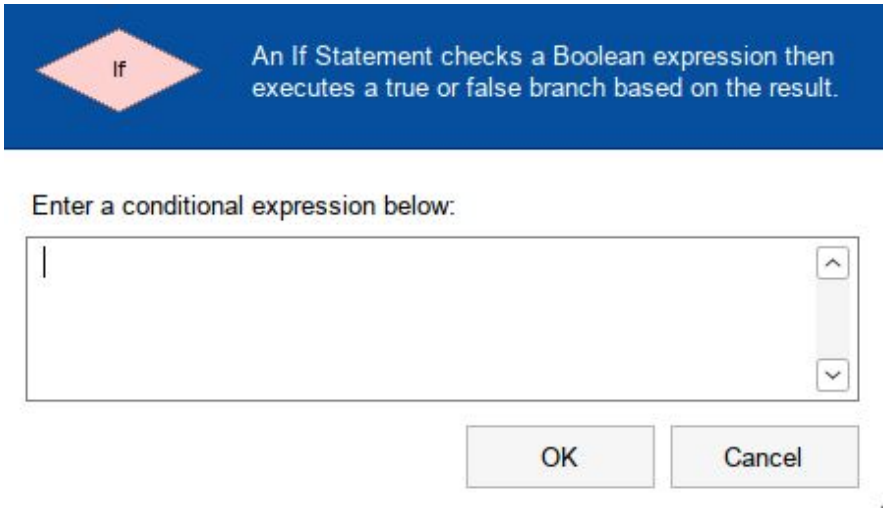


Nesting ‘if’ statements.

If you want to evaluate multiple conditional expressions, you can nest them inside each other

In this case, how we order our conditional statements matters. As soon as one block is executed, the remaining will be skipped

“num > 0 … num < 0” demo.



WARNING!

Nested or chained conditional statements are common, but can become very confusing!

In C# there is more powerful conditional statement functionality, we will cover that next class…

LAB TIME.