Building Blocks of Decision Making

When we made decisions throughout the day our brain goes through a process of comparisons.

You might compare the clothes your wear or the cell phone you own to what your friend has.

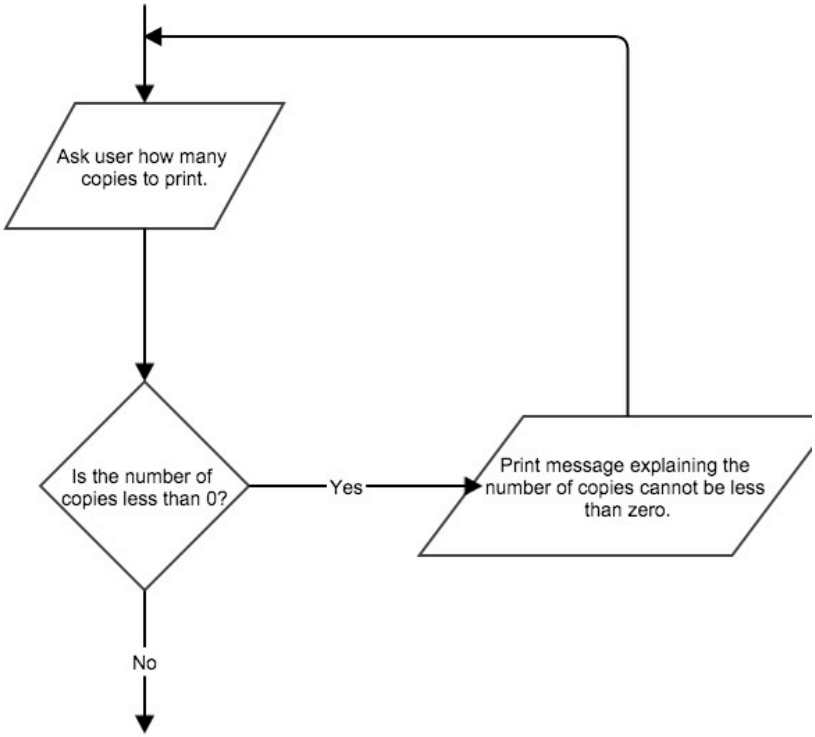
You might compare what different things you could buy if you had $20.

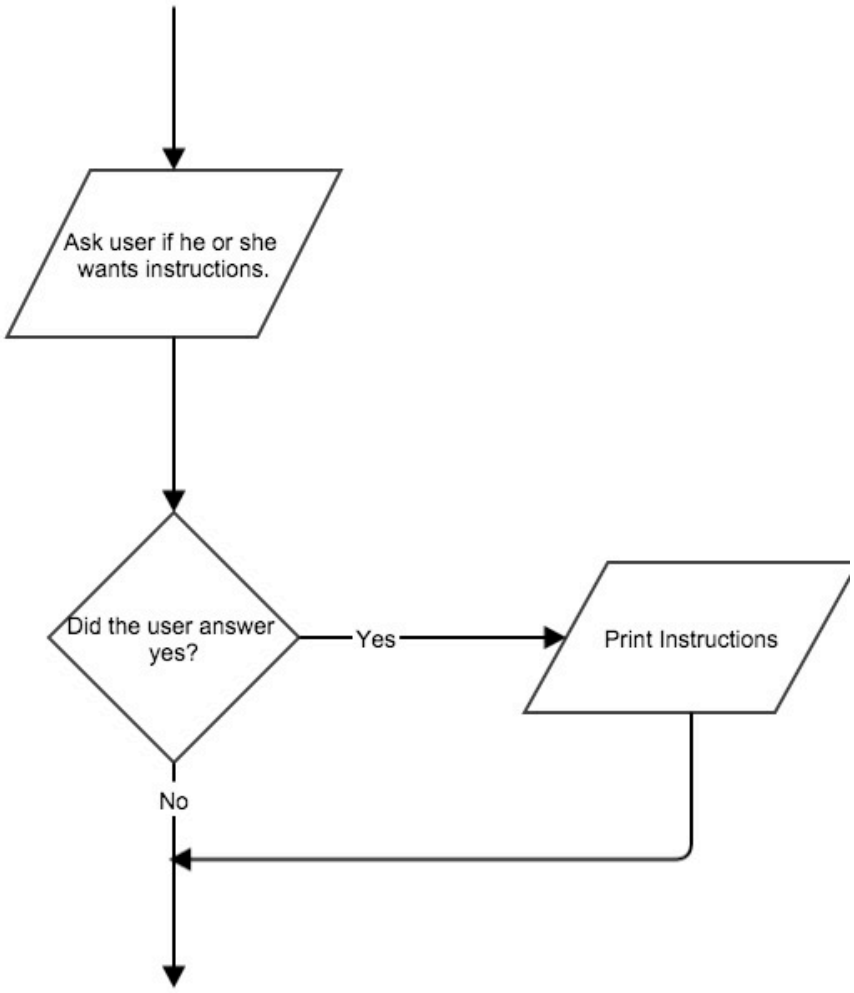
You might compare the different ways you can spend your time in the evenings (homework, activities, chores, etc.)

A program that is useful and user-friendly usually involves some type of decision making to help the user navigate through what they want to do.

Some programs can be written so that each next step happens directly after the previous but most need some type of decision-making and from there branches start giving the program lots of different directions.

It often helps to illustrate the flow of a program using a **flow chart**. It will map the decisions of a program to make a path where each decision leads.





Representing True and False

The computer has a very simple way of demonstrating true and false.

When the computer makes a comparison, the comparison results in a value of either 0 or 1.

If the result is 0, the comparison is proven false.

If the result is 1, the comparison is proven true.

Relational Operators

To make comparisons, C++ uses the following **relational operators**.

Operator	Meaning	Example
==	equal to	i == 1
>	greater than	i > 2
<	less than	i < 0
>=	greater than or equal to	i >= 6
<=	less than or equal to	i <= 10
!=	not equal to	i != 12

The result from one of these comparisons can either be printed with the `cout` statement or it can be stored in a Boolean variable.

Logical Operators

Sometimes it takes more than 2 comparisons to obtain the desired result.

For example, in order to see if an integer is between 1 and 10 you must do two comparisons.

It must be greater than 0 AND less than 11.

C++ provides 3 **logical operators**:

Operator	Meaning	Example
&&	and	(j == 1 && k == 2)
||	or	(j == 1 || k == 2)
!	not	result != (j == 1 && k == 2)

Truth Tables

One way to understand the result of these comparisons with logical operators is to use a truth table. A truth table shows all possible combinations using two or more statements.

There are entire branches of math that involve this type of logic and it should come up in other math classes.

The truth tables for our 3 logical operators are:

AND		
A	B	A && B
True (1)	True (1)	True (1)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	False (0)

OR		
A	B	A || B
True (1)	True (1)	True (1)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

A	!A
True (1)	False (0)
False (0)	True (1)

Combining More Than Two Comparisons

You can use logical operators to combine more than two comparisons.

Consider the following statement which decides whether its ok for a person to ride a roller coaster:

```
ok_to_ride = (height_in_inches > 45 && !back_trouble
&& !heart_trouble);
```

Back_ and heart_trouble hold either a 0 or 1 depending on whether the person has the problem.

In order to be able to ride the entire statement must be true and they must be tall enough AND not have either of the two problems.

Order of Logical Operations

Just like with arithmetic it's important to understand the order in which logical operators are applied.

The order is as follows:

! NOT

&& AND

|| OR

Consider the following statement:

```
dog_acceptable = (big || small && friendly);
```

At first glance it looks like this statement says the dog is acceptable if it's big or small as long as it's friendly.

When you consider the order of operations it says a big not friendly dog is acceptable because the && operation is done first.

To correct the statement we would do similar to arithmetic and add parenthesis:

```
dog_acceptable = ((big || small) && friendly);
```