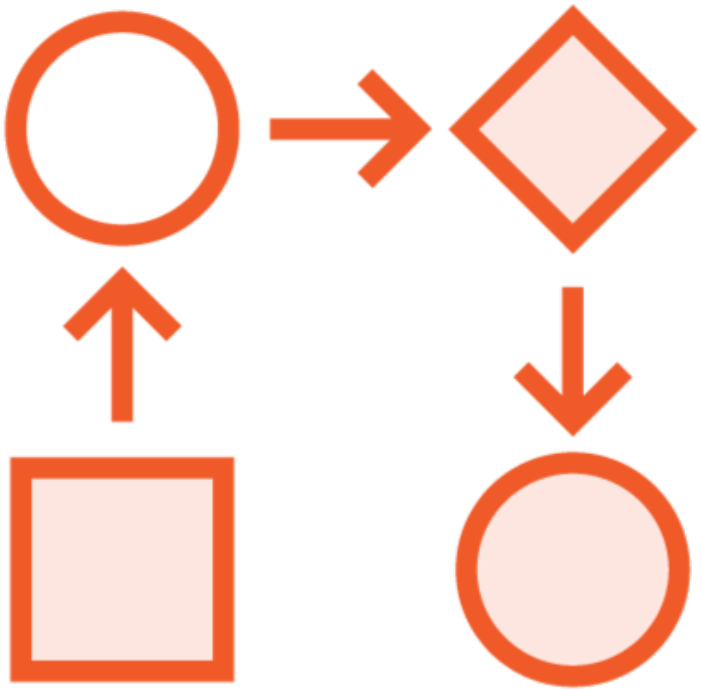
Flow of Control

Kate Gregory

www.gregcons.com/kateblog @gregcons

Flow of Control



Normally code is executed from top to bottom

A number of keywords change this

- if
- else
- while
- for

These keywords work with logical expressions

- `(x > 0)`
- `(y-2 < b)`

Operators to compare two operands:

- `> >= < <= == !=`
- Result is `true` or `false`

if

```
if (i<j)
{
    // statements
}
else
{
    // statements
}
```

Condition must always have ()

Statements that run if the condition is true have {}

- Optional if it's one line, but use them anyway

An else can only be used right after an if

while

```
while(keepgoing)
{
   // ...
   if (answer == 0)
   {
      keepgoing = false;
   }
}
```

- **Keeps going as long as the condition is true**
- **Statements to run should be surrounded by {}**
  - Optional for single line but use them anyway
- **Loop body must change something about the condition**
  - Otherwise infinite loop

for

```
for (int loop = 0;loop < 10;loop++)
{
        cout << loop << " ";
}
```

- **Traditional for loop has three parts**
  - Initializer
  - Continue condition
  - Incrementer
  - Separated by semi colons, not commas
- **Body of the loop doesn't have to change anything about the condition if the incrementer does**

Exercise

Write a “guess my number” game

- Hardcode the answer in your code
  - `int answer = 7;`
  - You can change the number, build, and run again
- Ask the user to enter a guess
- Let them know if they guessed too high, too low, or got it
- Keep going until they get it
- Don’t try error checking yet
  - When you test it, be nice

Other Keywords Exist

switch

Range based for

break

continue

do

goto

Summary

The keywords if, else, while, and for control the lines that execute in your program

Logical conditions are how an application makes decisions

- Comparing two things

These small building blocks can build a real application