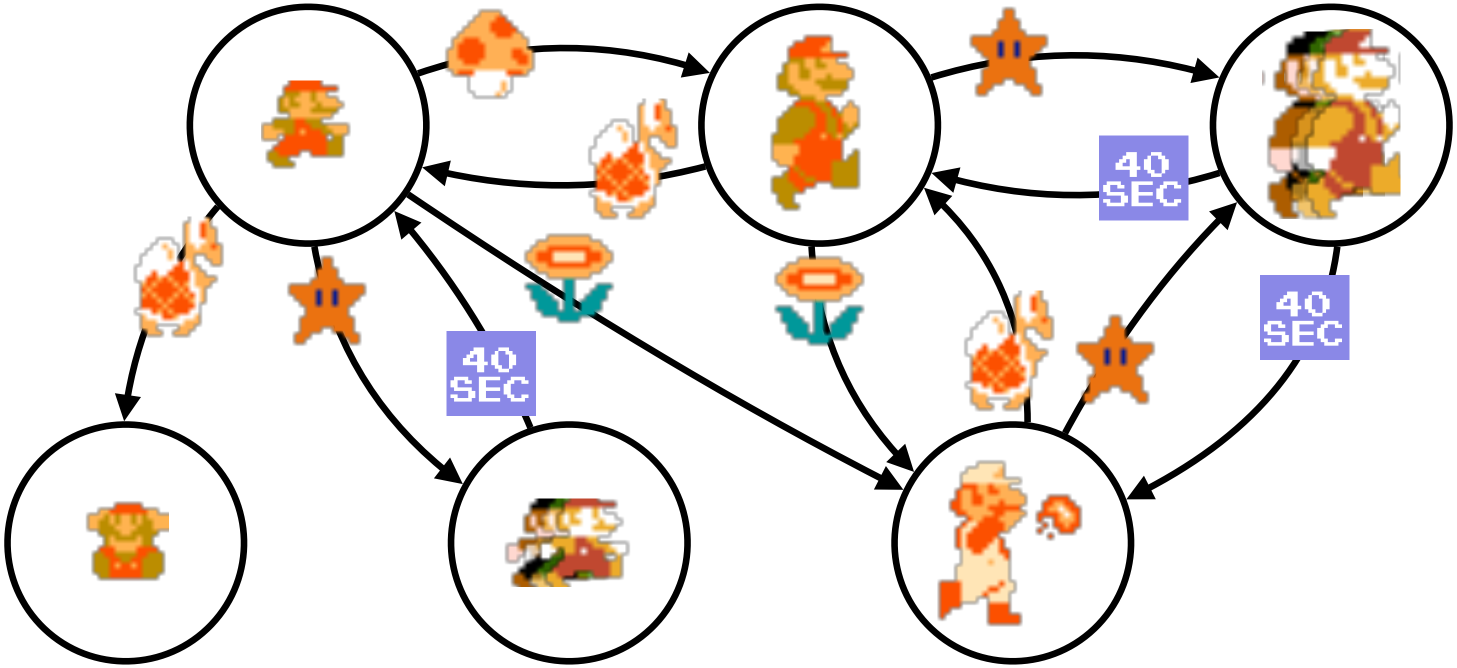
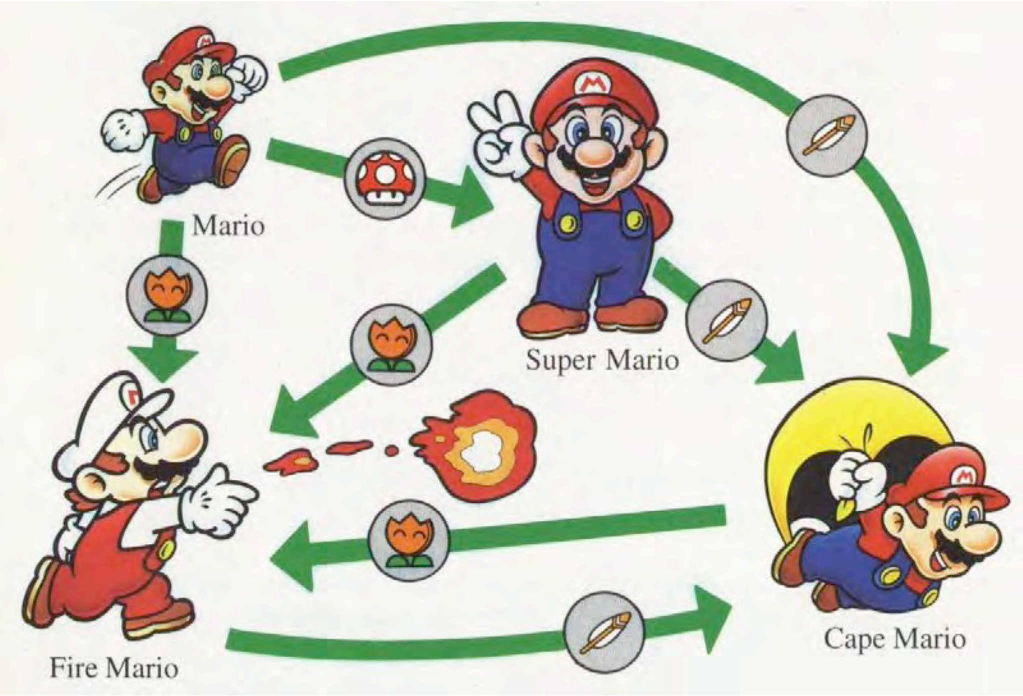


BRANCHING AND ITERATION

ROBOTICS 102
INTRO AI & PROOGRAMMING

FALL 2021
UNIVERSITY OF MICHIGAN

Finite State Machine



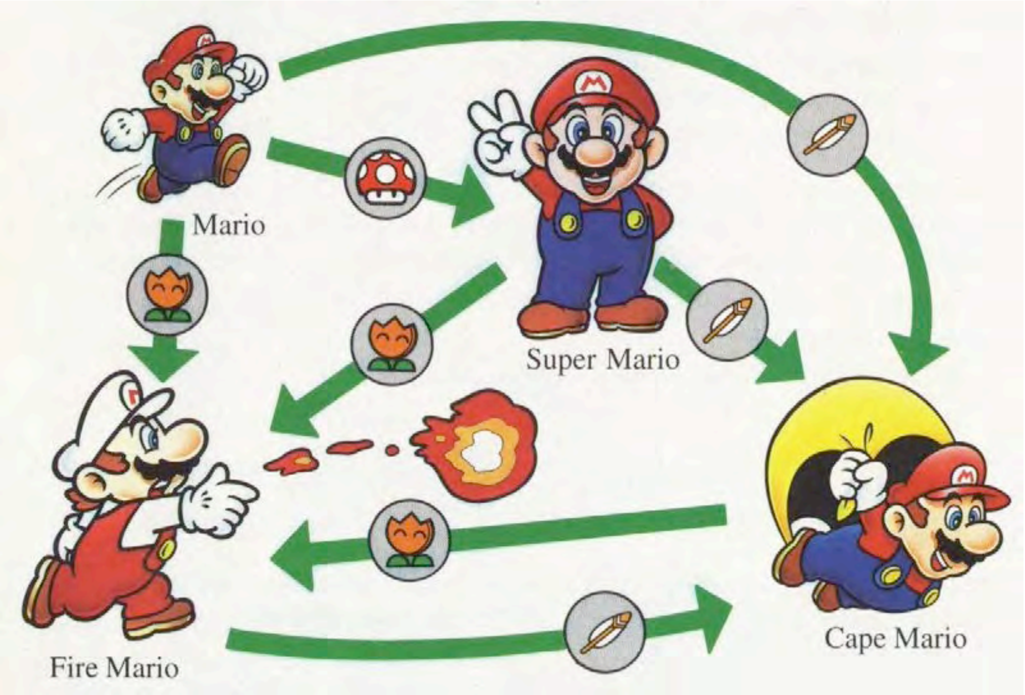
Inspired by: https://www.c-sharpcorner.com/article/understanding-state-design-pattern-by-implementing-finite-state/
Taken from: https://www.mariowiki.com/File:Smw_powerup_chart.jpg

What is a Finite State Machine?

A model of computation that describes the behavior of a system.

Expressed visually as a state diagram.

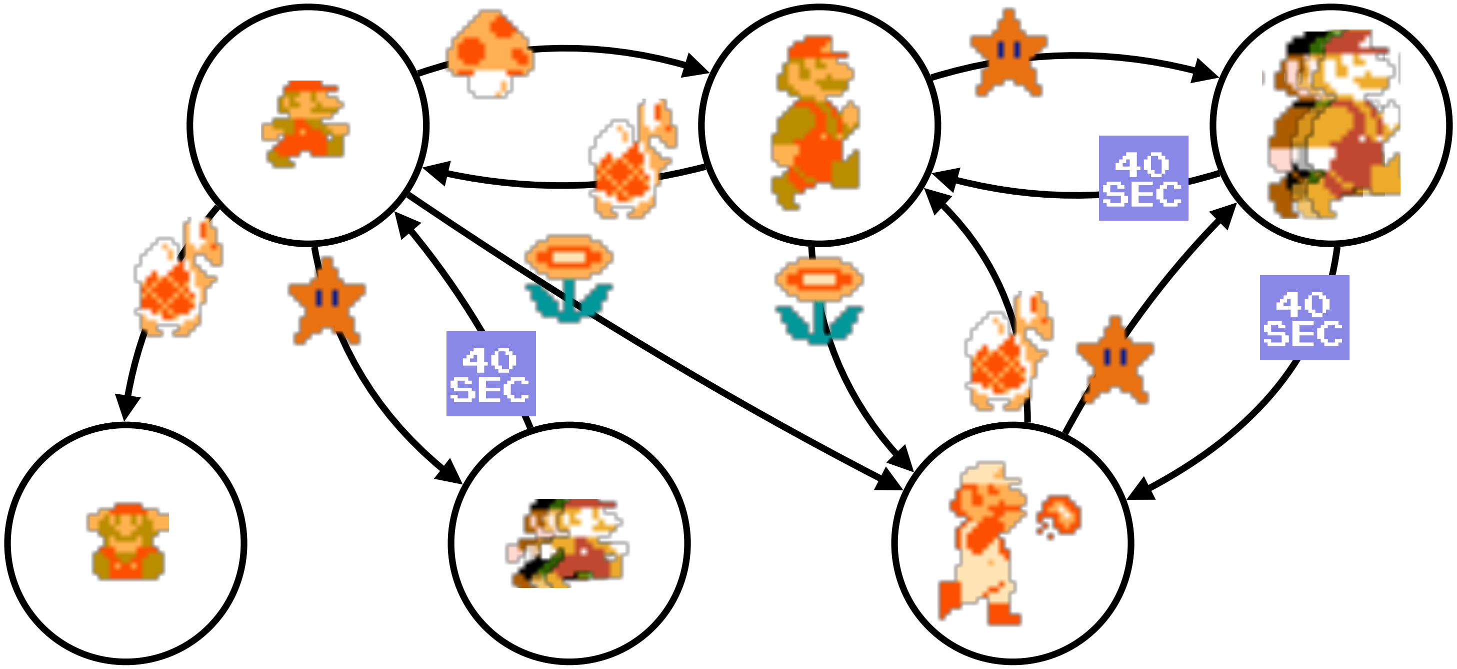
Represented as a graph of *nodes* connected by *edges*.



State diagram for Super Mario

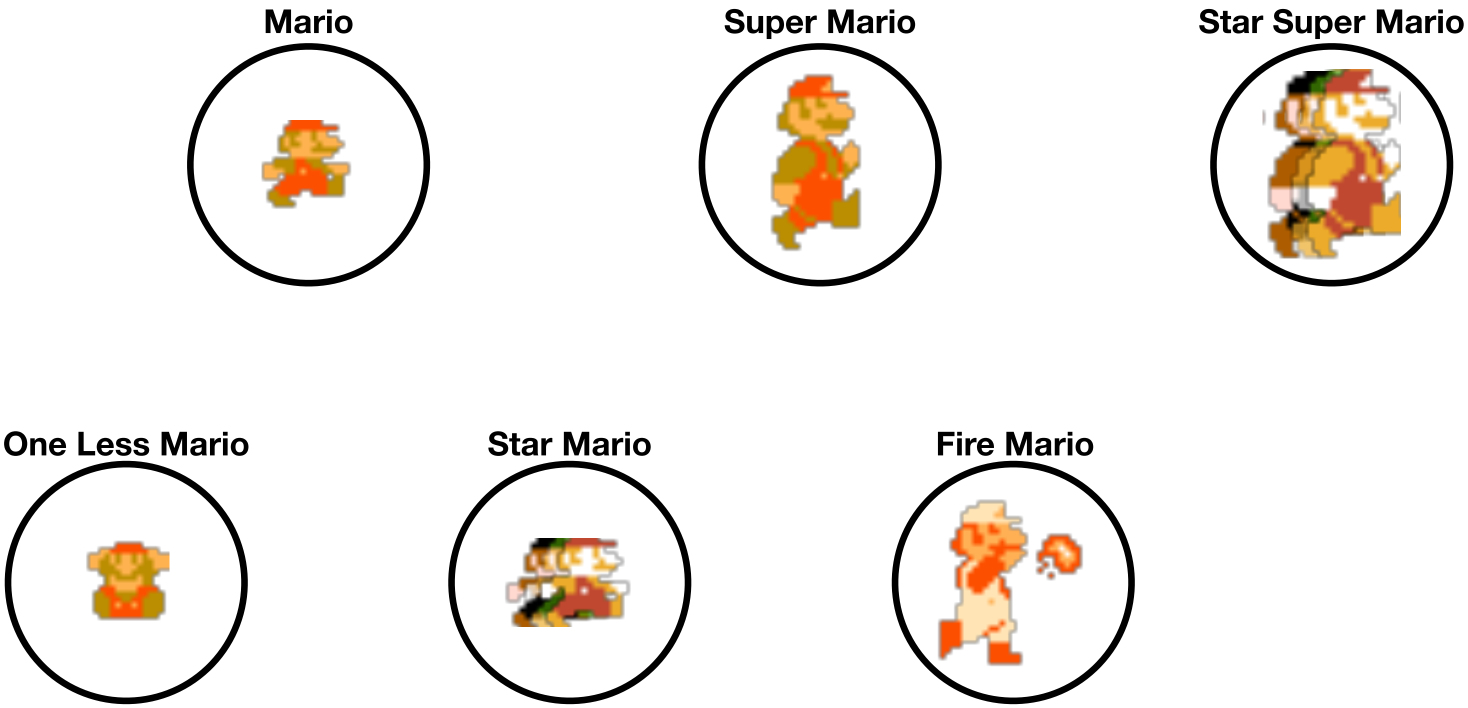
Finite State Machine

Represented as a graph of *nodes* connected by *edges*.



Finite State Machine

Represented as a graph of *nodes* connected by *edges*.



Mario

Super Mario

Star Super Mario

One Less Mario

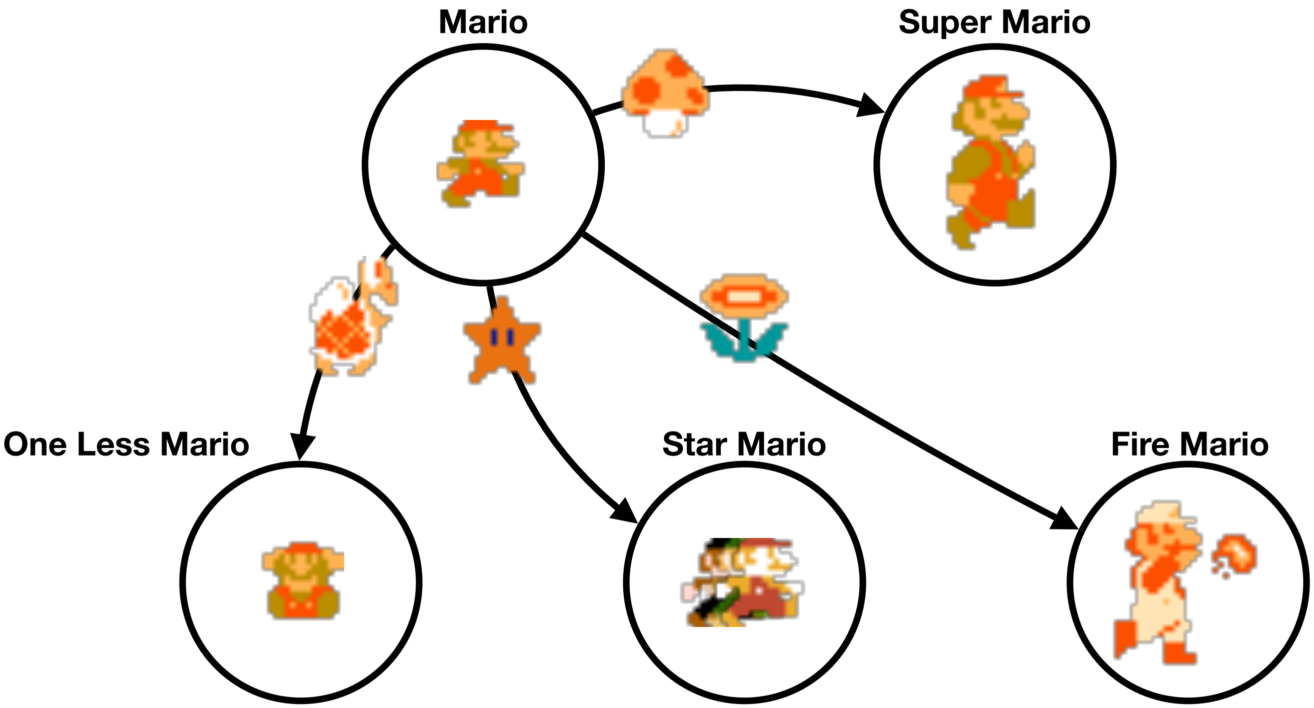
Star Mario

Fire Mario

Nodes represent the possible states of the system.

Finite State Machine

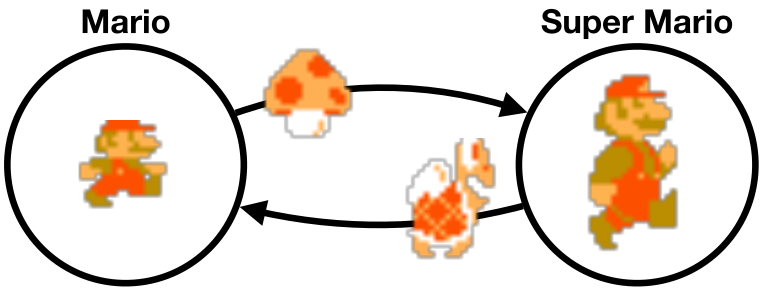
Represented as a graph of *nodes* connected by *edges*.



Edges represent how the system state changes.

Finite State Machine

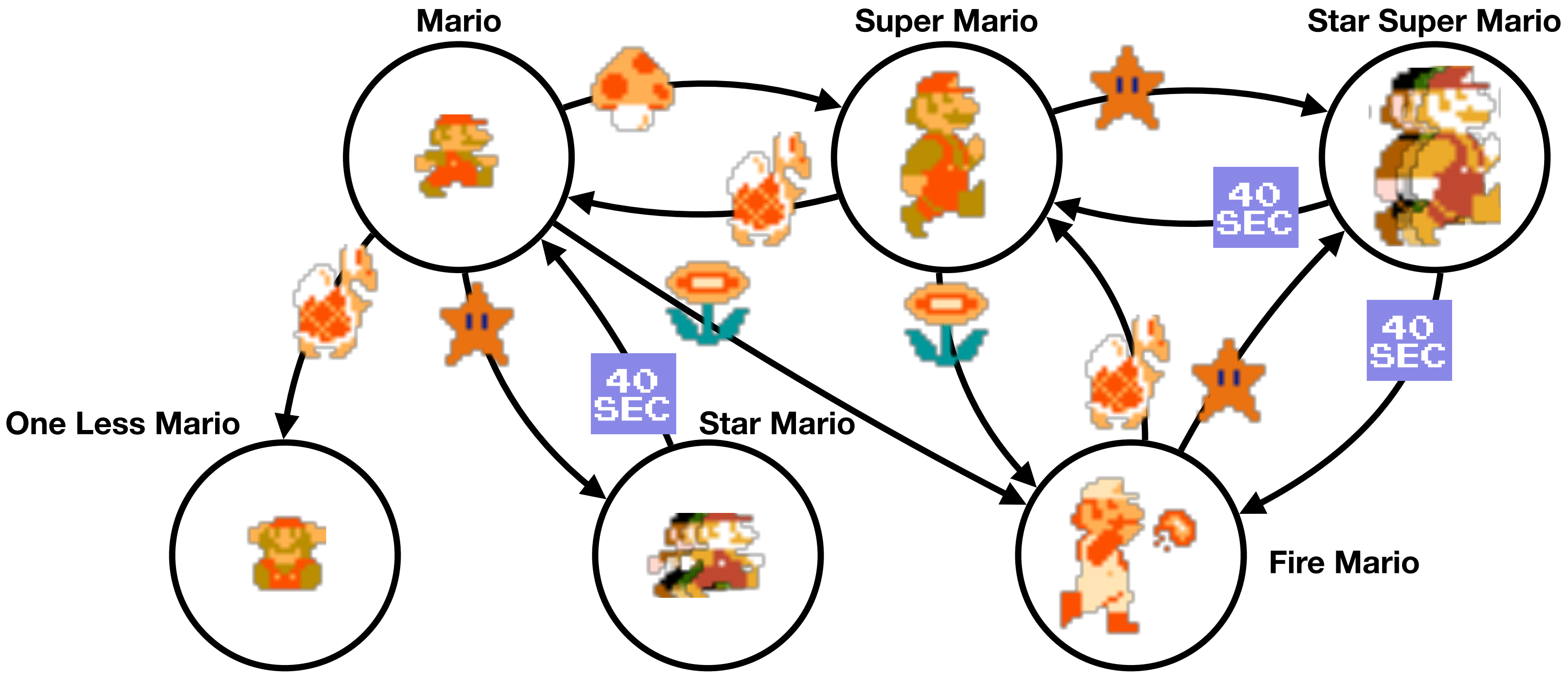
Represented as a graph of *nodes* connected by *edges*.



Edges are directional.
Their behavior depends on their direction.

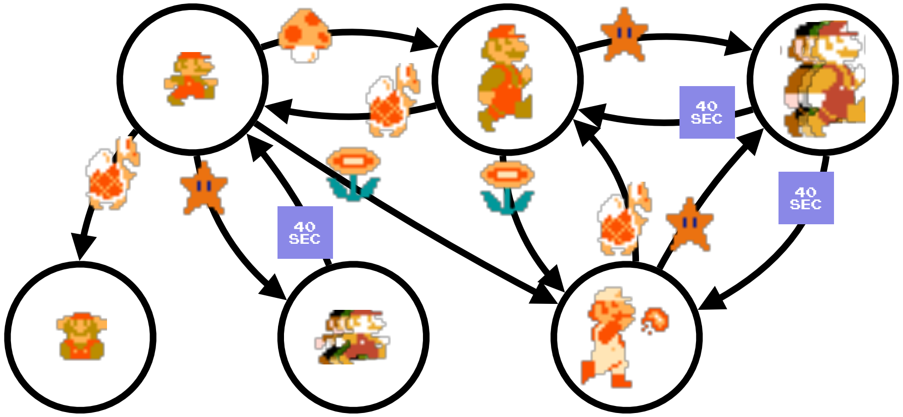
Finite State Machine

Represented as a graph of *nodes* connected by *edges*.



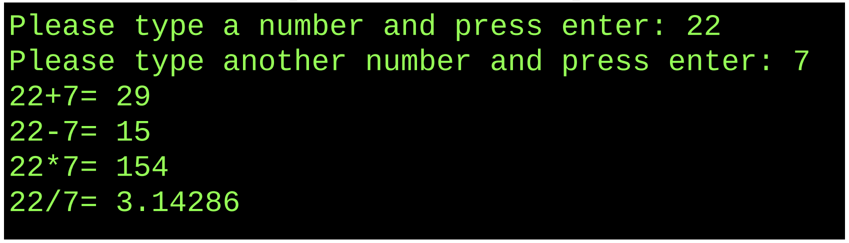
Nodes represent the possible states of the system.
Edges represent how the system state changes.

Finite state machines describe the behavior of programs



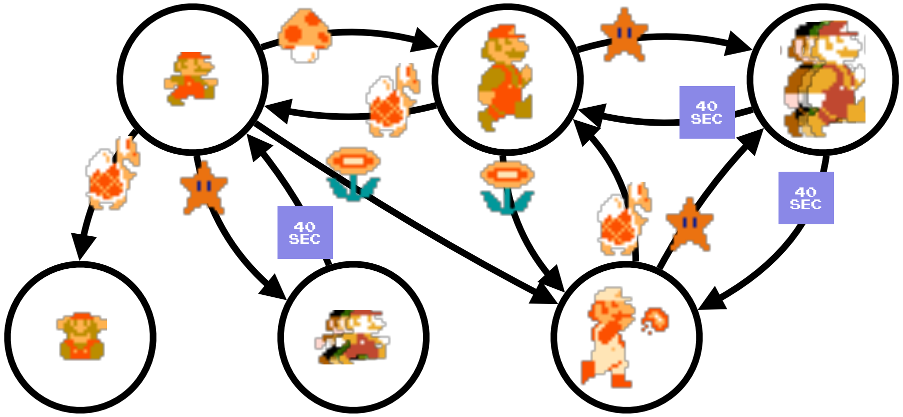
Mario state diagram

calculator (Version 41)


```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

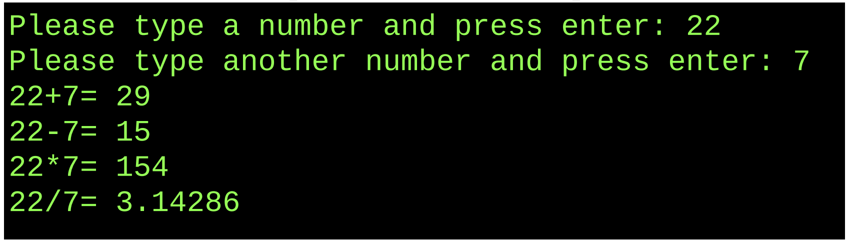
Can a Finite State Machine describe our current pocket calculator?

Finite state machines describe the behavior of programs



Mario state diagram

calculator (Version 41)



```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

What is this code doing ?

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
   // Let's declare our variables
   float myNumber, myOtherNumber;  // Calculation operands
   float sumNumber, differenceNumber, productNumber, quotientNumber;

   // Ask the user for the first operand
   getNumber(myNumber);

   // Ask the user for the second operand
   getNumber(myOtherNumber);

   // Perform all operations and store results in variables
   performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

   // Output operation results to screen
   outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

  return 0;
}
```

```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

Functions

```
addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
performOperations()
outputResults()
main()
```

Give us two numbers for our operands

Perform all operations

Output results to screen

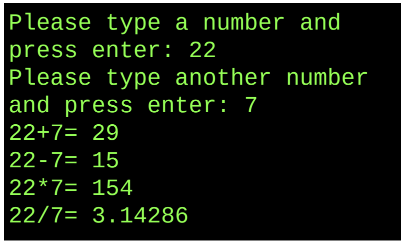
What is this code doing ?

Give us two numbers for our operands

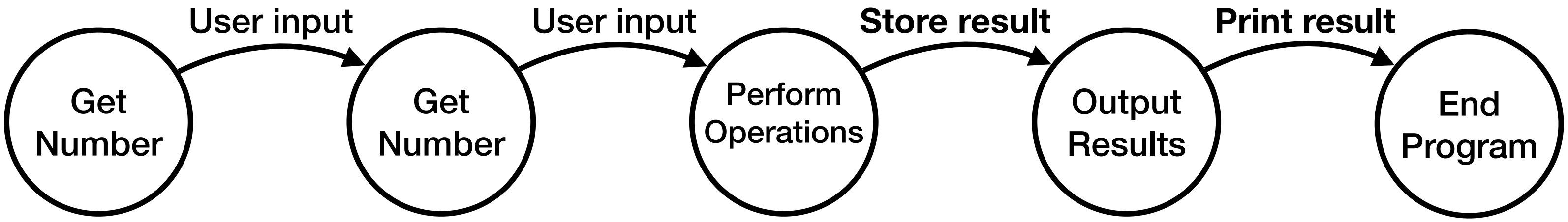
Perform all operations

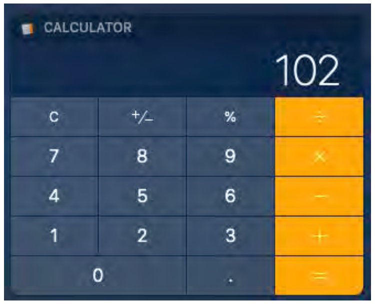
Output results to screen

calculator (Version 41)



```
Please type a number and
press enter: 22
Please type another number
and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```





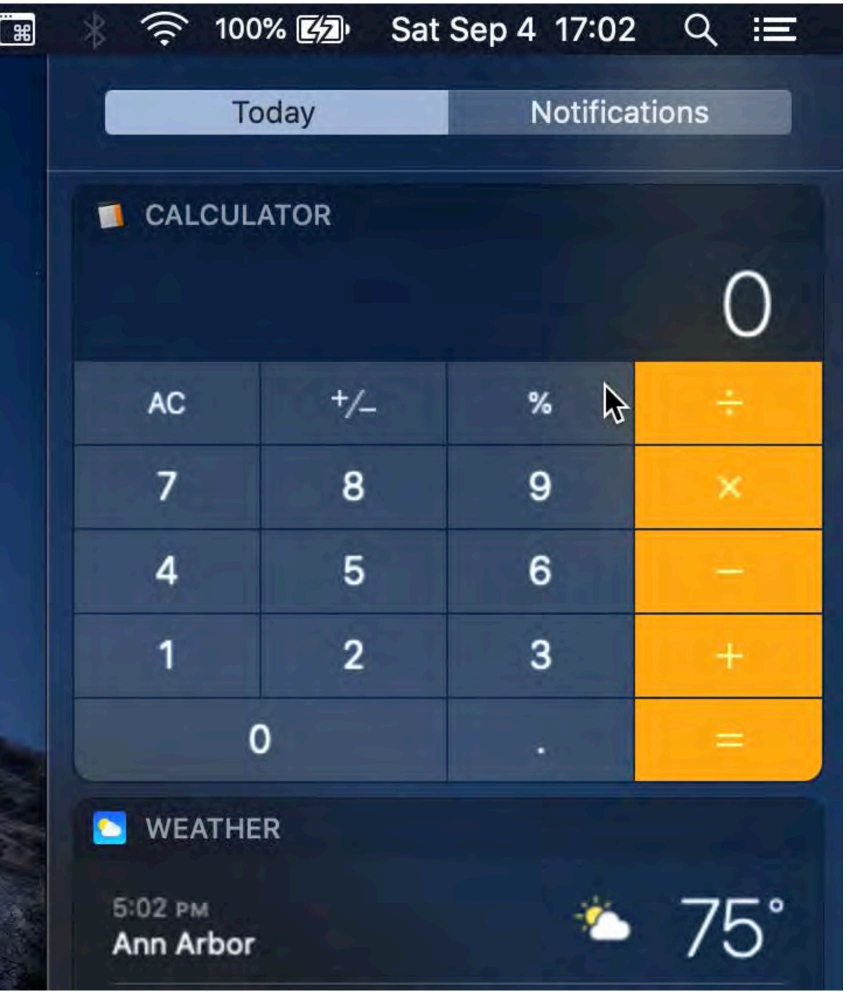
Pocket Calculator state machine is different

What would this "FSM" look like ?

Let’s walk through a calculation example

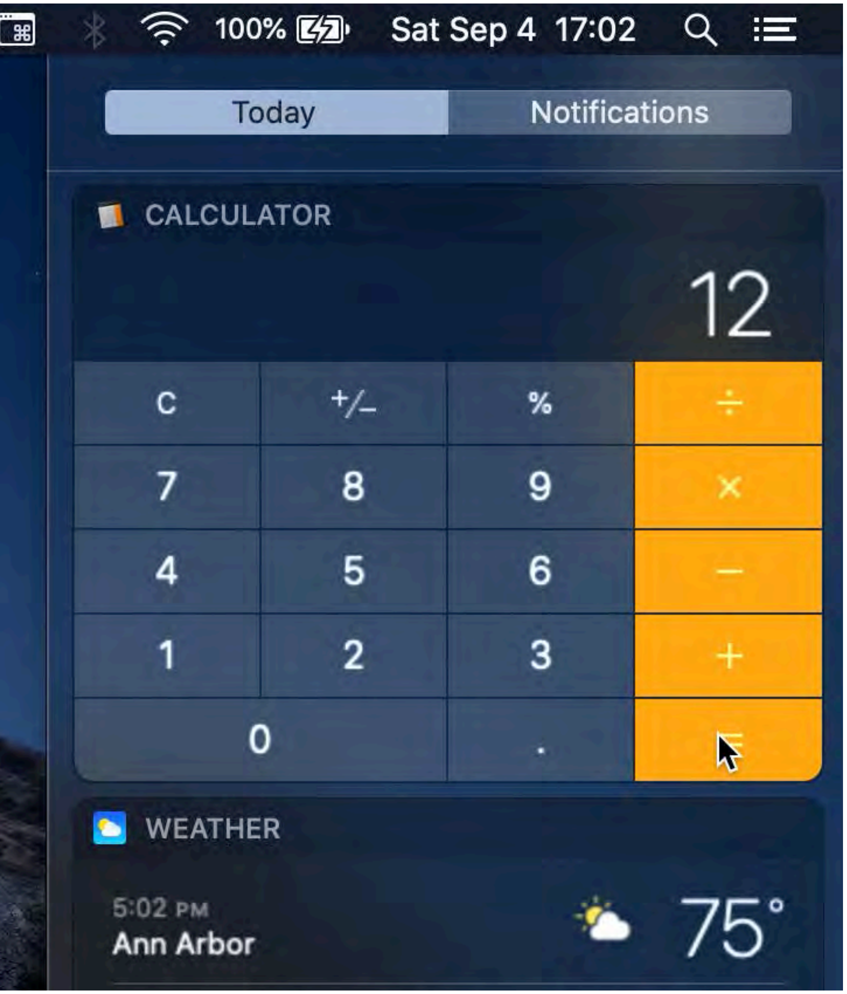
Let's walk through a calculation example

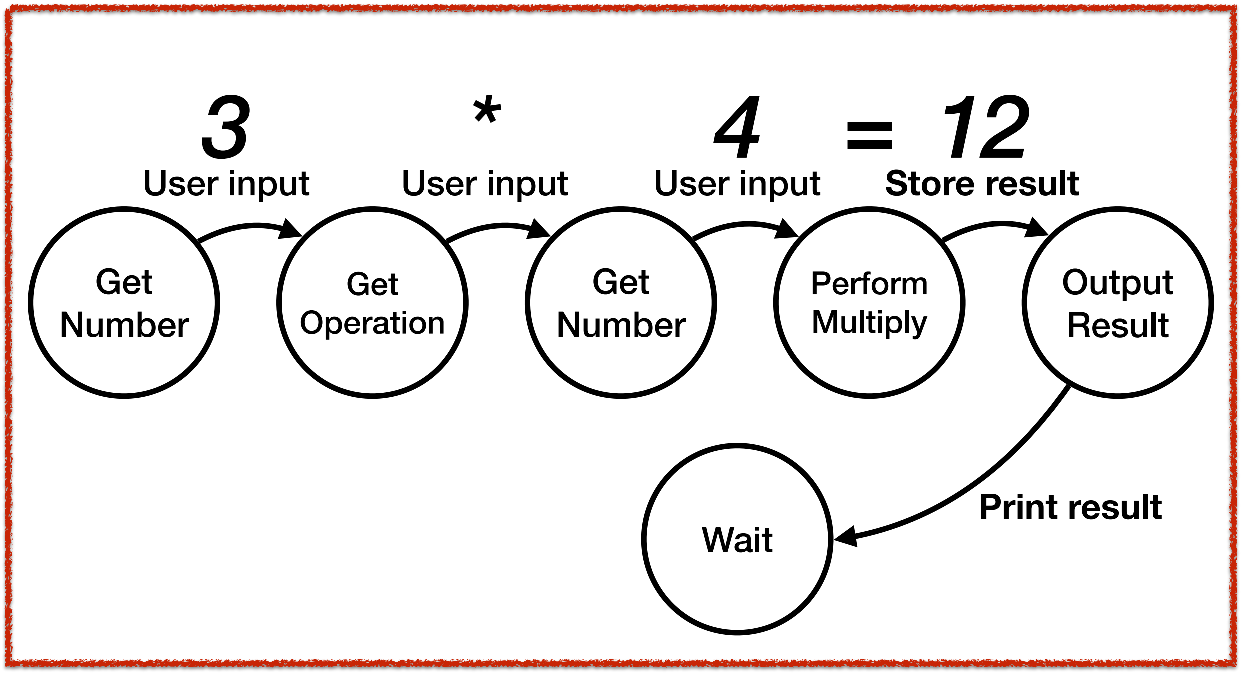
*3 * 4 = 12*



Let’s walk through a calculation example

*3 * 4 = 12*




3

User input

*

User input

4

User input

= *12*

Store result

Get Number

Get Operation

Get Number

Perform Multiply

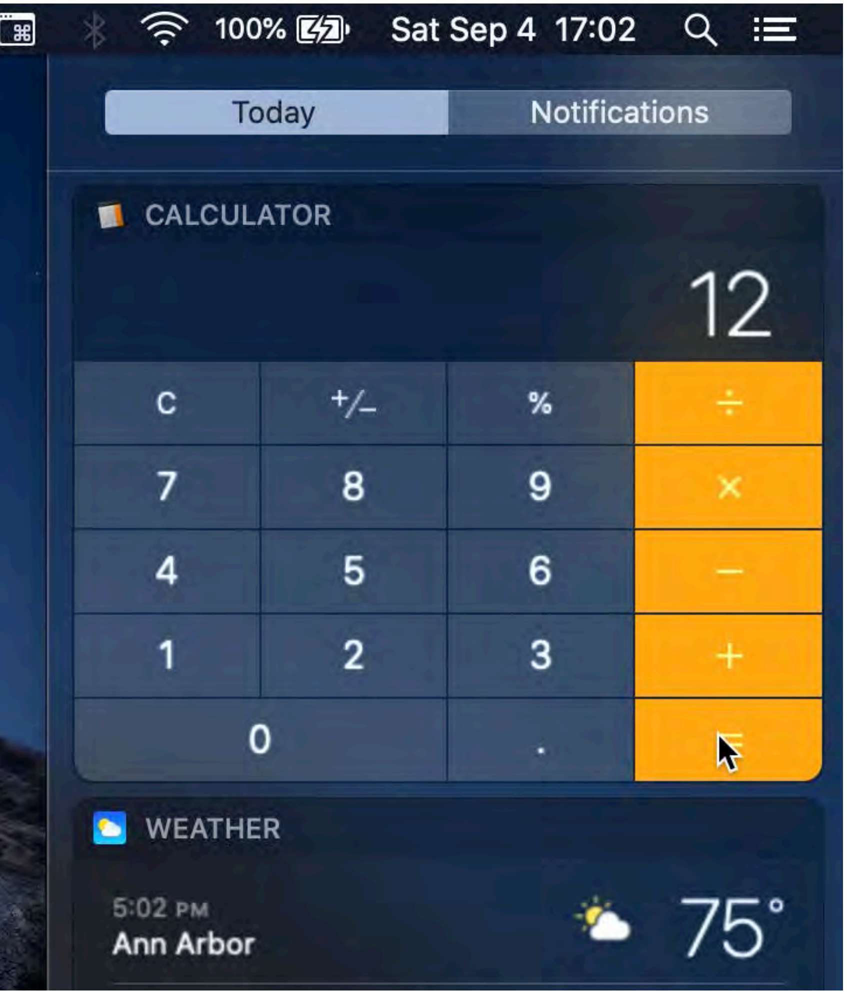
Output Result

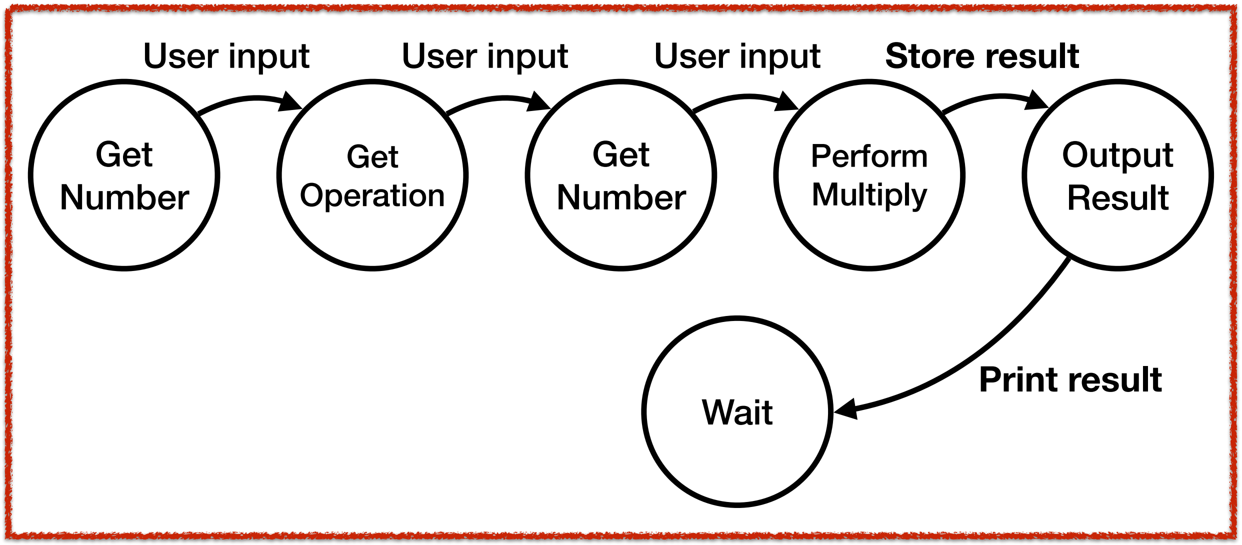
Print result

Wait

Let’s walk through a calculation example

*3 * 4 = 12*





Get Number

User input

Get Operation

User input

Get Number

User input

Perform Multiply

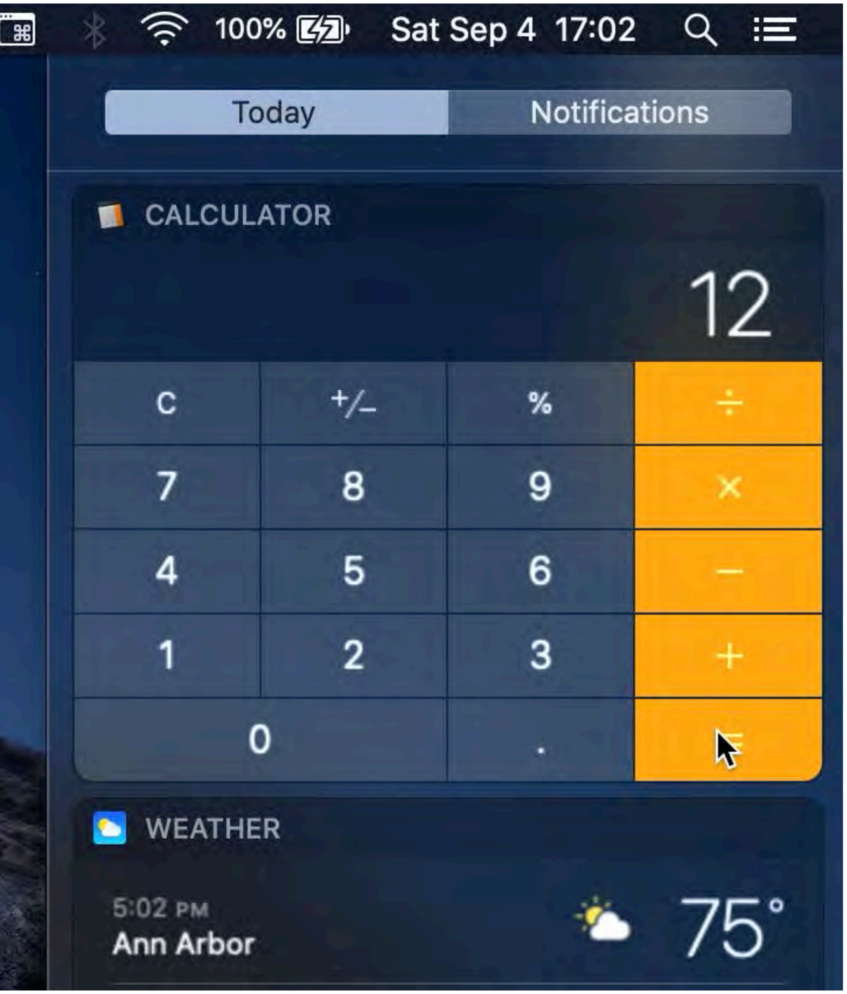
Store result

Output Result

Print result

Wait

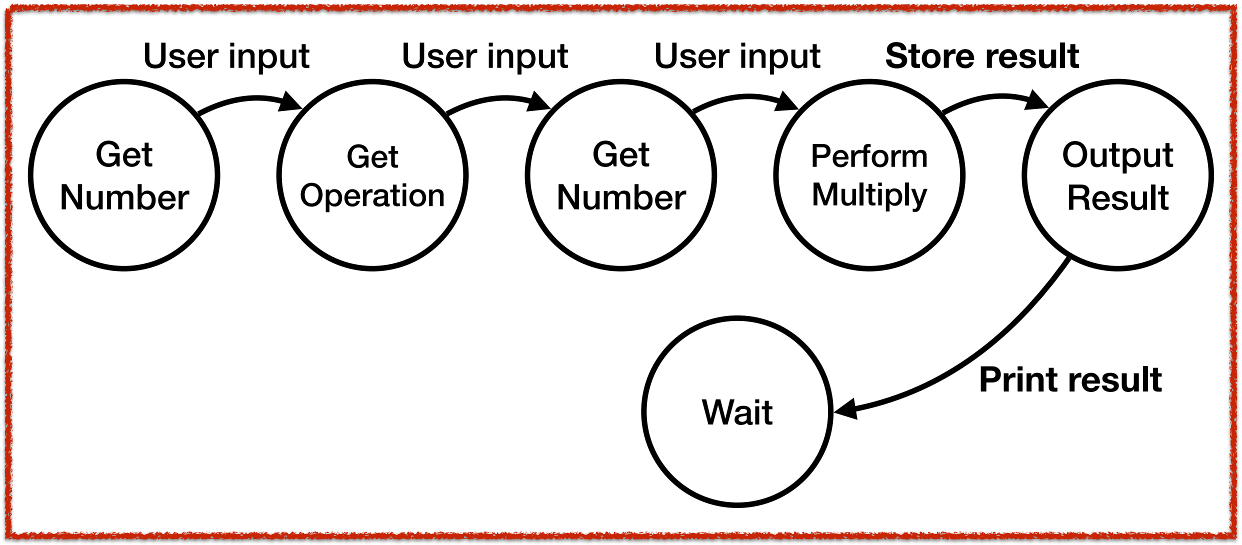
Let’s walk through a calculation example



*3 * 4 = 12*

12 + 8 = 20

The result of the first operation becomes the first operand of the second operation



User input

User input

User input

Store result

Get Number

Get Operation

Get Number

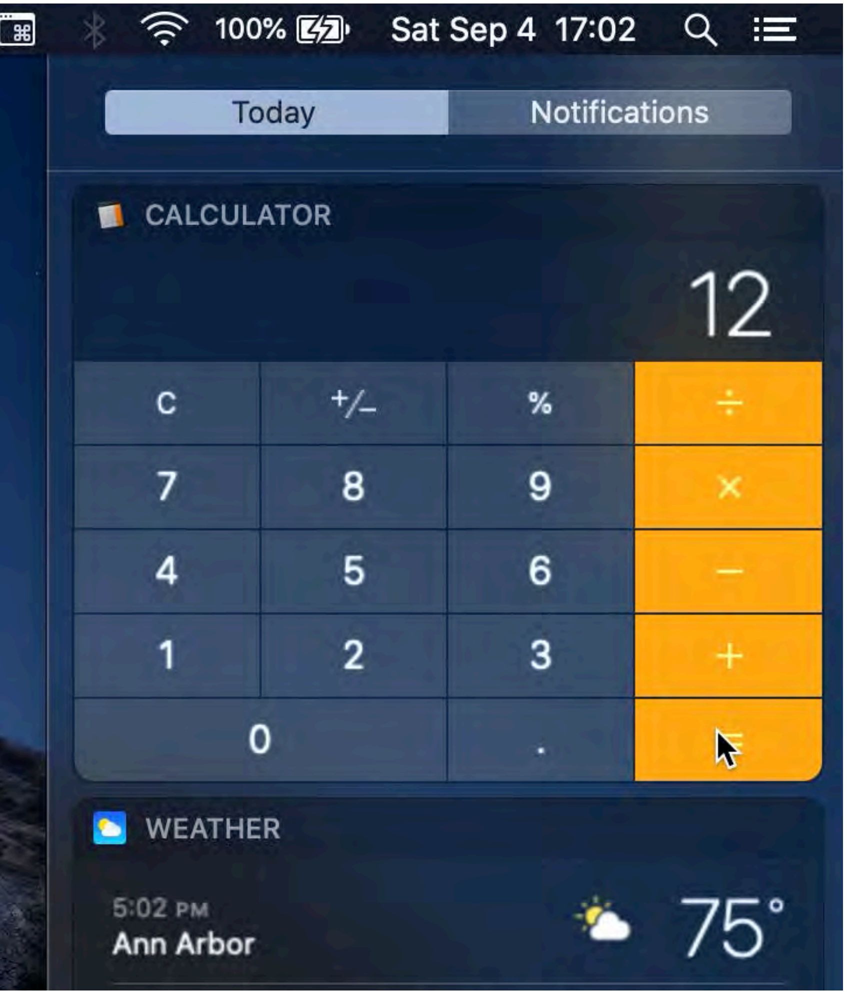
Perform Multiply

Output Result

Print result

Wait

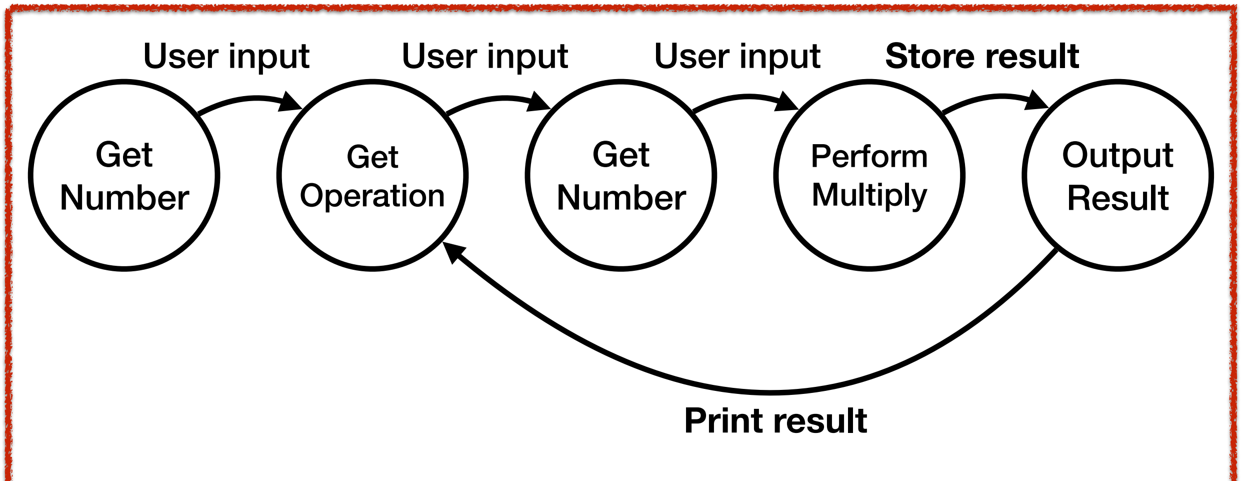
Let’s walk through a calculation example



*3 * 4 = 12*

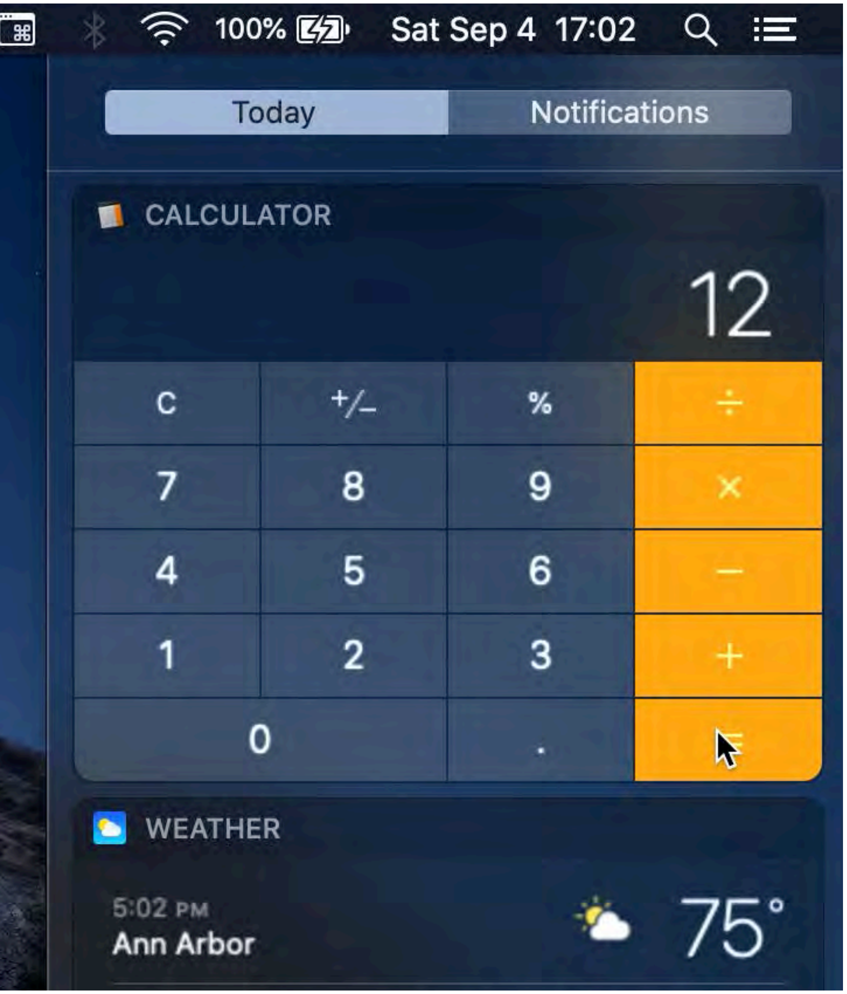
12 + 8 = 20

The result of the first operation becomes the first operand of the second operation



This program will iterate forever

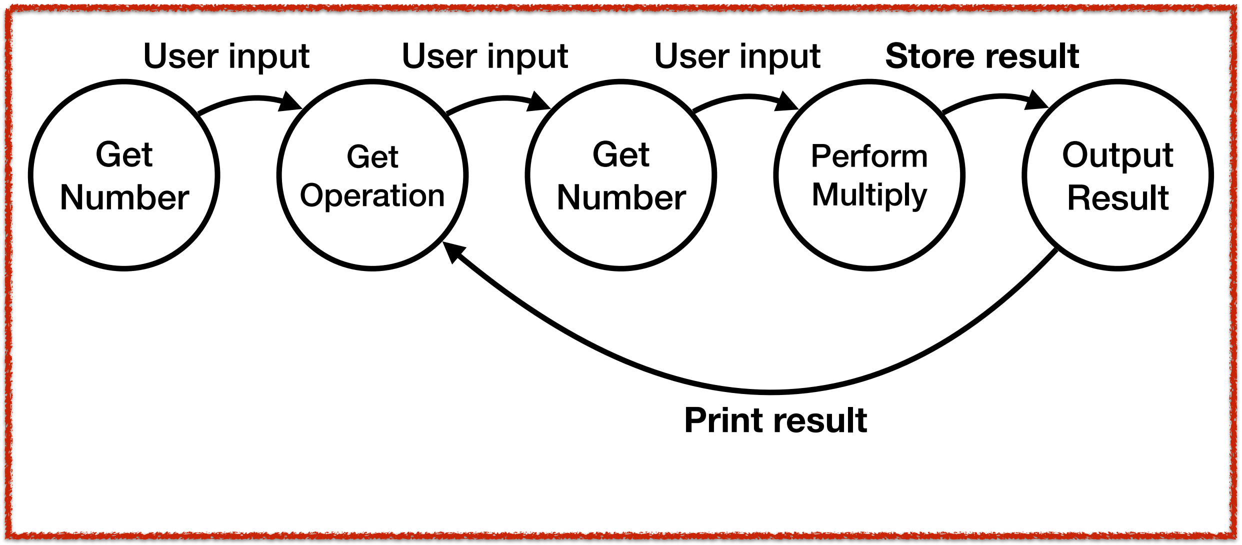
Let’s walk through a calculation example



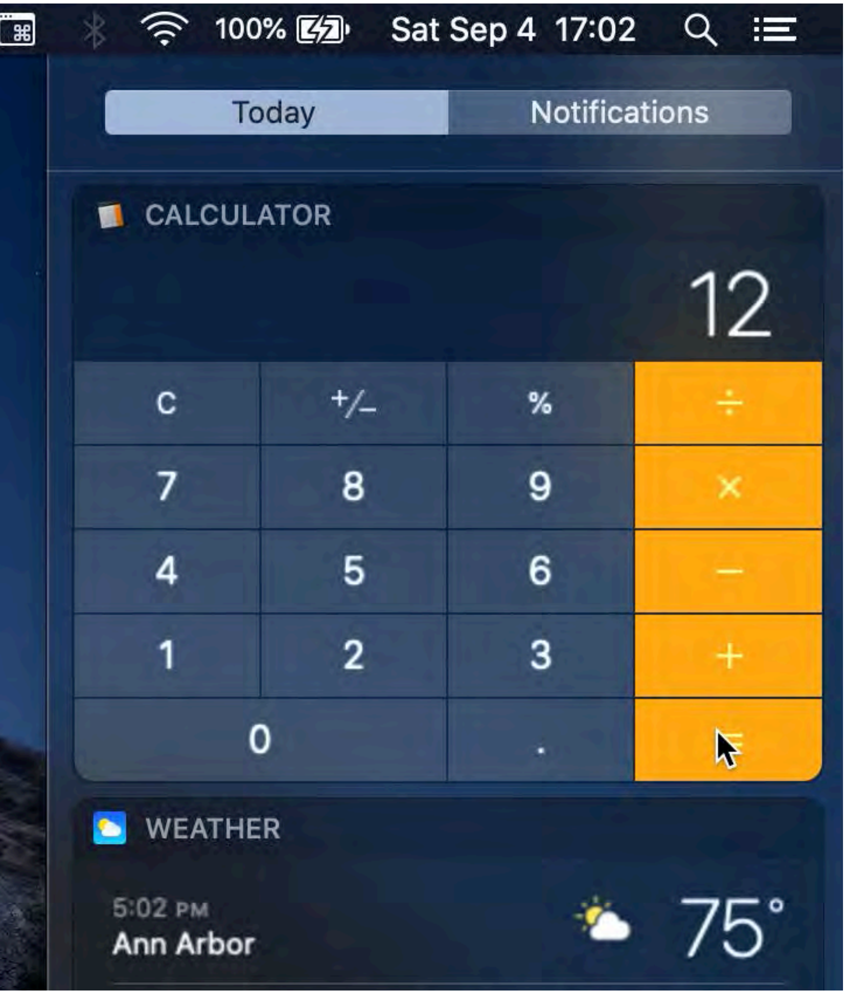
$$3 * 4 = 12$$

$$12 + 8 = 20$$

User chose addition instead of multiplication



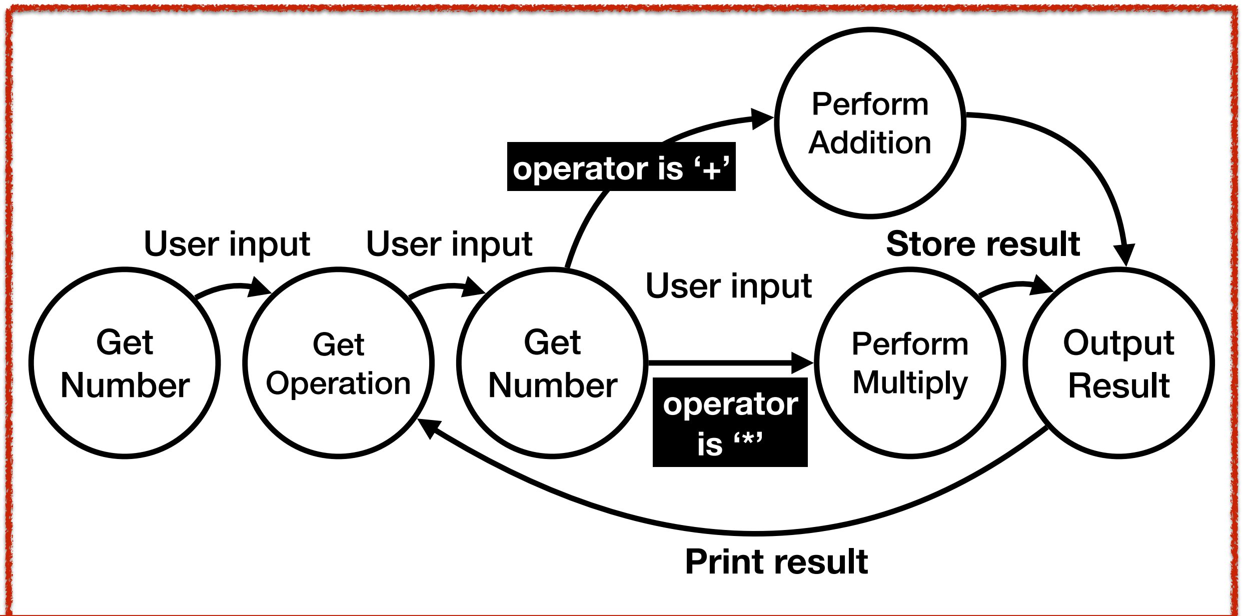
Let's walk through a calculation example



3 * 4 = 12

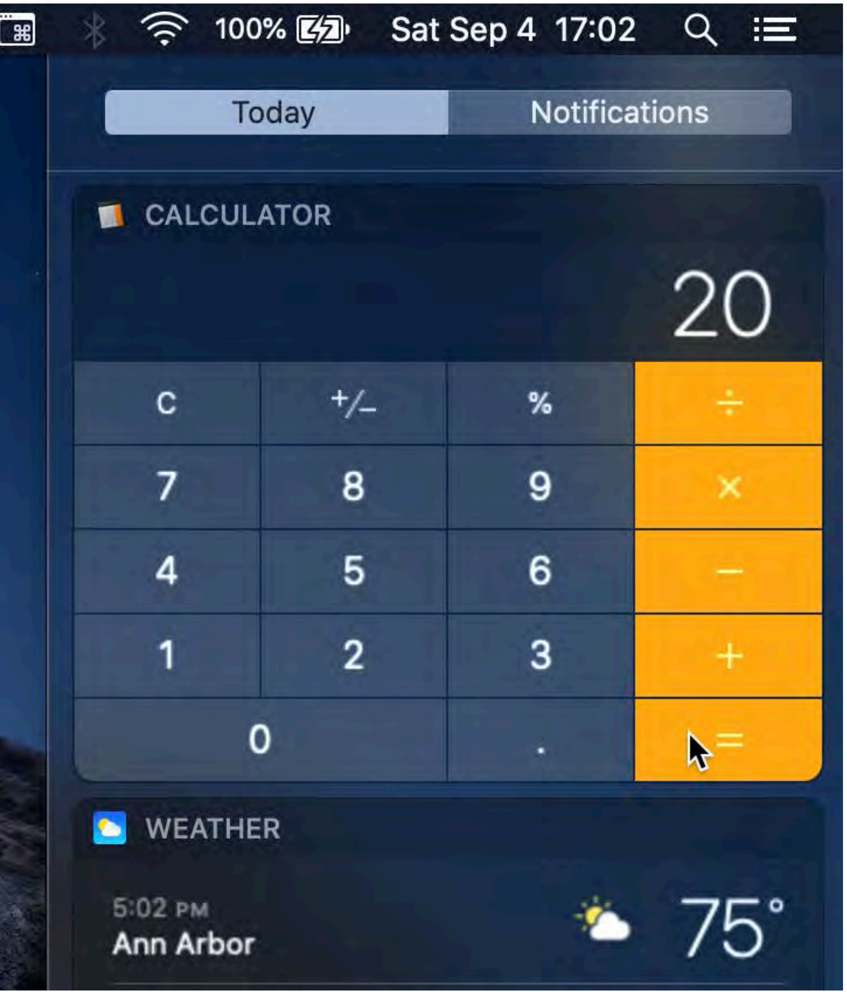
12 + 8 = 20

User chose addition instead of multiplication



This program branches for operators

Let’s walk through a calculation example

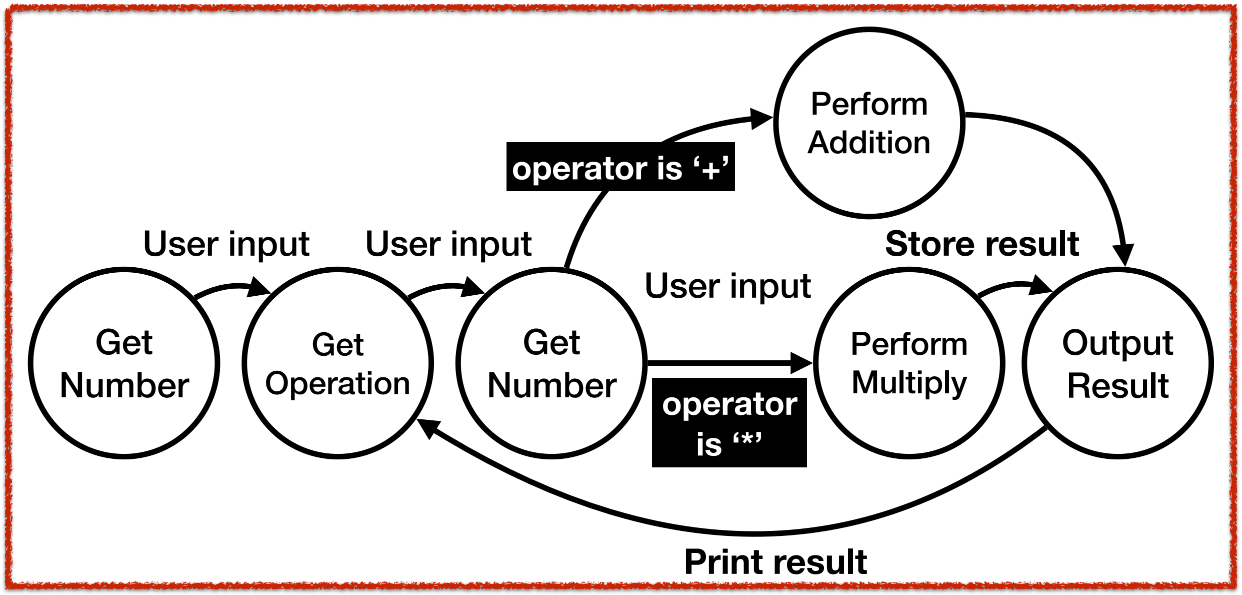


*3 * 4 = 12*

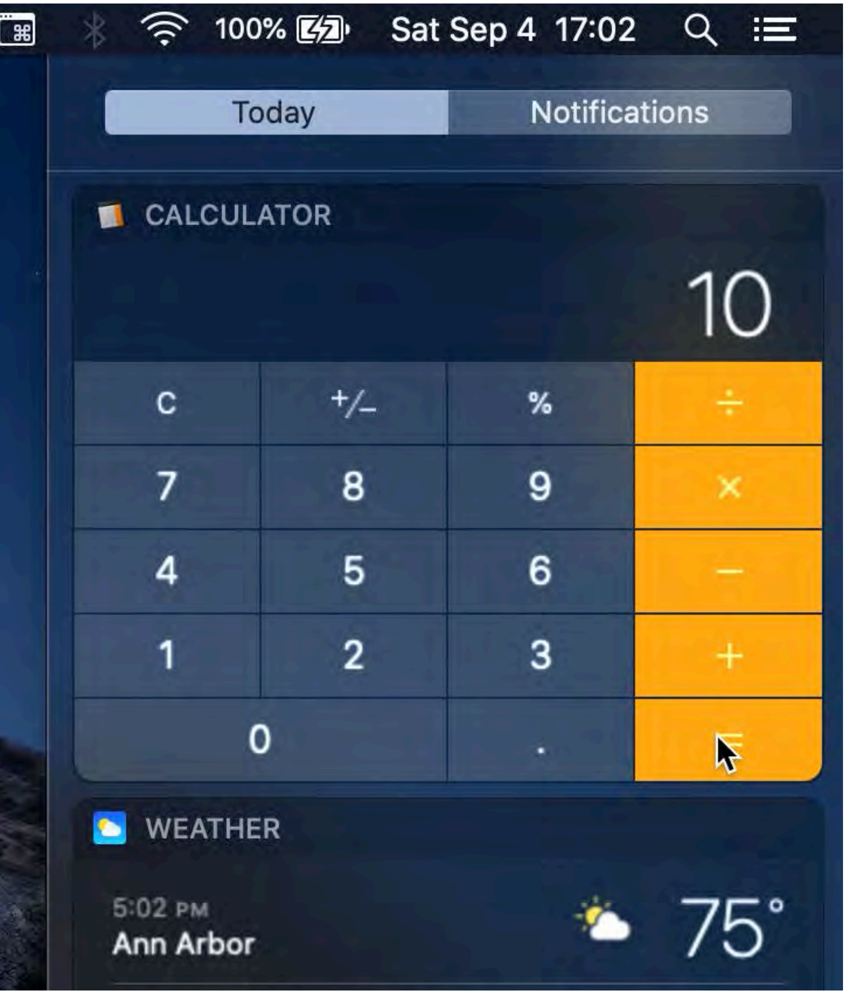
12 + 8 = 20

20 - 10 = 10

The result of the *current* operation becomes the first operand of the *next* operation



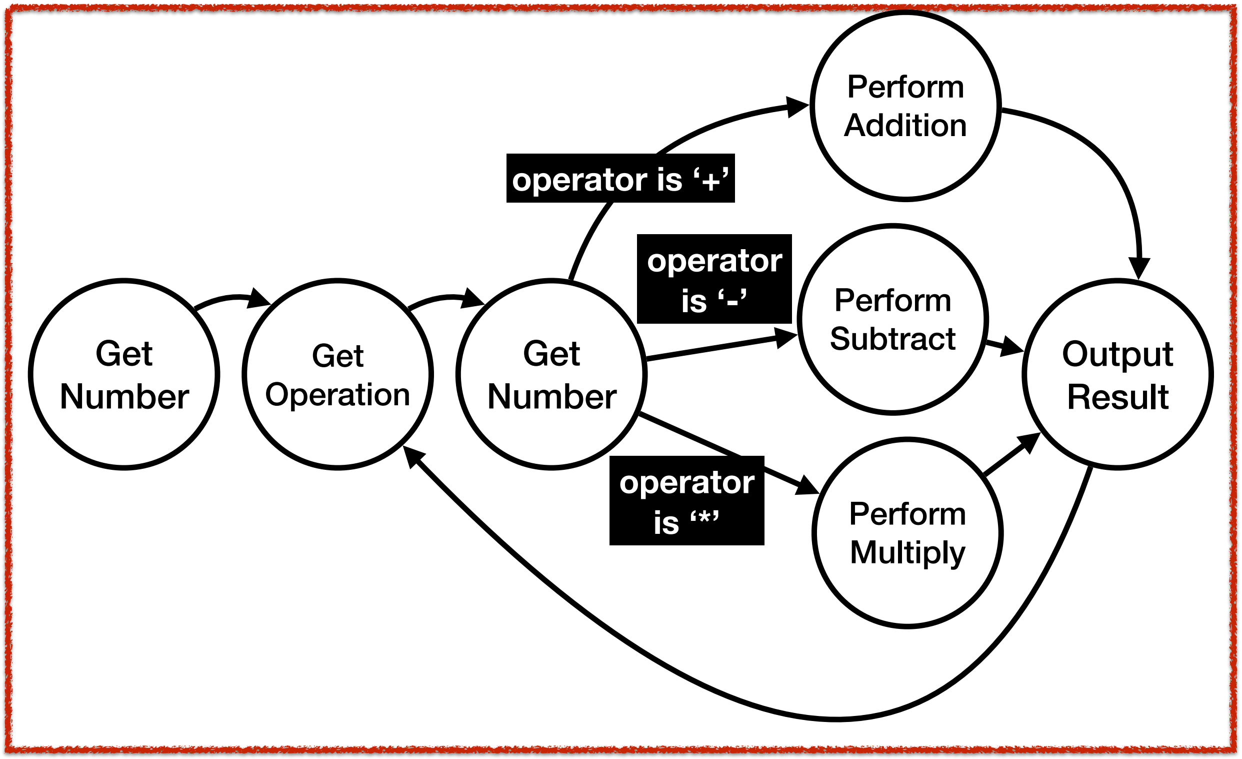
Let’s walk through a calculation example



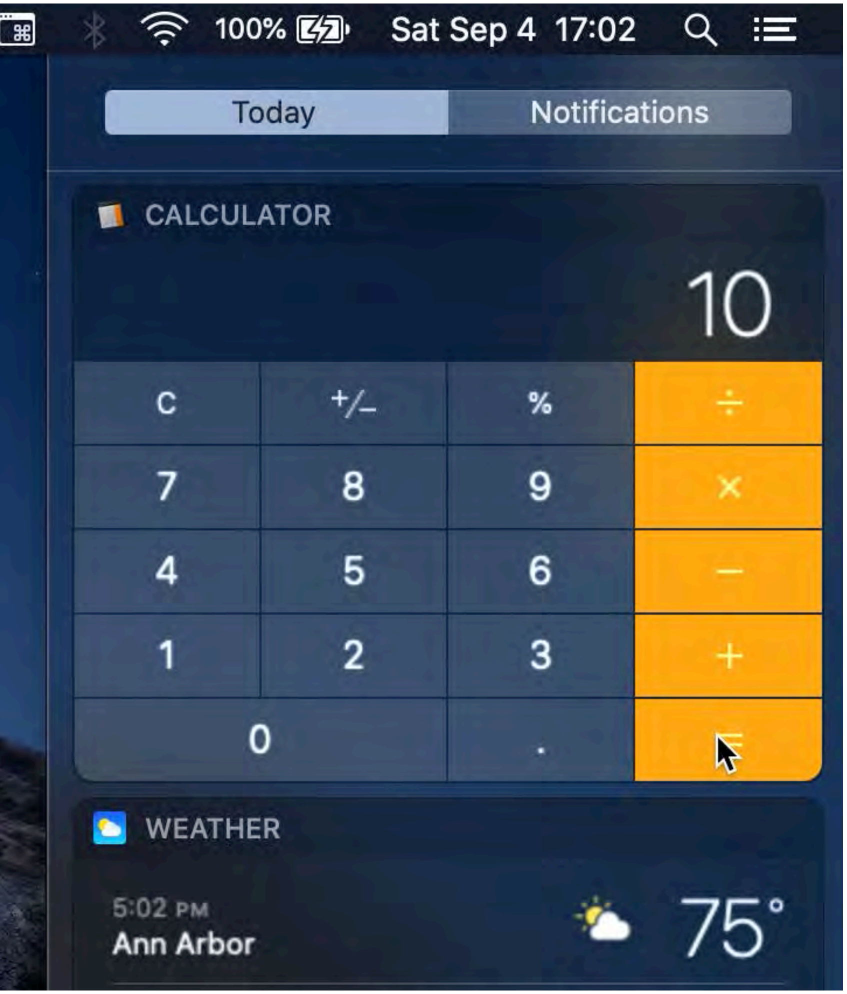
*3 * 4 = 12*

12 + 8 = 20

20 - 10 = 10



Let’s walk through a calculation example

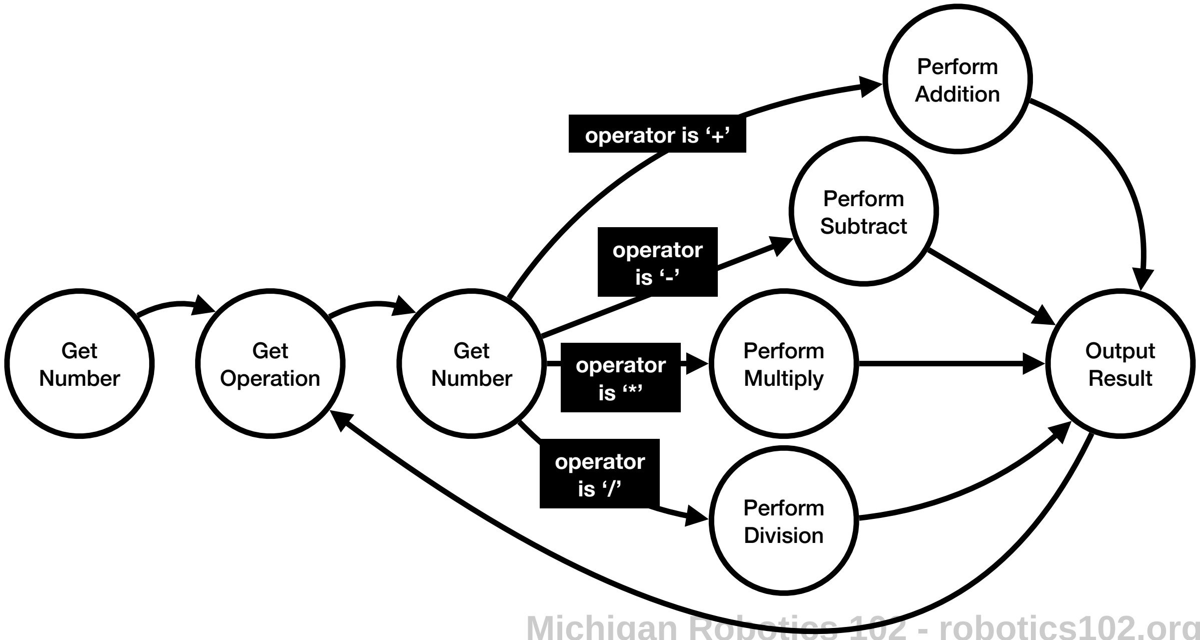


3 * 4 = 12

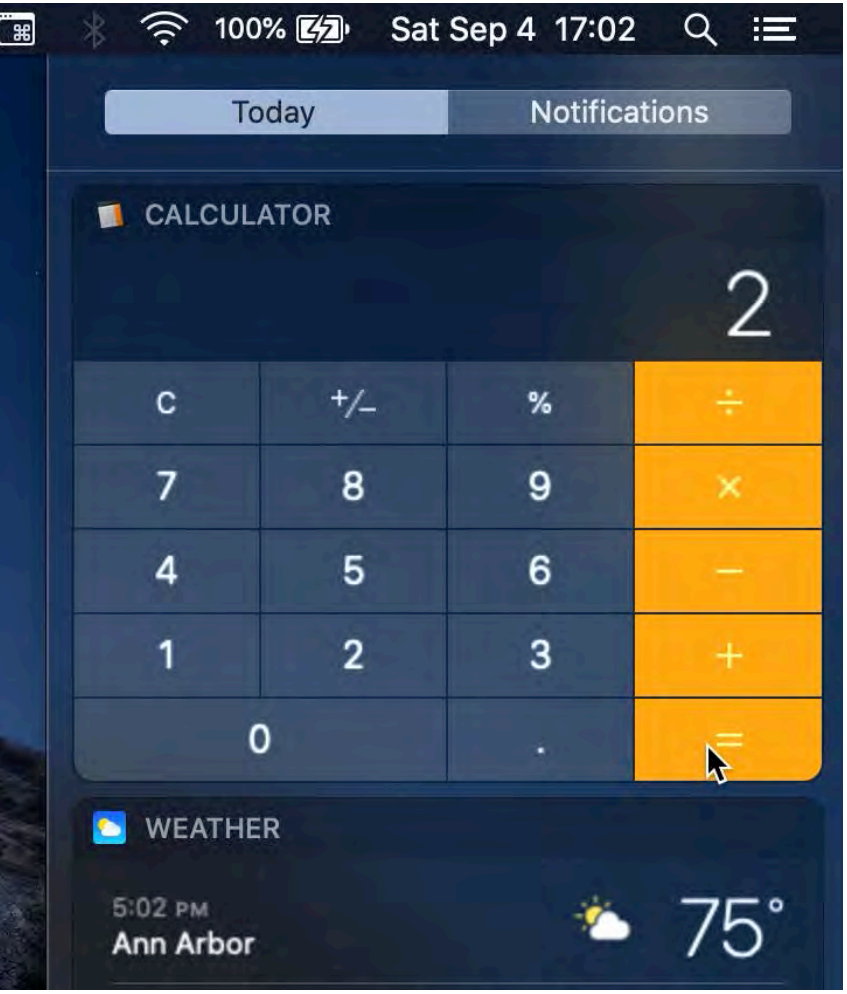
12 + 8 = 20

20 - 10 = 10

10 / 5 = 2



Let's walk through a calculation example



*3 * 4 = 12*

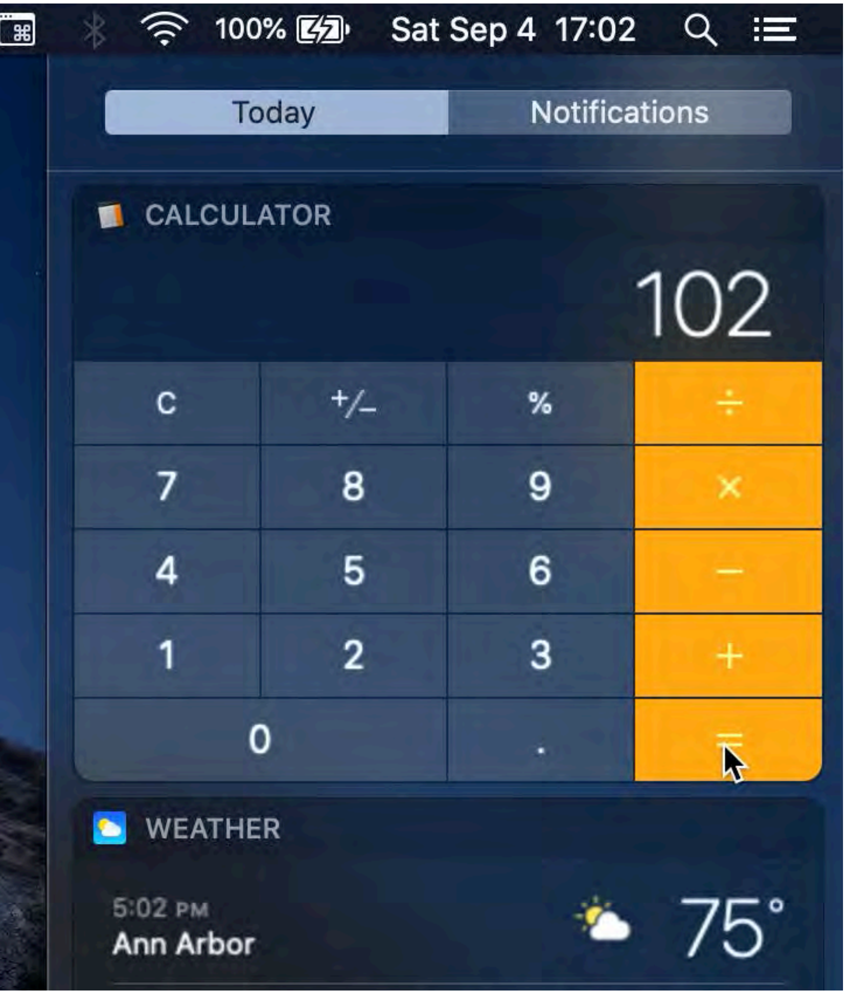
12 + 8 = 20

20 - 10 = 10

10 / 5 = 2

*2 * 51 = 102*

Let’s walk through a calculation example



*3 * 4 = 12*

12 + 8 = 20

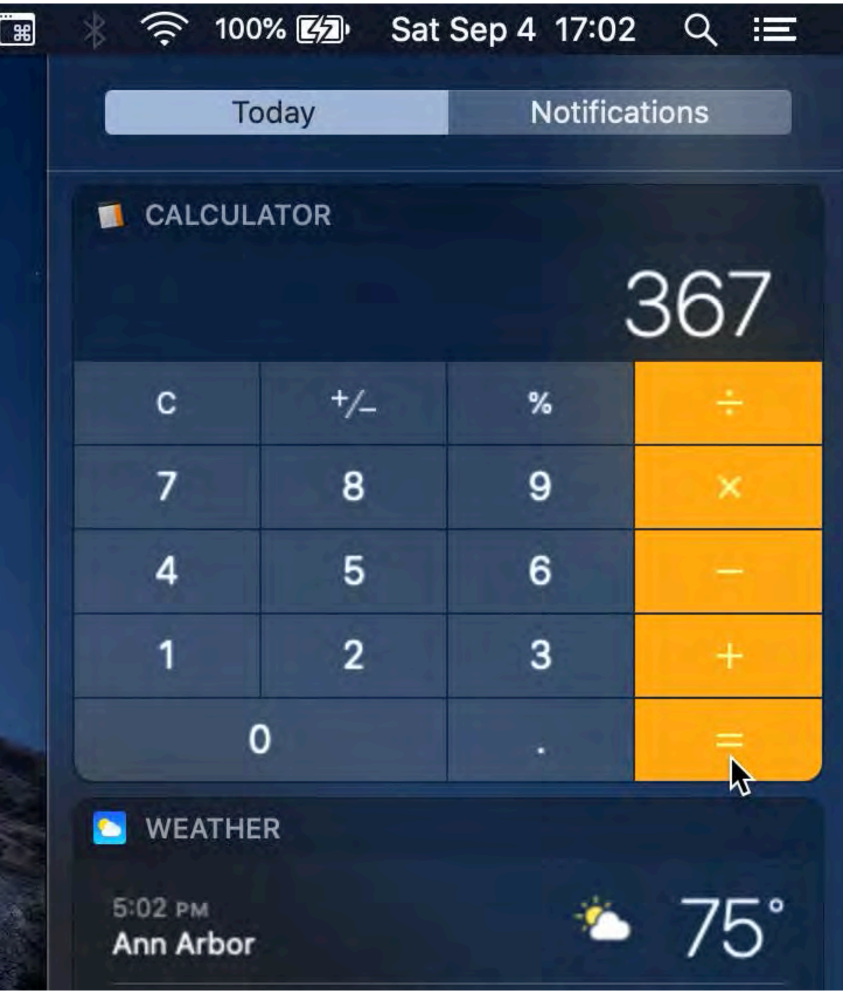
20 - 10 = 10

10 / 5 = 2

*2 * 51 = 102*

102 - -265 = 367

Let’s walk through a calculation example



3 * 4 = 12

12 + 8 = 20

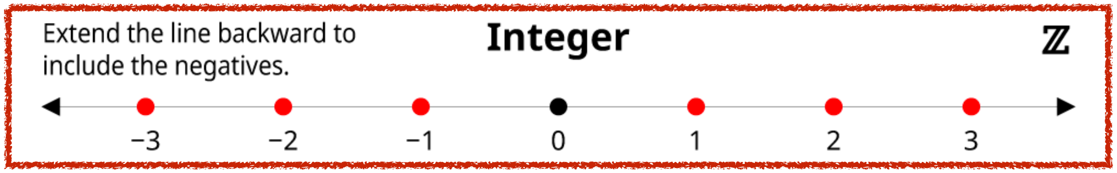
20 - 10 = 10

10 / 5 = 2

2 * 51 = 102

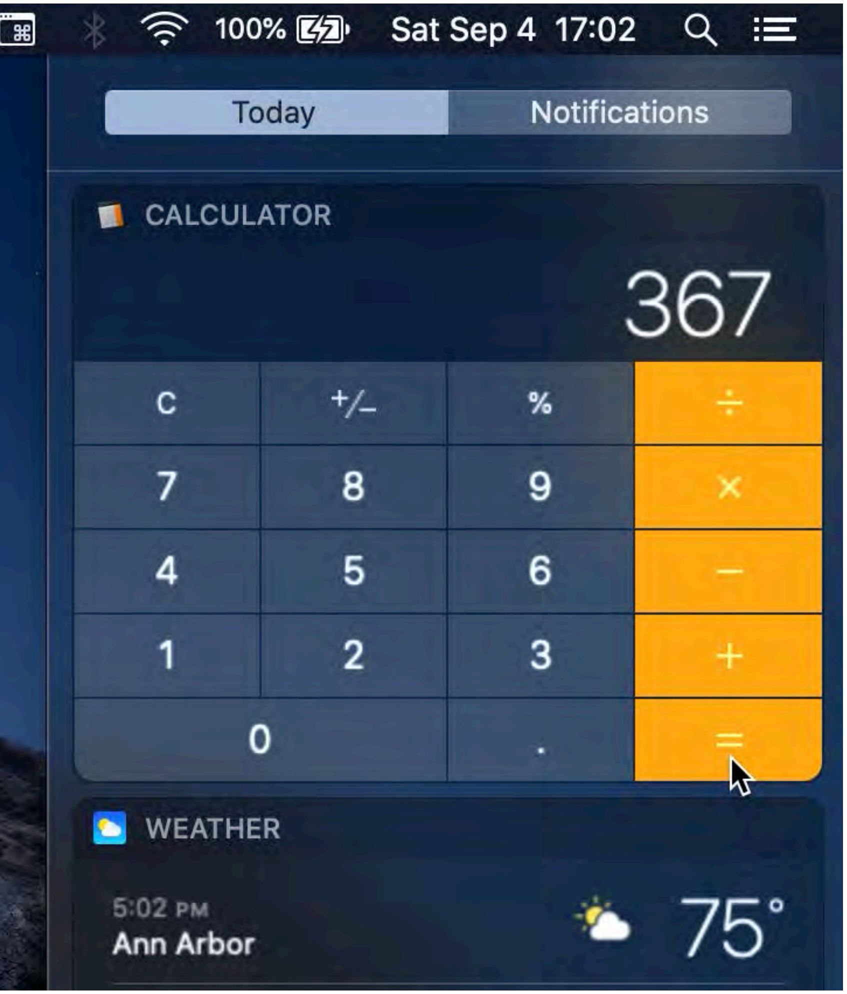
102 - -265 = 367

Integer Operations


Extend the line backward to include the negatives.

https://thinkzone.wlonk.com/Numbers/NumberSets.htm

Let's walk through a calculation example



*3 * 4 = 12*

12 + 8 = 20

20 - 10 = 10

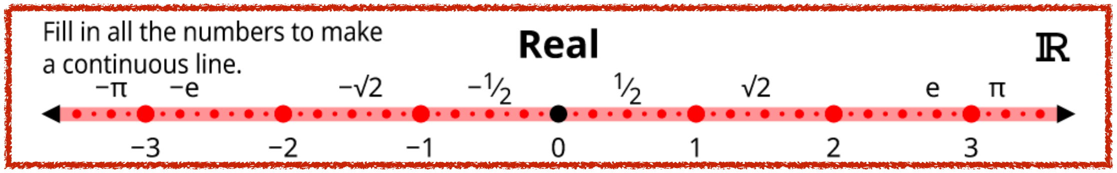
10 / 5 = 2

*2 * 51 = 102*

102 - -265 = 367

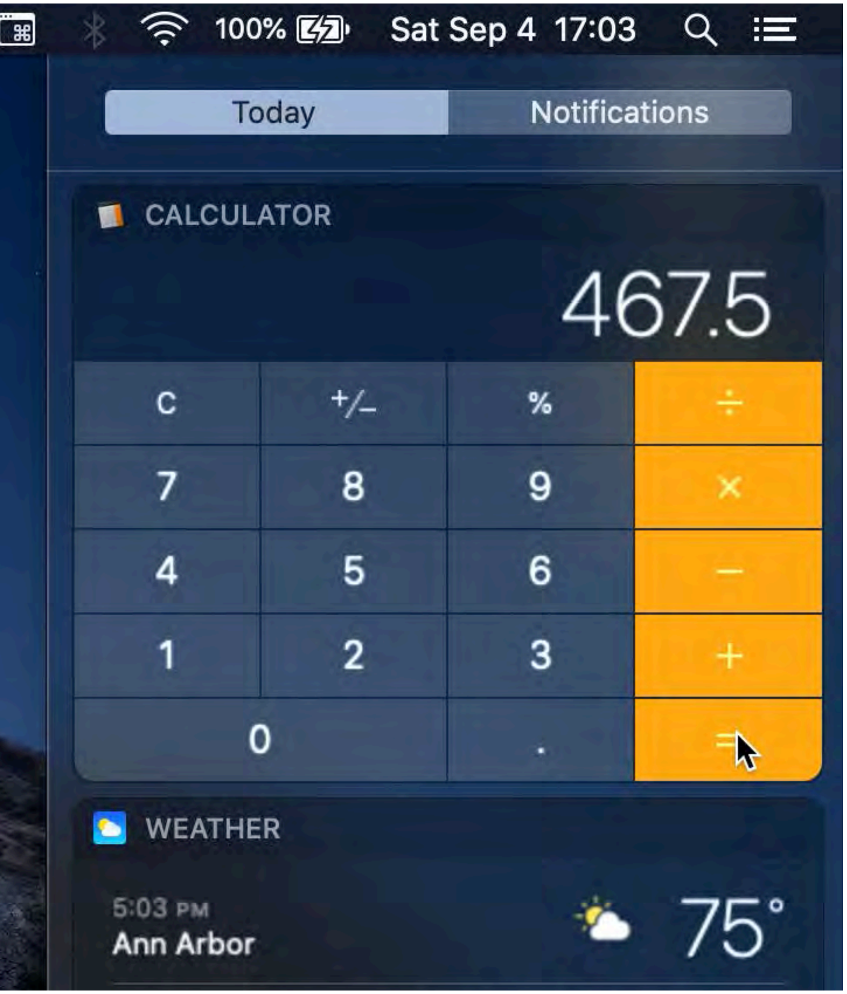
367 + 100.5 = 467.5

Floating Point Operations



https://thinkzone.wlonk.com/Numbers/NumberSets.htm

Let’s walk through a calculation example



*3 * 4 = 12*

12 + 8 = 20

20 - 10 = 10

10 / 5 = 2

*2 * 51 = 102*

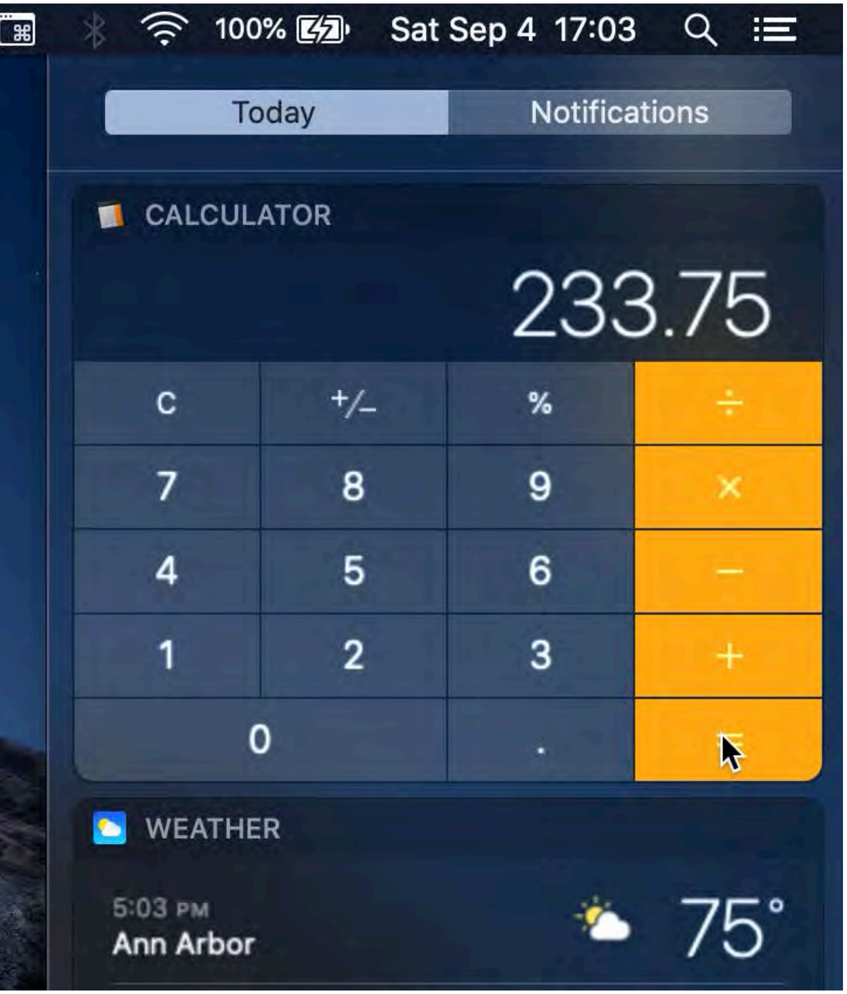
102 - -265 = 367

367 + 100.5 = 467.5

*467.5 * 0.5 = 233.75*

Floating Point Operations

Let's walk through a calculation example



3 * 4 = 12

12 + 8 = 20

20 - 10 = 10

10 / 5 = 2

2 * 51 = 102

102 - -265 = 367

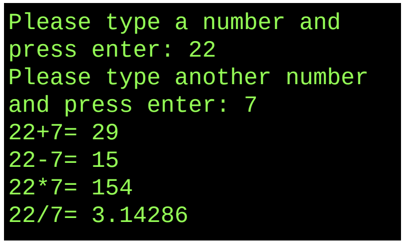
367 + 100.5 = 467.5

467.5 * 0.5 = 233.75

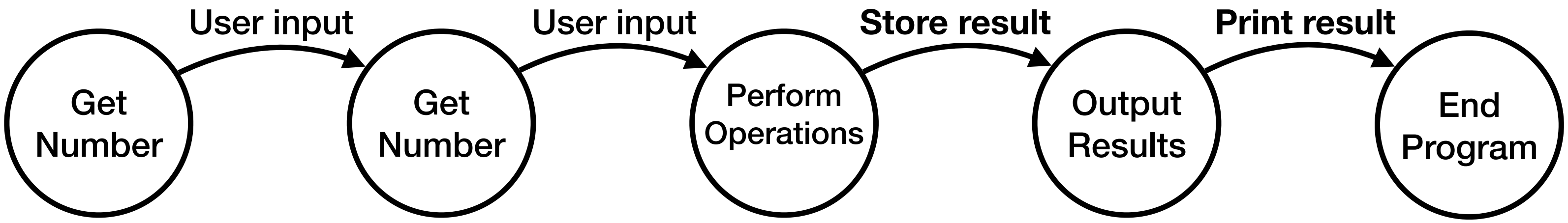
233.75 - 131.5526 = 102.1974

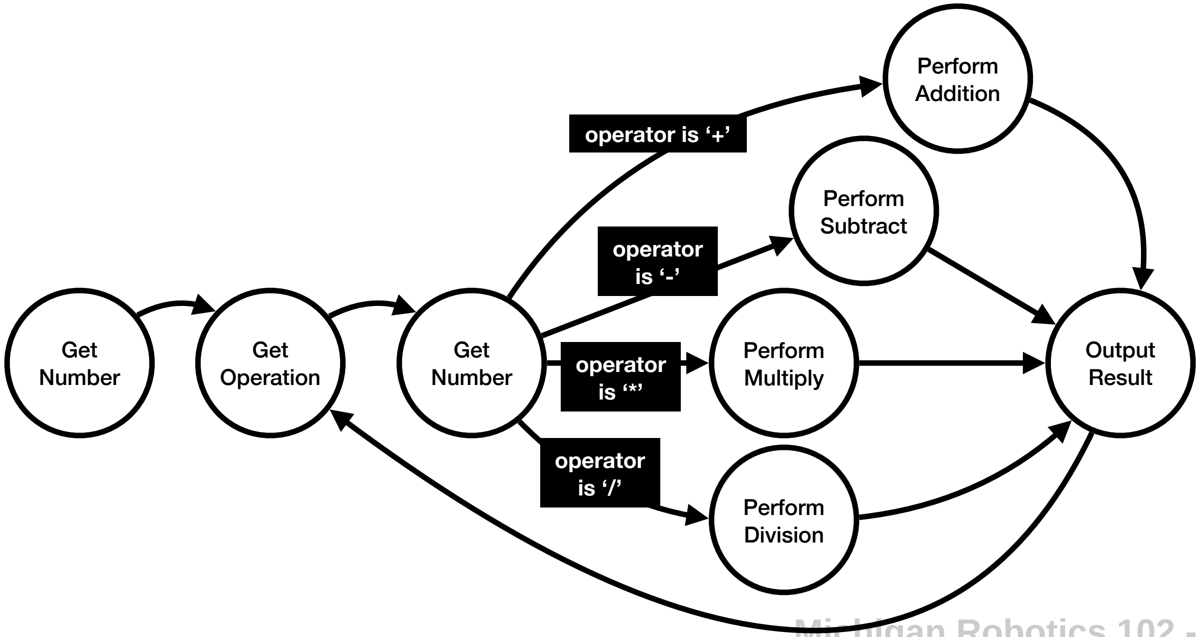
Floating Point Operations

calculator (Version 41)

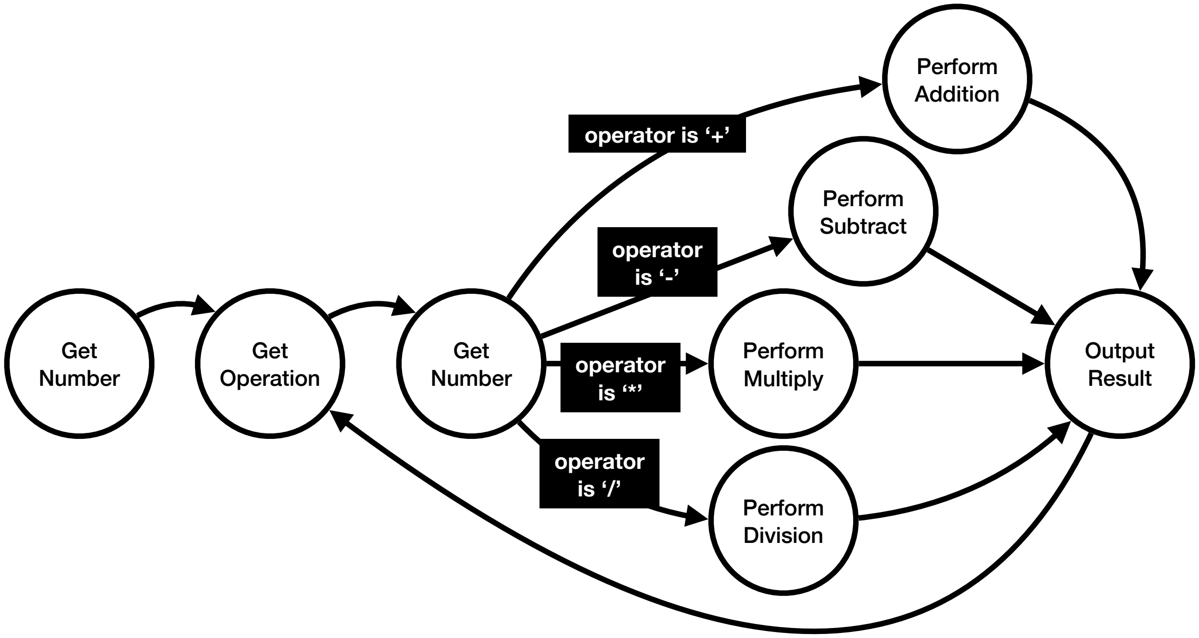


```
Please type a number and
press enter: 22
Please type another number
and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```





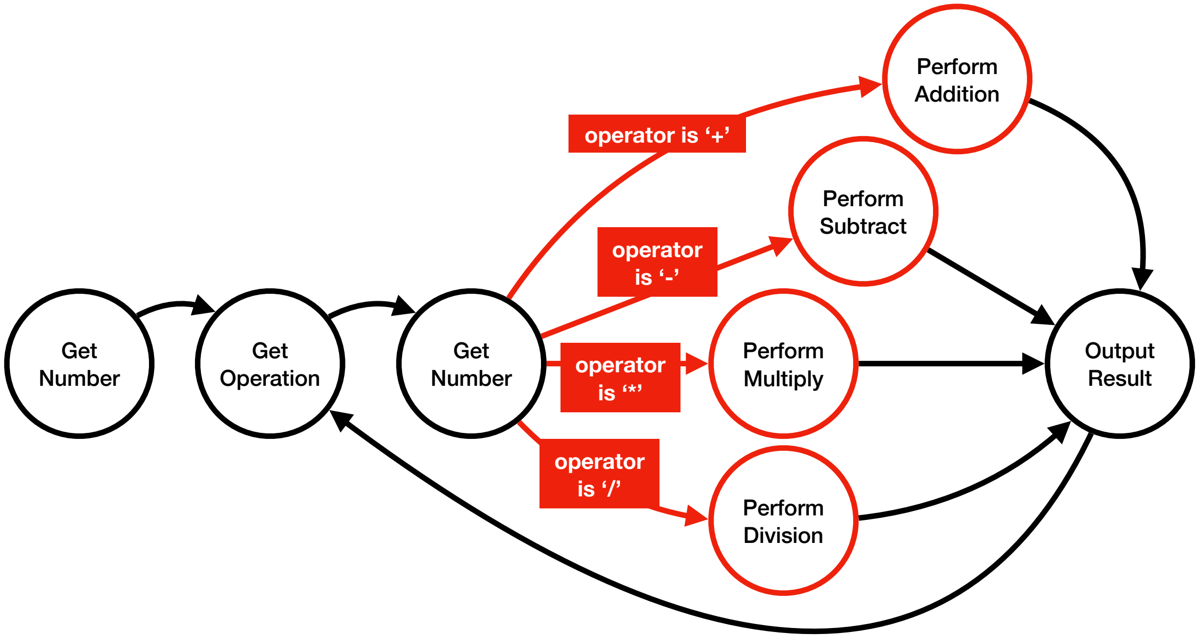
How can we do this in C++ ?



How can we do this in C++ ?

Branching

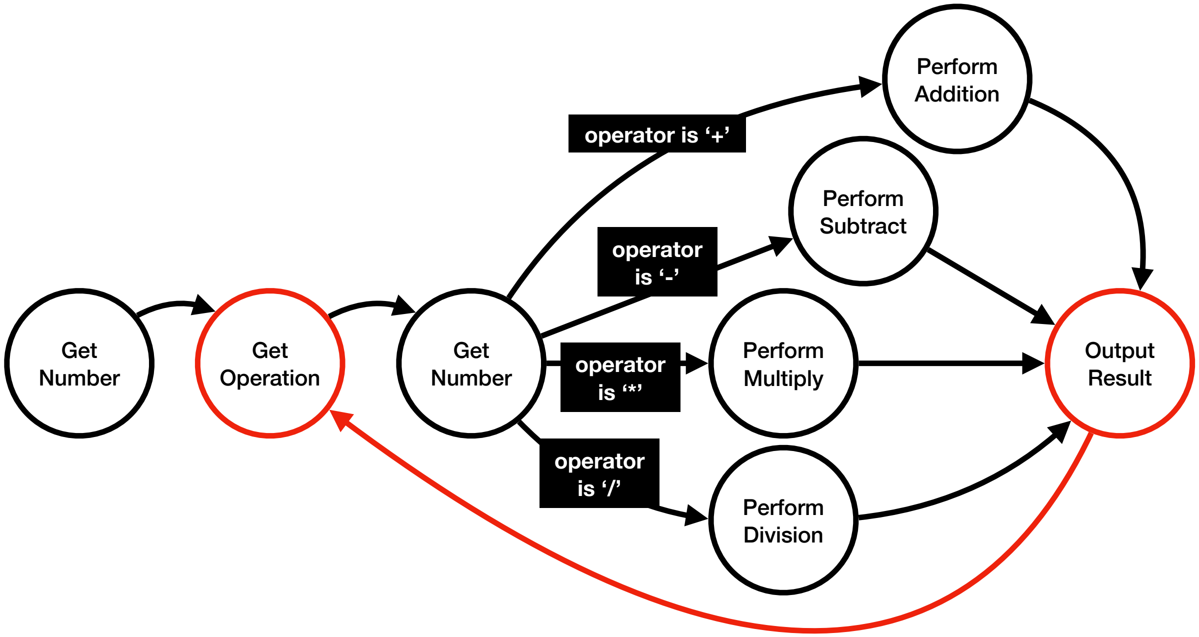
when a program chooses one of two (or more) execution options



How can we do this in C++ ?

Branching

when a program chooses one of two (or more) execution options

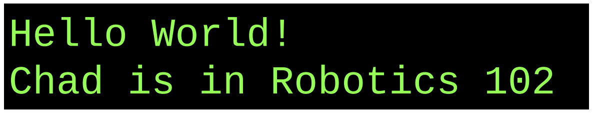


Iteration

when a program repeatedly executes (or loops) a segment of code until a condition is met

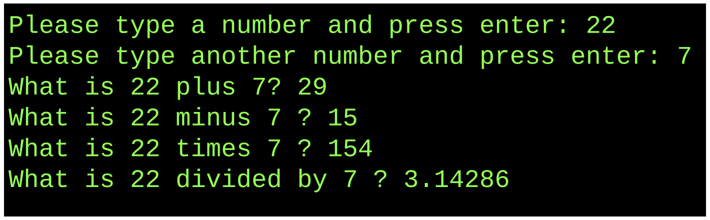
Done

hello



```
Hello World!
Chad is in Robotics 102
```

calculator (Version 24)



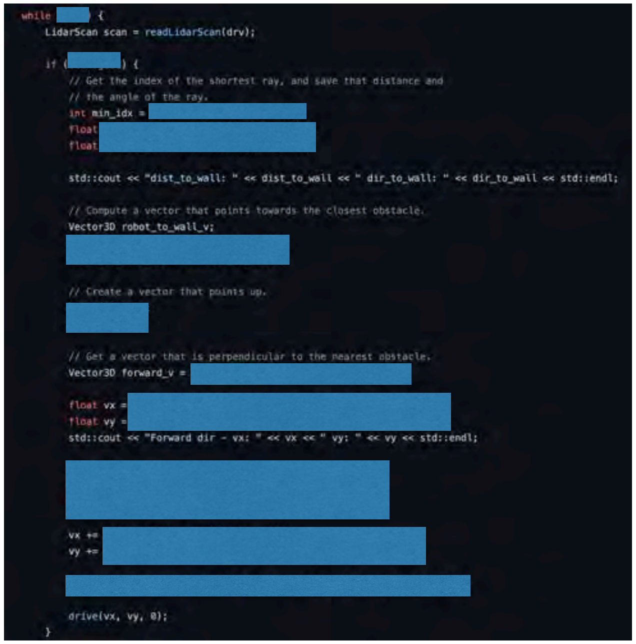
```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7?? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

calculator (Version 41)

- ☑ Program Structure
- ☑ Compile/Execute
- ☑ Operators
- ☑ Data Types
- ☑ Variables
- ☑ User Input/Output
- ☑ Functions
- □ Branching
- □ Iterators
- □ Vectors
- □ Structs
- □ File Input/Output

Coming

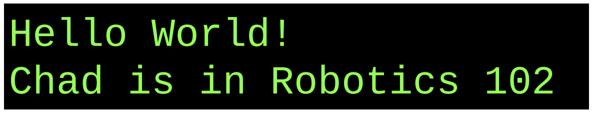
wall_follower.cpp - Project 1



```
while ([redacted]) {
    LidarScan scan = readLidarScan(drv);

    if ([redacted]) {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float

        std::cout << "dist_to_wall: " << dist_to_wall << " dir_to_wall: " << dir_to_wall << std::endl;

        // Compute a vector that points towards the closest obstacle.
        Vector3D robot_to_wall_v;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy =
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy +=

        [redacted]

        drive(vx, vy, 0);
    }
```

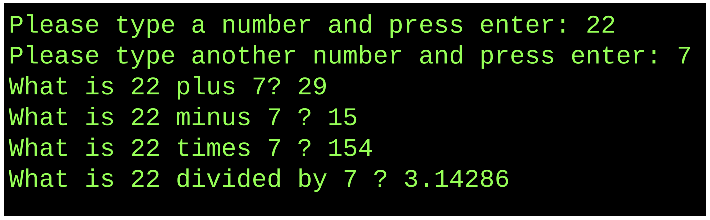
Done

hello



```
Hello World!
Chad is in Robotics 102
```

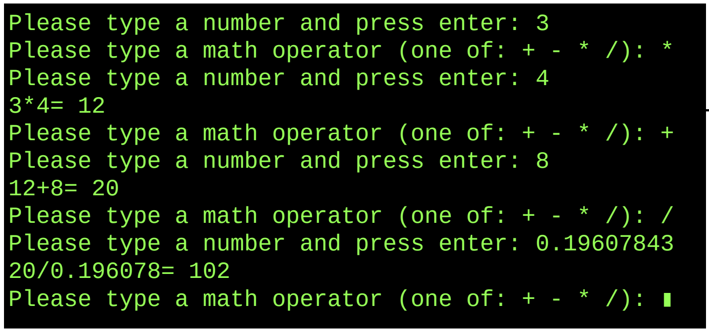
calculator (Version 24)



```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

calculator (Version 41)

This lecture: a working calculator!



```
Please type a number and press enter: 3
Please type a math operator (one of: + - * /): *
Please type a number and press enter: 4
3*4= 12
Please type a math operator (one of: + - * /): +
Please type a number and press enter: 8
12+8= 20
Please type a math operator (one of: + - * /): /
Please type a number and press enter: 0.19607843
20/0.196078= 102
Please type a math operator (one of: + - * /): ▮
```

☑ Program Structure

☑ Compile/Execute

☑ Operators

☑ Data Types

☑ Variables

☑ User Input/Output

☑ Functions

□ Branching

□ Iterators

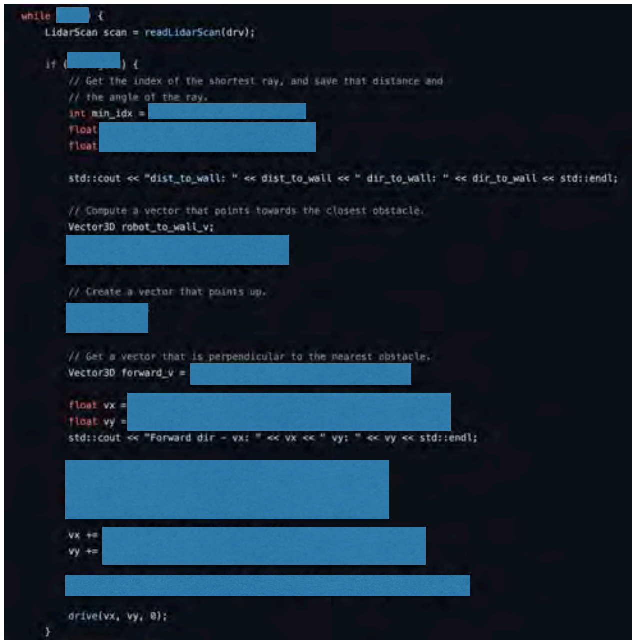
□ Vectors

□ Structs

□ File Input/Output

Coming

wall_follower.cpp - Project 1



```
while [redacted] {
    LidarScan scan = readLidarScan(drv);

    if ([redacted]) {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float

        std::cout << "dist_to_wall: " << dist_to_wall << " dir_to_wall: " << dir_to_wall << std::endl;

        // Compute a vector that points towards the closest obstacle.
        Vector3D robot_to_wall_v;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy =
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy +=

        [redacted]

        drive(vx, vy, 0);
    }
```

Let's write our first artificially intelligent program

Let’s write our first artificially intelligent program

I branch, therefore I am.

I branch, therefore I am.

iThink.cpp (Version 00)

```cpp
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   if (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

```
Therefore, I am.
```

I branch, therefore I am.

iThink.cpp (Version 00)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   if (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

If this variable is greater than 0, execute this statement

```
Therefore, I am.
```

I branch, therefore I am.

iThink.cpp (Version 00 - Branch 00)

```
#include <iostream>

int main()
{
   int thinkingAmount = 0;

   if (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

If this variable is greater than 0, execute this statement

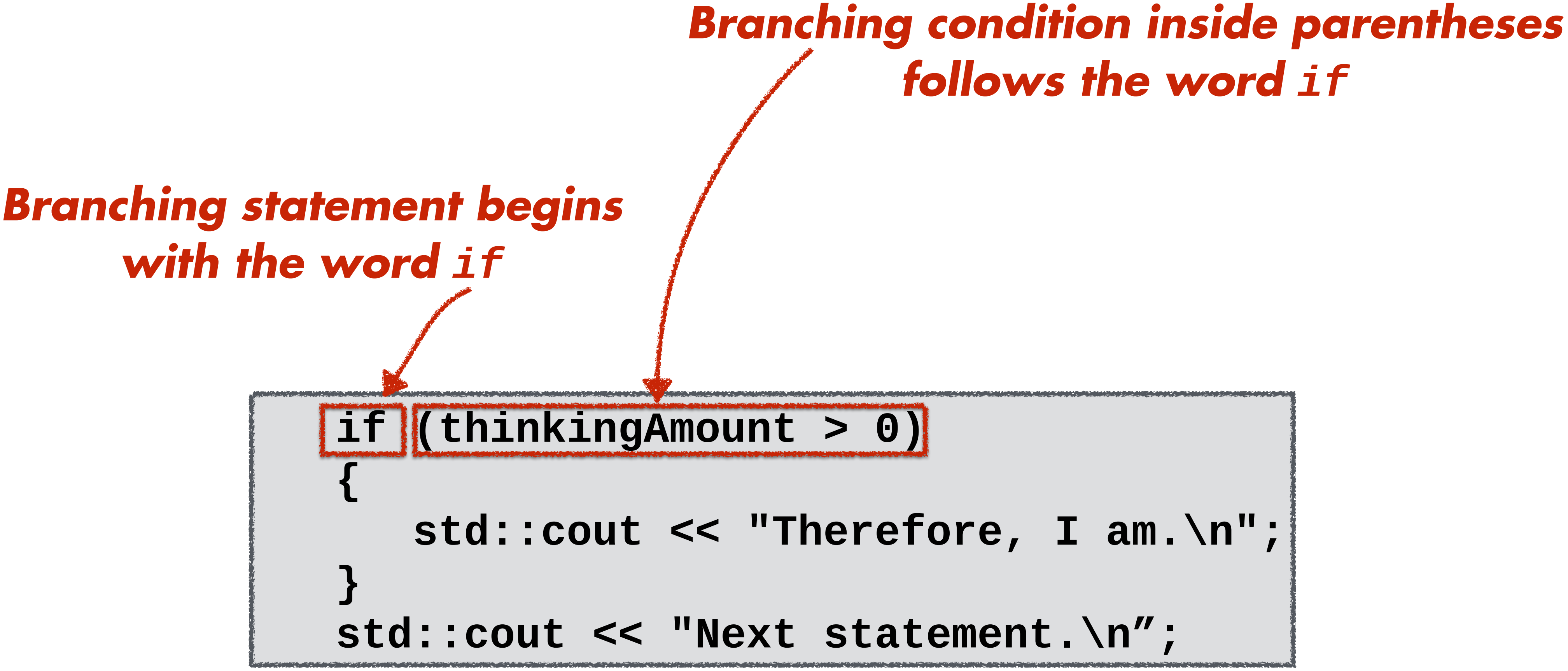
`if` statement performs branching

```
if (thinkingAmount > 0)
{
   std::cout << "Therefore, I am.\n";
}
std::cout << "Next statement.\n”;
```

`if` statement performs branching

Branching statement begins with the word `if`

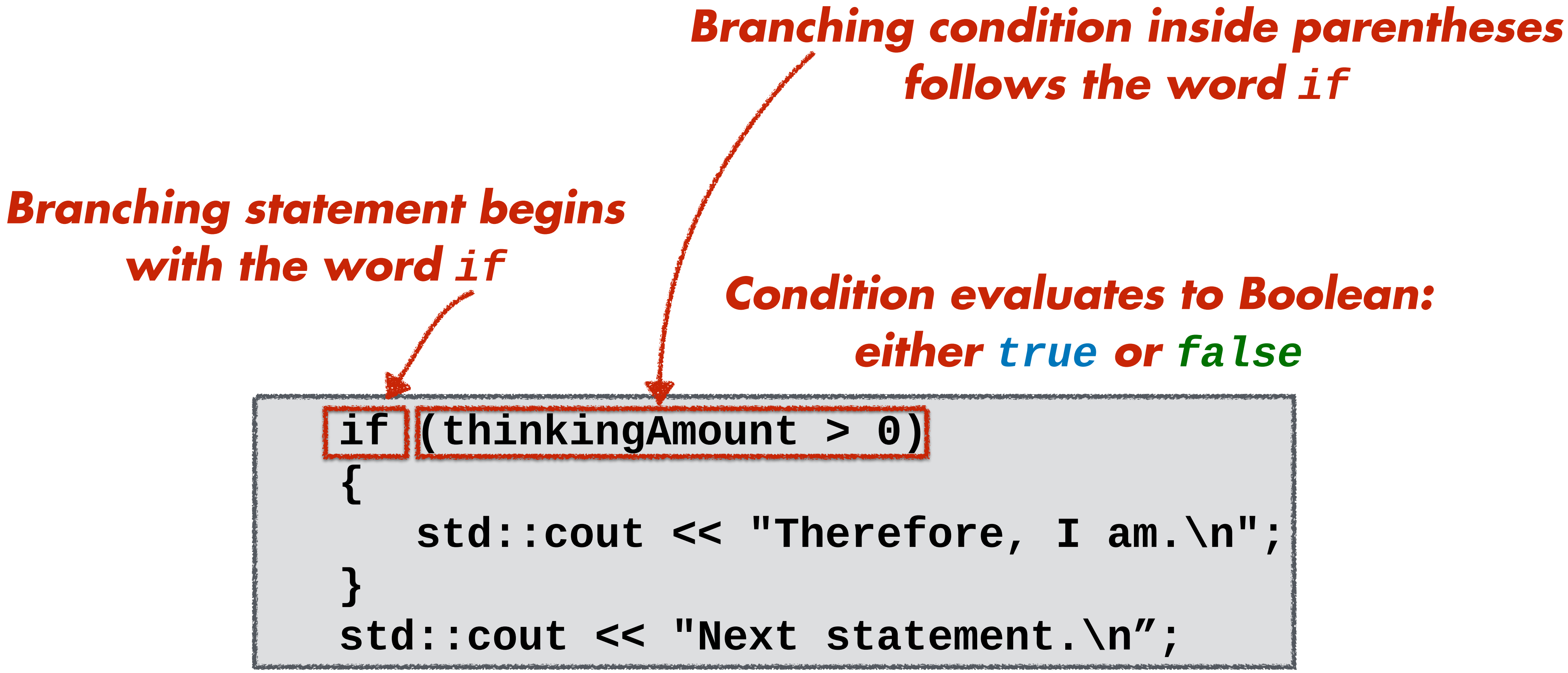
```
   if (thinkingAmount > 0)
   {
      std::cout << "Therefore, I am.\n";
   }
   std::cout << "Next statement.\n”;
```



Branching condition inside parentheses follows the word *`if`*

Branching statement begins with the word *`if`*

```
if (thinkingAmount > 0)
{
   std::cout << "Therefore, I am.\n";
}
std::cout << "Next statement.\n”;
```



Branching condition inside parentheses follows the word `if`

Branching statement begins with the word `if`

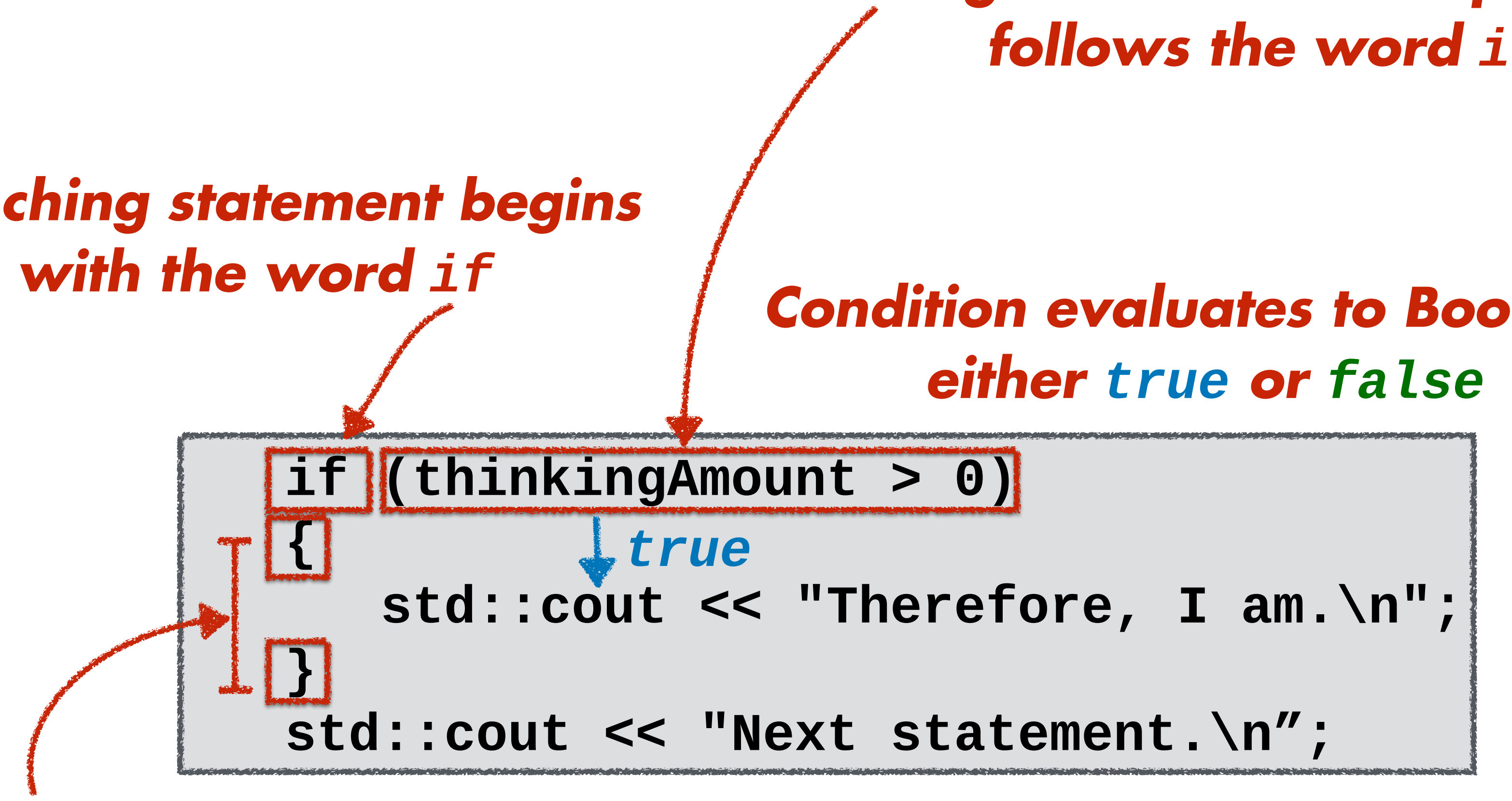
Condition evaluates to Boolean: either `true` or `false`

```
if (thinkingAmount > 0)
{
   std::cout << "Therefore, I am.\n";
}
std::cout << "Next statement.\n”;
```

Branching condition inside parentheses follows the word `if`

Branching statement begins with the word `if`

Condition evaluates to Boolean: either `true` or `false`



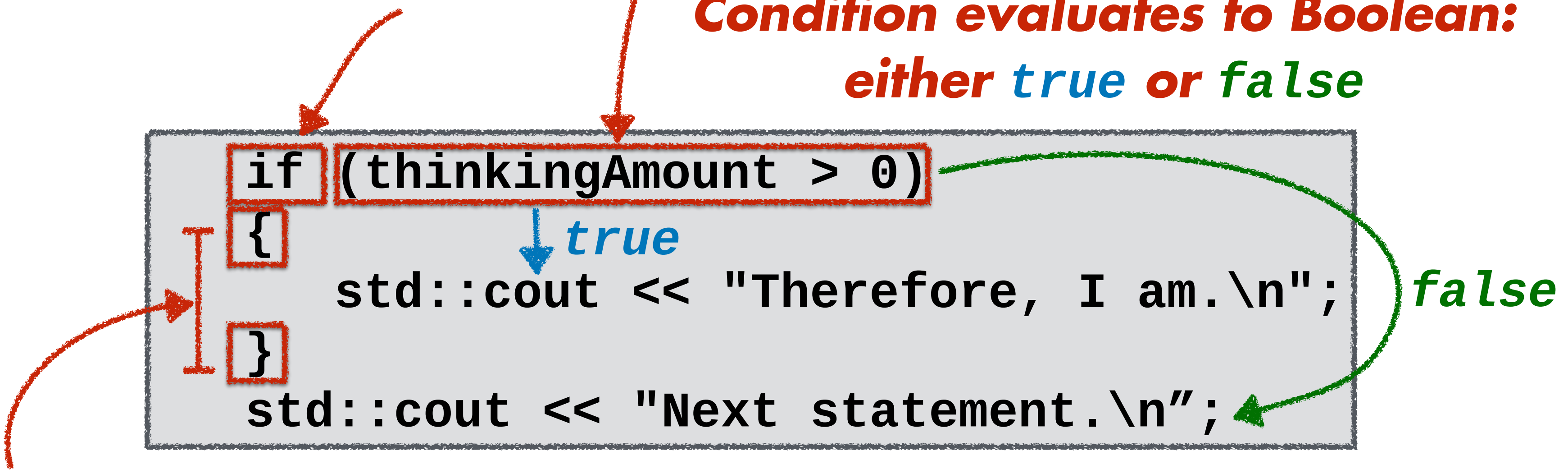
```
  if (thinkingAmount > 0)
  {
      std::cout << "Therefore, I am.\n";
  }
  std::cout << "Next statement.\n”;
```

If the branching condition is `true`, execute the next block of code

Branching condition inside parentheses follows the word `if`

Branching statement begins with the word `if`

Condition evaluates to Boolean: either `true` or `false`

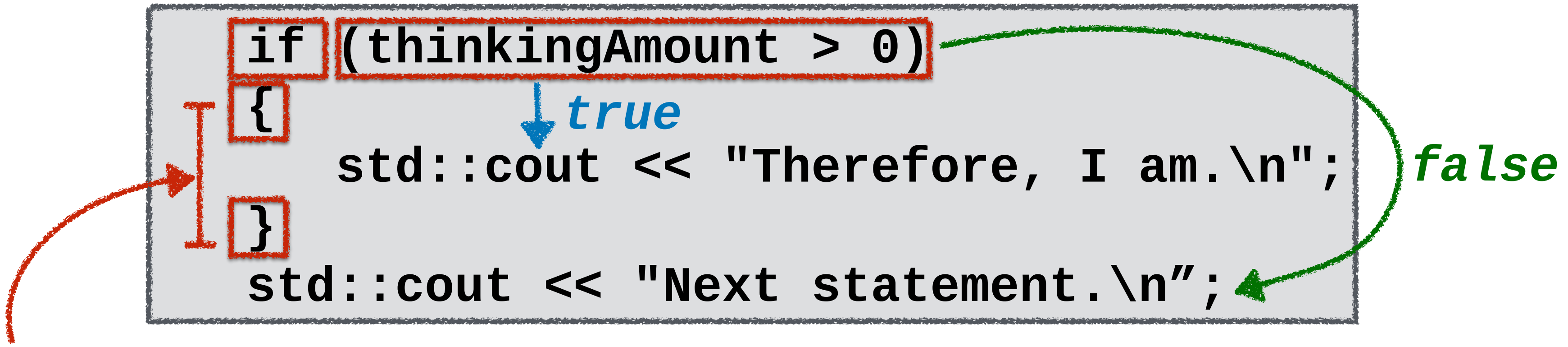


```
  if (thinkingAmount > 0)
  {
      std::cout << "Therefore, I am.\n";
  }
  std::cout << "Next statement.\n”;
```

If the branching condition is `true`, execute the next block of code

If the condition is `false`, skip the next block of code

Condition evaluates to Boolean:
either `true` *or* `false`



```
if (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
std::cout << "Next statement.\n”;
```

If the branching condition is `true`*,*
execute the next block of code

```
Therefore, I am.
Next statement.
```

If the condition is `false`*,*
skip the next block of code

```
Next statement.
```

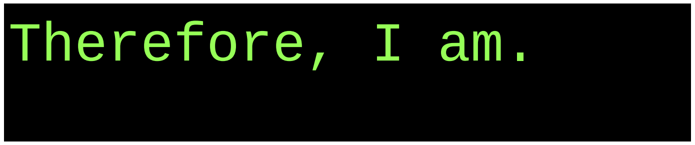
`if else` statements

```
if (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

true

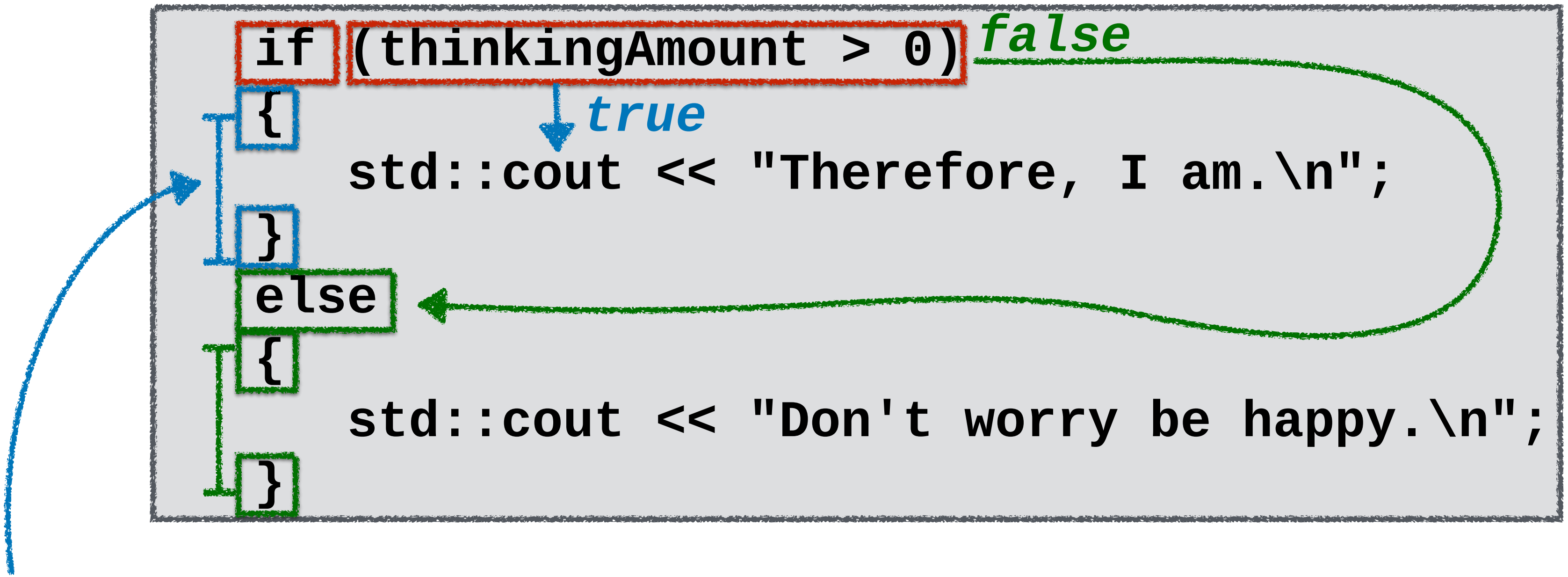
If the branching condition is `true`, execute the next block of code

If the condition is `false`, execute the `else` block of code



```
Therefore, I am.
```

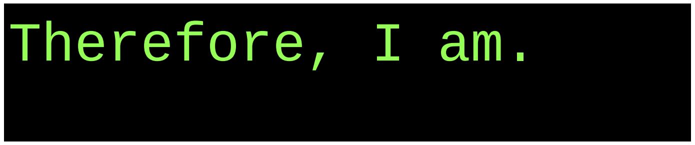
`if else` statements



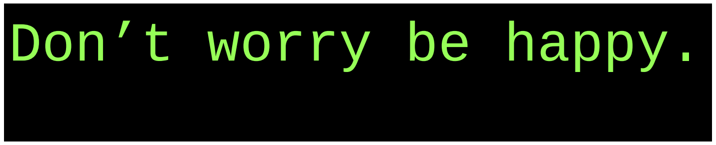
```
if (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

If the branching condition is `true`, execute the next block of code

If the condition is `false`, execute the `else` block of code



```
Therefore, I am.
```



```
Don’t worry be happy.
```

Branching Conditions

Condition evaluates to Boolean:
either `true` or `false`

Branching condition typically involve a comparison of two values

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	**Less than or equal to**	**Greater than**	**Greater than or equal to**	**Equal**	**Not equal**

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

Let's consider some examples

Comparison Operators

C++ uses these operators to compare two values

`<`	`<=`	`>`	`>=`	`==`	`!=`
Less than	**Less than or equal to**	**Greater than**	**Greater than or equal to**	**Equal**	**Not equal**

```
(102 < 101)
```

```
true or false ?
```

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

```
(102 < 101)
```

```
false
```

Comparison Operators

C++ uses these operators to compare two values

`<`	`<=`	`>`	`>=`	`==`	`!=`
Less than	**Less than or equal to**	**Greater than**	**Greater than or equal to**	**Equal**	**Not equal**

```
(102<=101+1)
```

```
true or false ?
```

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	**Less than or equal to**	**Greater than**	**Greater than or equal to**	**Equal**	**Not equal**

```
(102<=101+1)
```

```
true
```

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

```
('c'!='+')
```

```
true or false ?
```

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

Note: comparison operators work with all basic C++ data types

```
('c' != '+')
```

```
true
```

Comparison Operators

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

```
('c'=='+')
```

```
true or false ?
```

Comparison Operators

C++ uses these operators to compare two values

`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal
`!=`	Not equal

```
('c'=='+')
```

```
false
```

A condition with multiple comparisons ?

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

```
( (102 <= 101+1)  OR  ('c' == '+') )

       true                false
```

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

C++ uses these operators for Boolean logic

&&	||	!
Logical AND	Logical OR	Logical NOT

```
((102<=101+1) OR ('c'=='+') )

     true            false
```

Boolean Logic

C++ uses these operators for Boolean logic

&&	||	!
Logical AND	**Logical OR**	**Logical NOT**

```
((102<=101+1) || ('c'=='+'))
     true              false
```

```
true or false ?
```

Boolean Logic

C++ uses these operators for Boolean logic

`&&`	`||`	`!`
Logical AND	Logical OR	Logical NOT

```
( (102 <= 101+1) || ('c' == '+') )
```

`true` or `false` ?

Boolean Logic

C++ uses these operators for Boolean logic

`&&`	`||`	`!`
Logical AND	**Logical OR**	**Logical NOT**

```
((102<=101+1) || ('c'=='+'))
```

```
true
```

Boolean Logic

C++ uses these operators for Boolean logic

`&&` **Logical AND**

`||` **Logical OR**

`!` **Logical NOT**

Logical OR evaluates as `true` ***if either operand is*** `true`

```
( (102 <= 101+1) || ('c' == '+') )
      true                false
                 true
```

Boolean Logic

C++ uses these operators for Boolean logic

&&	||	!
Logical AND	**Logical OR**	**Logical NOT**

***Logical AND evaluates as* `false` *if either operand is* `false`**

```
( (102<=101+1) && ('c'=='+') )
     true                false
```

```
true or false ?
```

Boolean Logic

C++ uses these operators for Boolean logic

`&&`	**`||`**	**`!`**
Logical AND	**Logical OR**	**Logical NOT**

Logical AND evaluates as `false` ***if either operand is*** `false`

```
((102<=101+1) && ('c'=='+'))
     true               false
              false
```

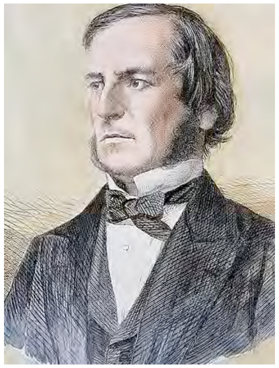
Boolean Logic

Logical AND

OPERAND 1	OPERAND 2	&&
false	false	false
false	true	false
true	false	false
true	true	true

Logical OR

OPERAND 1	OPERAND 2	||
false	false	false
false	true	true
true	false	true
true	true	true



George Boole
(1815-1864)

Logical NOT

OPERAND 1	!
false	true
true	false

C++ uses these operators to compare two values

<	<=	>	>=	==	!=
Less than	Less than or equal to	Greater than	Greater than or equal to	Equal	Not equal

C++ uses these operators for Boolean logic

&&	||	!
Logical AND	Logical OR	Logical NOT

Operators and Precedence

- A subset of C++ operators in order of precedence
  - grouping:

/*	*/	//	(	)
open comment	close comment	comment to end of line	open parenthesis	close parenthesis

  - increment/decrement:

++	- -
increment variable	decrement variable

  - arithmetic:

*	/	%	+	-
multiplication	division	modulus	addition/ concatenation	subtraction

  - comparison:

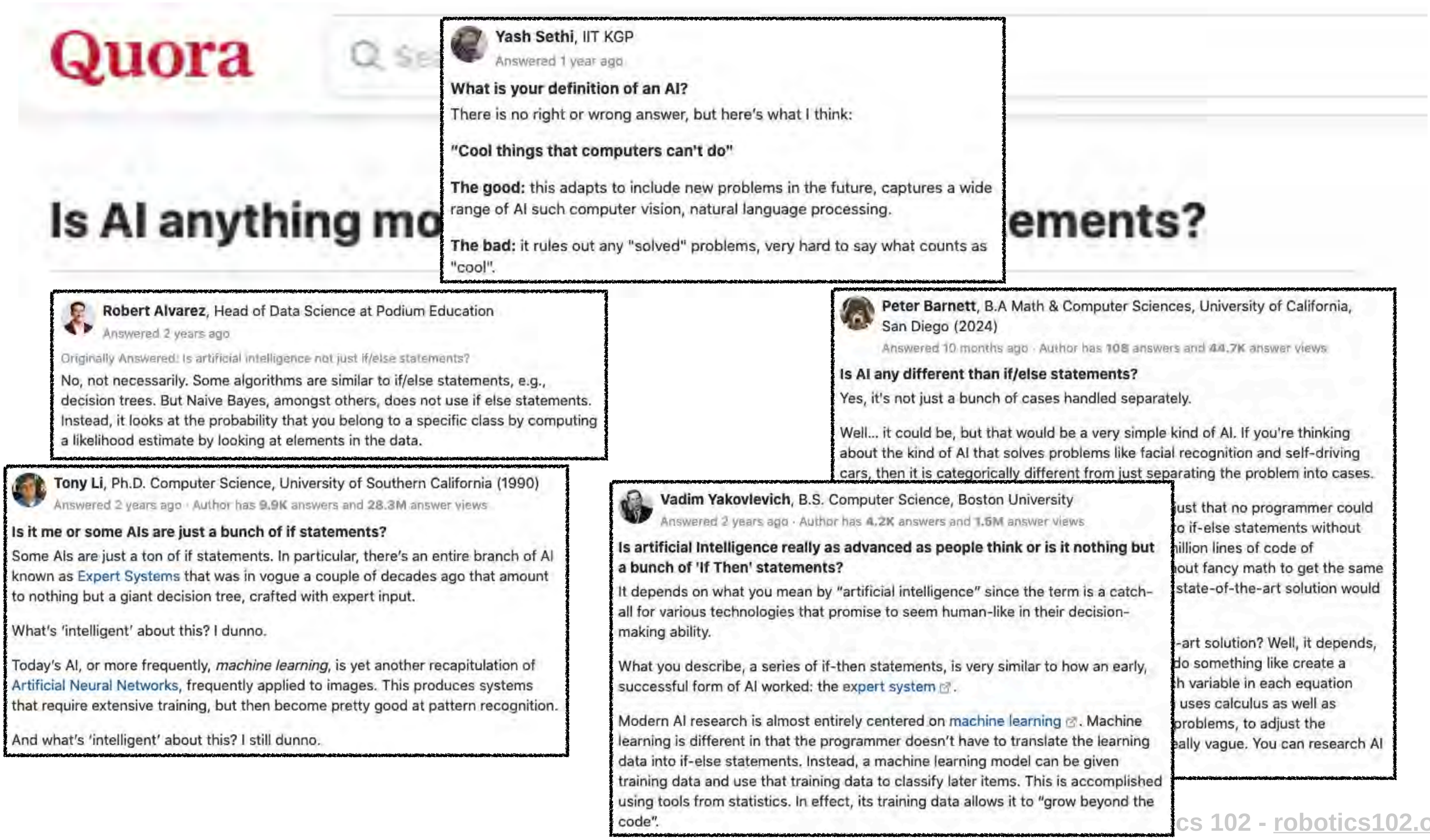
<	<=	>	>=	==	!=	&&	||	!
less than	less than or equal	greater than	greater than or equal	equality	inequality	logical AND	logical OR	logical NOT

  - assignment:

=	+=	*=
assignment	add to variable	multiply to variable

Is an `if` statement really AI ?

Is AI anything more than a bunch of IF statements?



Quora

Is AI anything mo ements?

Yash Sethi, IIT KGP
Answered 1 year ago

What is your definition of an AI?

There is no right or wrong answer, but here's what I think:

"Cool things that computers can't do"

The good: this adapts to include new problems in the future, captures a wide range of AI such computer vision, natural language processing.

The bad: it rules out any "solved" problems, very hard to say what counts as "cool".

Robert Alvarez, Head of Data Science at Podium Education
Answered 2 years ago

Originally Answered: Is artificial intelligence not just if/else statements?

No, not necessarily. Some algorithms are similar to if/else statements, e.g., decision trees. But Naive Bayes, amongst others, does not use if else statements. Instead, it looks at the probability that you belong to a specific class by computing a likelihood estimate by looking at elements in the data.

Peter Barnett, B.A Math & Computer Sciences, University of California, San Diego (2024)
Answered 10 months ago · Author has 108 answers and 44.7K answer views

Is AI any different than if/else statements?

Yes, it's not just a bunch of cases handled separately.

Well... it could be, but that would be a very simple kind of AI. If you're thinking about the kind of AI that solves problems like facial recognition and self-driving cars, then it is categorically different from just separating the problem into cases.

ust that no programmer could
o if-else statements without
illion lines of code of
out fancy math to get the same
state-of-the-art solution would

-art solution? Well, it depends,
do something like create a
h variable in each equation
uses calculus as well as
problems, to adjust the
ally vague. You can research AI

Tony Li, Ph.D. Computer Science, University of Southern California (1990)
Answered 2 years ago · Author has 9.9K answers and 28.3M answer views

Is it me or some AIs are just a bunch of if statements?

Some AIs are just a ton of if statements. In particular, there's an entire branch of AI known as Expert Systems that was in vogue a couple of decades ago that amount to nothing but a giant decision tree, crafted with expert input.

What's 'intelligent' about this? I dunno.

Today's AI, or more frequently, *machine learning*, is yet another recapitulation of Artificial Neural Networks, frequently applied to images. This produces systems that require extensive training, but then become pretty good at pattern recognition.

And what's 'intelligent' about this? I still dunno.

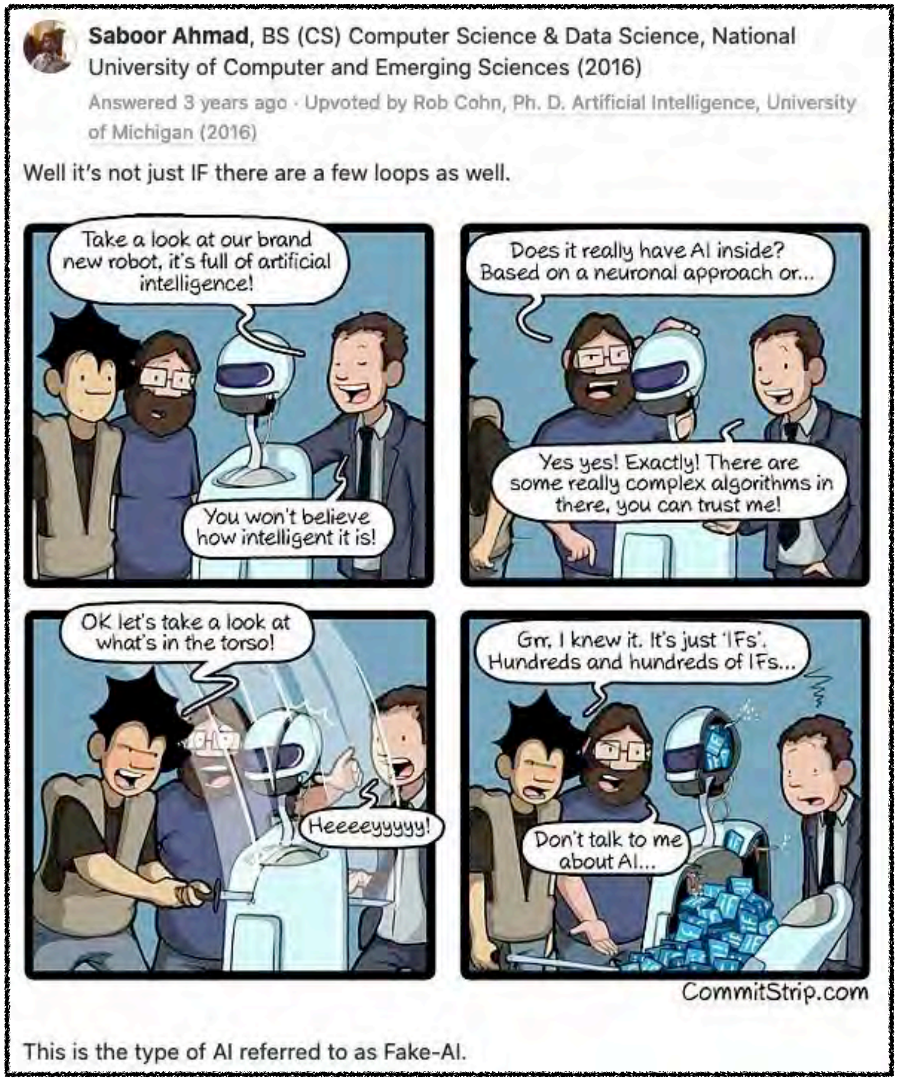
Vadim Yakovlevich, B.S. Computer Science, Boston University
Answered 2 years ago · Author has 4.2K answers and 1.6M answer views

Is artificial Intelligence really as advanced as people think or is it nothing but a bunch of 'If Then' statements?

It depends on what you mean by "artificial intelligence" since the term is a catch-all for various technologies that promise to seem human-like in their decision-making ability.

What you describe, a series of if-then statements, is very similar to how an early, successful form of AI worked: the expert system.

Modern AI research is almost entirely centered on machine learning. Machine learning is different in that the programmer doesn't have to translate the learning data into if-else statements. Instead, a machine learning model can be given training data and use that training data to classify later items. This is accomplished using tools from statistics. In effect, its training data allows it to "grow beyond the code".



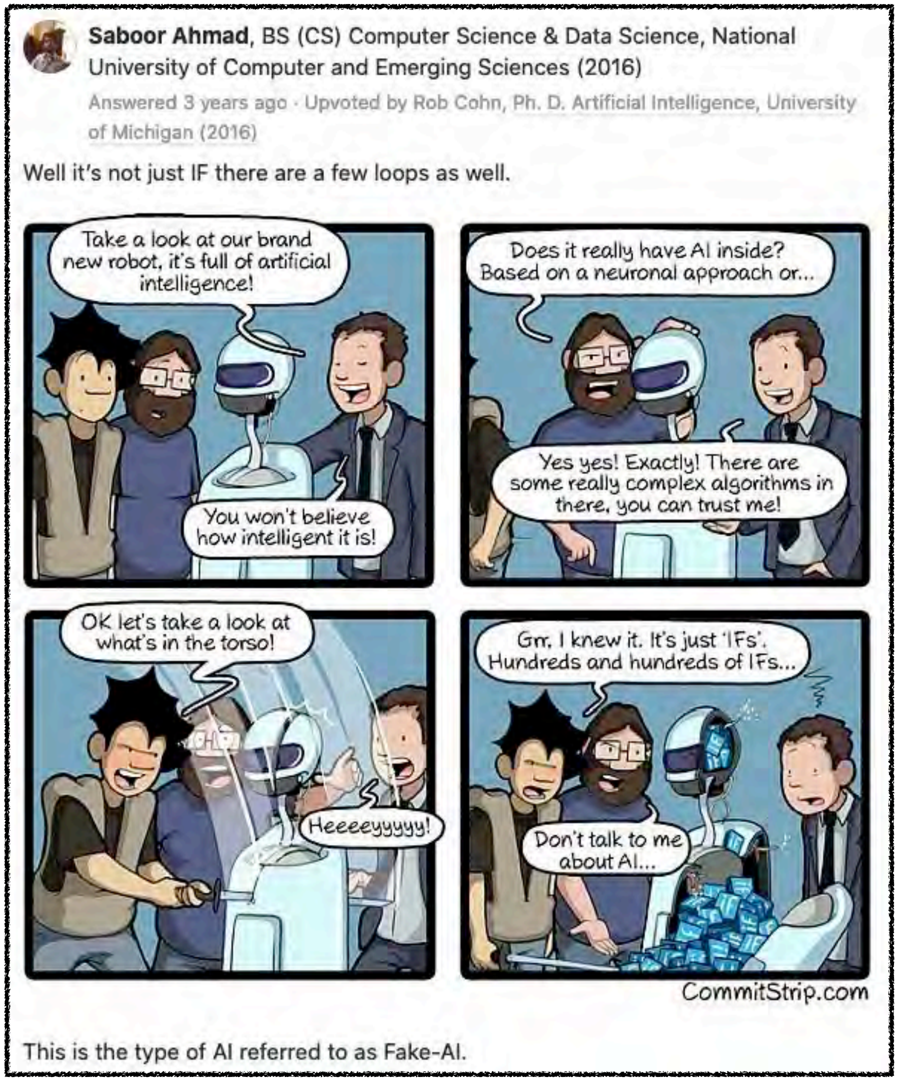
Saboor Ahmad, BS (CS) Computer Science & Data Science, National University of Computer and Emerging Sciences (2016)

Answered 3 years ago · Upvoted by Rob Cohn, Ph. D. Artificial Intelligence, University of Michigan (2016)

Well it's not just IF there are a few loops as well.

Take a look at our brand new robot, it's full of artificial intelligence!
You won't believe how intelligent it is!
Does it really have AI inside? Based on a neuronal approach or...
Yes yes! Exactly! There are some really complex algorithms in there, you can trust me!
OK let's take a look at what's in the torso!
Heeeeyyyyy!
Grr, I knew it. It's just 'IFs'. Hundreds and hundreds of IFs...
Don't talk to me about AI...
CommitStrip.com

This is the type of AI referred to as Fake-AI.



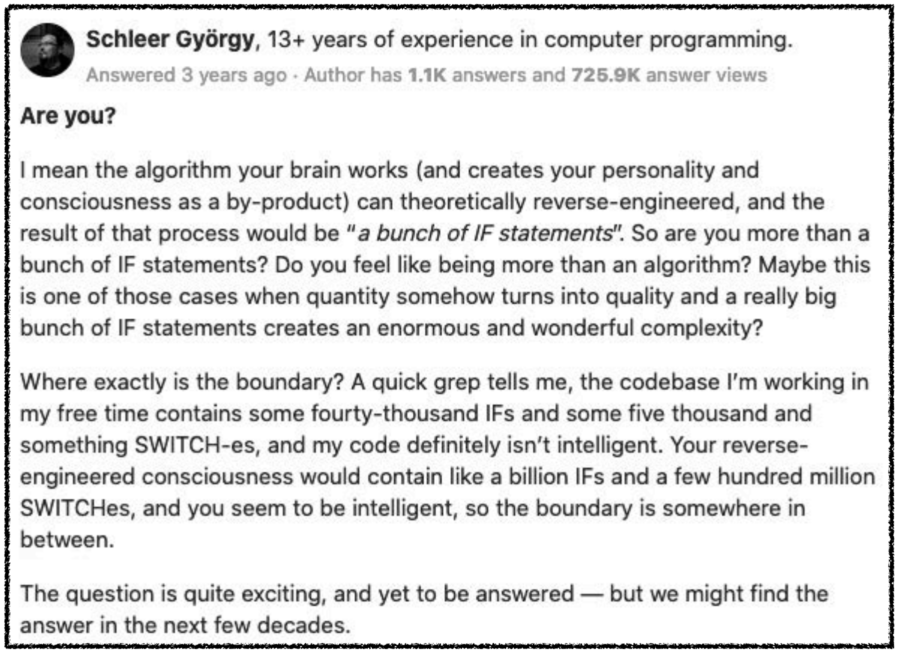
Saboor Ahmad, BS (CS) Computer Science & Data Science, National University of Computer and Emerging Sciences (2016)

Answered 3 years ago · Upvoted by Rob Cohn, Ph. D. Artificial Intelligence, University of Michigan (2016)

Well it's not just IF there are a few loops as well.

Take a look at our brand new robot, it's full of artificial intelligence!
You won't believe how intelligent it is!
Does it really have AI inside? Based on a neuronal approach or...
Yes yes! Exactly! There are some really complex algorithms in there, you can trust me!
OK let's take a look at what's in the torso!
Heeeeyyyyy!
Grr. I knew it. It's just 'IFs'. Hundreds and hundreds of IFs...
Don't talk to me about AI...
CommitStrip.com

This is the type of AI referred to as Fake-AI.



Schleer György, 13+ years of experience in computer programming.

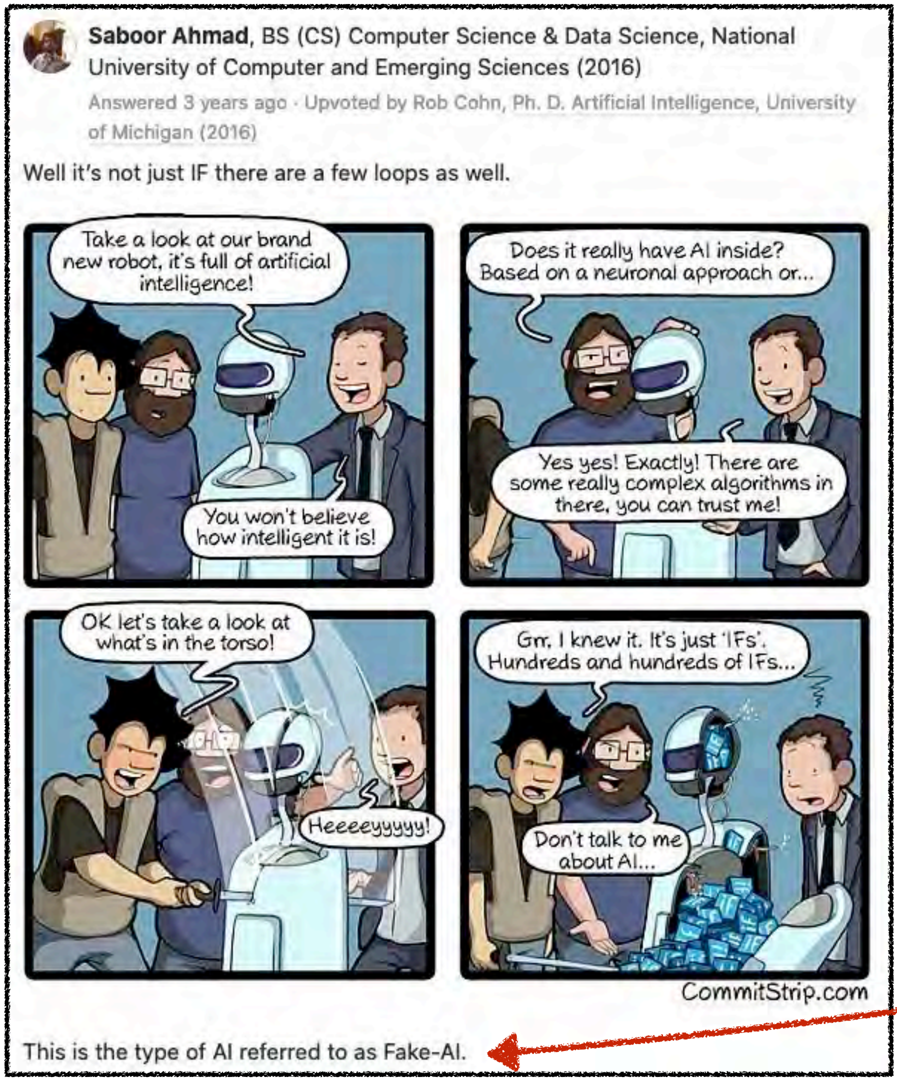
Answered 3 years ago · Author has **1.1K** answers and **725.9K** answer views

Are you?

I mean the algorithm your brain works (and creates your personality and consciousness as a by-product) can theoretically reverse-engineered, and the result of that process would be "*a bunch of IF statements*". So are you more than a bunch of IF statements? Do you feel like being more than an algorithm? Maybe this is one of those cases when quantity somehow turns into quality and a really big bunch of IF statements creates an enormous and wonderful complexity?

Where exactly is the boundary? A quick grep tells me, the codebase I'm working in my free time contains some fourty-thousand IFs and some five thousand and something SWITCH-es, and my code definitely isn't intelligent. Your reverse-engineered consciousness would contain like a billion IFs and a few hundred million SWITCHes, and you seem to be intelligent, so the boundary is somewhere in between.

The question is quite exciting, and yet to be answered — but we might find the answer in the next few decades.



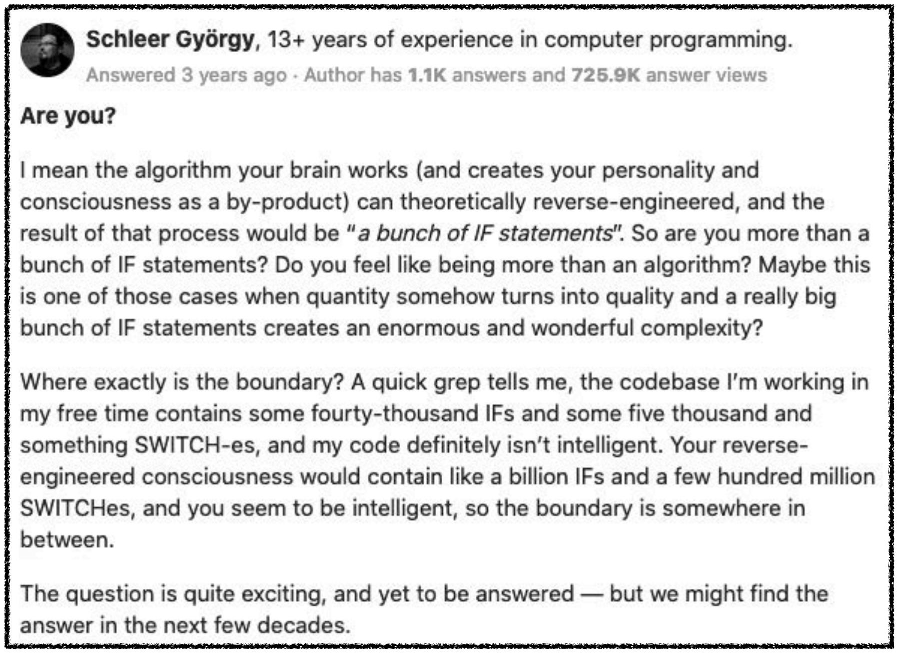
Saboor Ahmad, BS (CS) Computer Science & Data Science, National University of Computer and Emerging Sciences (2016)

Answered 3 years ago · Upvoted by Rob Cohn, Ph. D. Artificial Intelligence, University of Michigan (2016)

Well it's not just IF there are a few loops as well.

Take a look at our brand new robot, it's full of artificial intelligence!
You won't believe how intelligent it is!
Does it really have AI inside? Based on a neuronal approach or...
Yes yes! Exactly! There are some really complex algorithms in there, you can trust me!
OK let's take a look at what's in the torso!
Heeeeyyyyy!
Grr, I knew it. It's just 'IFs'. Hundreds and hundreds of IFs...
Don't talk to me about AI...
CommitStrip.com

This is the type of AI referred to as Fake-AI.



Schleer György, 13+ years of experience in computer programming.

Answered 3 years ago · Author has **1.1K** answers and **725.9K** answer views

Are you?

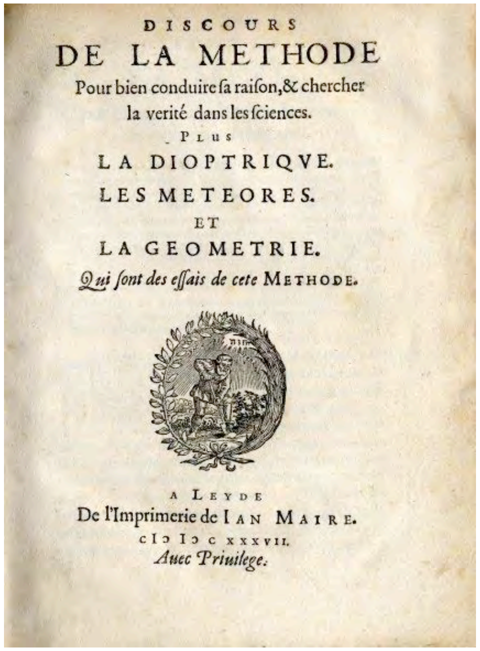
I mean the algorithm your brain works (and creates your personality and consciousness as a by-product) can theoretically reverse-engineered, and the result of that process would be "*a bunch of IF statements*". So are you more than a bunch of IF statements? Do you feel like being more than an algorithm? Maybe this is one of those cases when quantity somehow turns into quality and a really big bunch of IF statements creates an enormous and wonderful complexity?

Where exactly is the boundary? A quick grep tells me, the codebase I'm working in my free time contains some fourty-thousand IFs and some five thousand and something SWITCH-es, and my code definitely isn't intelligent. Your reverse-engineered consciousness would contain like a billion IFs and a few hundred million SWITCHes, and you seem to be intelligent, so the boundary is somewhere in between.

The question is quite exciting, and yet to be answered — but we might find the answer in the next few decades.

Can you define AI without defining intelligence?

René Descartes
(1596-1650)



DISCOURS
DE LA METHODE
Pour bien conduire ſa raiſon, & chercher
la verité dans les ſciences.
PLUS
LA DIOPTRIQVE.
LES METEORES.
ET
LA GEOMETRIE.
Qui ſont des eſſais de cete METHODE.

A LEYDE
De l'Imprimerie de IAN MAIRE.
cIɔ Iɔ c xxxvii.
Auec Priuilege.

Discourse on the Method (1637)

"*Je pense, donc je suis*"
("*I think, therefore I am*")

René Descartes
(1596-1650)

"*Je pense, donc je suis*"
("*I think, therefore I am*")

Antoine Léonard Thomas
(1732-1785)

"*Puisque je doute, je pense; puisque je pense, j'existe*"
("*I doubt, therefore I think, therefore I am*")

"*Puisque je doute, je pense; puisque je pense, j'existe*"
("*I doubt, therefore I think, therefore I am*")

I doubt, therefore I think, therefore I am

I doubt, therefore I think, therefore I am

```
if (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

This is not quite right

I doubt, therefore I think, therefore I am

```
if (  ???  (thinkingAmount > 0))
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

I doubt, therefore I think, therefore I am

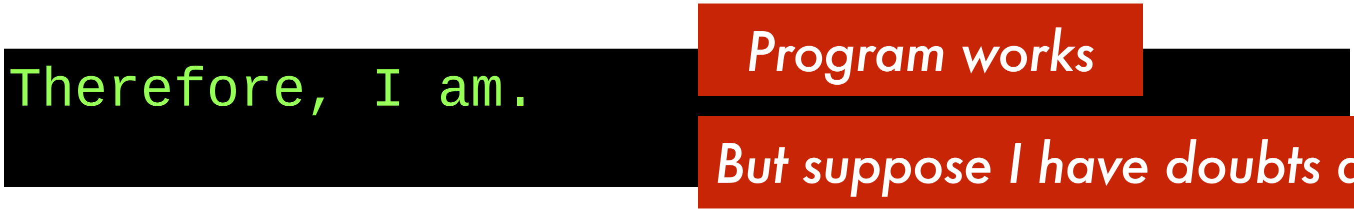
```
if ((doubt ???) ??? (thinkingAmount > 0))
{
   std::cout << "Therefore, I am.\n";
}
else
{
   std::cout << "Don't worry be happy.\n";
}
```

I doubt, therefore I think, therefore I am

```
if ((doubt > 0) ??? (thinkingAmount > 0))
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

I doubt, therefore I think, therefore I am

```
if ((doubt > 0) && (thinkingAmount > 0))
{
    std::cout << "Therefore, I am.\n";
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```



```
Therefore, I am.
```

Program works

But suppose I have doubts and still exist ?

else if statement

```
if (doubt > 0)
{
    std::cout << "Therefore, I am.\n";
}
else if (thinkingAmount > 0)
{
    std::cout << “I am someone without doubts.\n”;
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

else if statement

```
if (doubt > 0)
{
    std::cout << "Therefore, I am.\n";
}
else if (thinkingAmount > 0)
{
    std::cout << “I am someone without doubts.\n”;
}
else
{
    std::cout << "Don't worry be happy.\n";
}
```

`else if` ***statement can branch on another condition***

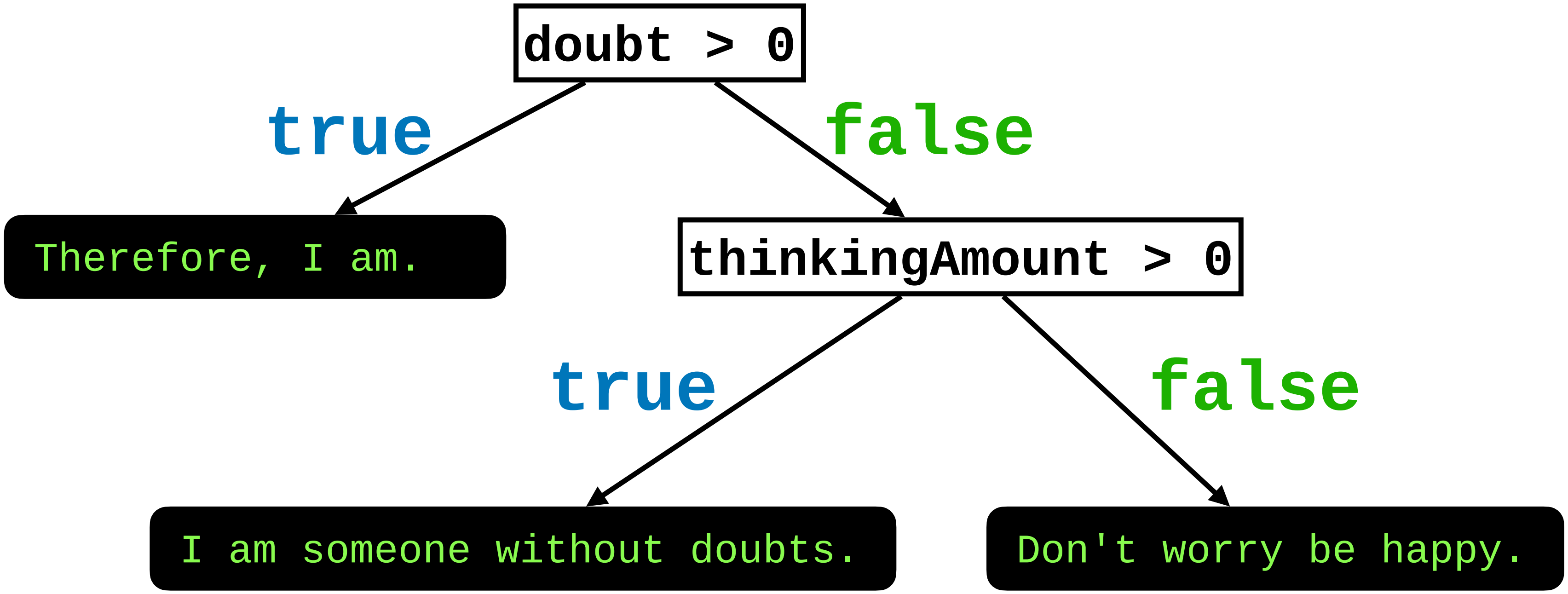
else if statement

```
if (doubt > 0)
{
   std::cout << "Therefore, I am.\n";
}
else if (thinkingAmount > 0)
{
   std::cout << “I am someone without doubts.\n”;
}
else
{
   std::cout << "Don't worry be happy.\n";
}
```

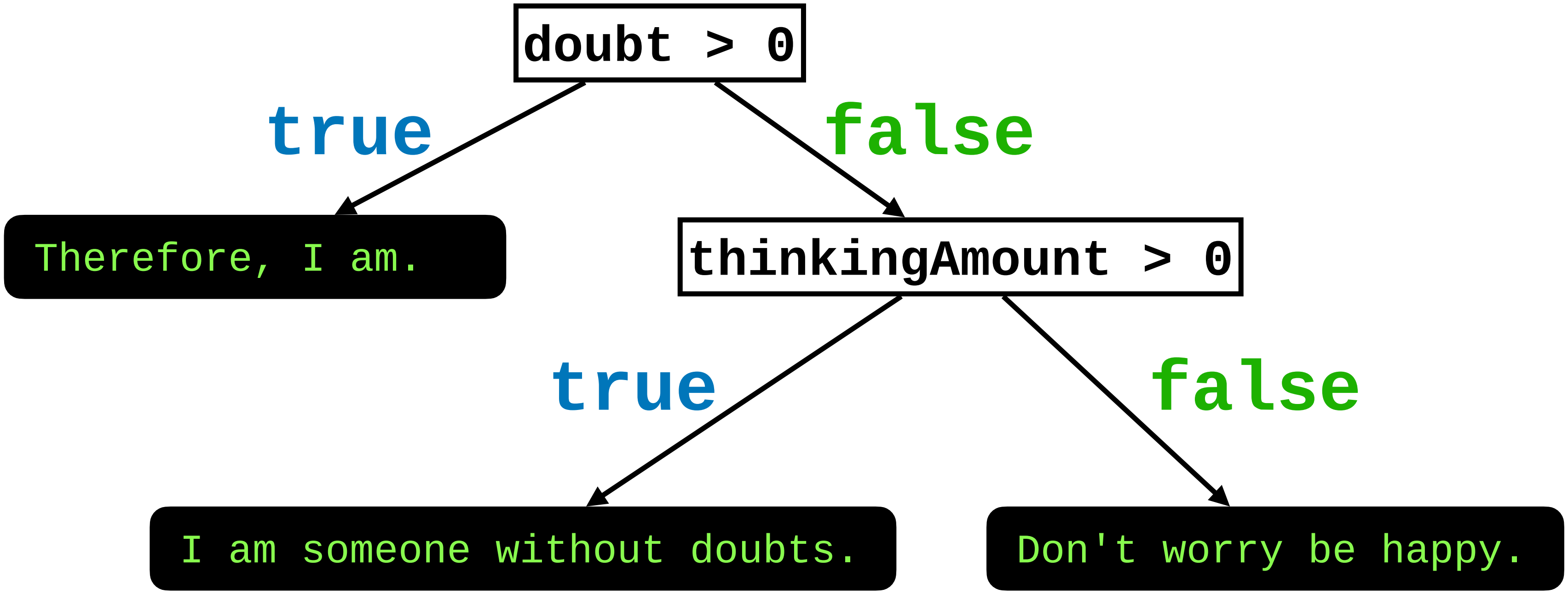
else if statement

```
if (doubt > 0)
{
    std::cout << "T Therefore, I am. n";
}
else if (thinkingAmount > 0)
{
    std::cout << “I I am someone without doubts. ts.\n”;
}
else
{
    std::cout << "I Don't worry be happy. y.\n";
}
```

Decision Tree



Decision Tree



`doubt > 0`

true

`Therefore, I am.`

false

`thinkingAmount > 0`

true

`I am someone without doubts.`

false

`Don't worry be happy.`

We can program this another way

Nested `if` statements

```
if (doubt > 0)
{
    std::cout << "Therefore, I am.\n";
}
else
    if (thinkingAmount > 0) {
        std::cout << "I am someone without doubts.\n";
    }
    else {
        std::cout << "Don't worry be happy.\n";
    }
}
```

`if` statements can branch inside of another `if` or `else` statement

Nested `if` statements

```
if (doubt > 0)
{
   std::cout << "Therefore, I am.\n";
}
else
   if (thinkingAmount > 0) {
      std::cout << "I am someone without doubts.\n";
   }
   else {
      std::cout << "Don't worry be happy.\n";
   }
}
```

`if` ***statements can branch inside of another*** `if` ***or*** `else` ***statement***

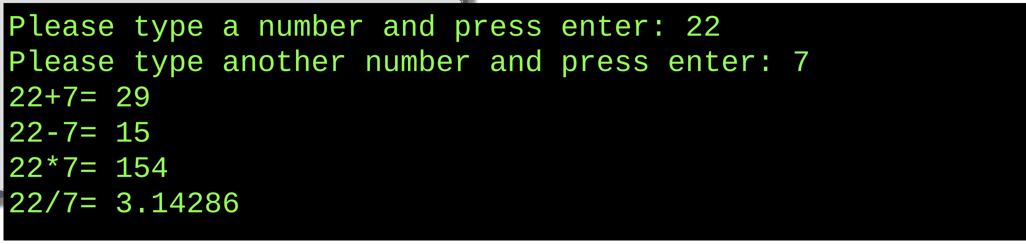
Pocket calculators branch for operations

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
   // Let's declare our variables
   float myNumber, myOtherNumber;  // Calculation operands
   float sumNumber, differenceNumber, productNumber, quotientNumber;

   // Ask the user for the first operand
   getNumber(myNumber);

   // Ask the user for the second operand
   getNumber(myOtherNumber);

   // Perform all operations and store results in variables
   performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

   // Output operation results to screen
   outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

  return 0;
}
```

Functions

```
    addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
  divideTwoNumbers()
         getNumber()
 performOperations()
     outputResults()
              main()
```



```
Please type a number and press enter: 22
Please type another number and press enter: 7
22+7= 29
22-7= 15
22*7= 154
22/7= 3.14286
```

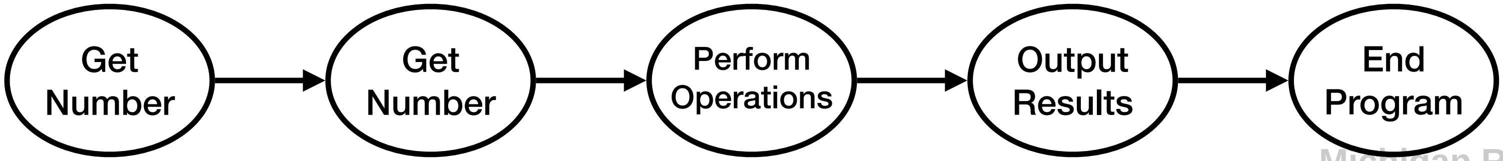
Our previous calculator did not consider which operation to perform

calculator.cpp (Version 41)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
   // Let's declare our variables
   float myNumber, myOtherNumber;  // Calculation operands
   float sumNumber, differenceNumber, productNumber, quotientNumber;

   // Ask the user for the first operand
   getNumber(myNumber);

   // Ask the user for the second operand
   getNumber(myOtherNumber);

   // Perform all operations and store results in variables
   performOperations(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

   // Output operation results to screen
   outputResults(myNumber,myOtherNumber,sumNumber,differenceNumber,
      productNumber,quotientNumber);

  return 0;
}
```

Functions

addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
performOperations()
outputResults()
main()



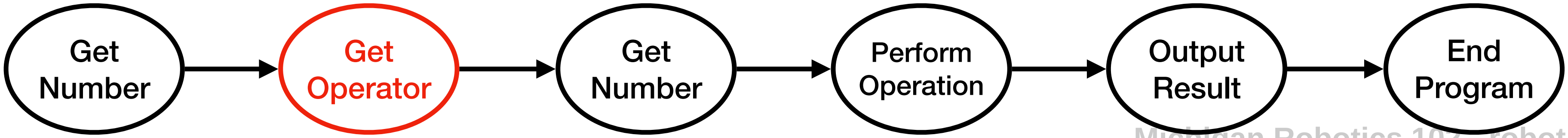
calculator.cpp (Version 46)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
   // Let's declare our variables
   float myNumber, myOtherNumber;  // Calculation operands
   float resultNumber;  // Calculation result
   char myOperator;  // Character for the operation to perform

   // Ask the user for the first operand
   getNumber(myNumber);

   // Ask the user for the operator to perform
   getOperator(myOperator);

   // Ask the user for the second operand
   getNumber(myOtherNumber);

   // Perform single operation and store result in variable
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);

   // Output operation result to screen
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);

  return 0;
}
```

Functions

addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
getOperation()
performOperation()
outputResult()
main()

Let's get an operator for branching

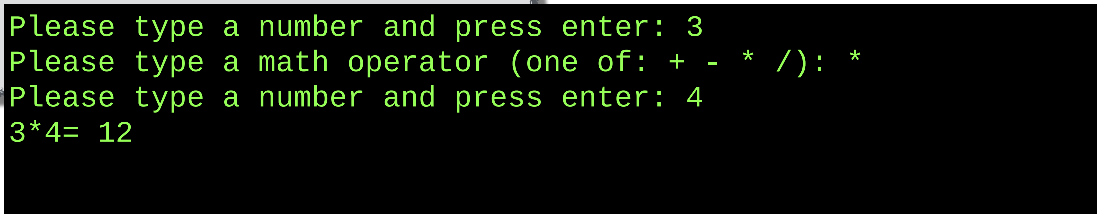


calculator.cpp (Version 46)

```
// Main function declaration, returns 0 if no errors encountered
int main()
{
   // Let's declare our variables
   float myNumber, myOtherNumber;  // Calculation operands
   float resultNumber;  // Calculation result
   char myOperator;  // Character for the operation to perform

   // Ask the user for the first operand
   getNumber(myNumber);

   // Ask the user for the operator to perform
   getOperator(myOperator);

   // Ask the user for the second operand
   getNumber(myOtherNumber);

   // Perform single operation and store result in variable
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);

   // Output operation result to screen
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);

  return 0;
}
```

Functions

addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
getOperation()
performOperation()
outputResult()
main()



```
Please type a number and press enter: 3
Please type a math operator (one of: + - * /): *
Please type a number and press enter: 4
3*4= 12
```

The correct operation is performed

calculator.cpp (Version 46)

```
// Function defined to perform specified operation on operands
bool performOperation( What function arguments are needed ? ) {

   // Compute result for only the specified operation
   If operation is addition, provide sum as result

   If operation is subtraction, provide difference as result

   If operation is multiplication, provide product as result

   If operation is division, provide quotient as result

   return false;  // return false if no errors
}
```

Functions

addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
divideTwoNumbers()
getNumber()
getOperation()
performOperation()
outputResult()
main()

main()

```
   // Perform single operation and store result in variable
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
```

calculator.cpp (Version 46)

```
bool divideTwoNumbers(float operand1, float operand2, float &quotient) {
   quotient = operand1 / operand2;
   return false;
```

*If divide by zero attempted here,
print a error message and exit the program*

```
exit(-1);  // this function call will terminate the program immediately
```

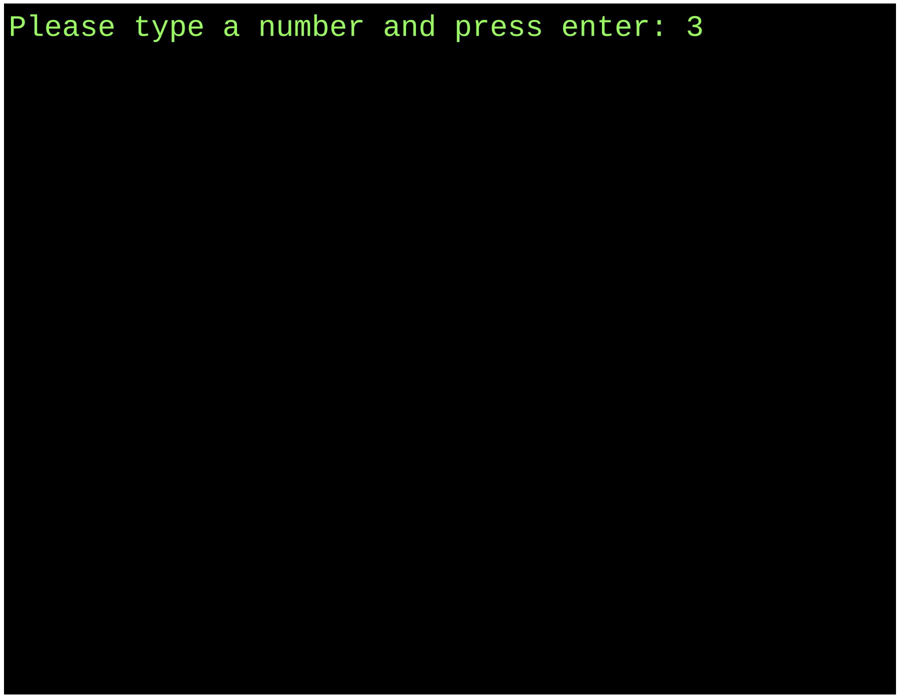
Functions

```
     addTwoNumbers()
subtractTwoNumbers()
multiplyTwoNumbers()
  divideTwoNumbers()
         getNumber()
      getOperation()
  performOperation()
      outputResult()
              main()
```

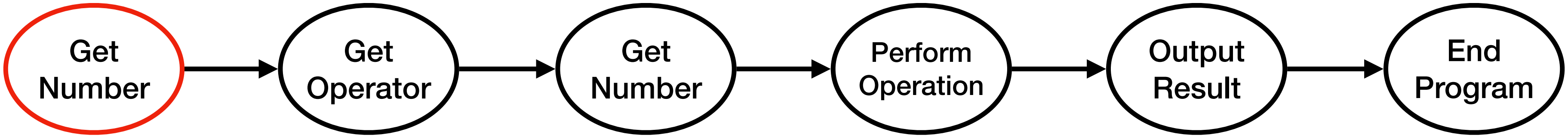
When should our calculator stop ?

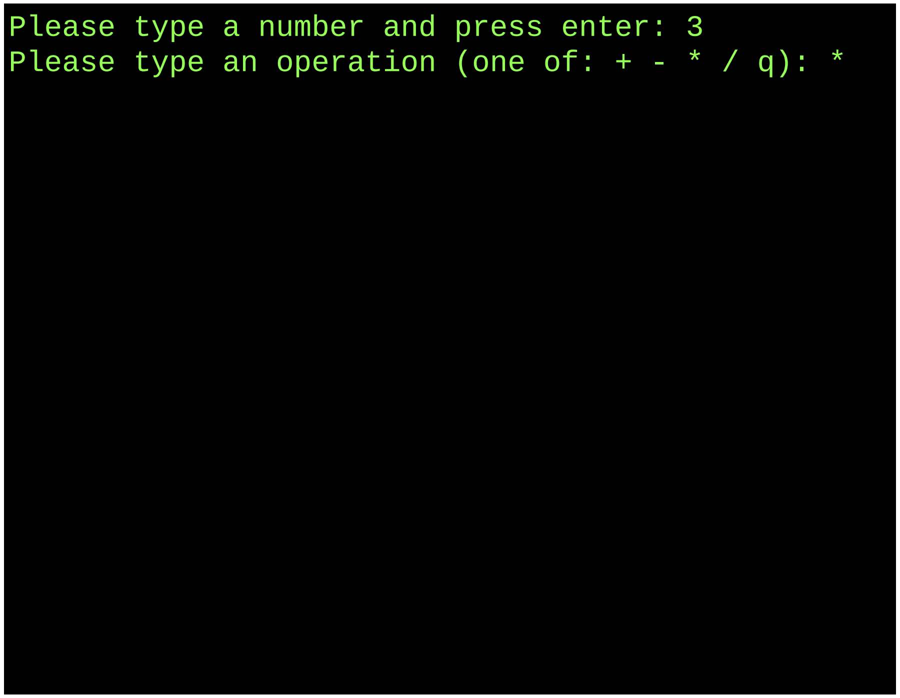
When should our calculator stop ?

When we tell it to

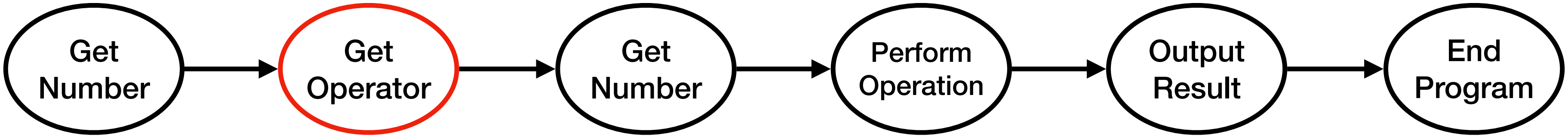


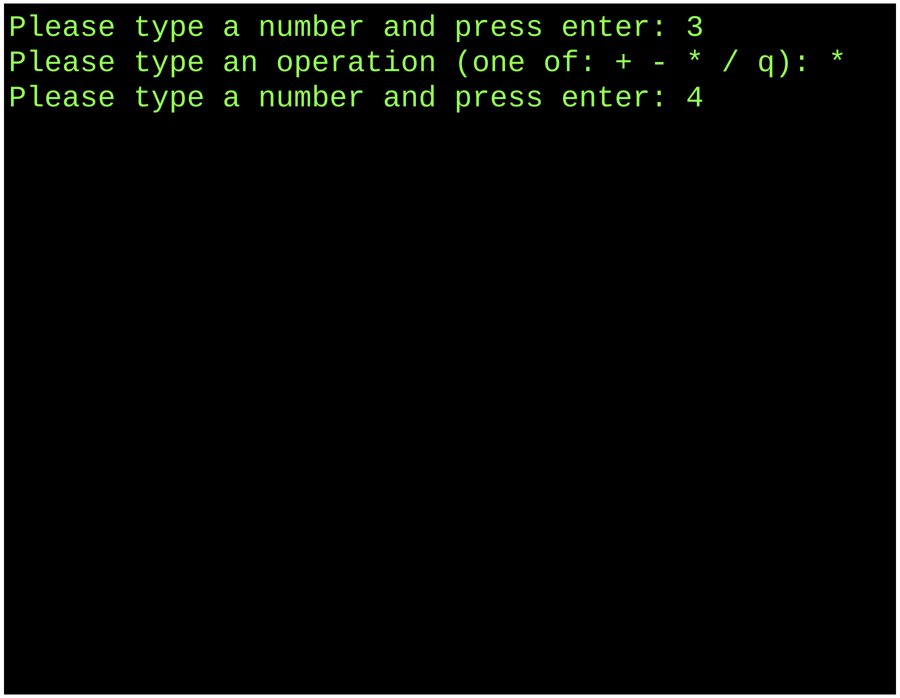
```
Please type a number and press enter: 3
```



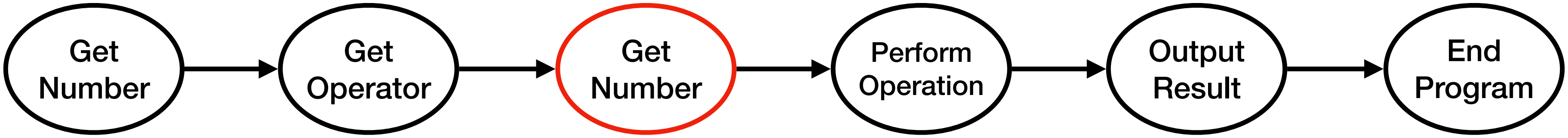


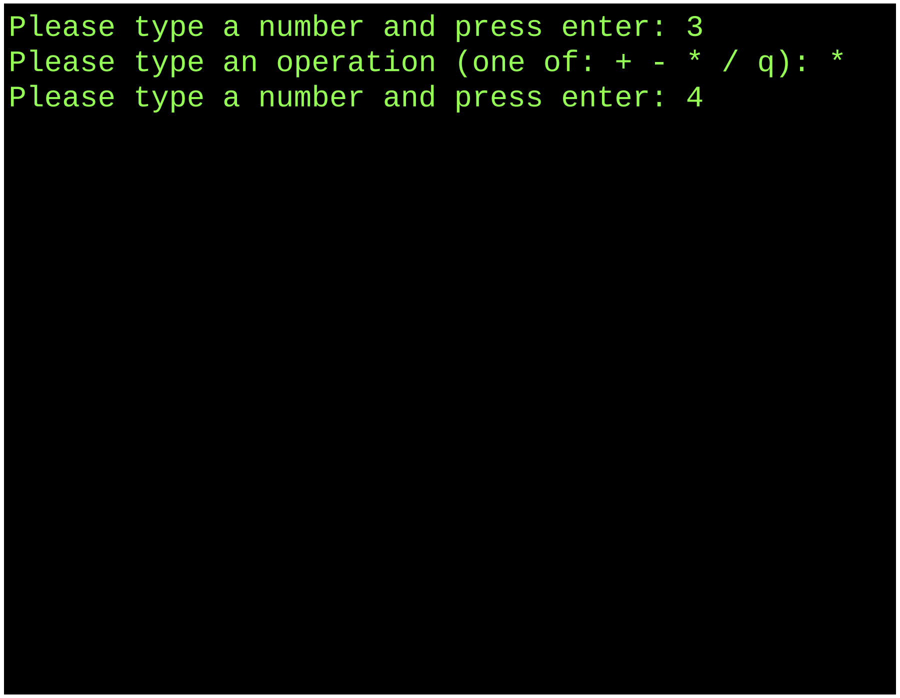
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
```



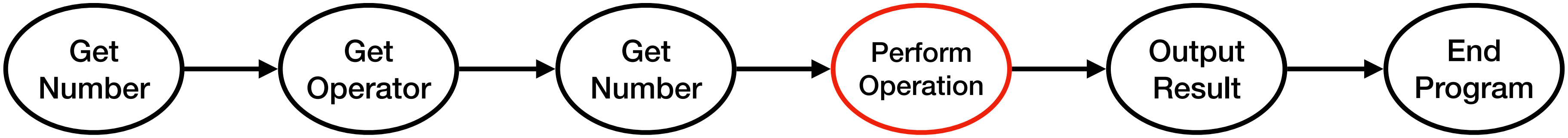


```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
```

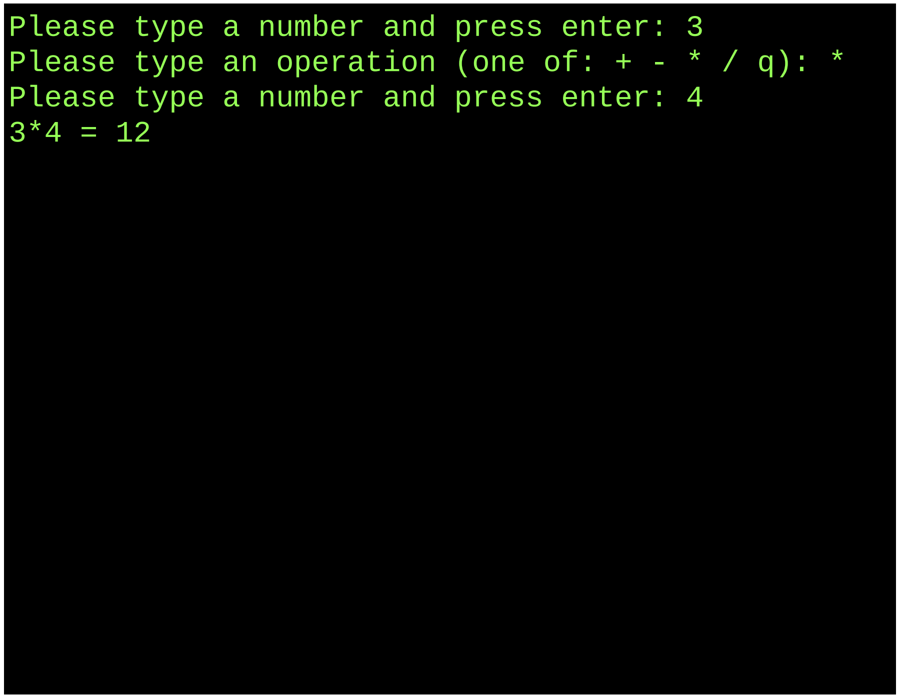




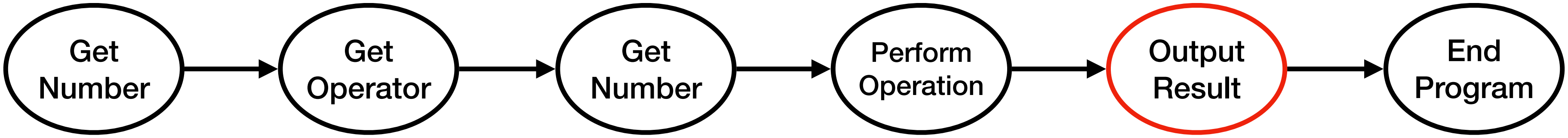
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
```

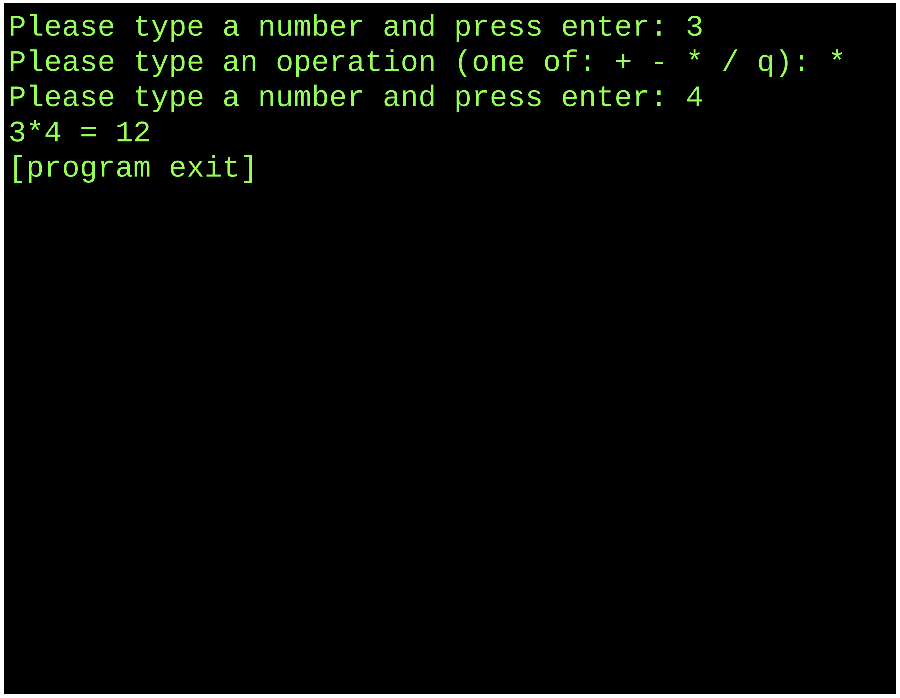


Get Number → Get Operator → Get Number → Perform Operation → Output Result → End Program

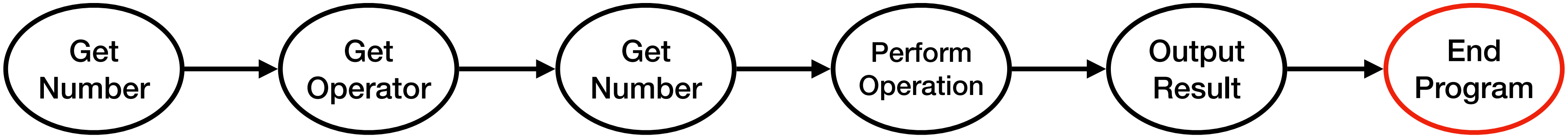


```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
```

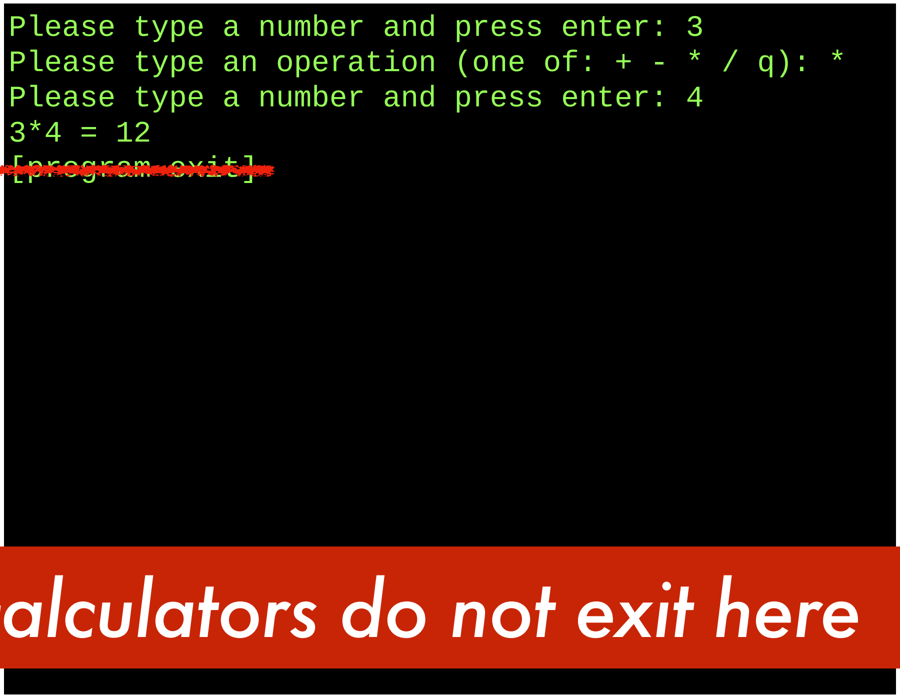




```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
[program exit]
```

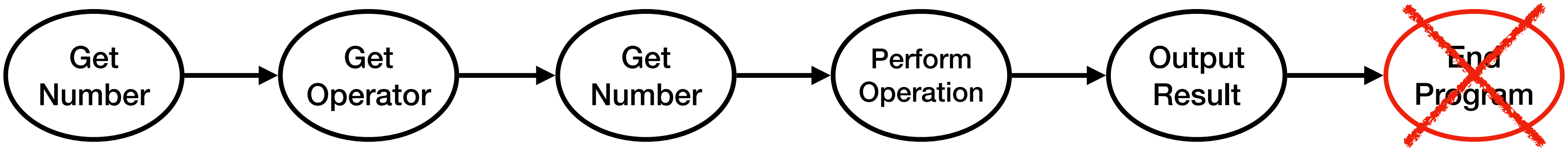


Get Number → Get Operator → Get Number → Perform Operation → Output Result → End Program

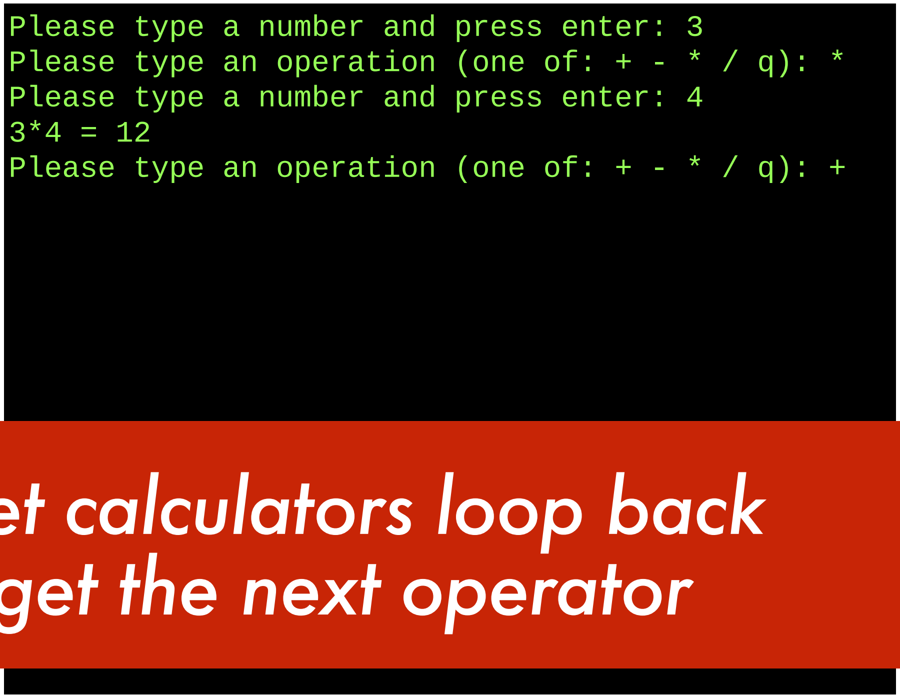


```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
~~[program exit]~~
```

Pocket calculators do not exit here

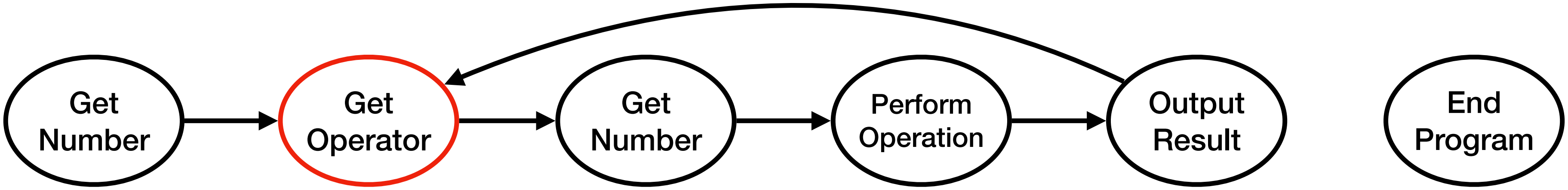


Get Number → Get Operator → Get Number → Perform Operation → Output Result → ~~End Program~~



```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
```

Pocket calculators loop back to get the next operator



Get Number

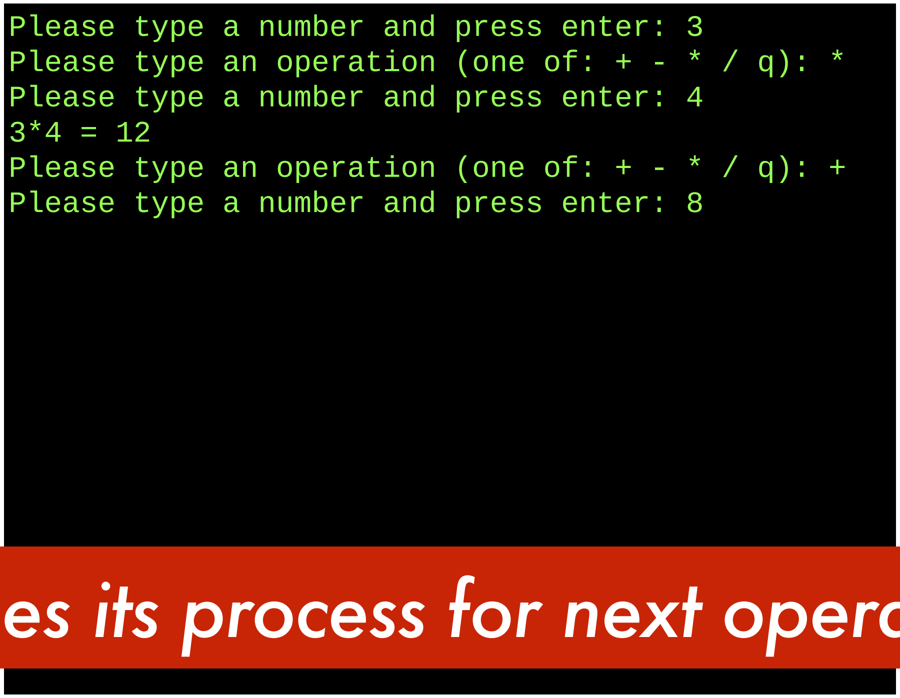
Get Operator

Get Number

Perform Operation

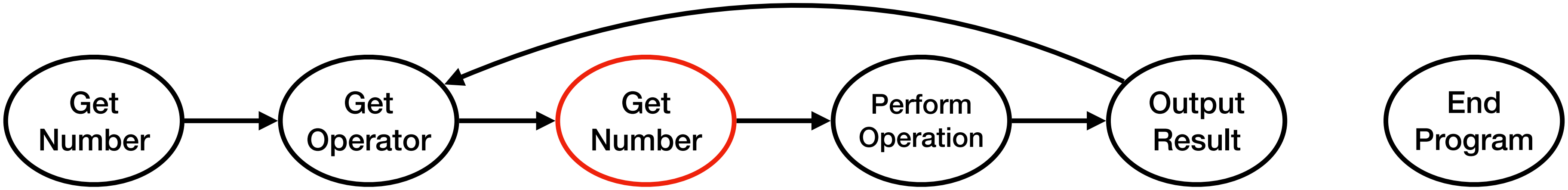
Output Result

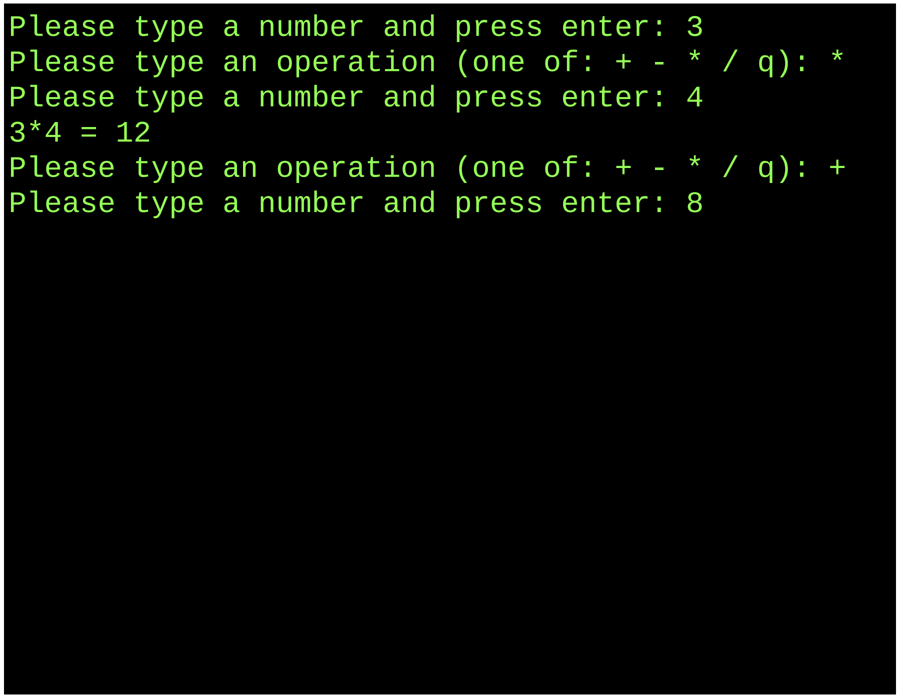
End Program



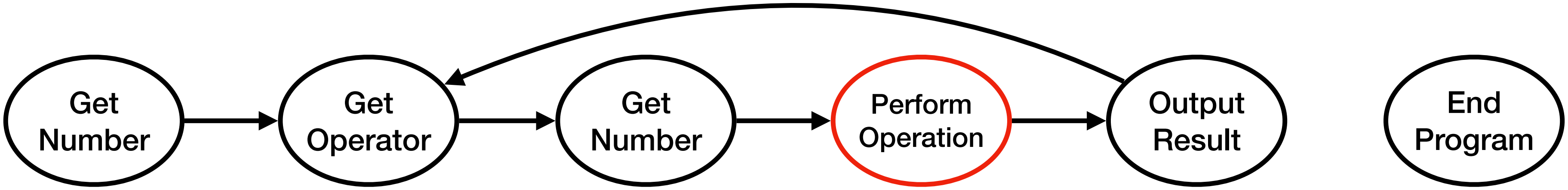
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
```

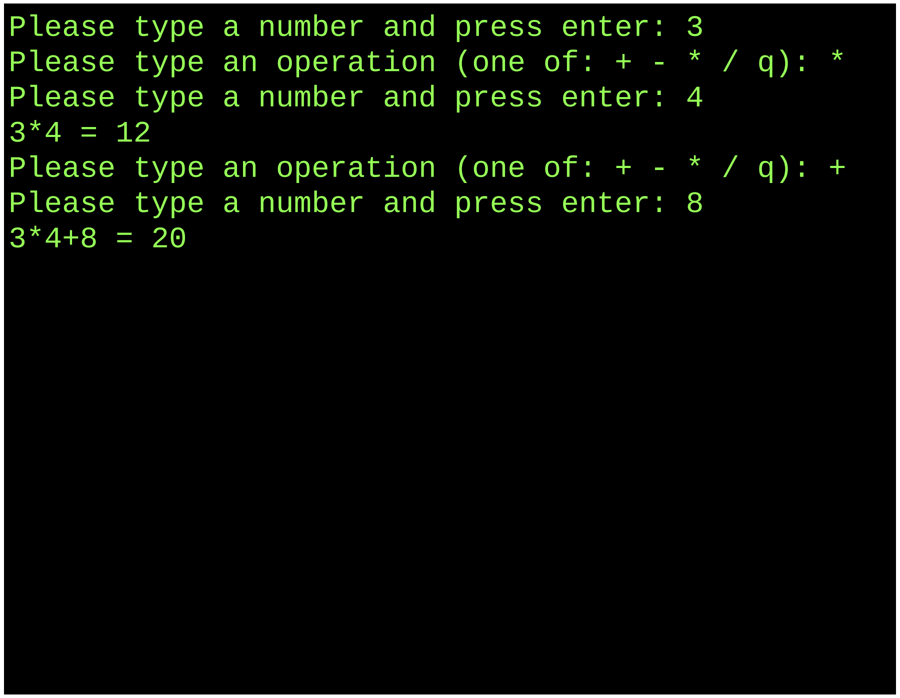
And, continues its process for next operand



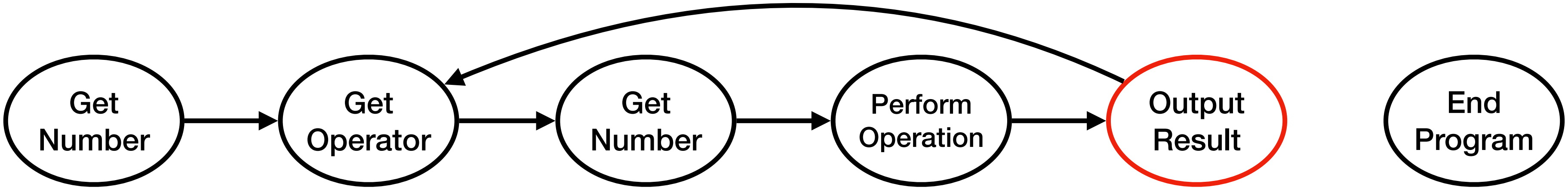


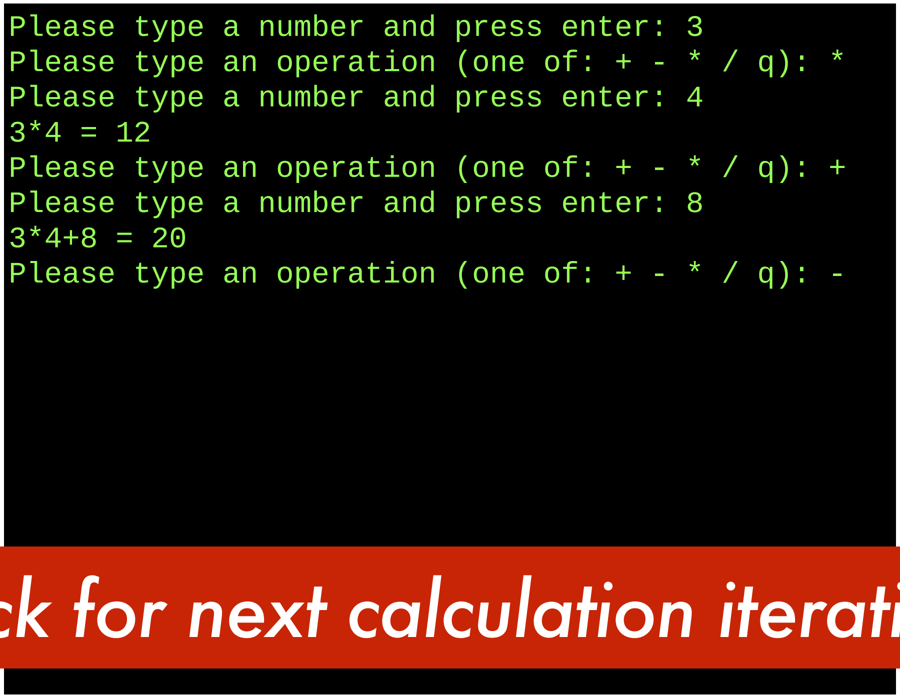
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
```





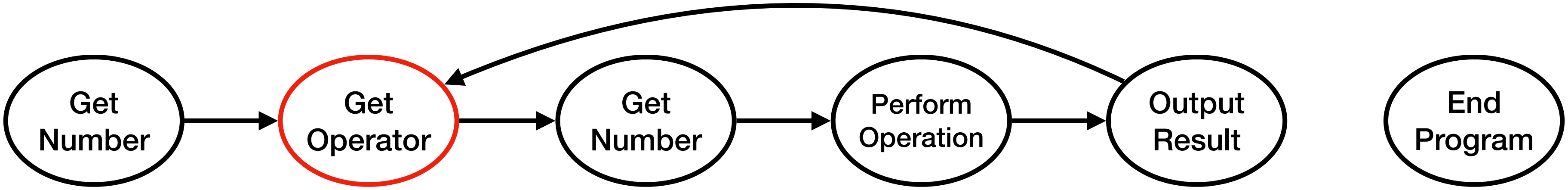
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
```

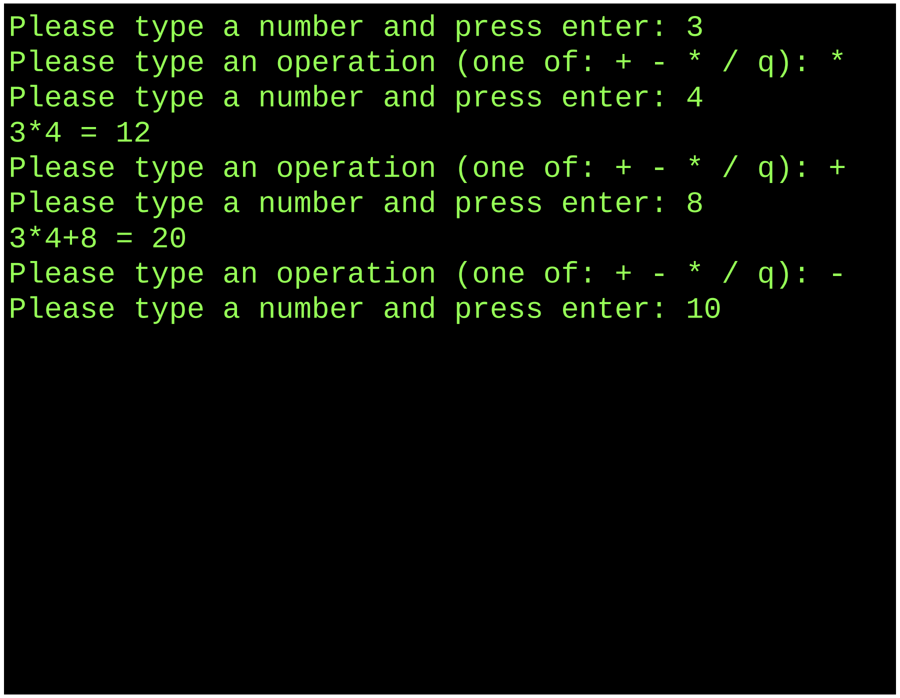




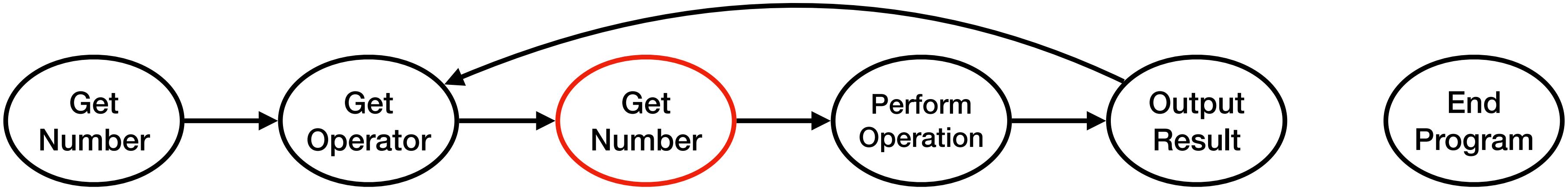
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
```

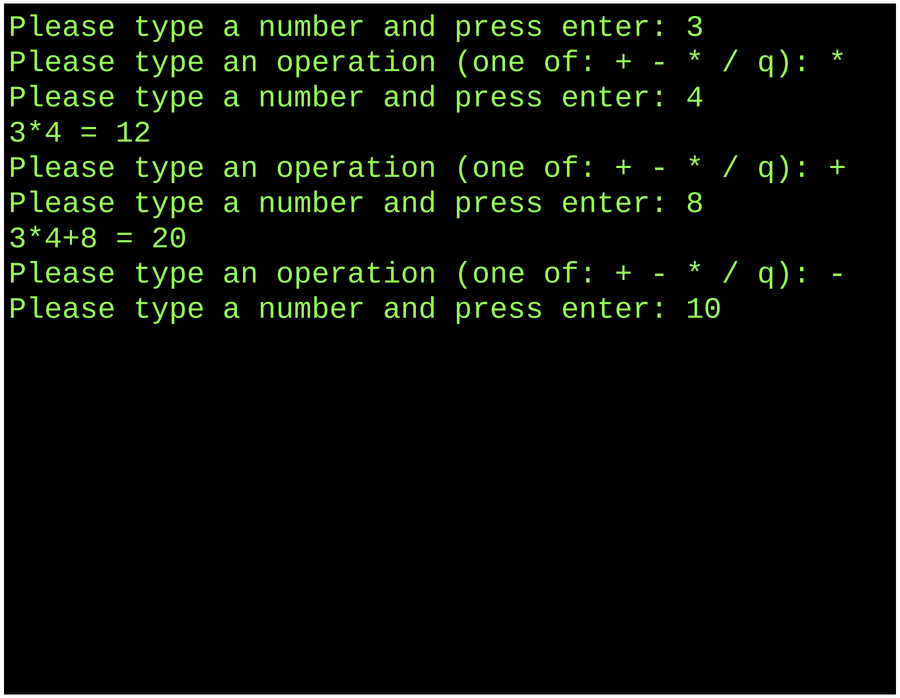
Looping back for next calculation iteration



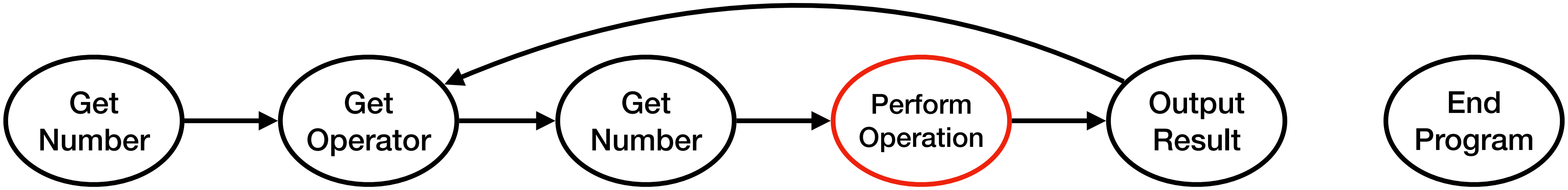


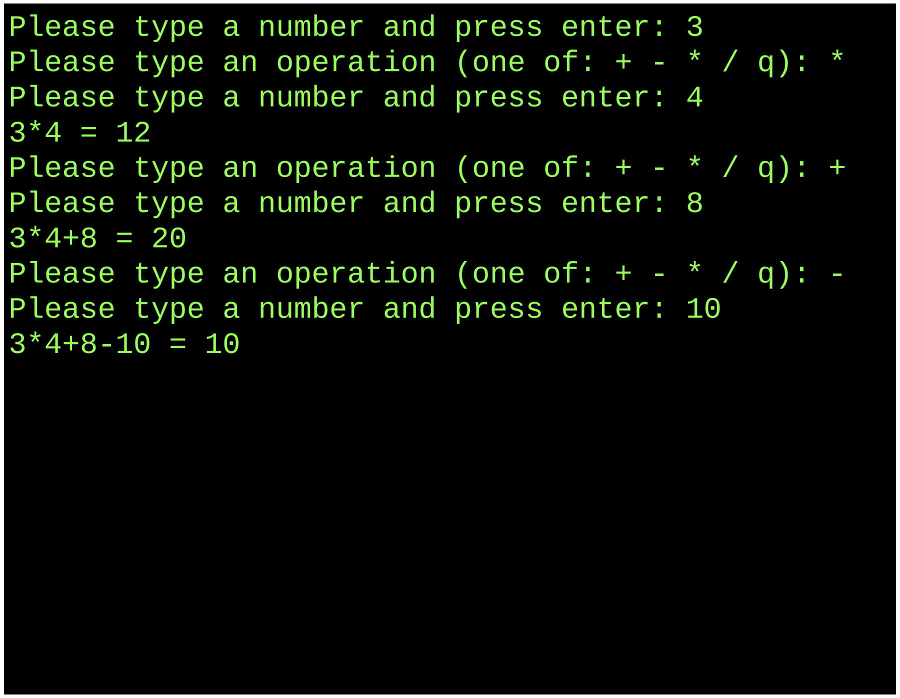
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
```



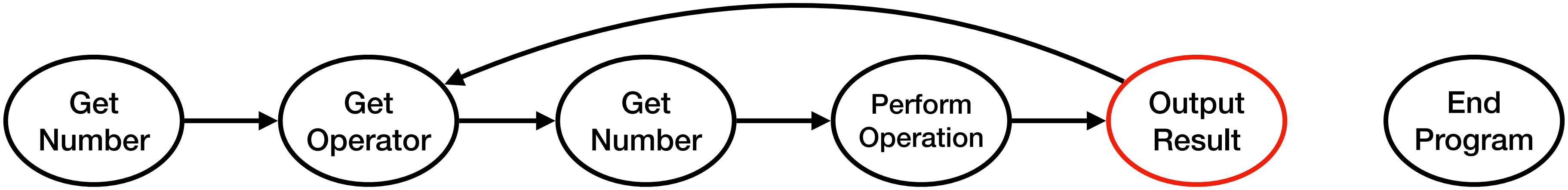


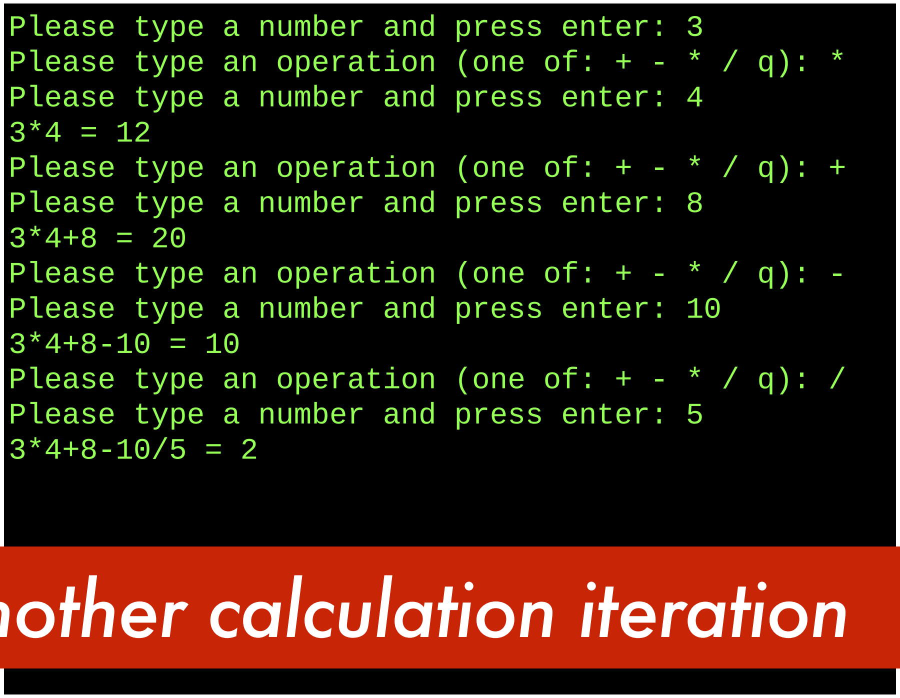
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
```





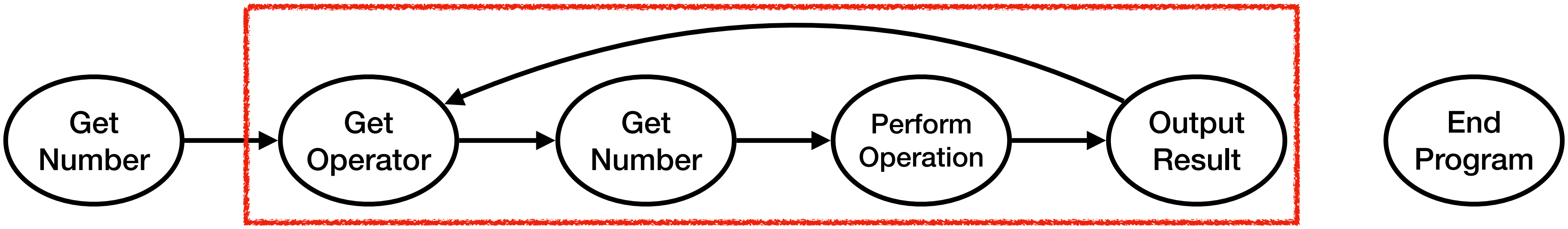
```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
```

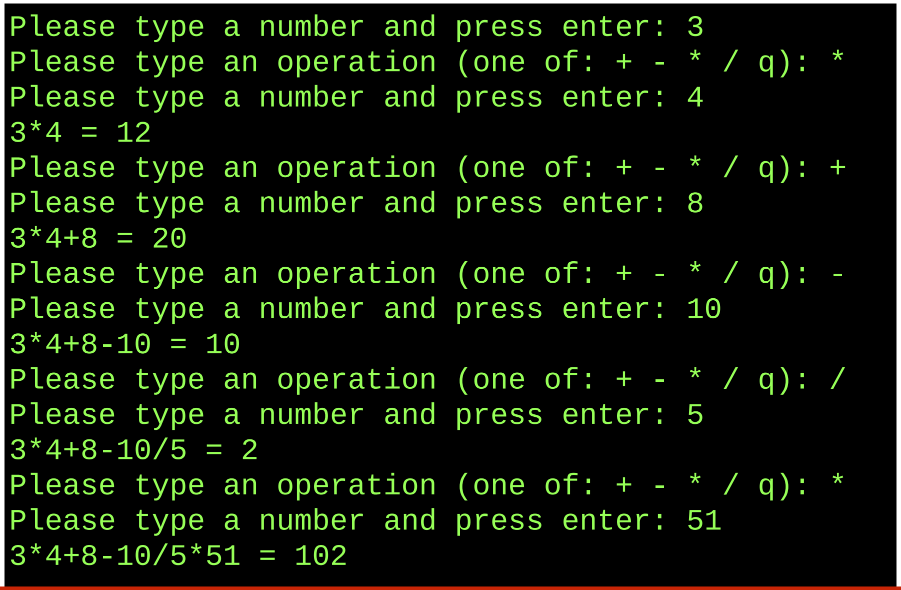




```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
```

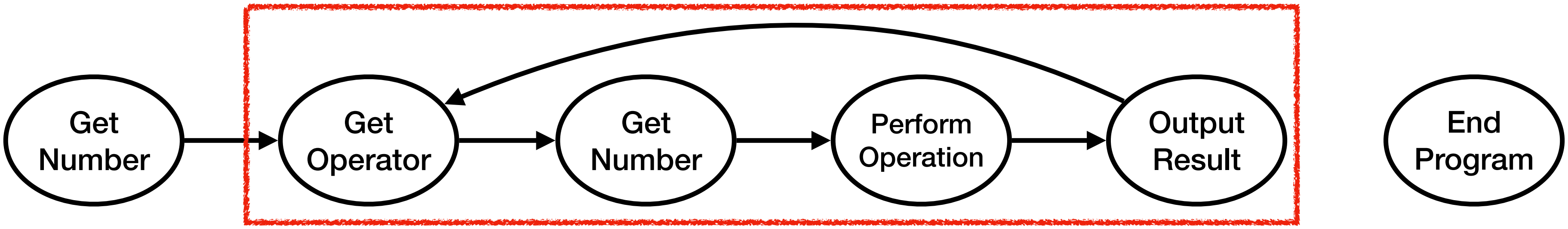
After another calculation iteration

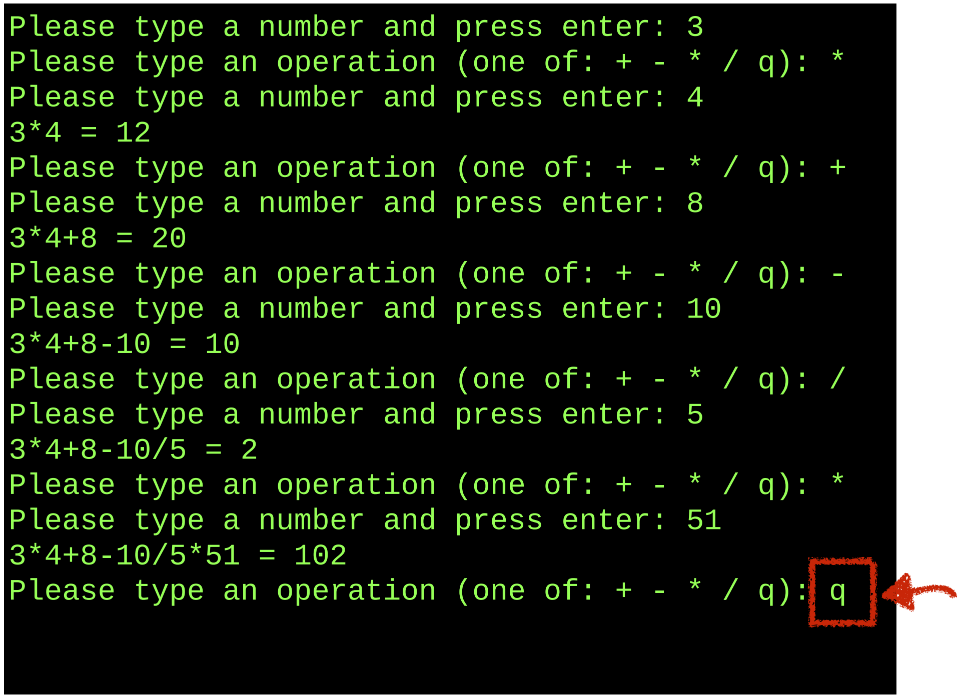




```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
```

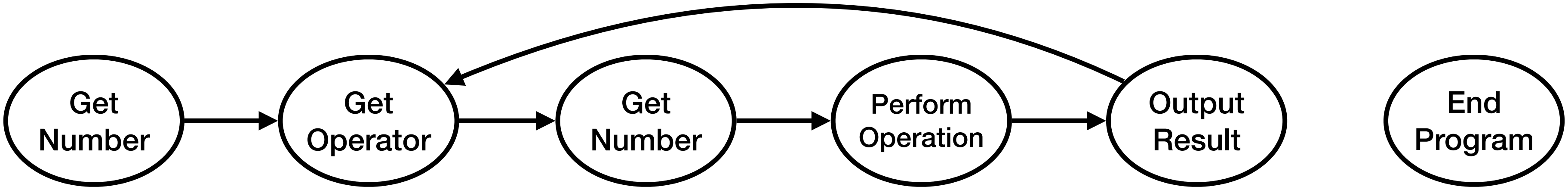
And, one more iteration

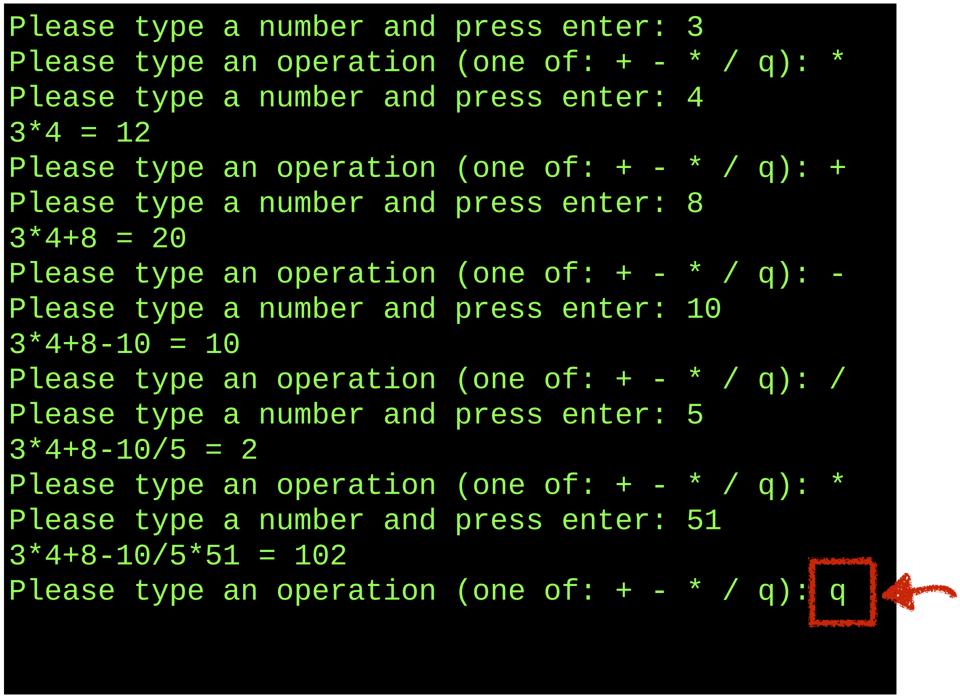




```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

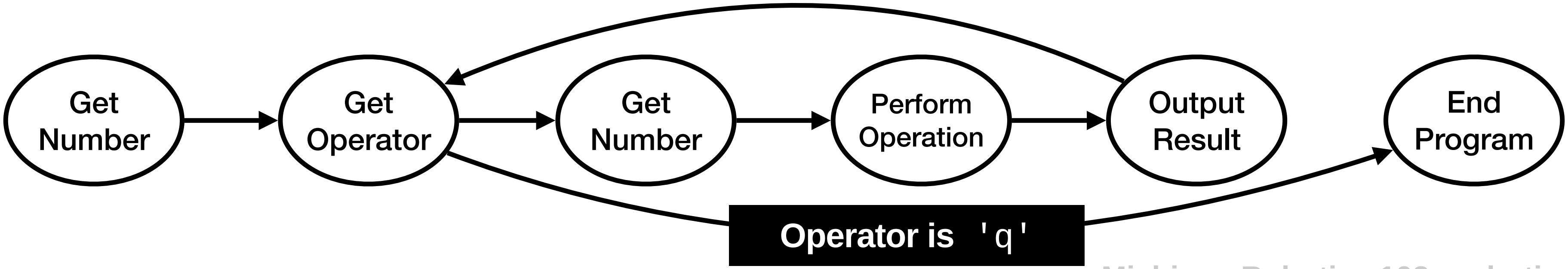
The user input is to quit the program



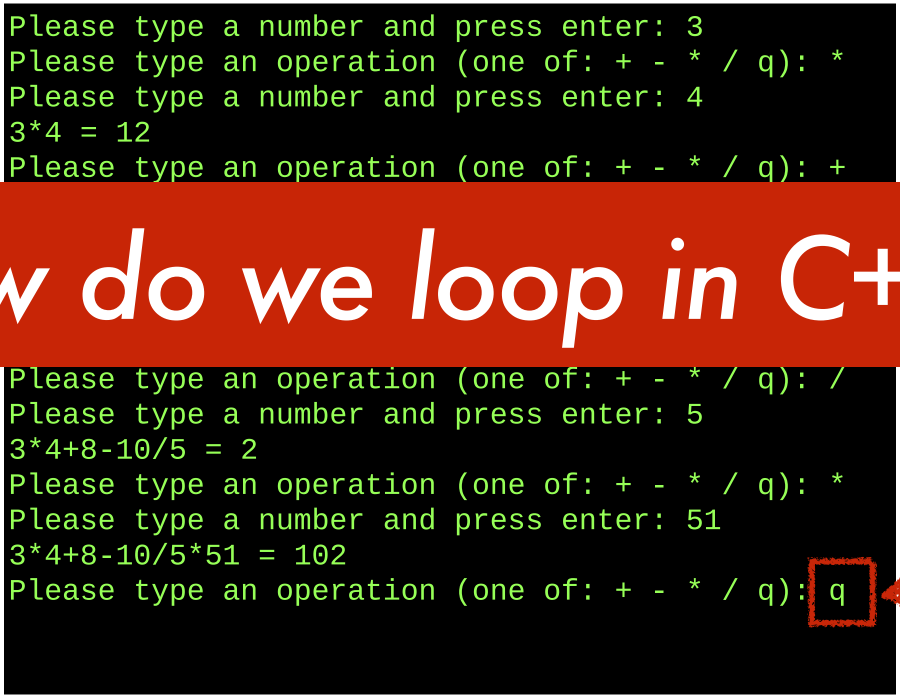


```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

The user input is to quit the program



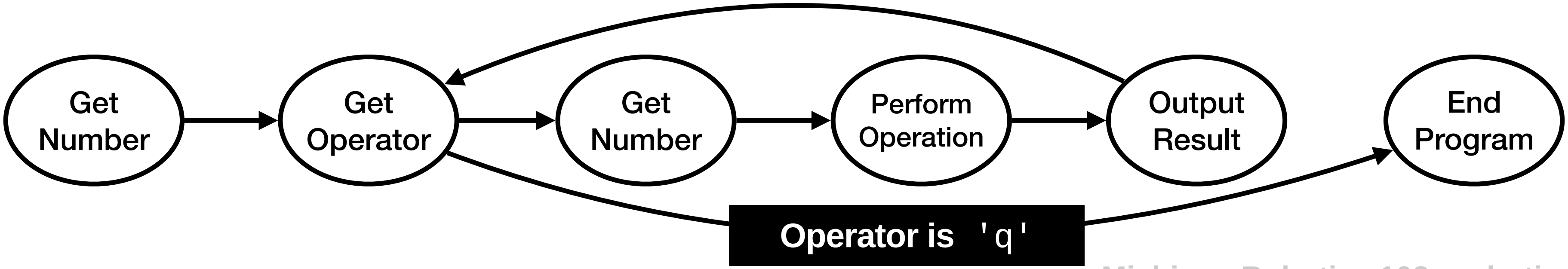
How do we loop in C++ ?



```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
```

```
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

The user input is to quit the program



How do we loop in C++ ?

One option:

A `while` loop

iThink.cpp (Version 00)

```cpp
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   if (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

```
Therefore, I am.
```

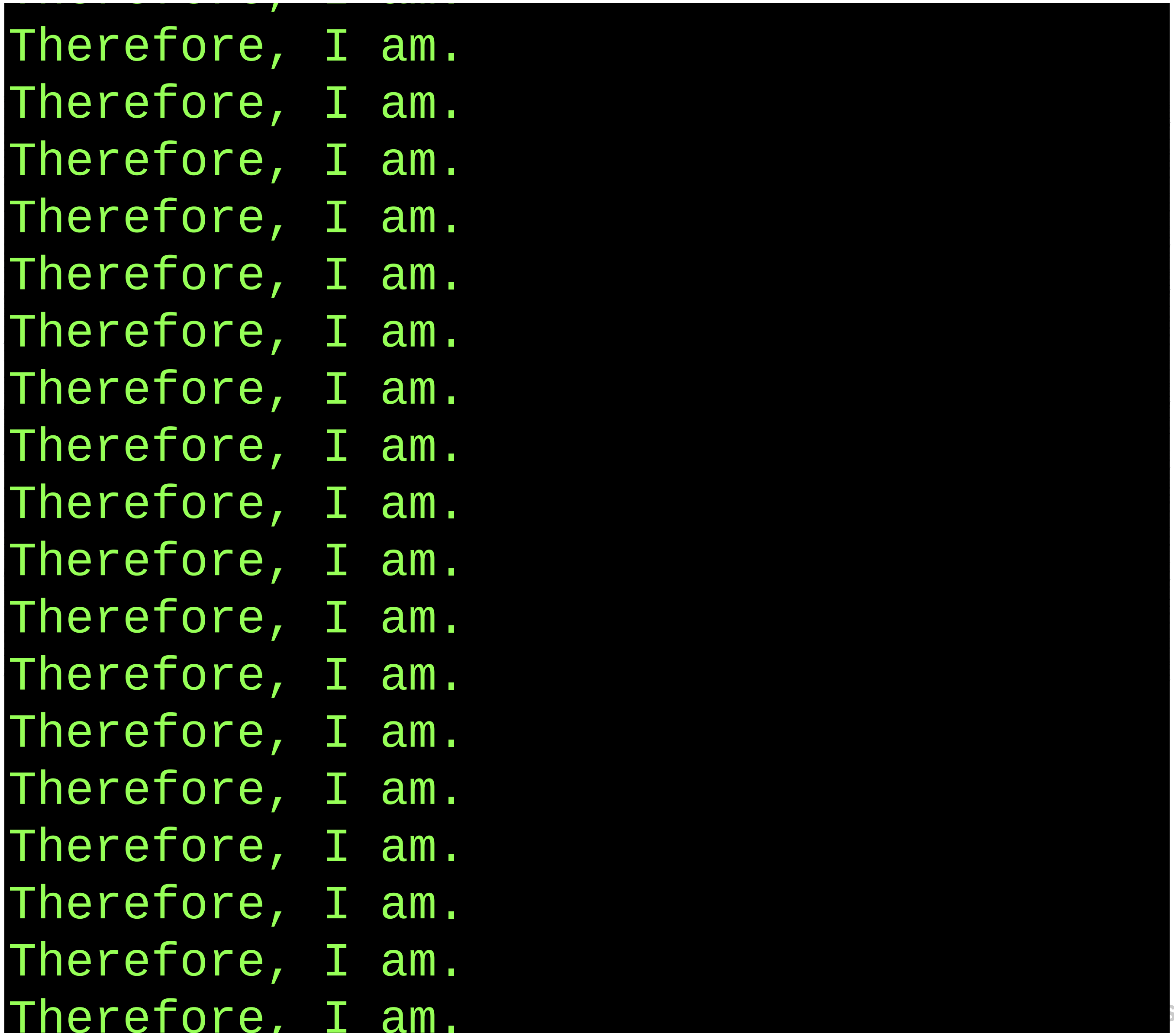
iThink.cpp (Version 04)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

iThink.cpp (Version 04)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
```



```
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 04)

```
#include <iostream>

int main()
{
    int thinkingAmount = 1;

    while (thinkingAmount > 0) {
        std::cout << "Therefore, I am.\n";
    }
}
```

This is an infinite loop

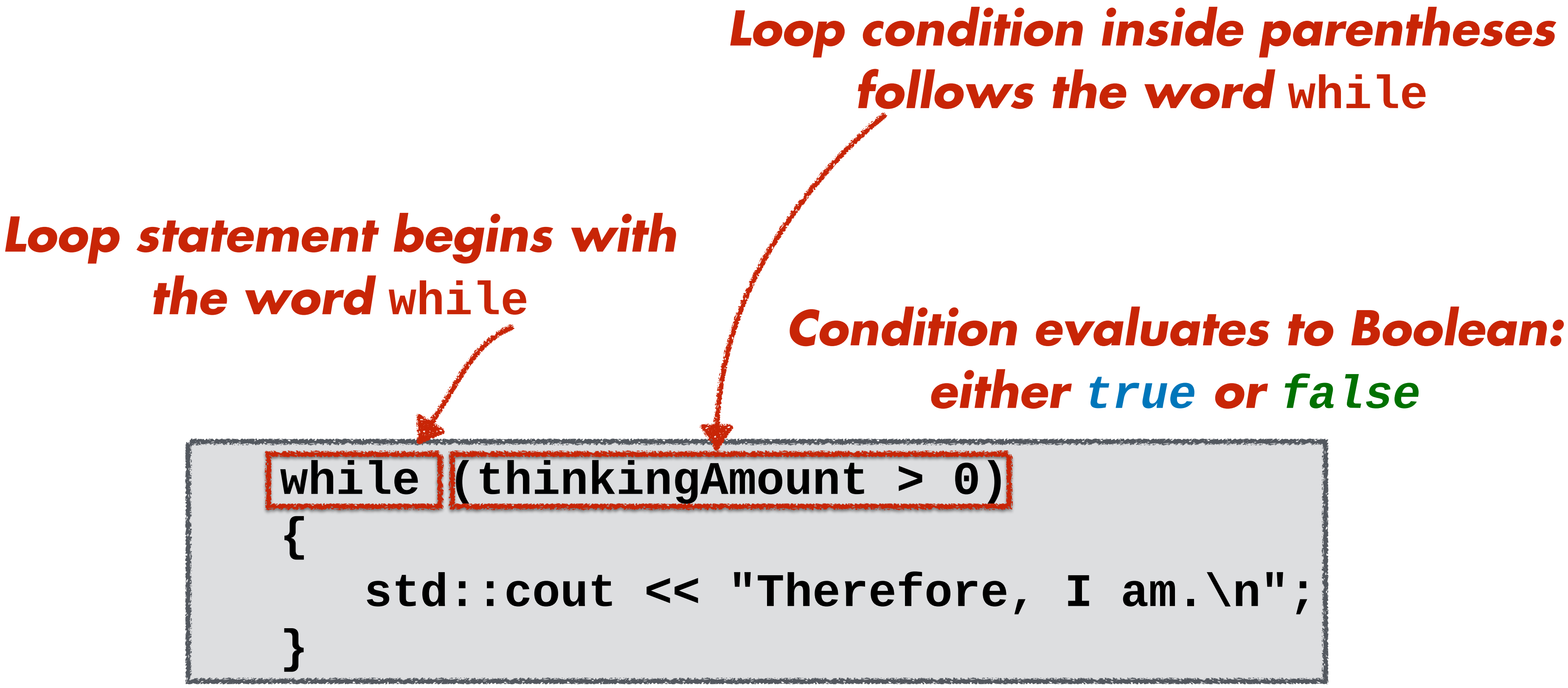
```
Therefore, I am.
Therefore, I am.
Therefore, I am.
^C
```

Press Control and C keys together to interrupt and terminate

```
while (thinkingAmount > 0)
{
   std::cout << "Therefore, I am.\n";
}
```

Loop statement begins with the word `while`

```
while (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
```



Loop condition inside parentheses follows the word `while`

Loop statement begins with the word `while`

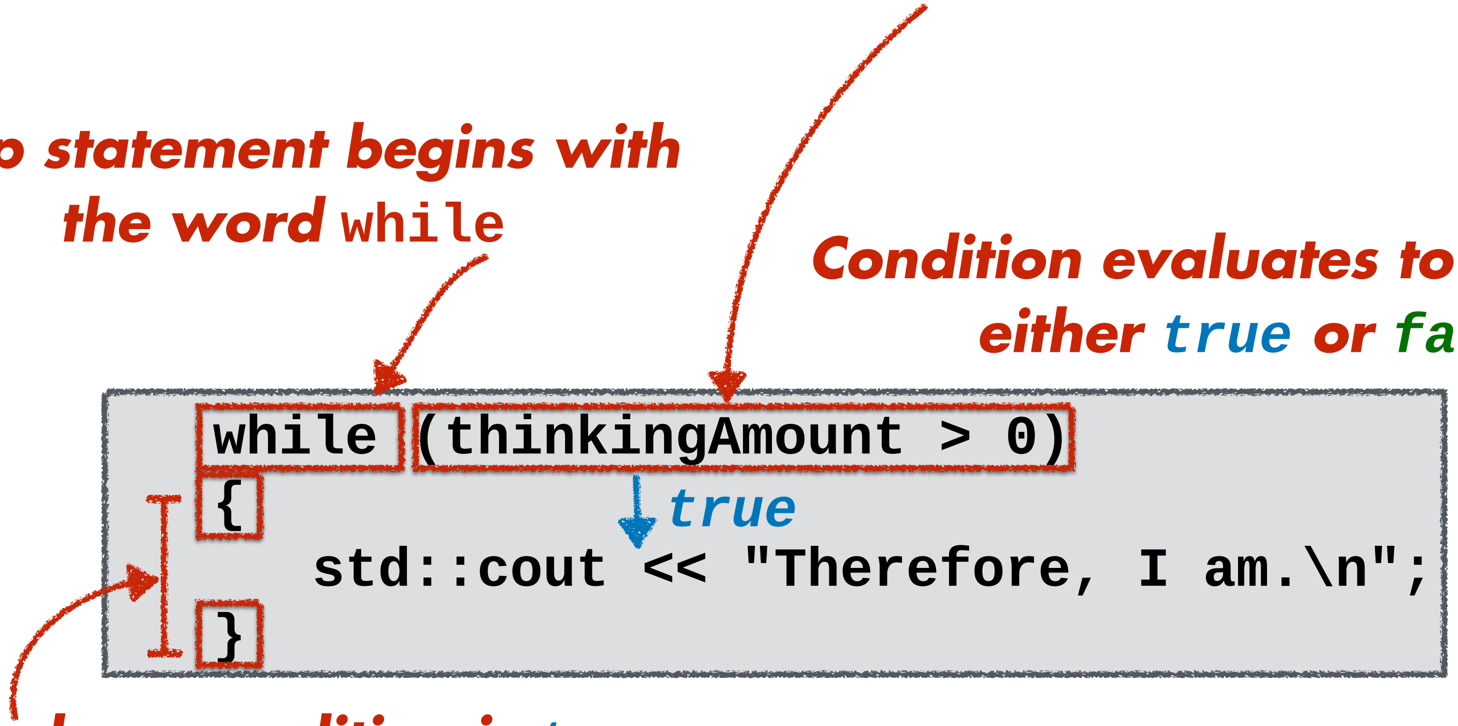
Condition evaluates to Boolean: either `true` or `false`

```
while (thinkingAmount > 0)
{
    std::cout << "Therefore, I am.\n";
}
```

***Loop condition inside parentheses follows the word* `while`**

***Loop statement begins with the word* `while`**

***Condition evaluates to Boolean: either* `true` *or* `false`**



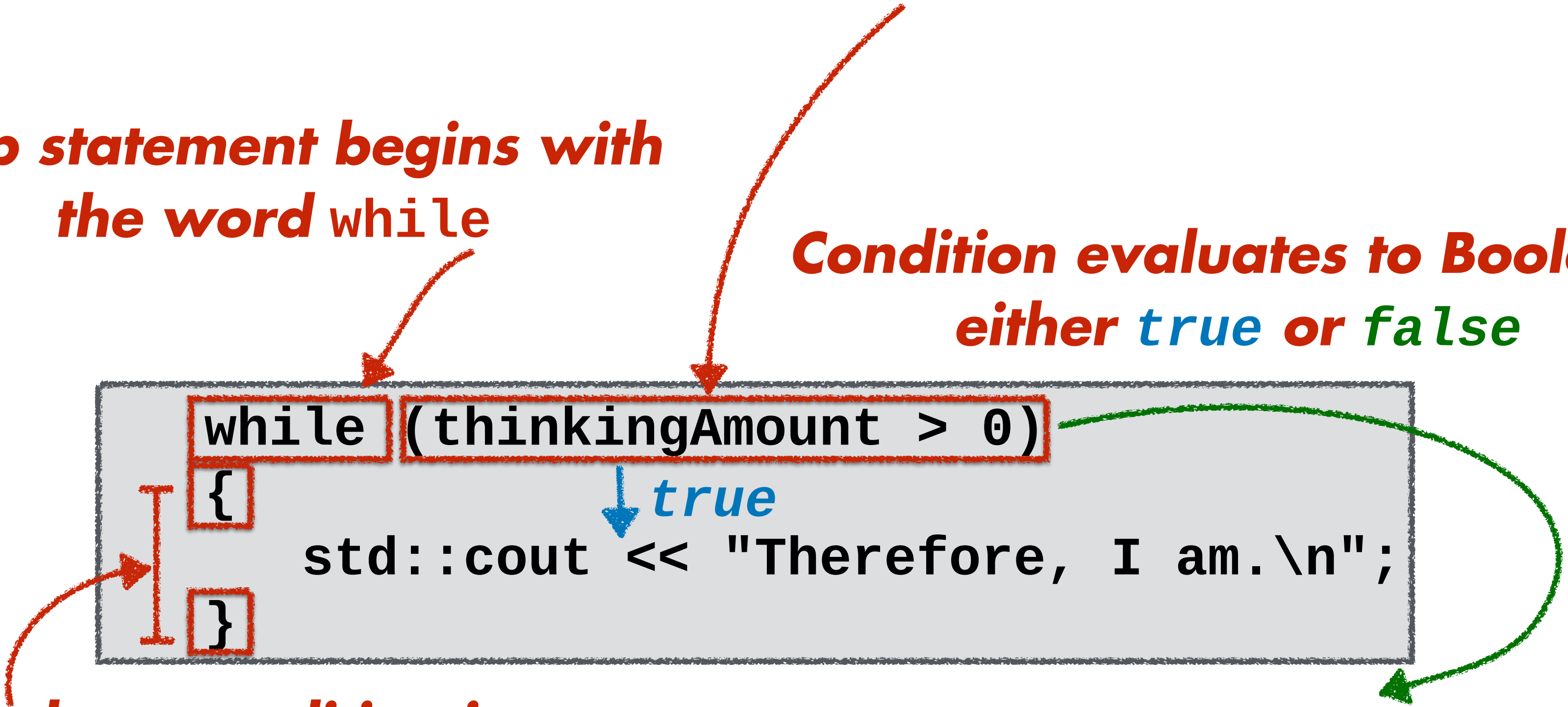
```
  while (thinkingAmount > 0)
  {
      std::cout << "Therefore, I am.\n";
  }
```

If the loop condition is* `true`*, execute the next block of code and come back to the loop condition afterwards

Loop condition inside parentheses follows the word `while`

Loop statement begins with the word `while`

Condition evaluates to Boolean: either `true` *or* `false`



```
  while (thinkingAmount > 0)
  {
      std::cout << "Therefore, I am.\n";
  }
```

If the loop condition is `true`*, execute the next block of code and come back to the loop condition afterwards*

If the condition is `false`*, leave the loop and skip the next block of code*

iThink.cpp (Version 04)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n";
   }
}
```

Suppose we decreased our thinking amount after each iteration

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
^C
```

iThink.cpp (Version 05)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount = thinkingAmount - 1;
   }
}
```

iThink.cpp (Version 05)

```cpp
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount = thinkingAmount - 1;
   }
}
```

```
Therefore, I am.
```

iThink.cpp (Version 06)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Decrement operator:

Subtracts one from an integer variable

[redacted]

Operators and Precedence

- A subset of C++ operators in order of precedence
  - grouping: `/*` open comment, `*/` close comment, `//` comment to end of line, `(` open parenthesis, `)` close parenthesis
  - increment/decrement: `++` increment variable, `--` decrement variable
  - arithmetic: `*` multiplication, `/` division, `%` modulus, `+` addition/concatenation, `-` subtraction
  - comparison: `<` less than, `<=` less than or equal, `>` greater than, `>=` greater than or equal, `==` equality, `!=` inequality, `&&` logical AND, `||` logical OR, `!` logical NOT
  - assignment: `=` assignment, `+=` add to variable, `*=` multiply to variable

Decrement operator: Subtracts one from an integer variable

iThink.cpp (Version 06)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

```
Therefore, I am.
```

Great! Same result

iThink.cpp (Version 06)

```
#include <iostream>

int main()
{
   int thinkingAmount = 1;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Suppose we had more thought at the beginning

```
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

[redacted]

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

Our loop iterated 3 times

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Let’s walkthrough each step

[redacted]

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution → `int thinkingAmount = 3;`

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution → `while (thinkingAmount > 0) {`

Variables

3

thinkingAmount

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;
                  true
   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

3

thinkingAmount

Current point in Program execution

Loop condition is true because 3 is greater than 0

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution → `std::cout << "Therefore, I am.\n”;`

Variables

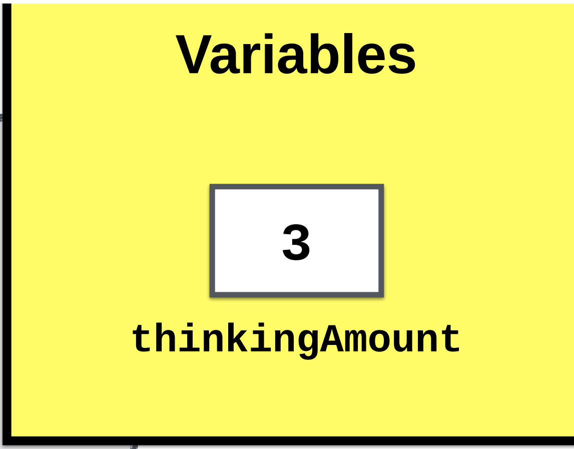
3

thinkingAmount

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution →



Variables

3

thinkingAmount

```
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

2

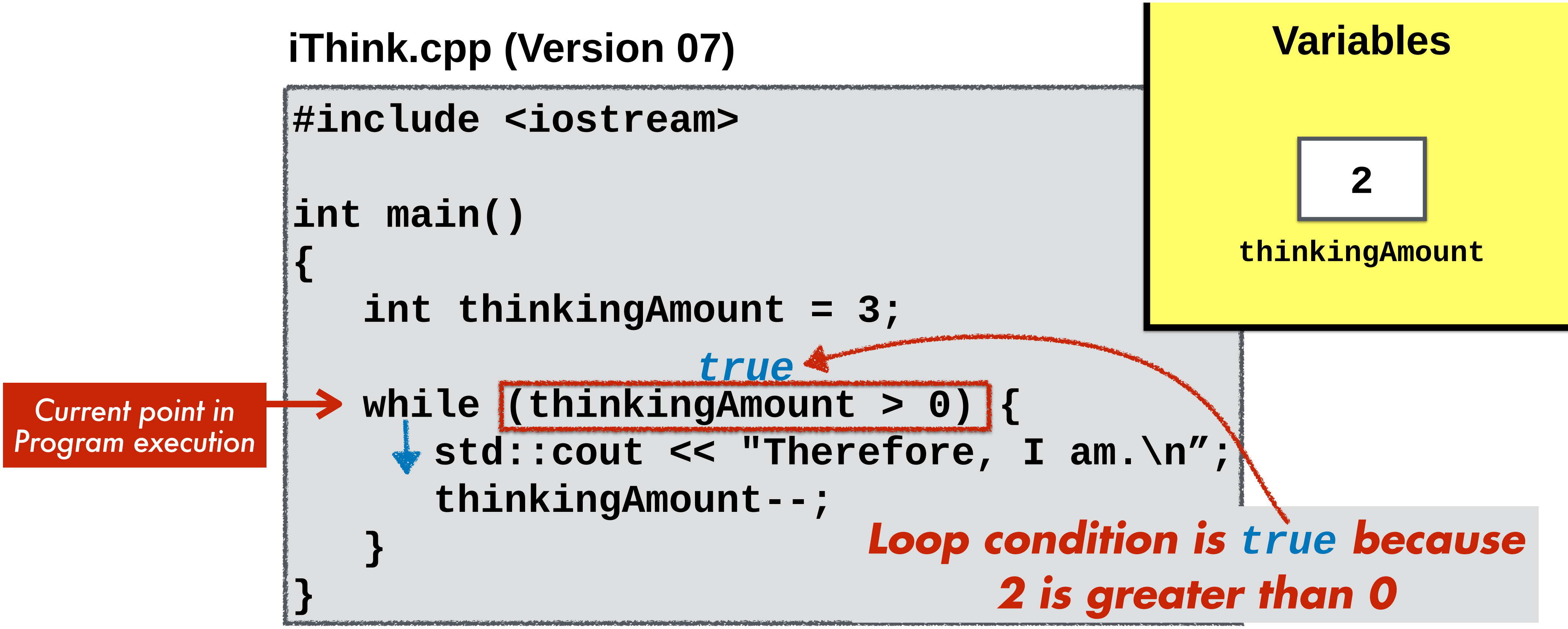
thinkingAmount

Current point in Program execution

Iteration done. Go back to loop start

```
Therefore, I am.
```

iThink.cpp (Version 07)



```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

Current point in Program execution

Loop condition is `true` because 2 is greater than 0

```
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution → `std::cout << "Therefore, I am.\n”;`

Variables

2

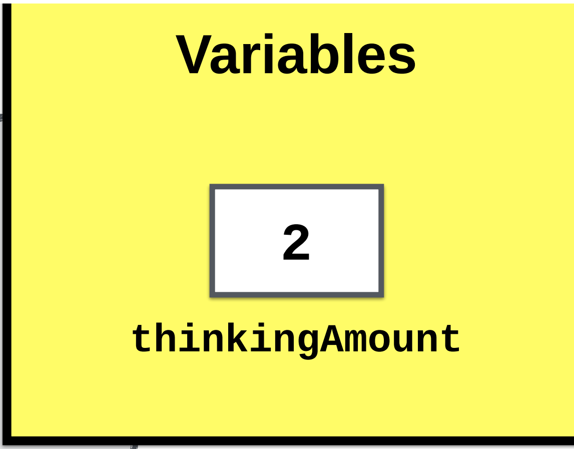
thinkingAmount

```
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution


Variables

2

thinkingAmount

```
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution

Iteration done. Go back to loop start

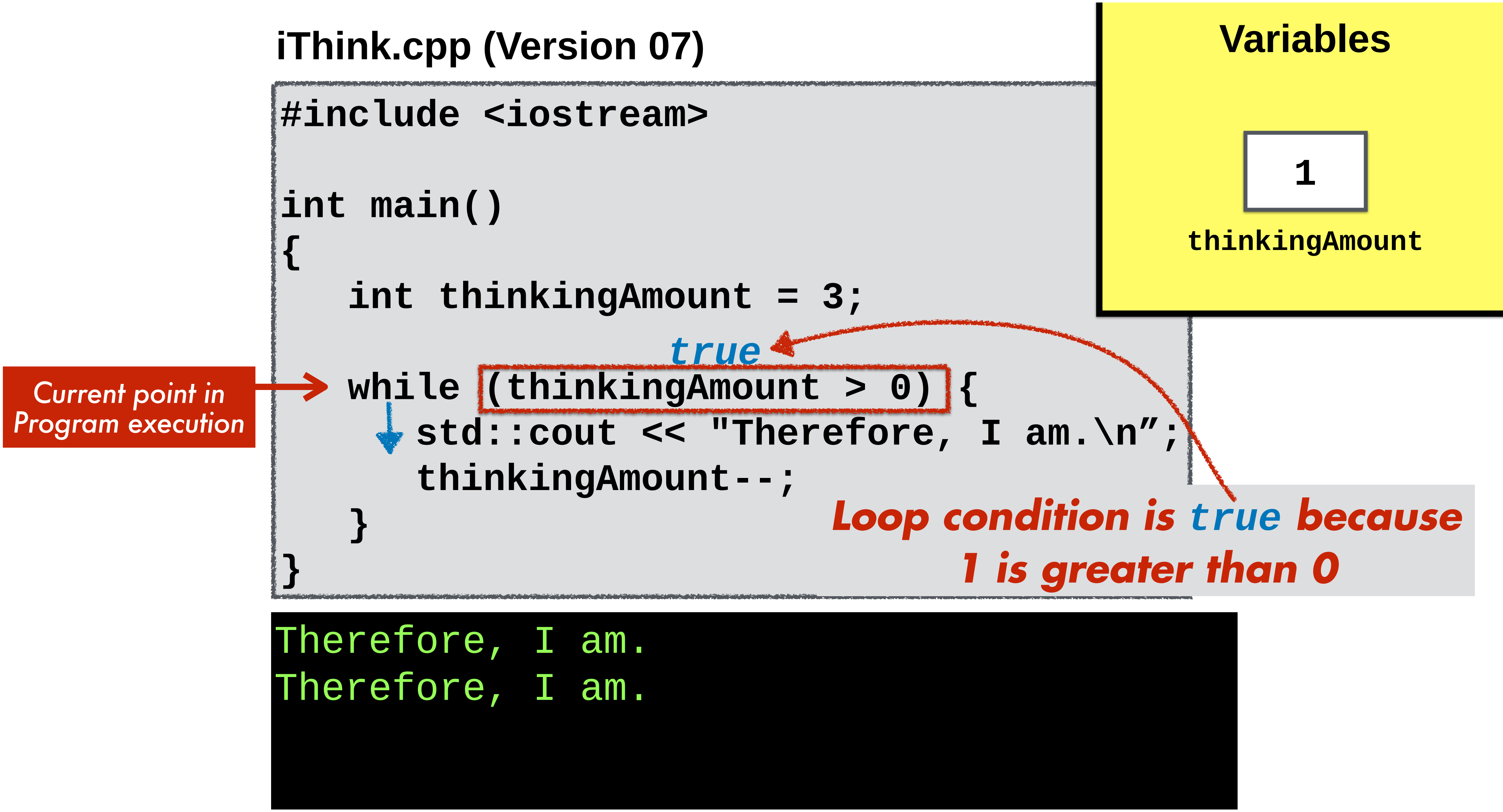
Variables

1

thinkingAmount

```
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)



```
#include <iostream>

int main()
{
   int thinkingAmount = 3;
                              true
   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

Loop condition is true because 1 is greater than 0

```
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Current point in Program execution → `std::cout << "Therefore, I am.\n”;`

```
Therefore, I am.
Therefore, I am.
```

Variables

`1`

`thinkingAmount`

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

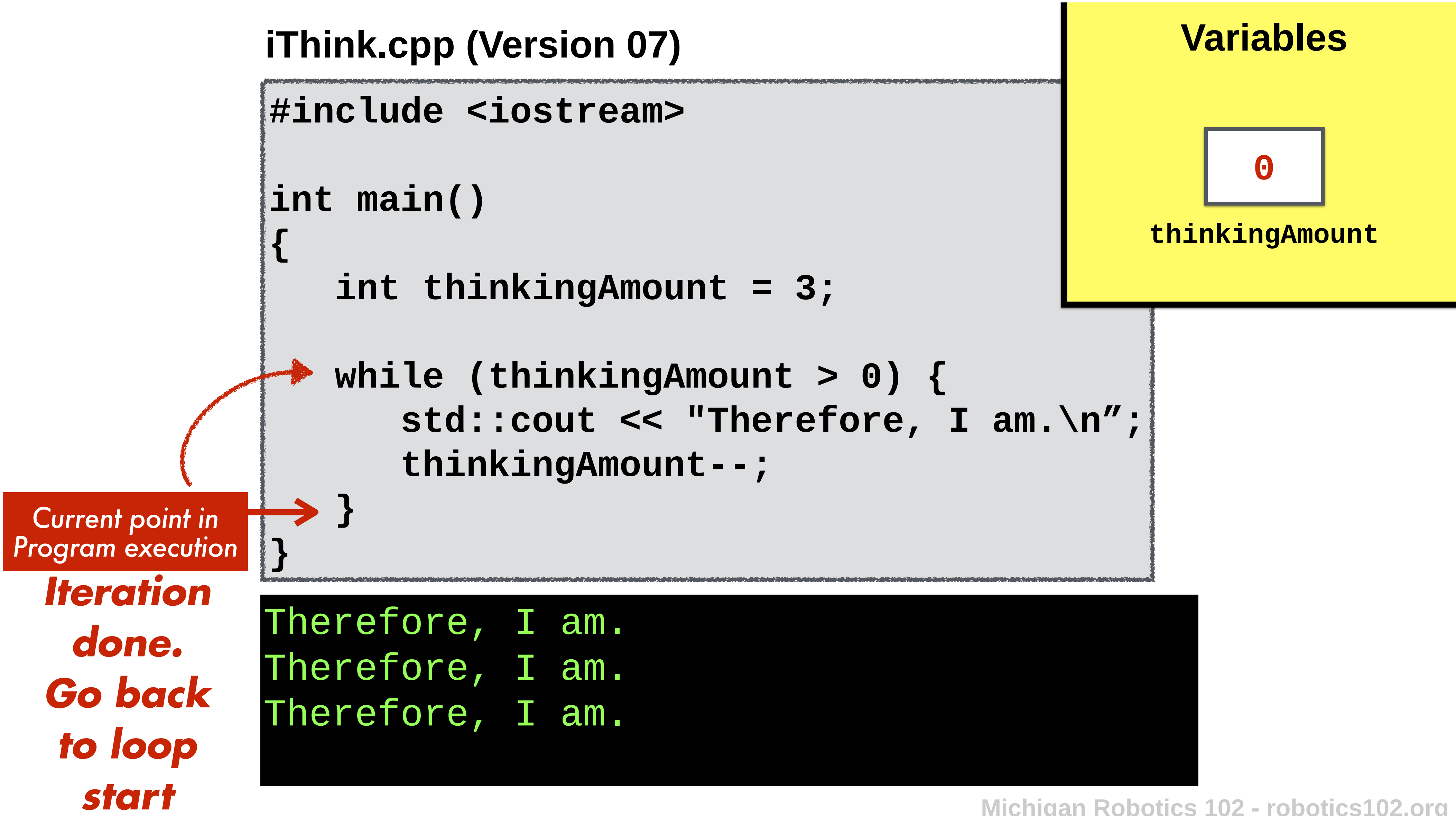
Current point in Program execution → `thinkingAmount--;`

Variables

thinkingAmount: 1

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)



```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

Iteration done. Go back to loop start

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

0

thinkingAmount

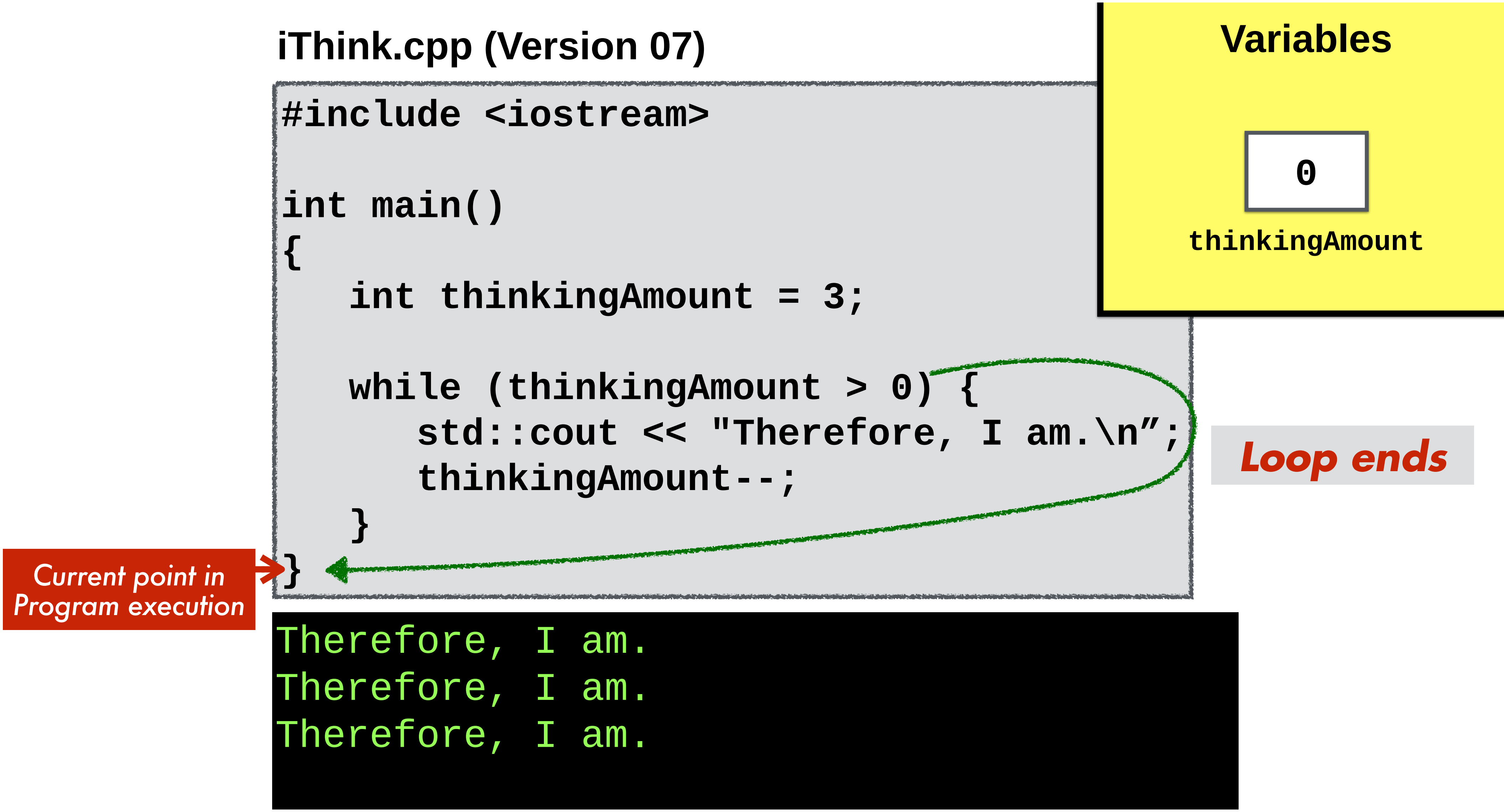
Current point in Program execution

false

Loop condition is *false* because 0 is not greater than 0

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)



```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Variables

0

thinkingAmount

Loop ends

Current point in Program execution

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

iThink.cpp (Version 07)

```cpp
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Program ends

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

How do we loop in C++ ?

One option:

A `while` loop

How do we loop in C++ ?

Another option:

A `for` loop

iThink.cpp (Version 07)

Elements of a typical loop

```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

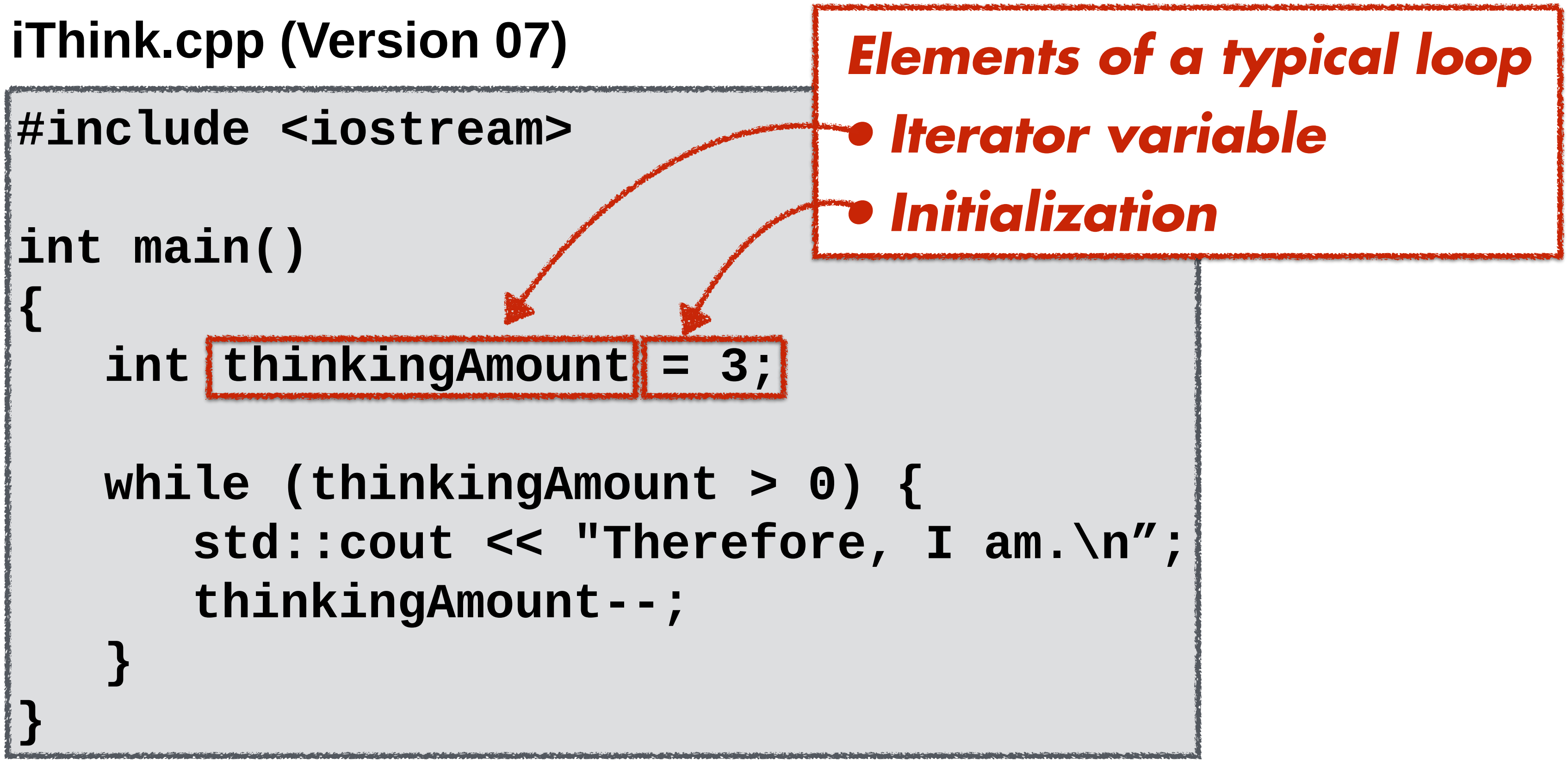
iThink.cpp (Version 07)

```
#include <iostream>

int main()
{
    int thinkingAmount = 3;

    while (thinkingAmount > 0) {
        std::cout << "Therefore, I am.\n”;
        thinkingAmount--;
    }
}
```

Elements of a typical loop

- ***Iterator variable***

iThink.cpp (Version 07)

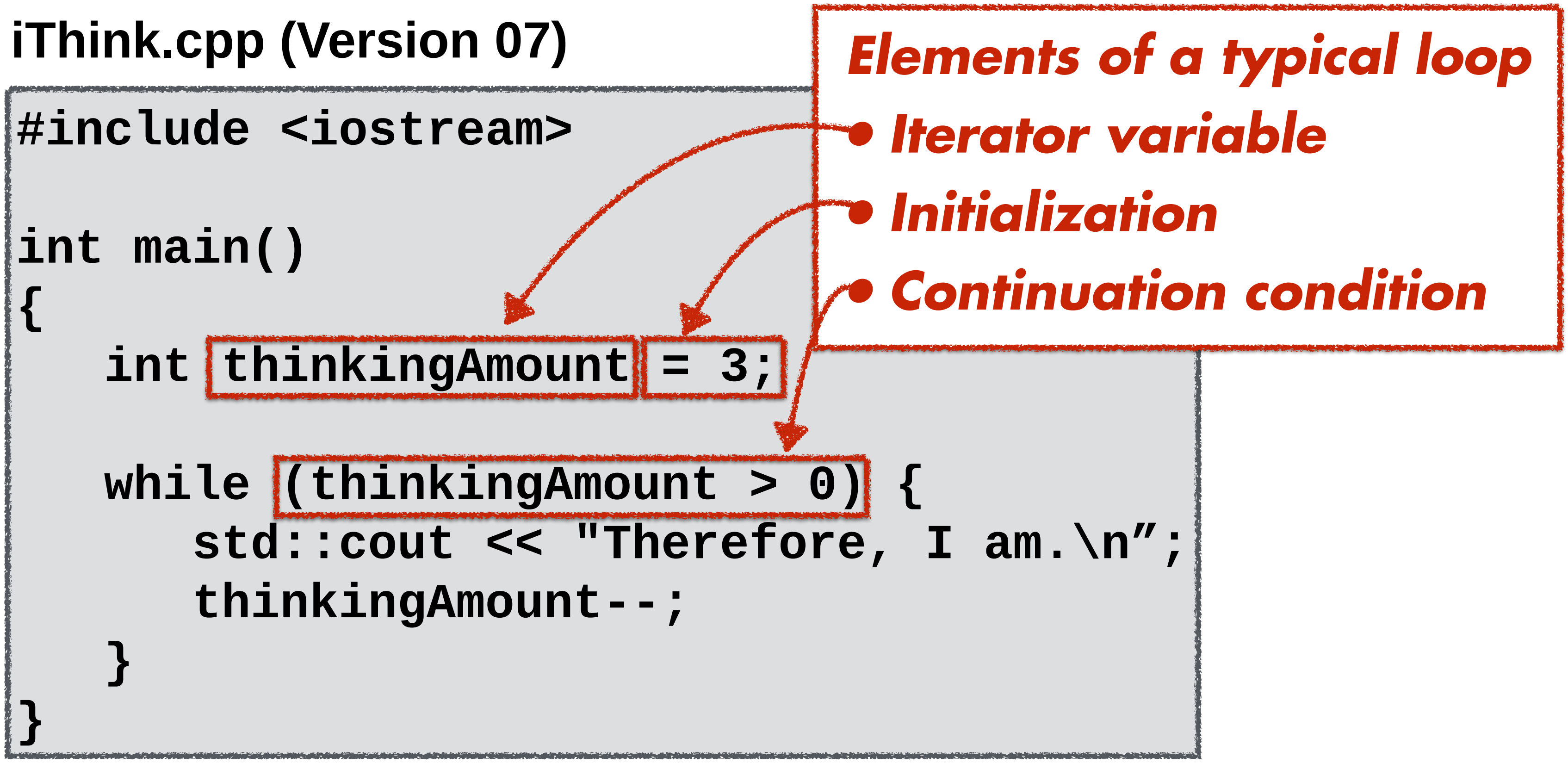


```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Elements of a typical loop

- *Iterator variable*
- *Initialization*

iThink.cpp (Version 07)

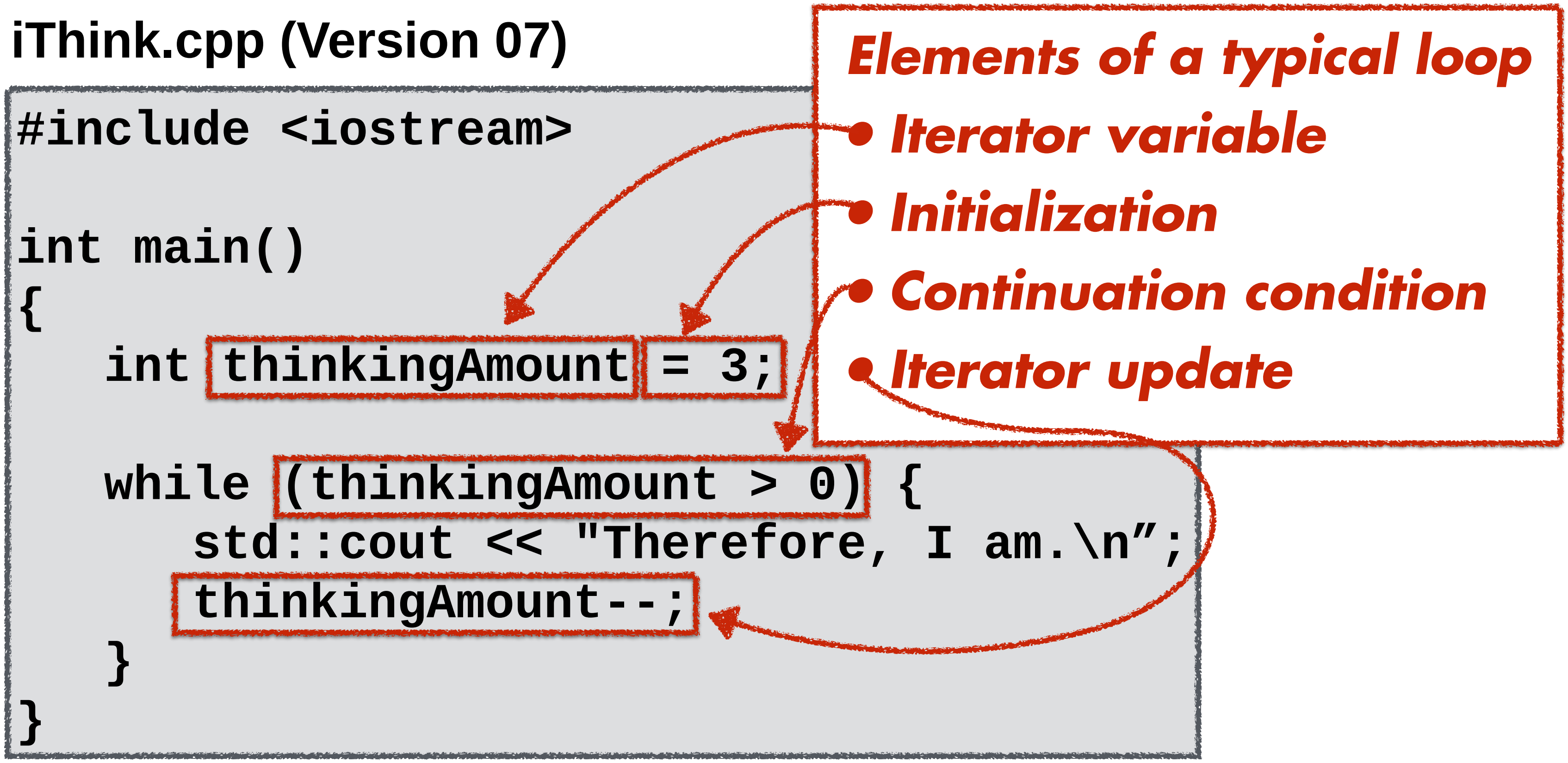


```
#include <iostream>

int main()
{
   int thinkingAmount = 3;

   while (thinkingAmount > 0) {
      std::cout << "Therefore, I am.\n”;
      thinkingAmount--;
   }
}
```

Elements of a typical loop

- ***Iterator variable***
- ***Initialization***
- ***Continuation condition***

iThink.cpp (Version 07)

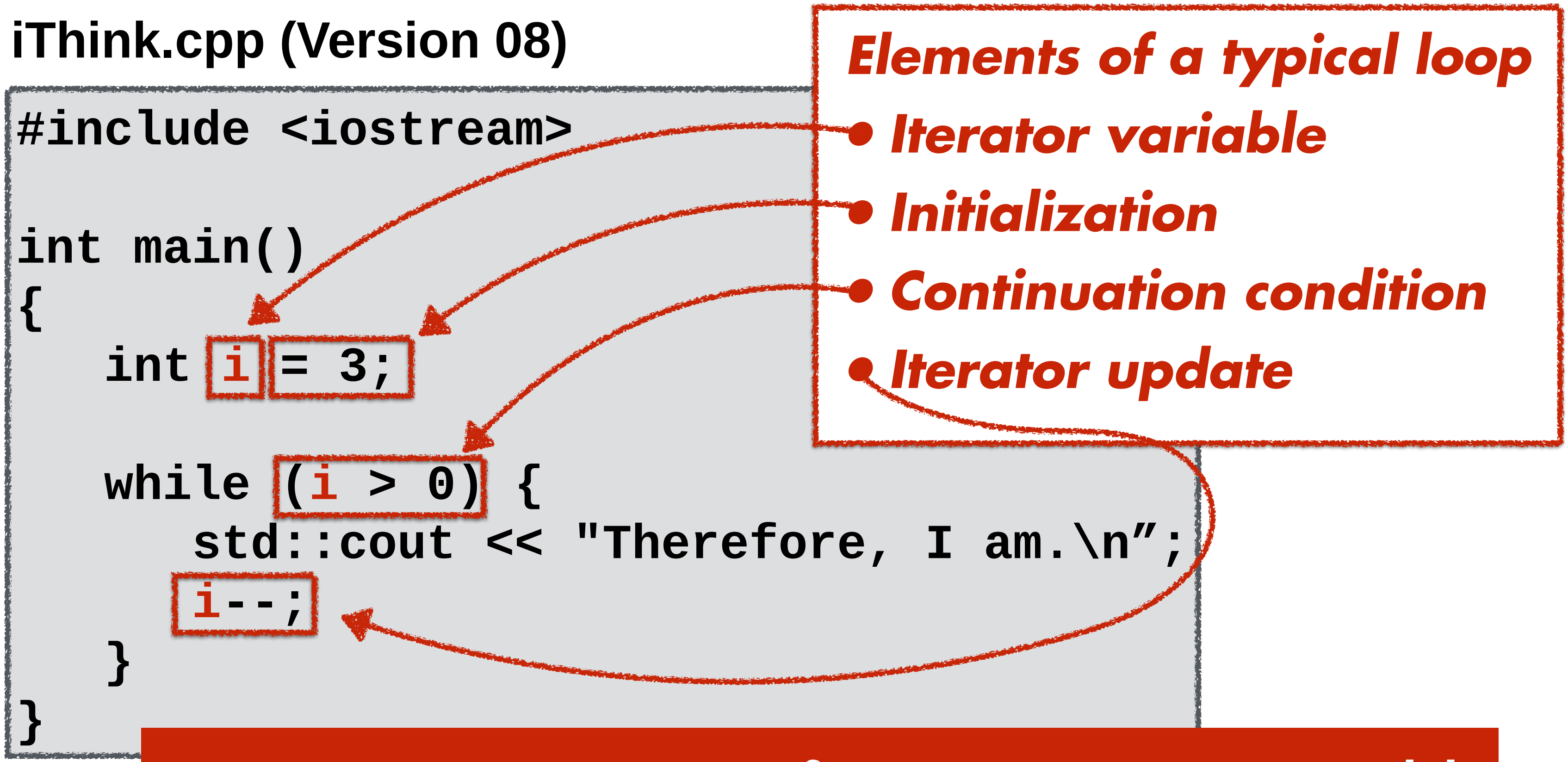


```
#include <iostream>

int main()
{
    int thinkingAmount = 3;

    while (thinkingAmount > 0) {
        std::cout << "Therefore, I am.\n”;
        thinkingAmount--;
    }
}
```

Elements of a typical loop

- *Iterator variable*
- *Initialization*
- *Continuation condition*
- *Iterator update*

iThink.cpp (Version 08)



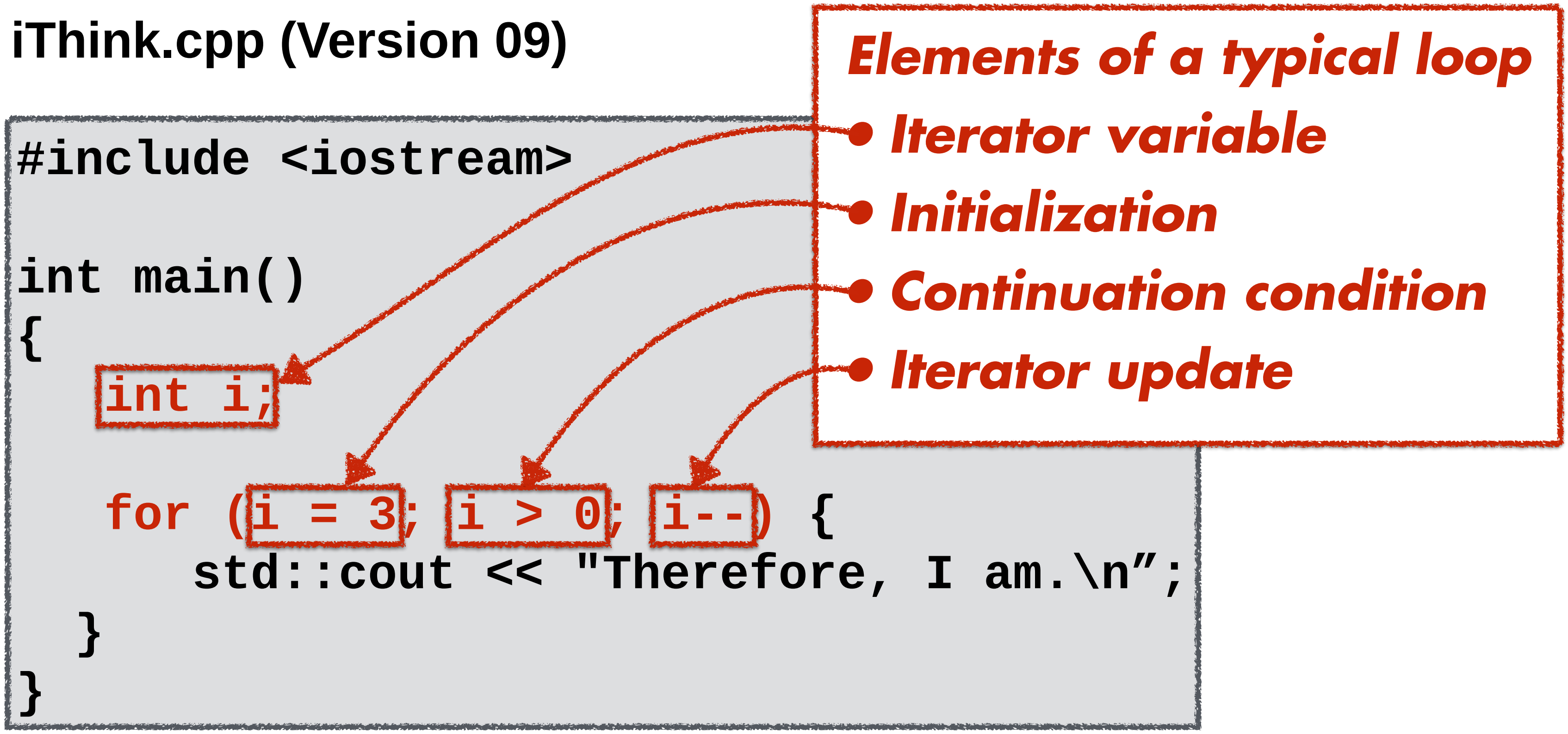
```
#include <iostream>

int main()
{
    int i = 3;

    while (i > 0) {
        std::cout << "Therefore, I am.\n”;
        i--;
    }
}
```

Elements of a typical loop

- **_Iterator variable_**
- **_Initialization_**
- **_Continuation condition_**
- **_Iterator update_**

`i` *is a common name for an iterator variable*

iThink.cpp (Version 09)



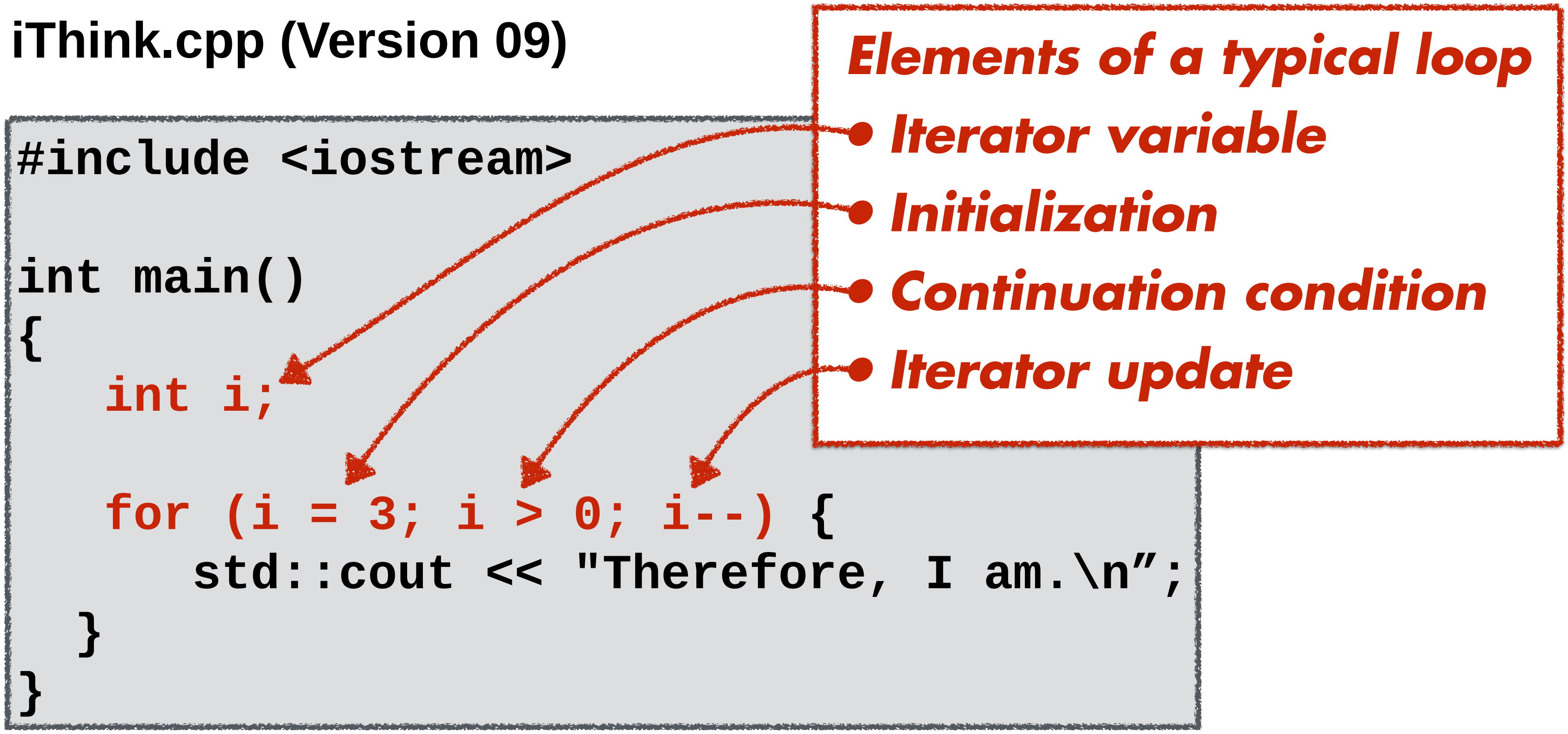
```
#include <iostream>

int main()
{
   int i;

   for (i = 3; i > 0; i--) {
      std::cout << "Therefore, I am.\n”;
   }
}
```

Elements of a typical loop

- *Iterator variable*
- *Initialization*
- *Continuation condition*
- *Iterator update*

[redacted]

iThink.cpp (Version 09)



```
#include <iostream>

int main()
{
    int i;

    for (i = 3; i > 0; i--) {
        std::cout << "Therefore, I am.\n”;
  }
}
```

Elements of a typical loop

- *Iterator variable*
- *Initialization*
- *Continuation condition*
- *Iterator update*

[redacted]

iThink.cpp (Version 09)

```
#include <iostream>

int main()
{
   int i;

   for (i = 3; i > 0; i--) {
      std::cout << "Therefore, I am.\n”;
  }
}
```

```
Therefore, I am.
Therefore, I am.
Therefore, I am.
```

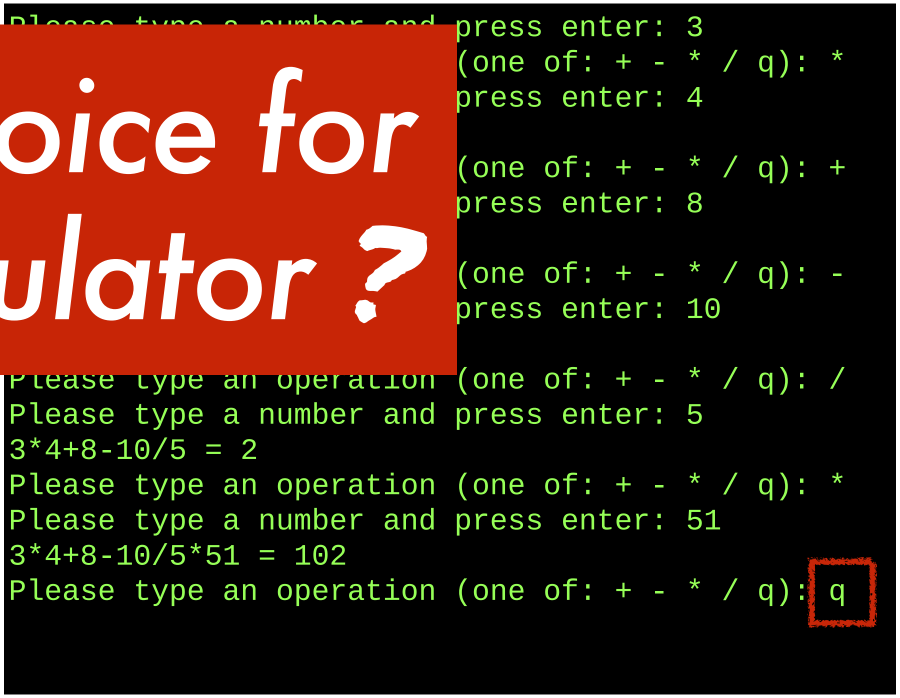
Still correct output

How do we loop in C++ ?

```
for (i=3;i>0;i--) {
  // for loop
}
```

```
i = 3;
while (i > 0) {
  // while loop
  i--;
}
```

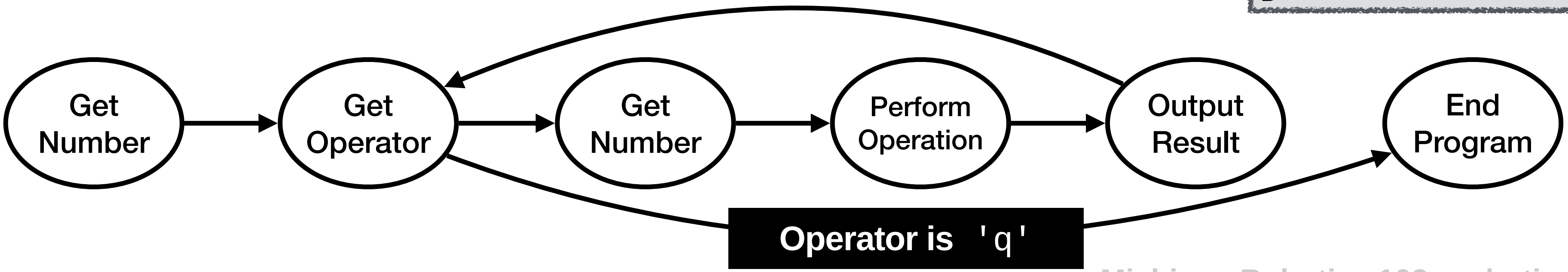
Right choice for our calculator ?



```
Please type a number and press enter: 3
                         (one of: + - * / q): *
                         press enter: 4

                         (one of: + - * / q): +
                         press enter: 8

                         (one of: + - * / q): -
                         press enter: 10

Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

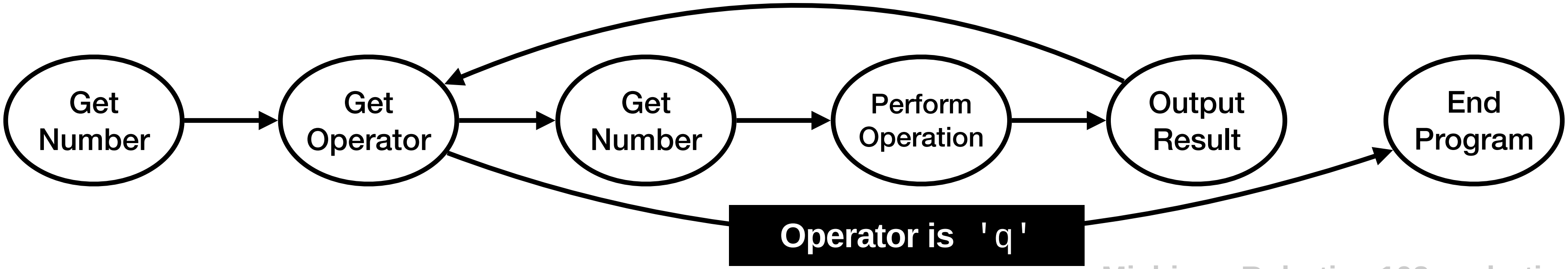
```
for (i=3;i>0;i--) {
  // for loop
}
```

```
i = 3;
while (i > 0) {
  // while loop
  i--;
}
```



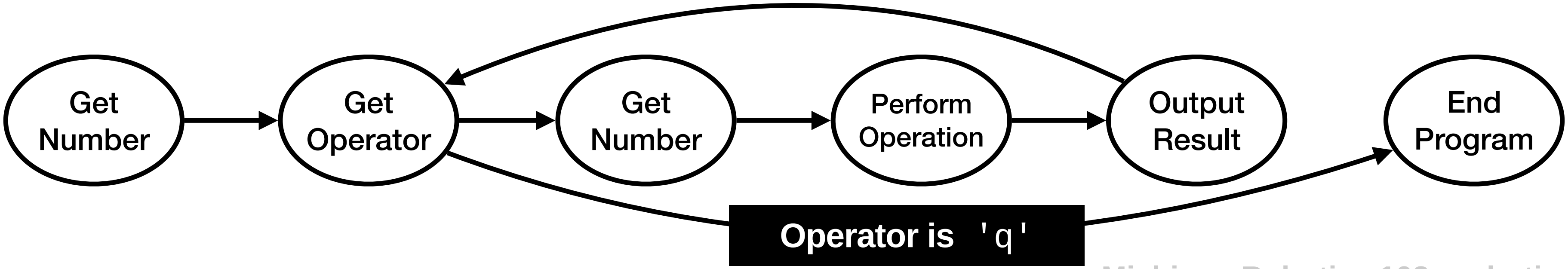
Right choice for our calculator ?

```
i = 3;
while (i > 0) {
  // while loop
  i--;
}
```



calculator.cpp (Version 54) - Condensed version

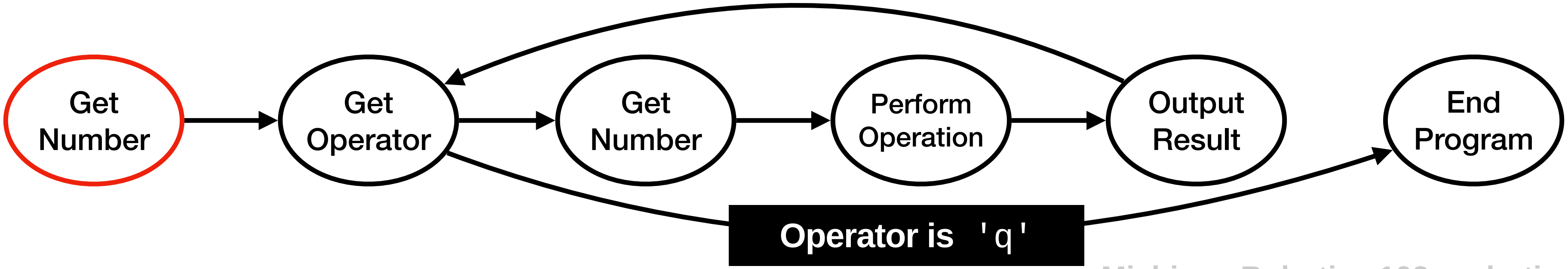
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed version

Current point in Program execution

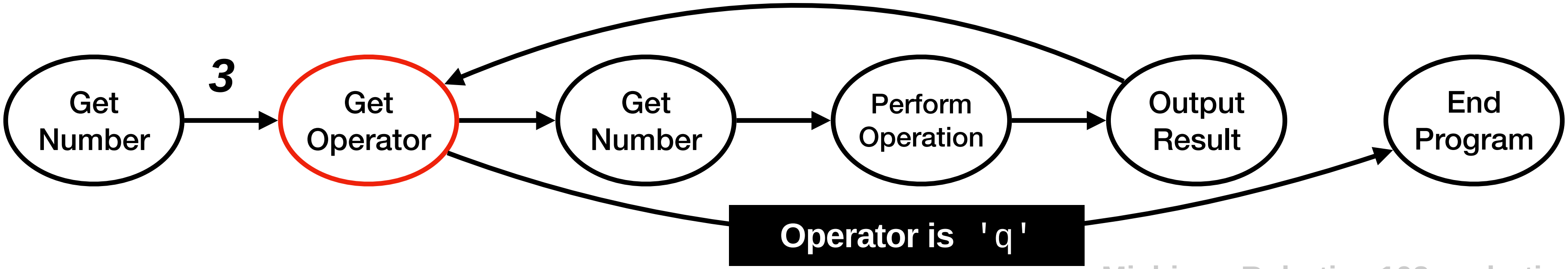
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed version

Current point in Program execution

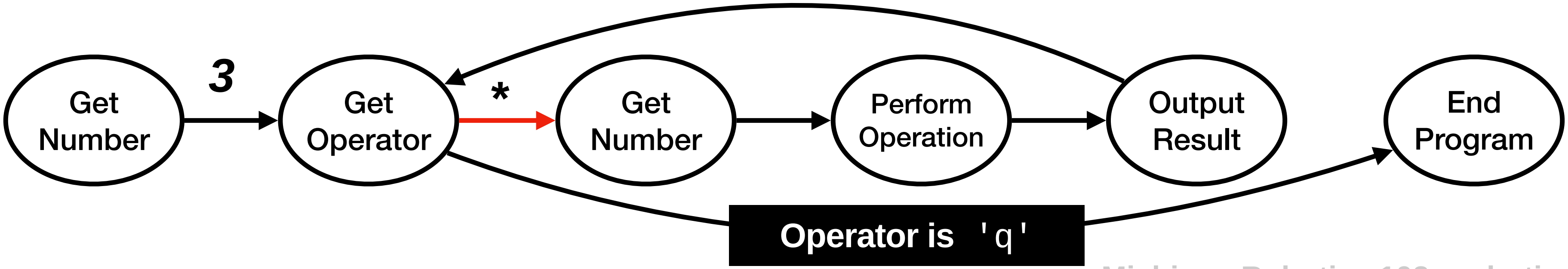
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed version

Current point in Program execution →

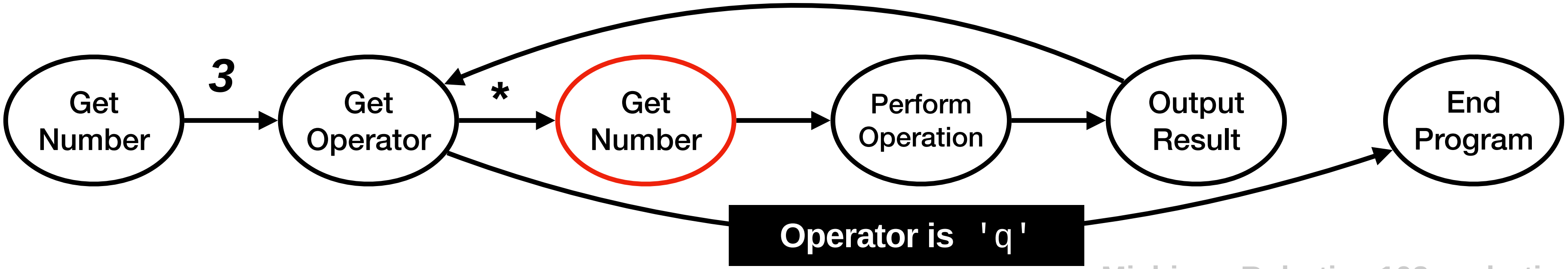
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

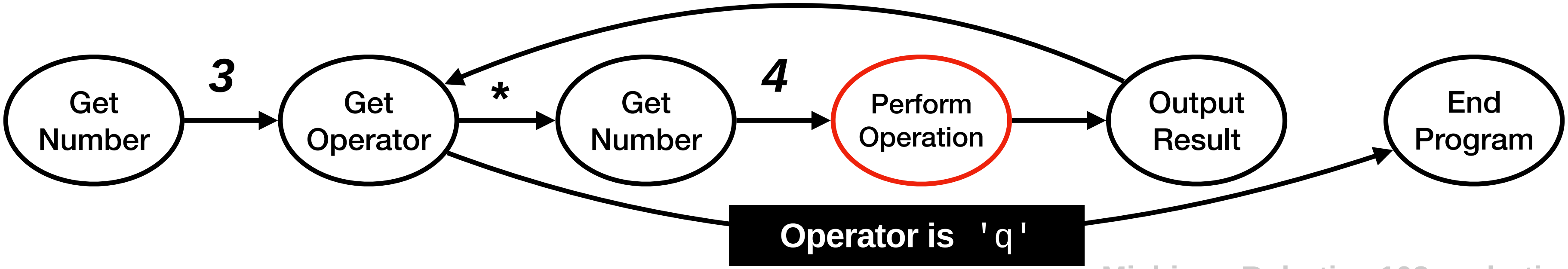
Current point in Program execution →



calculator.cpp (Version 54) - Condensed version

Current point in Program execution →

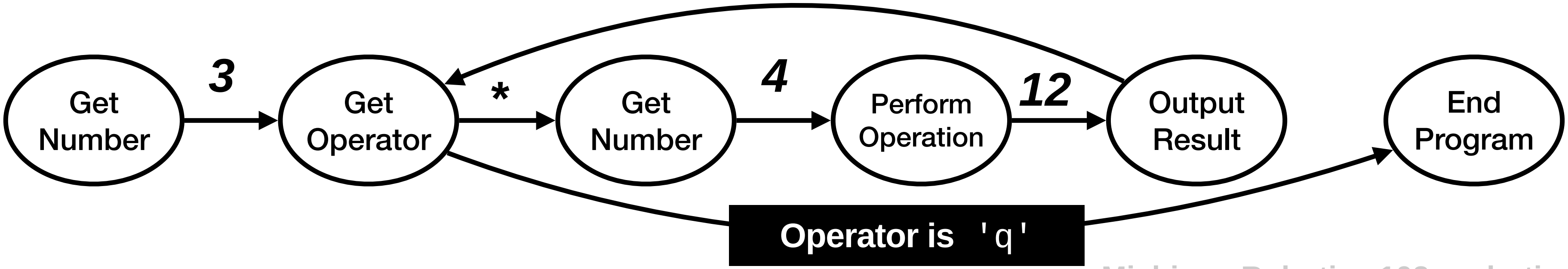
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

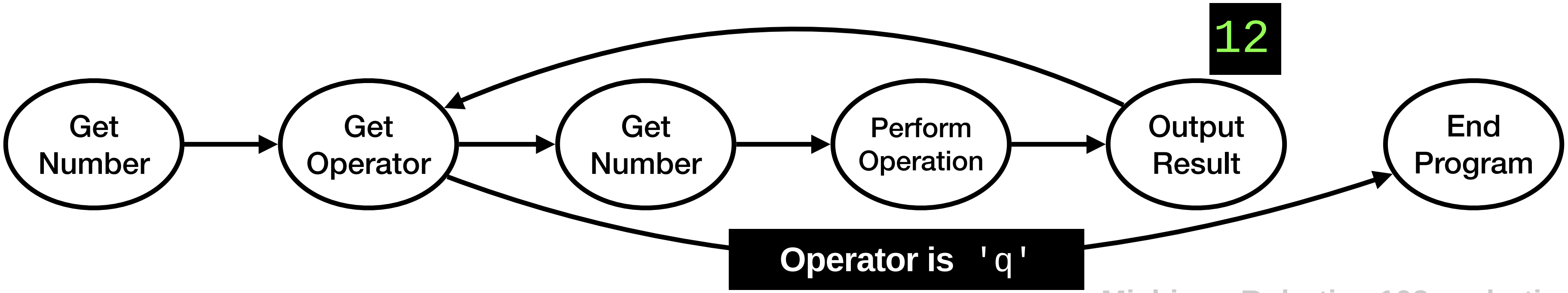
Current point in Program execution



calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

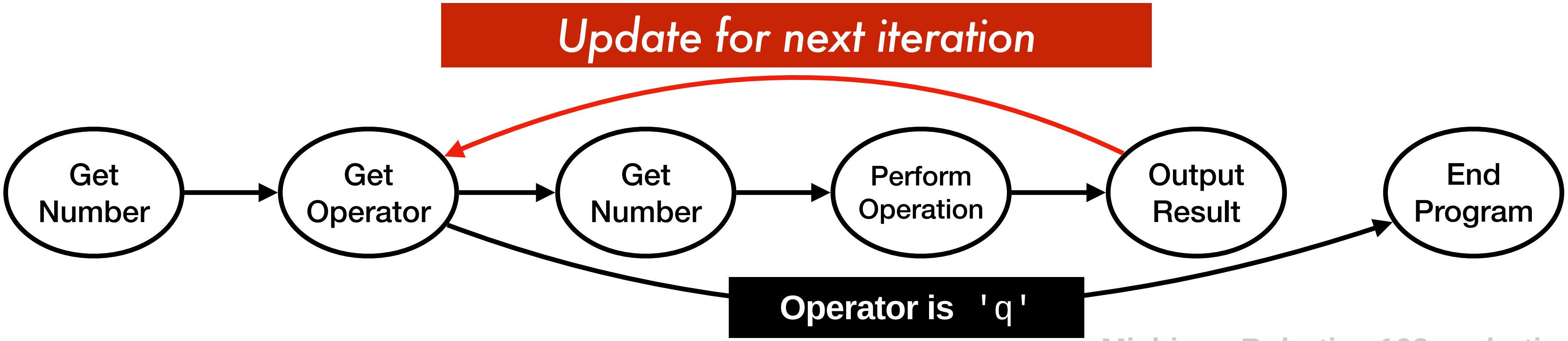
Current point in Program execution →



calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

Current point in Program execution



Update for next iteration

Get Number

Get Operator

Get Number

Perform Operation

Output Result

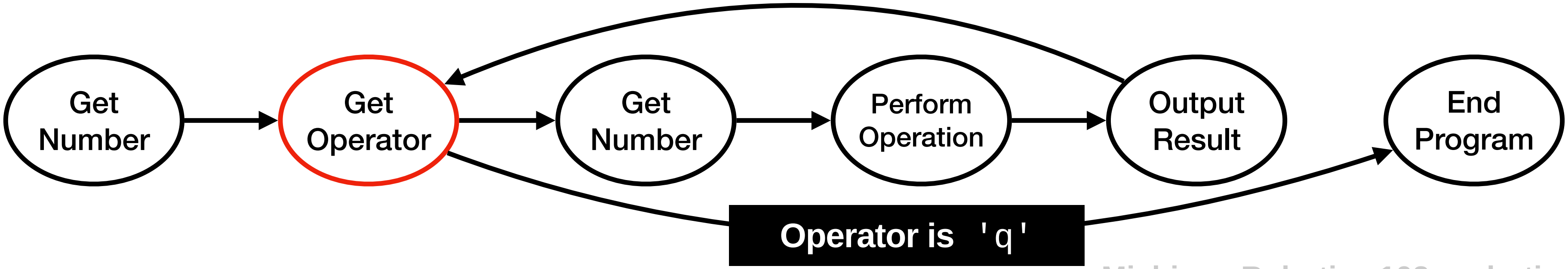
End Program

Operator is `'q'`

calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

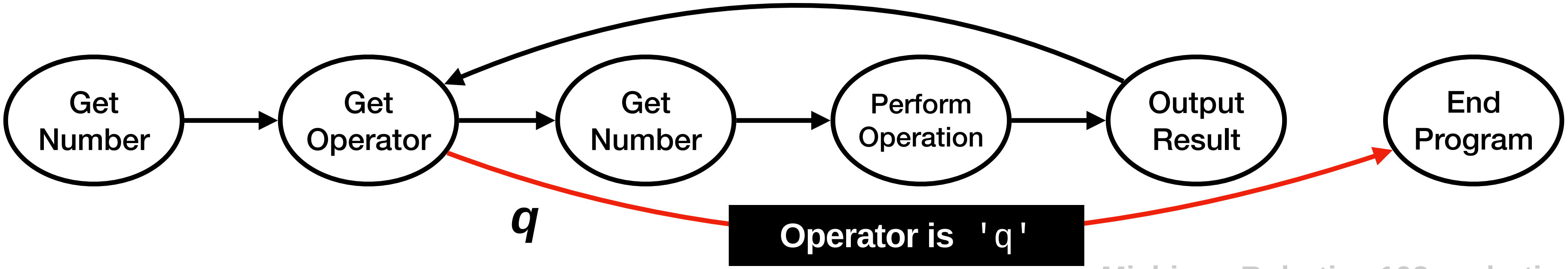
Current point in Program execution



calculator.cpp (Version 54) - Condensed version

Current point in Program execution

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



Get Number

Get Operator

Get Number

Perform Operation

Output Result

End Program

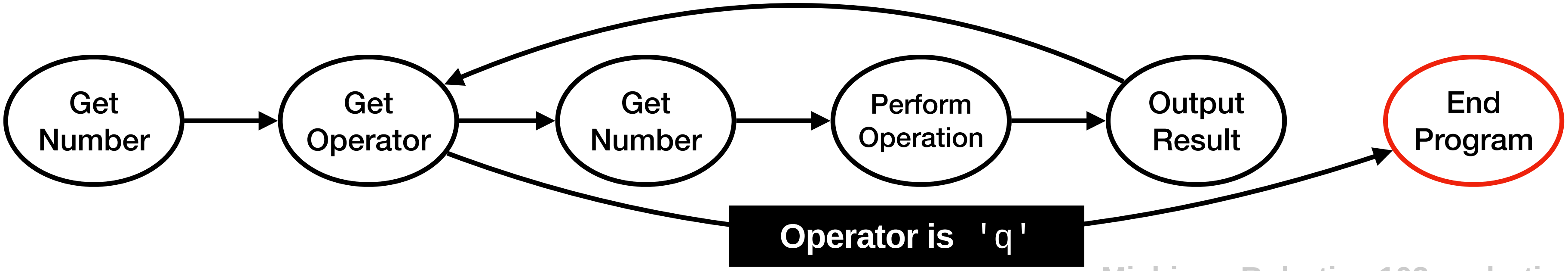
q

Operator is `'q'`

calculator.cpp (Version 54) - Condensed version

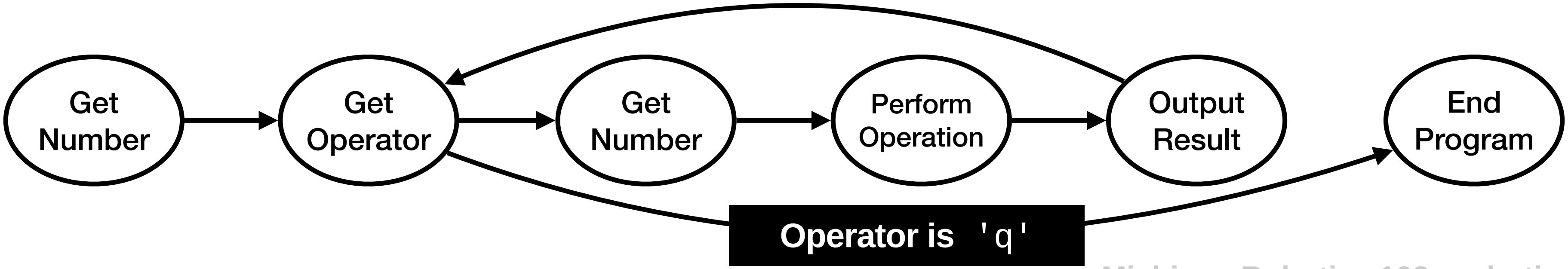
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```

Current point in Program execution



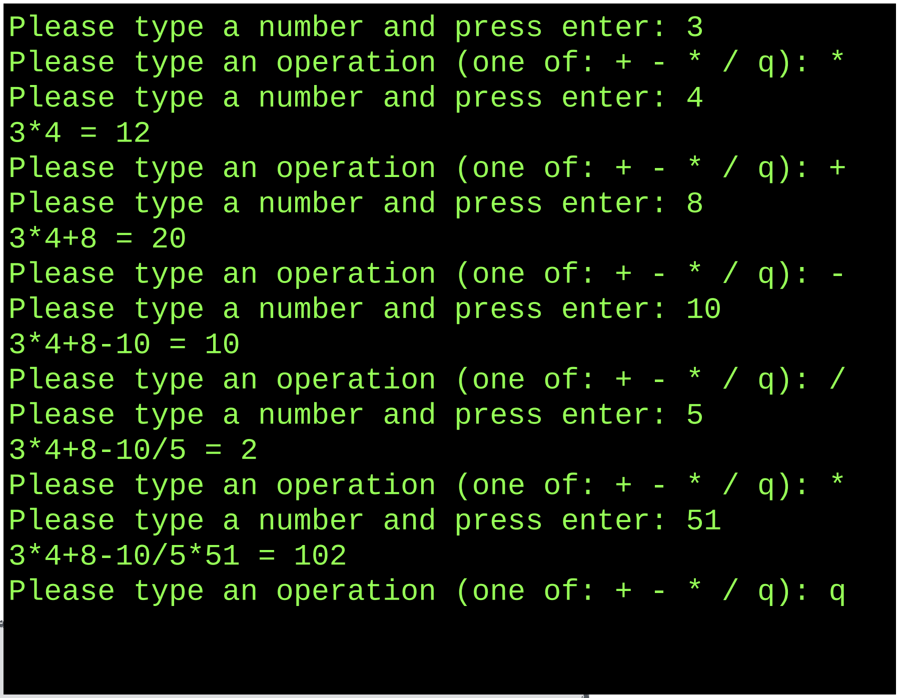
calculator.cpp (Version 54) - Condensed version

```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



calculator.cpp (Version 54) - Condensed

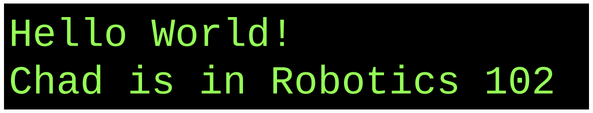
```
getNumber(myOtherNumber);
getOperator(myOperator);
while (myOperator != 'q') {
   getNumber(myOtherNumber);
   performOperation(myNumber,myOperator,myOtherNumber,resultNumber);
   outputResult(myNumber,myOperator,myOtherNumber,resultNumber);
   myNumber = resultNumber;
   getOperator(myOperator);
}
```



```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

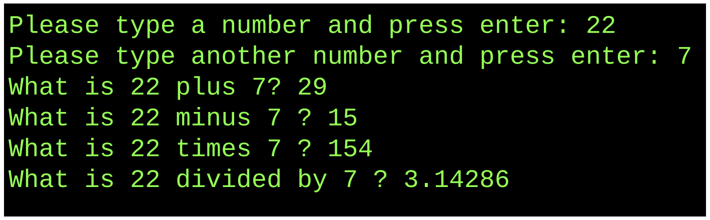
Done

hello



```
Hello World!
Chad is in Robotics 102
```

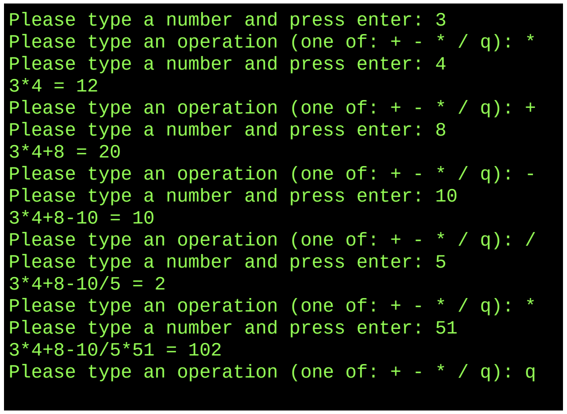
calculator (Version 24)



```
Please type a number and press enter: 22
Please type another number and press enter: 7
What is 22 plus 7? 29
What is 22 minus 7 ? 15
What is 22 times 7 ? 154
What is 22 divided by 7 ? 3.14286
```

calculator (Version 41)

calculator (Version 54)



```
Please type a number and press enter: 3
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 4
3*4 = 12
Please type an operation (one of: + - * / q): +
Please type a number and press enter: 8
3*4+8 = 20
Please type an operation (one of: + - * / q): -
Please type a number and press enter: 10
3*4+8-10 = 10
Please type an operation (one of: + - * / q): /
Please type a number and press enter: 5
3*4+8-10/5 = 2
Please type an operation (one of: + - * / q): *
Please type a number and press enter: 51
3*4+8-10/5*51 = 102
Please type an operation (one of: + - * / q): q
```

☑ Program Structure

☑ Compile/Execute

☑ Operators

☑ Data Types

☑ Variables

☑ User Input/Output

☑ Functions

☑ Branching

☑ Iterators

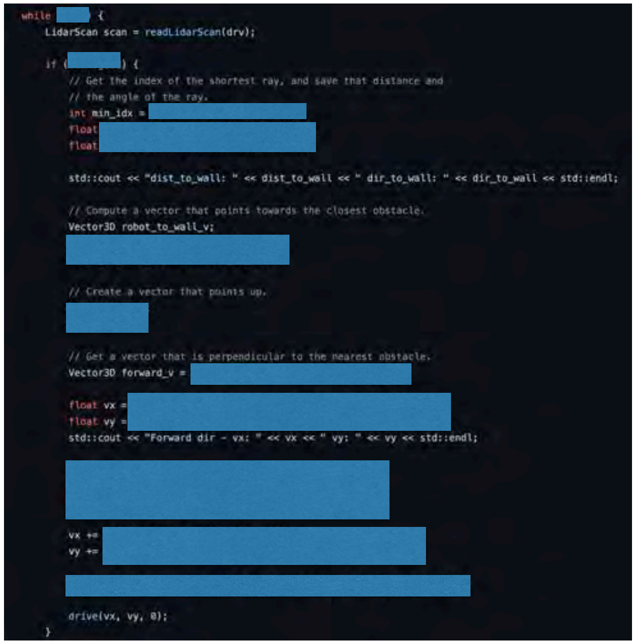
□ Vectors

□ Structs

□ File Input/Output

Coming

wall_follower.cpp - Project 1

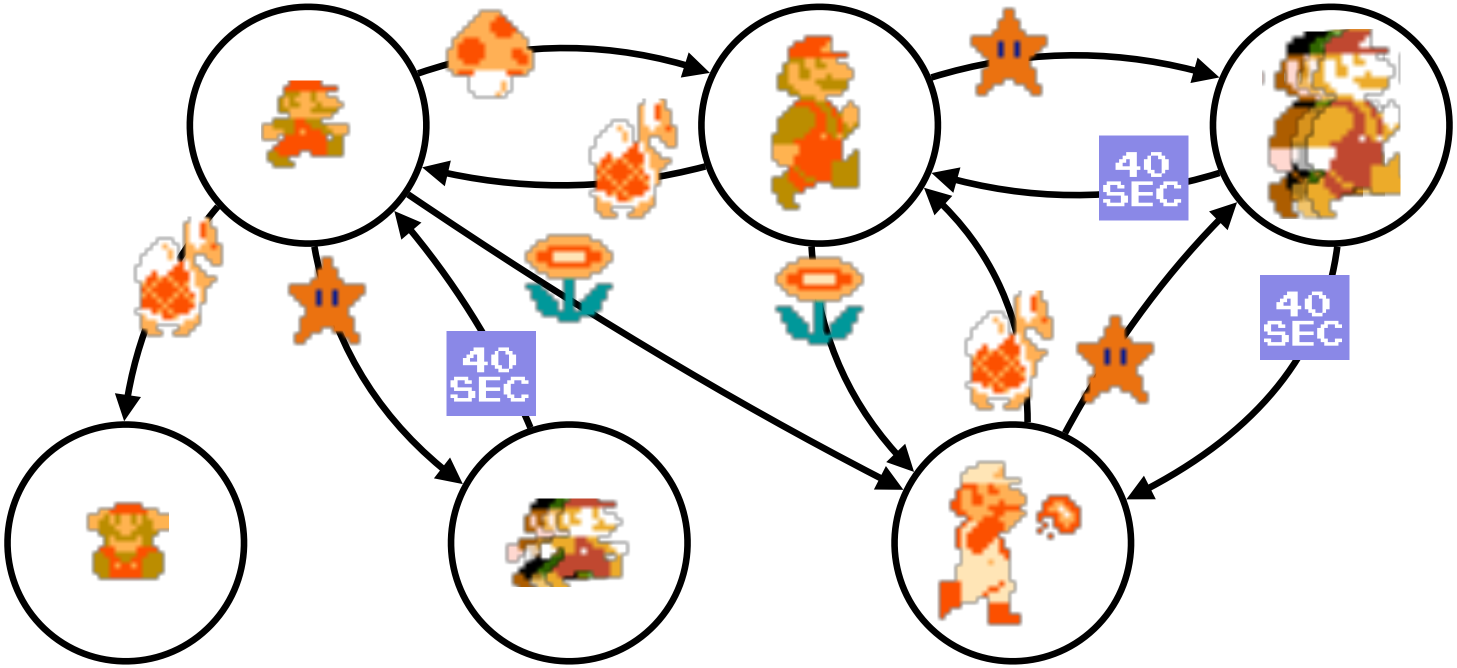


```
while [redacted] {
    LidarScan scan = readLidarScan(drv);

    if ([redacted]) {
        // Get the index of the shortest ray, and save that distance and
        // the angle of the ray.
        int min_idx = [redacted]
        float [redacted]
        float

        std::cout << "dist_to_wall: " << dist_to_wall << " dir_to_wall: " << dir_to_wall << std::endl;

        // Compute a vector that points towards the closest obstacle.
        Vector3D robot_to_wall_v;
        [redacted]

        // Create a vector that points up.
        [redacted]

        // Get a vector that is perpendicular to the nearest obstacle.
        Vector3D forward_v = [redacted]

        float vx = [redacted]
        float vy =
        std::cout << "Forward dir - vx: " << vx << " vy: " << vy << std::endl;

        [redacted]

        vx += [redacted]
        vy +=

        [redacted]

        drive(vx, vy, 0);
    }
```

Can we keep a history of operations?

Can we undo the last operation?

This `3*4+8-10/5*51 = 102` does not look right



BRANCHING AND ITERATORS

ROBOTICS 102
INTRO AI & PROOGRAMMING

FALL 2021
UNIVERSITY OF MICHIGAN