CSC 211: Computer Programming

Introducing loops (for)

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Summer 2025

THINK BIG WE DO℠

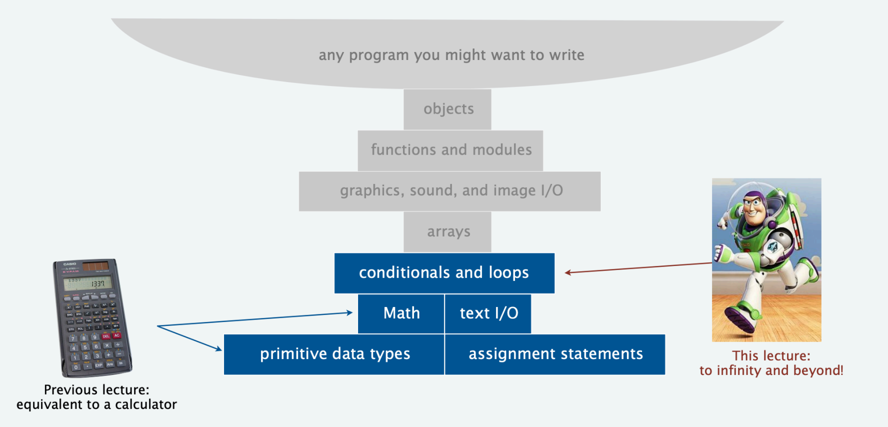
Administrative Announcements

- Exam#01 ~ Thursday 10/10
  - ✓ Same time / place as lecture
  - ✓ One 11x8 notes sheet
  - ✓ Calculator allowed
- A01 Due 09/29 (only 20 submissions?)
- MC02 Due 09/29

Administrative Announcements

- Exam#01 ~ Tuesday 02/21
  - ✓ Same time / place as lecture
  - ✓ One 11x8 notes sheet
  - ✓ Calculator allowed
- A01 Due 02/19
- MC03 due 02/21

Basic building blocks



any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

Math

text I/O

primitive data types

assignment statements

Previous lecture:
equivalent to a calculator

This lecture:
to infinity and beyond!

https://introcs.cs.princeton.edu/java/lectures/

Flowchart of `if` statements

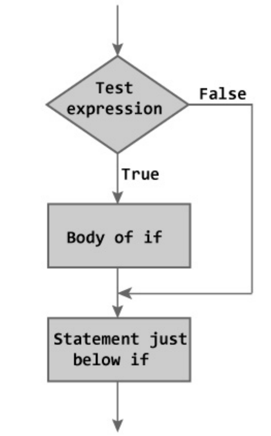


Figure: Flowchart of if Statement

```
// ...
// statements above
// ...

if (test_expression) {
    // body of if
}

// ...
// statements below
// ...
```

https://www.programiz.com/cpp-programming/if-else

Flowchart of `if` statements

```
// ...
// statements above
// ...

if (test_expression) {
    // body of if
} else {
    // body of else
}

// ...
// statements below
// ...
```

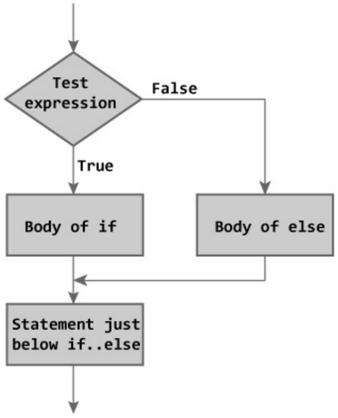
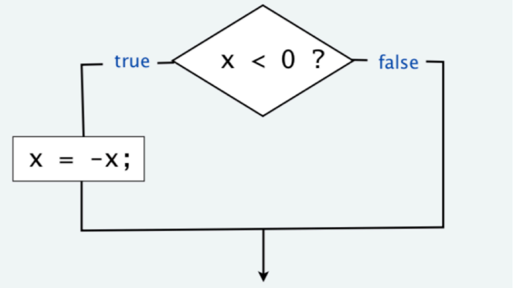


Figure: Flowchart of if...else Statement

https://www.programiz.com/cpp-programming/if-else

`if` statement examples

Example: `if (x < 0) x = -x;`



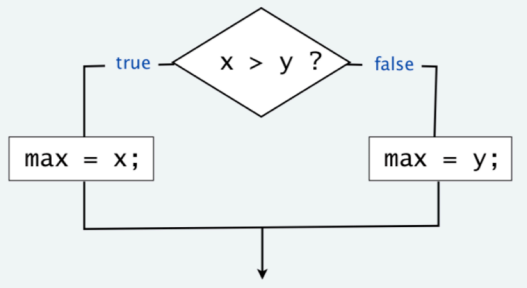
Replaces x with the absolute value of x

https://introcs.cs.princeton.edu/java/lectures/

`if` statement examples

Example:

```
if (x > y) max = x;
else       max = y;
```



Computes the maximum of x and y

https://introcs.cs.princeton.edu/java/lectures/

The increment / decrement operators

- Increment (**++**) and decrement (**––**) are **unary** operators that add or subtract one, to or from their operand, respectively
  - ✓ **pre-increment** and **pre-decrement** operators increment (or decrement) their operand by **1**, and the value of the expression is the resulting incremented (or decremented) value
  - ✓ **post-increment** and **post-decrement** operators increase (or decrease) the value of their operand by **1**, but the value of the expression is the operand's original value prior to the increment (or decrement) operation

from: wikipedia

The increment / decrement operators

- Example:

```
int a = 5;
```

```
std::cout << 5 + ++a
```

v.s

```
int a = 5;
```

```
std::cout << 5 + a++;
```

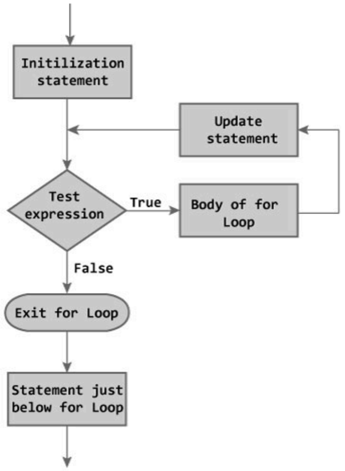
Trace the code

```
int  x;
int  y;

x = 1;

y = ++x;

// Checkpoint a (status of x and y?)

y = x++;

// Checkpoint b (status of x and y?)

x = 5;

y = x––;

// Checkpoint c (status of x and y?)

y = ––x;

// Checkpoint d (status of x and y?)
```

from: wikipedia

the `for` loop

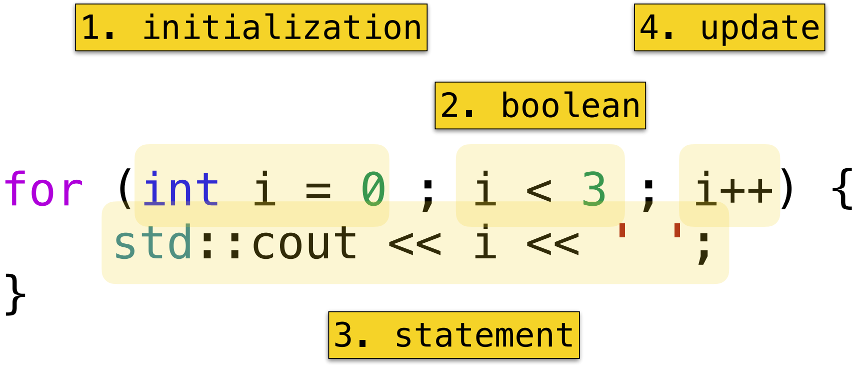
Flowchart of `for` statement



```
// ...
// statements above
// ...

for (init ; test ; update) {
    // body of for
}

// ...
// statements below
// ...
```

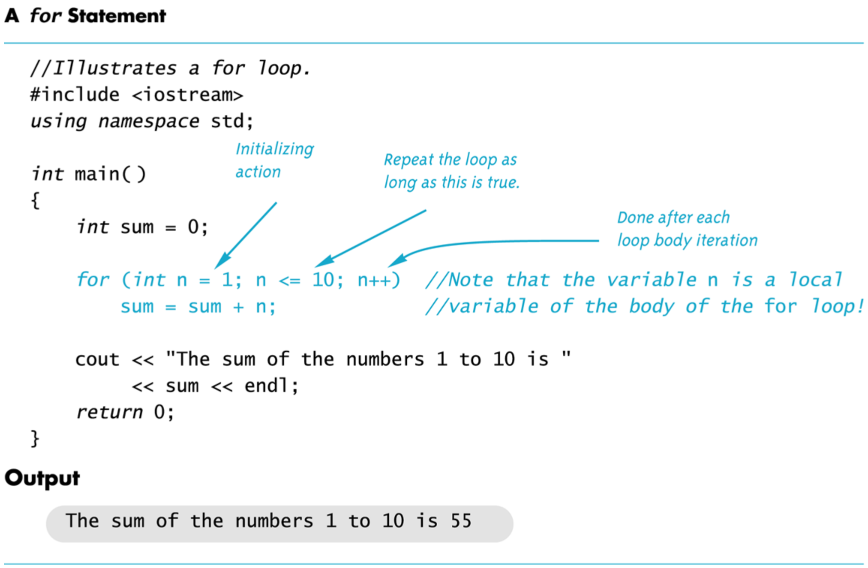
Figure: Flowchart of for Loop

https://www.programiz.com/cpp-programming/for-loop



```
for (int i = 0 ; i < 3 ; i++) {
    std::cout << i << ' ';
}
```

then go back to step 2



A *for* Statement

```
//Illustrates a for loop.
#include <iostream>
using namespace std;

int main()
{
    int sum = 0;

    for (int n = 1; n <= 10; n++)  //Note that the variable n is a local
        sum = sum + n;             //variable of the body of the for loop!

    cout << "The sum of the numbers 1 to 10 is "
         << sum << endl;
    return 0;
}
```

Initializing action
Repeat the loop as long as this is true.
Done after each loop body iteration

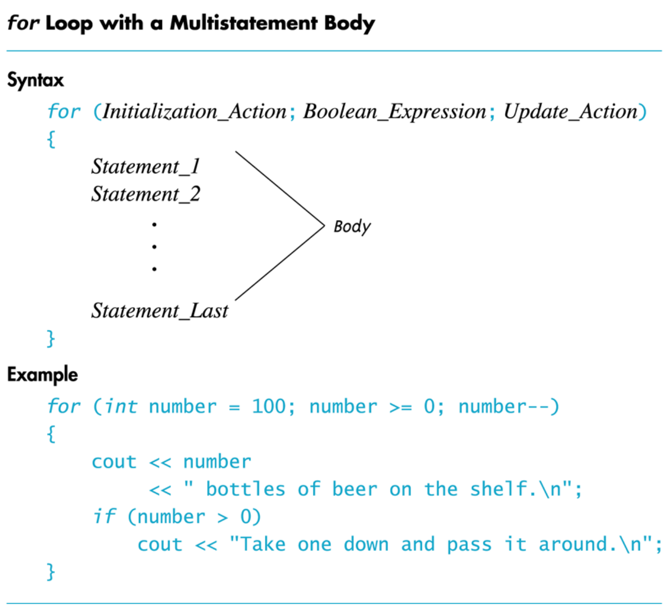
Output

```
The sum of the numbers 1 to 10 is 55
```

from: Problem Solving with C++, 10th Edition, Walter Savitch

What is the output?

```
int value = 0;

for (int i = 0 ; i < 5 ; i++) {
    value += (i * 10);
}

std::cout << value << std::endl;
```



***for* Loop with a Multistatement Body**

Syntax

```
for (Initialization_Action; Boolean_Expression; Update_Action)
{
     Statement_1
     Statement_2
        .
        .
        .
     Statement_Last
}
```

Example

```
for (int number = 100; number >= 0; number--)
{
     cout << number
          << " bottles of beer on the shelf.\n";
     if (number > 0)
         cout << "Take one down and pass it around.\n";
}
```

from: Problem Solving with C++, 10th Edition, Walter Savitch

Examples

```
int n = 1;

for ( ; n <= 10 ; n = n + 2)
    std::cout << n << std::endl;

for (n = 10 ; n > 0 ; n -= 2) std::cout << n << std::endl;

for (n = 0 ; n > -30 ; n = n - 7) {
    std::cout << n << std::endl;
}

for (double x = 16.0 ;  x >= 2.0 ;  x = sqrt(x)) {
    std::cout << x;
    std::cout << std::endl;
}
```

What is the output?

```
for (int count = 1 ; count <= 10 ; count++);
    std::cout << "Hello\n";
```

Careful with the semi-colon

- Semi-colon is used to end statements
- Placing it after the parenthesis of a `for` loop creates an **empty statement**

Different output?

```
for (int count = 1 ; count <= 10 ; count++){
    std::cout << count << std::endl;
}
```

V.S

```
for (int count = 1 ; count <= 10 ; ++count){
    std::cout << count << std::endl;
}
```

Question

- Write a single for loop to print the first 50 even numbers

Question

- Write a single for loop to print the average of the first 25 multiples of 3