CSCI 200: Foundational Programming Concepts & Design Lecture 05

Repetition using Loops

Open VS Code, iClicker,
Download lecture example code,
and Find a Partner!

Programming Task #3

- Complete Programming Task #3
  - Run program with sample inputs below to verify results
  - Correct the condition, using `1e-6` as a tolerance
  - See https://bit.ly/comparefloats for reference

Sample Input	Expected Out
`2`	`sqrt(2) squared is 2`
`3`	`sqrt(3) squared is 3`
`4`	`sqrt(4) squared is 4`

Comparing Floats Error

```
double num1 = 0.3 * 3.0 + 1.0;
double num2 = 1.0;


if( num1 == num2 ) {
    cout << "it's true! :)" << endl;
} else {
    cout << "it's false :(" << endl;
}
```

Comparing Floats FIXED

```
double num1 = 0.3 * 3.0 + 1.0;
double num2 = 1.0;
const double EPSILON = 1e-6;

if( fabs(num1 - num2) < EPSILON ) {
    cout << "it's true! :)" << endl;
} else {
    cout << "it's false :(" << endl;
}
```

Tip #5!

Compare floats for equality against an error tolerance

Clarifying a small note

- && vs & || vs | == vs =
- && and || are logical operators, eval T / F
- ~ & | ^ are bitwise operators

  Take integers and convert to binary

  Perform NOT AND OR XOR operation bit by bit

6 0110
~6 == 9 1001

7 0111	4 0100	6 0110
13 1101	2 0010	3 0011
7 & 13 == 5 0101	4 | 2 == 6 0110	6 ^ 3 == 5 0101

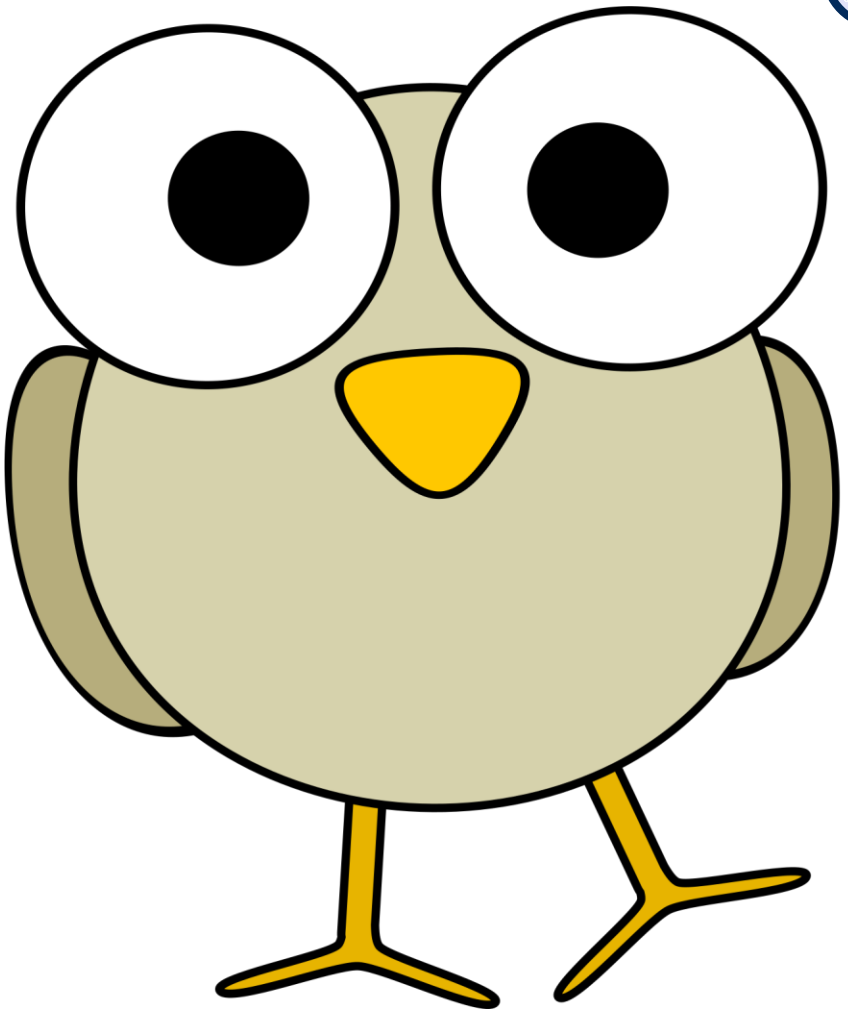
Precedence Table

Category	Precedence	Operator	Associativity
Parenthesis	1	( )	Innermost First
Unary Operators	2	+a -a ~a (type)a !a	Right to Left
Binary Operators	3	a*b a/b a%b	Left to Right
	4	a+b a-b	
Relational Operators	5	a<b a>b a<=b a>=b	
	6	a==b a!=b	
Bitwise Operators	7	a&b	
	8	a^b	
	9	a|b	
Logical Operators	10	a&&b	
	11	a||b	
Assignment Operators	12	a=b a+=b a-=b a*=b a/=b a%=b a&=b a^=b a|=b	Right to Left

Additional Clarification

- Despite `and` and `or` being C++ keywords...
- Use `&&` and `||` for logical operators
  - Be explicit about which and/or you want
    - Logical, not bitwise
- Logical operators return a Boolean
- Bitwise operators return an integer

Questions?



Learning Outcomes For Today

- Identify C++ repetition structures and explain the following terms: looping parameter, stopping condition, and looping parameter modification.
- Explain the appropriate use and differences between a while loop, for loop, and a do-while loop.
- Convert a program written with a for loop to a program using a while loop and vice versa.

On Tap For Today

- Repetition Structures
- Scope
- Practice

On Tap For Today

- Repetition Structures
- Scope
- Practice

Turing Machine

- Given infinite time and memory, if a machine has the following features:
  1. Sequence
  2. Control
  3. Iteration
  4. Output
  5. Input
  6. Variables
- It can solve any mathematical problem

Three Questions To Ask With Loops

1. What is the initial value of the looping parameter?
2. What condition must be met for the looping sequence to execute? (What condition causes the loop to exit?)
3. How is the looping parameter modified in the looping sequence? (What happens if it doesn't change?)

Three Kinds of Loops

- Condition at the start
  - `while` loop
    - When you're not sure how many times to loop
  - `for` loop
    - When you know how many times to loop
- Condition at the end
  - `do-while` loop
    - When you need to loop at least once

These are rules of thumb, other scenarios exist, and all can be used equivalently

while Loop

```
01 int x = 0;
02 while( x < 100 ) {
03    cout << "What am I doing?" << endl;
04 }
```

while Loop

```
01 int x = 0;
02 while( x < 100 ) {
03    cout << "I'll run 100 times!" << endl;
04    x++;
05 }
```

Parts of a `for` Loop

```
int cookies;
for( cookies = 0; cookies <= 8; ++cookies ) {
    Loop Body
    cout << "I ate " << cookies
         << " cookies." << endl;
}
```

Looping Parameter

Stopping Condition

Parameter Modification

Increment / Decrement

- Another shorthand
  - Increment

```
x++; ++x;  |  x += 1;  |  x = x + 1;
```

  - Decrement

```
x--; --x;  |  x -= 1;  |  x = x - 1;
```

Increment / Decrement

- Two versions

Postfix	Prefix	
`x++;`	`++x;`	`x = x + 1;`
`x--;`	`--x;`	`x = x - 1;`

Order Matters!

```
int x, y;
```

Postfix

```
x = 5;
y = x++;
```

x	y
6	5

Prefix

```
x = 5;
y = ++x;
```

x	y
6	6

Precedence Table

Category	Precedence	Operator	Associativity
Parenthesis	1	( )	Innermost First
Postfix Unary Operators	2	a++ a--	Left to Right
Prefix Unary Operators	3	++a --a +a -a !a ~a (type)a	Right to Left
Binary Operators	4	a*b a/b a%b	Left to Right
	5	a+b a-b	
Relational Operators	6	a<b a>b a<=b a>=b	
	7	a==b a!=b	
Bitwise Operators	8	a&b	
	9	a^b	
	10	a|b	
Logical Operators	11	a&&b	
	12	a||b	
Assignment Operators	13	a=b a+=b a-=b a*=b a/=b a%=b a&=b a^=b a|=b	Right to Left

Practice

```
int x, y;
x = 10;
y = ++x - 3;
cout << y << endl;
```

```
x = x + 1;
```

`x == 11`

```
y = x - 3;
```

`y == 8`

```
int a, b;
a = 10;
b = a++ - 3;
cout << b << endl;
```

```
b = a - 3;
```

`b == 7`

```
a = a + 1;
```

`a == 11`

On Tap For Today

- Repetition Structures
- Scope
- Practice

Scope

- Determines where variables can be referenced
  - Referenceable in ALL code blocks
    - Global Scope
  - Referenceable in A SINGLE code block
    - Local Scope

Global Scope

- Variables that are available anywhere in our program that follows the definition
  - Defined above **`main()`** outside all code blocks

```
#include <iostream>
using namespace std;

const double PI_CONSTANT = 3.14159;

int main() {
    double area = 5.0 * 5.0 * PI_CONSTANT;
    return 0;
}
```

Local Scope

```
int main()
{// start scope 1
  int x = 4, z = 6;
  cout << x << endl;    // prints 4
  int x = 5;            // error! redefinition of x
  if( true )
  {// start scope 2
    int z = 2;          // warning, "shadows" prior declaration
    int y = 3;
    cout << x << endl; // prints 4
    cout << z << endl; // prints 2
  }// end scope 2
  cout << z << endl;    // prints 6
  cout << y << endl;    // error! y undeclared
  return 0;
}// end scope 1
```

Local Scope & `for` Loops

- Can define looping parameter in the `for` loop declaration

```
int main() { // begin code block 1
    int x = 0;
    // begin CB 2
    for( x = 0; x < 10; ++x ) {
        cout << x << endl;  // OK :)
    } // end CB 2
    cout << x << endl;  // OK :)
    return 0;
} // end code block 1
```

```
int main() { // begin code block 1
    // begin CB 2
    for( int x = 0; x < 10; ++x ) {
        cout << x << endl;  // OK :)
    } // end CB 2
    cout << x << endl;  // ERROR!  :(
    return 0;
} // end code block 1
```

On Tap For Today

- Repetition Structures
- Scope
- Practice

Practice

```
01 int sum(0);
02 for( int i = 1; i <= 10; ++i ) {
03     sum += i;
04 }
05 cout << sum;
```

Practice

```
01 int sum(0);
02 for( int i = 1; i <= 10; i += 2 ) {
03     sum += i;
04 }
05 cout << sum;
```

Practice

```
01 int count(0);
02 for( int i = 3; i <= 5; ++i ) {
03   for( int j = 10; j >= 5; --j ) {
04     count++;
05   }
06 }
07 cout << count;
```

Practice

```
01 int count(0);
02 for( int i = 0; i <= 5; ++i ) {
03   for( int j = 0; j < i; ++j ) {
04     count++;
05   }
06 }
07 cout << count;
```

Programming Task

- Fix the five loop errors

Section	Sample Input	Expected Output
I	4	10
I	6	21
II	3 7 2 8	17
II	4 2 3 8 10	23
III	5	15
III	7	28
IV	4	30
IV	7	140
V	3	36
V	5	225

To Do For Next Time

- Wednesday
  - L1B due before class
- Thursday
  - A1 due at 11:59 PM
- Friday:
  - Structured Programming Quiz in class