CSCI 200: Foundational Programming Concepts & Design Lecture 04

Conditionals

Open Canvas 8/30 Quiz To Follow Along

Access Code:

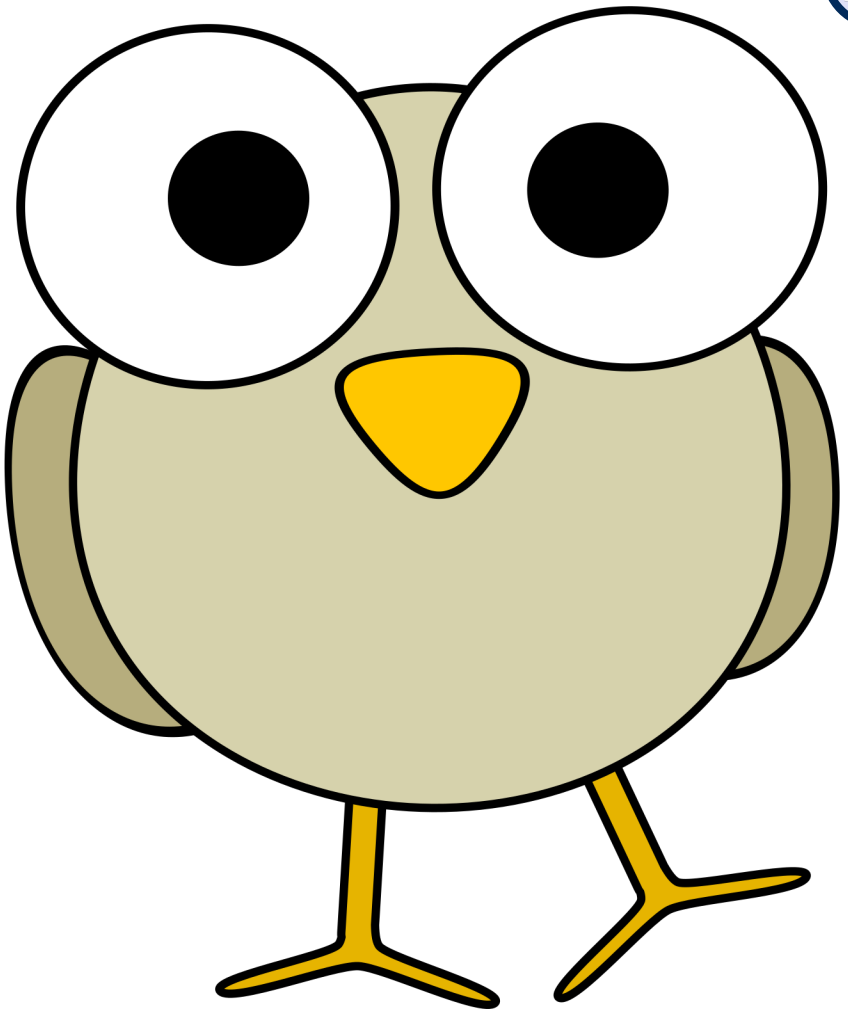
Previously in CSCI 200

- Random Number Generation
  1. Seed RNG with current time
     - Do this once per program
  2. Throw away first `rand()` value
  3. Call `rand()` for a random integer
     - Need to manipulate value into desired range & type

Previously in CSCI 200

- Build Process
  - Compile each cpp file separately
  - Link all object files and libraries
- Makefile
  - Batches build commands
  - Only rebuilds files that have changed since last build

Questions?



Learning Outcomes For Today

- Identify C++ control structures and conclude which branch a sample program will execute.
- List C++ logic operators and evaluate Boolean expressions consisting of multiple logic operators.
- Evaluate the resultant output of a code block containing a control structure.

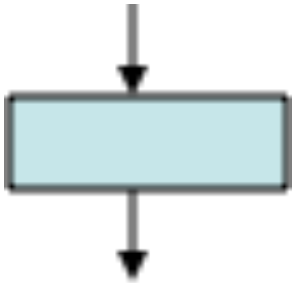
On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

Statement Types



Sequential

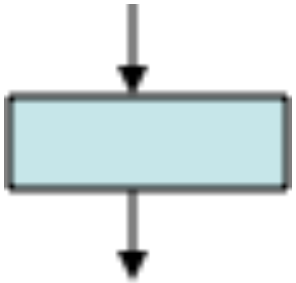
Imperative Programming

- Explicit sequence of steps to perform one at a time
  - Shows how the computation takes place
- Each step changes the **state** of the program
  - **state** comprised of stack information
    - Current line of execution
    - Variables that are in scope

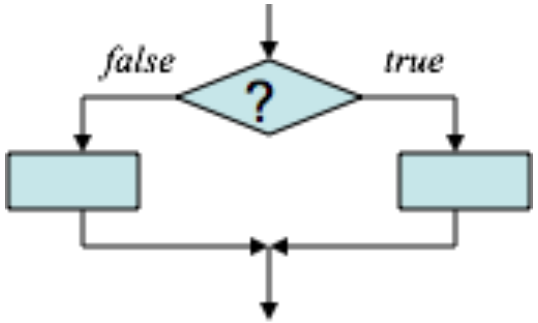
Imperative Programming

```
int main() {
  int sum = 0;
  sum += 1;
  sum += 2;
  sum += 3;
  sum += 4;
  sum += 5;
  sum += 6;
  sum += 7;
  sum += 8;
  sum += 9;
  sum += 10;
  cout << "The sum is: " << sum << endl;
  return 0;
}
```

Statement Types



Sequential



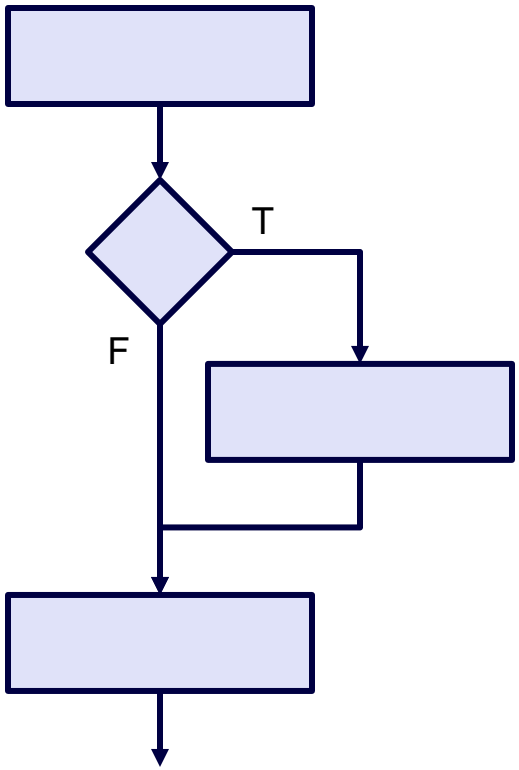
Conditional

Structured Programming

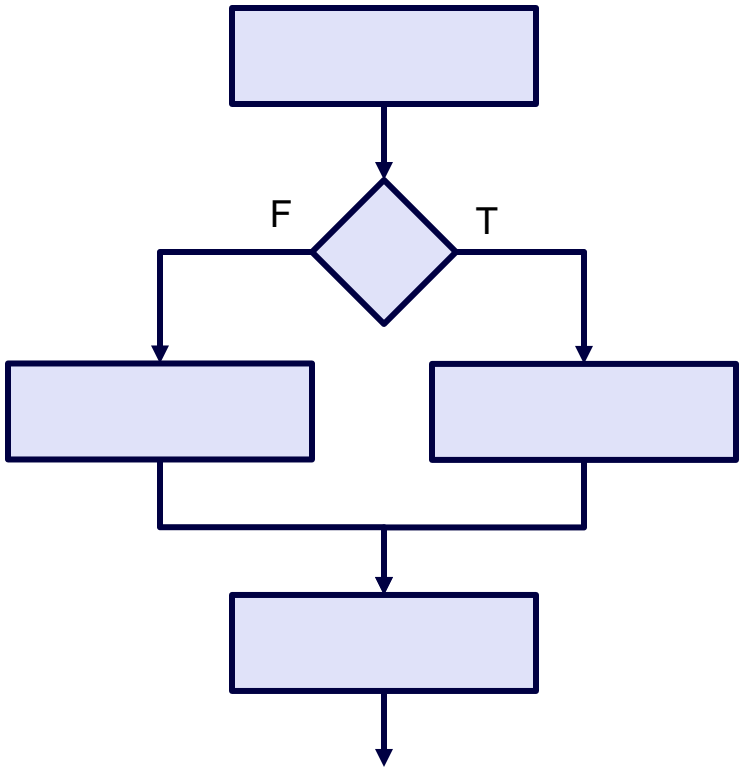
- Imperative Programming where flow is defined by control structures (e.g. conditionals, loops)

```
int main() {
  int sum = 0;
  cout << "Enter sum: ";
  cin >> sum;
  if(sum > 0) {
    cout << "Sum is positive" << endl;
  } else {
    cout << "The sum is: " << sum << endl;
  }
  return 0;
}
```

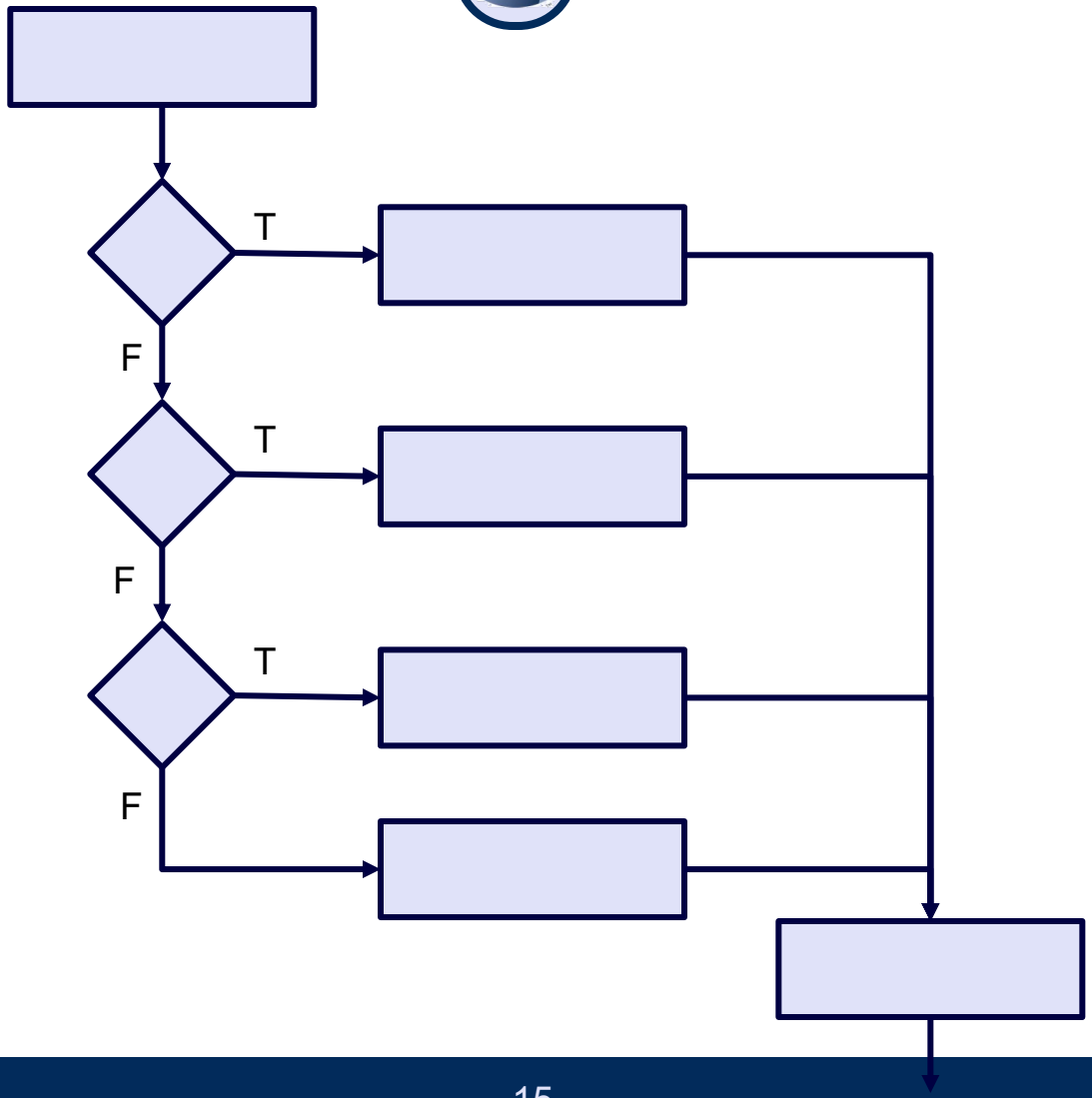
`if` Program Flow



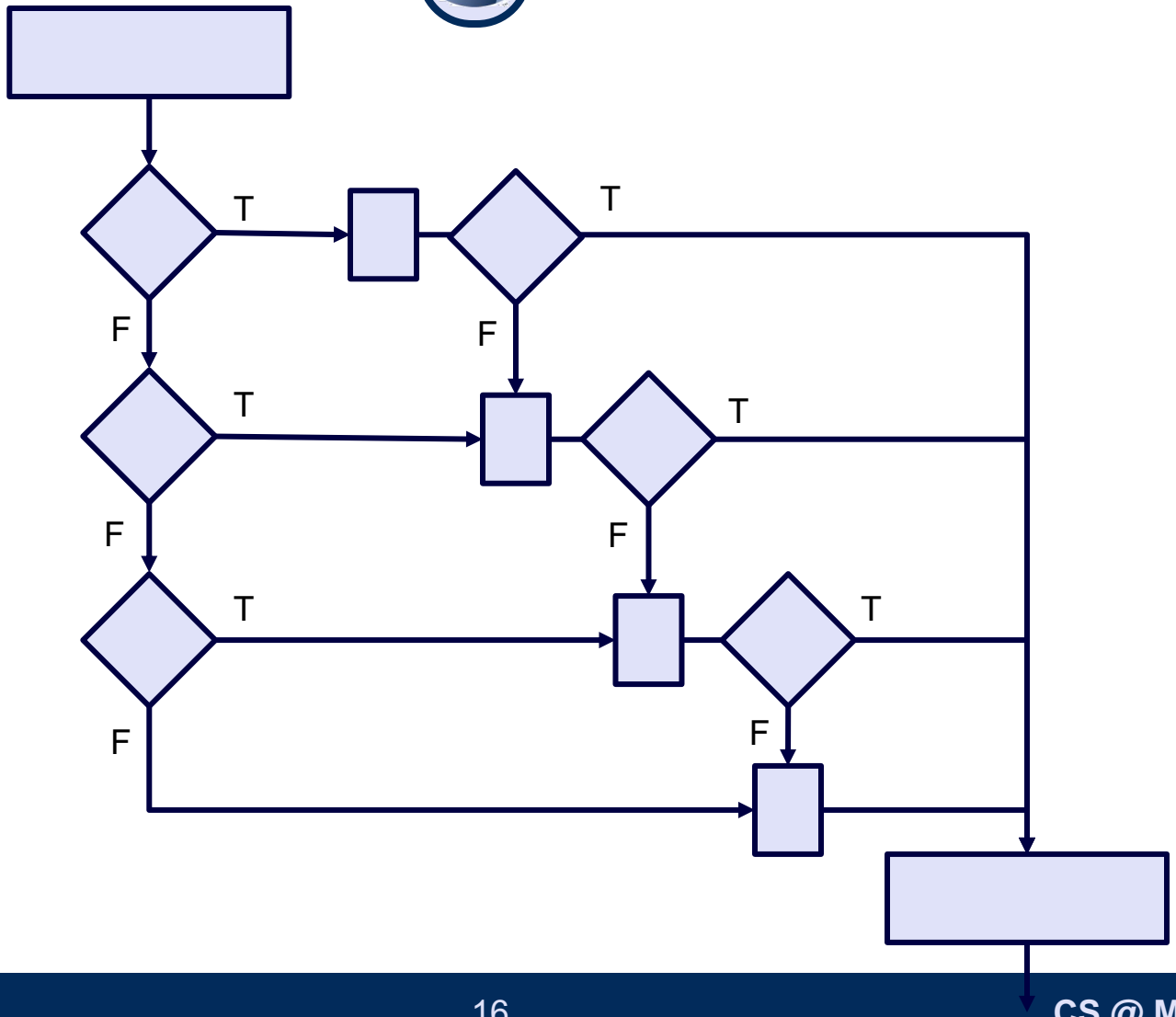
`if`/`else` Program Flow



`if/else if/else` Program Flow



`switch` Program Flow



On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

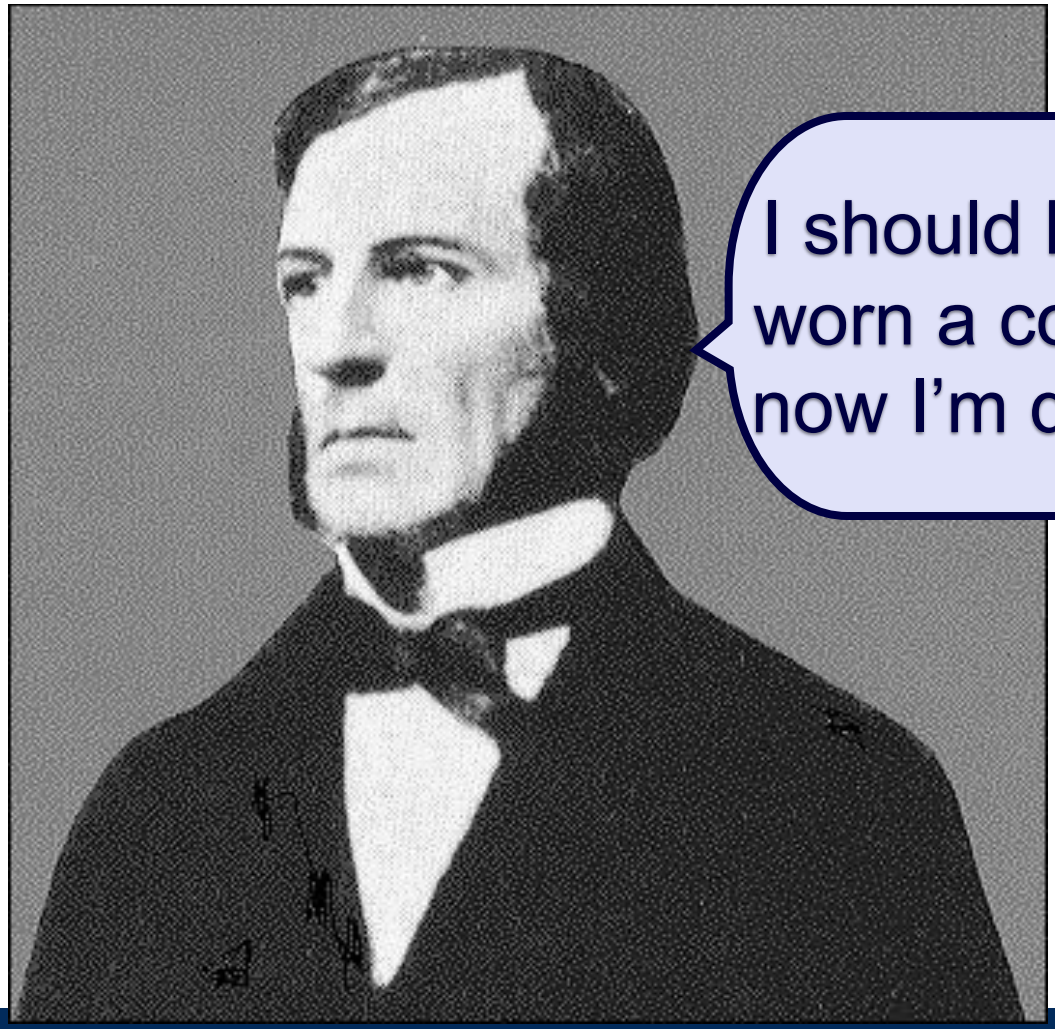
Making a Decision

- Need to teach the computer to make a decision
- Use
  - Boolean Logic

    +

    Control Structures

George Boole (1815 – 1864)



On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

if Syntax

```
if( condition )
    statement;


// only one statement
// so no code block needed
```

if Syntax

```
if( condition ) {
    statement;
}
// only one statement
// so no code block needed
// but code block IS recommended
```

if Syntax

```
if( condition ) {
    statementOne;
    statementTwo;
    statementThree;
    ...
}
```

if / else Syntax

```
if( condition )
    statementOne;
else
    statementTwo;


// one statement each
// no code blocks needed
```

if / else Syntax

```
if( condition ){
    statementOne;
} else {
    statementTwo;
}
// but is recommended
```

if / else Syntax

```
if( condition ){
    statementOne;
    statementTwo;
} else {
    statementThree;
    statementFour;
    statementFive;
}
```

Nested if else

```
if( condition1 ) {
    statementOne;
} else {
    if( condition2 ) {
        statementTwo;
    } else {
        if( condition7 ) {
            statementThree;
        } else {
            statementFour;
        }
    }
}
```

if / else if / else Syntax

```
if( condition1 ) {
   statementOne;
} else if( condition2 ) {
   statementTwo;
} else if( condition7 ) {
   statementThree;
} else {
   statementFour;
}
// same code block rules as before
```

On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

Conditions

- A boolean value
  - A true / false value

- The result of a boolean expression
  - A test using a relational operator
  - **`variableOne == variableTwo`**
  - **`valueOne < valueTwo`**

Relational Operators

- Perform a test between two values
- Result is a boolean value

Operator	Test
==	Equality
!=	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Precedence Table

Precedence	Operator	Associativity
1	Parenthesis: ()	Innermost First
2	Unary Operators: +a -a (type)a	Right to Left
3	Binary Operators: a*b a/b a%b	Left to Right
4	Binary Operators: a+b a-b	Left to Right
5	Relational Operators: a<b a>b a<=b a>=b	Left to Right
6	Relational Operators: a==b a!=b	Left to Right
7	Assignment Operators: a=b a+=b a-=b a*=b a/=b a%=b	Right to Left

What Gets Printed?

```
int x = 15;


if( x < 100 ) {
    cout << "x is less than 100" << endl;
} else {
    cout << "x is greater than or equal to 100" << endl;
}
```

Practice - What Gets Printed?

```
int x = 15;


if( x + 95 < 100 ) {
    cout << "x is less than 100" << endl;
} else {
    cout << "x is greater than or equal to 100" << endl;
}
```

Practice - What Gets Printed?

```
int x = 15;


if( x < 100 ) {
    cout << "alligator";
}
if( x < 50 ) {
    cout << "crocodile";
}
```

Practice - What Gets Printed?

```
int x = 15;
if( x < 20 )
    cout << "apple";
    if( x < 10 )
        cout << "banana";
else
    cout << "orange";
```

White Space Doesn't Matter

- else always belongs to the closest if behind it

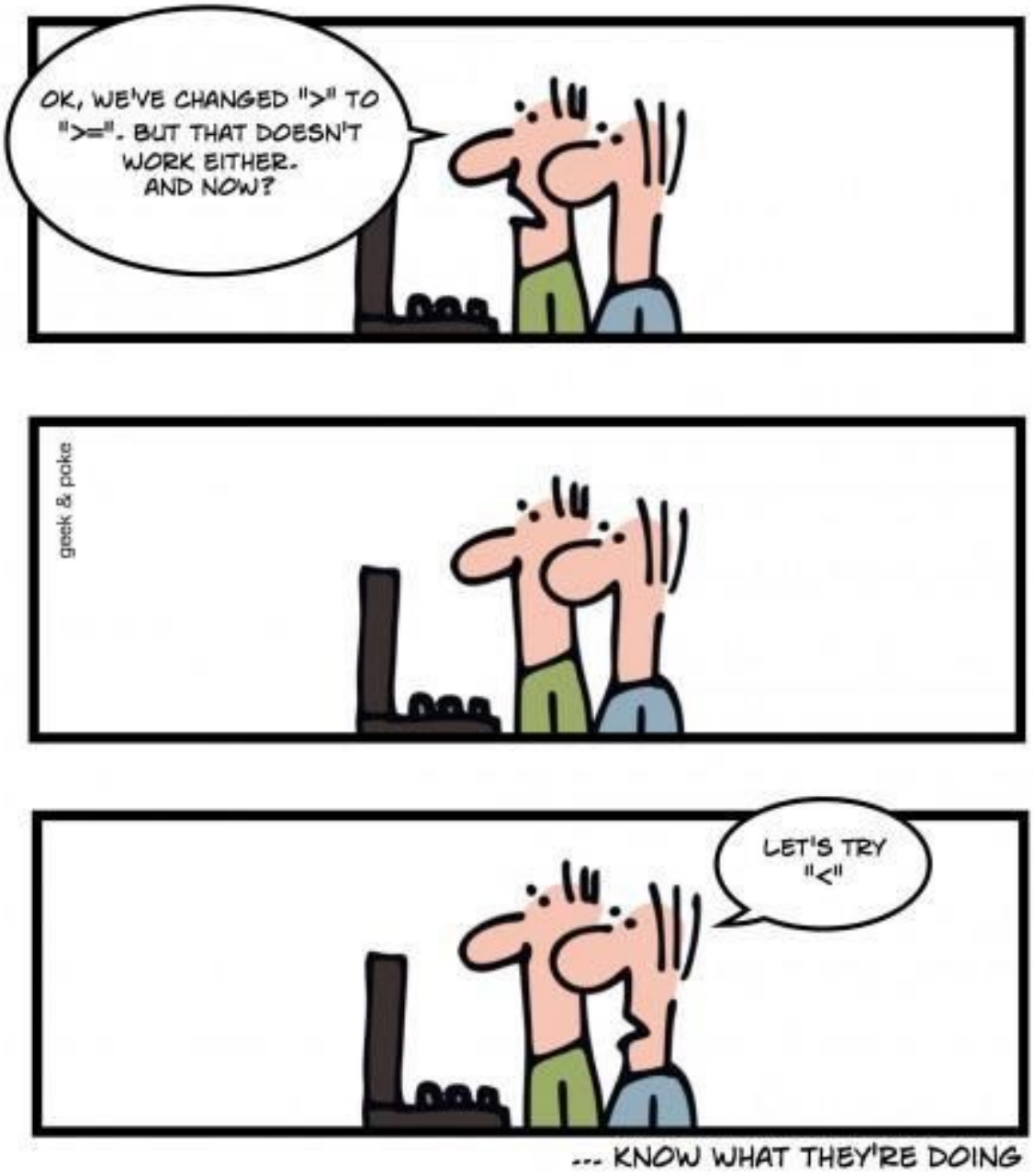
```
int x = 15;
if( x < 20 )
    cout << "apple";
if( x < 10 )
    cout << "banana";
else
    cout << "orange";
```

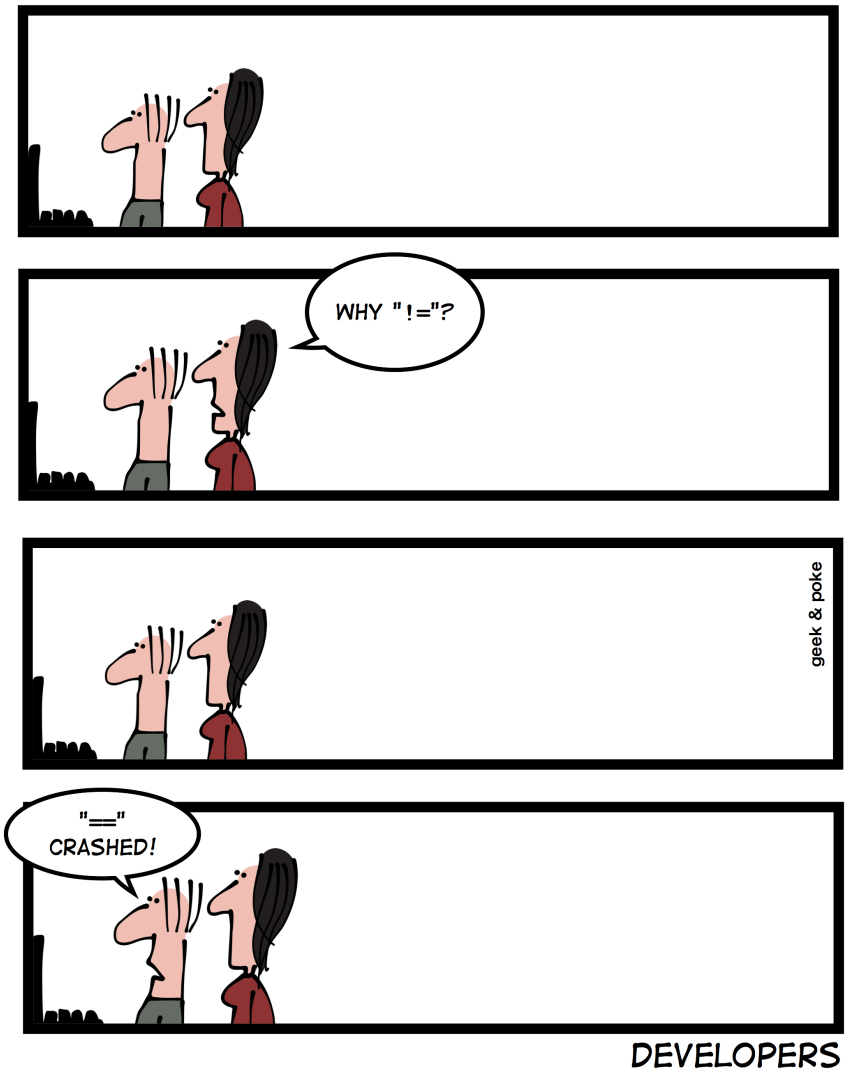
Very Important Point

- else always belongs to the closest if behind it
  - Use code blocks { } to denote sections

```
int x = 15;
if( x < 20 ) {
    cout << "apple";
    if( x < 10 ) {
        cout << "banana";
    }
} else {
    cout << "orange";
}
```

GOOD CODERS...


OK, WE'VE CHANGED ">" TO ">=". BUT THAT DOESN'T WORK EITHER. AND NOW?
geek & poke
LET'S TRY "<"
... KNOW WHAT THEY'RE DOING


WHY "!="?
geek & poke
"=="
CRASHED!
DEVELOPERS

Logical Operators

- Used to combine multiple conditions

- and &&
- or ||
- not !

x	y	x && y	x || y	!x
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

and, or, not

- Expressions get evaluated and yield a value

```
int x = 2, y = 3, z = 3;


x < y && x < z;
x > y || y > z;
y == z && !(x < y);
```

Precedence Table

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

Practice!

```
int x = 0, y = 3, z = 3;
```

1. `y <= 3 + x;` — T
2. `z != y - x;` — F
3. `x < y && x >= z;` — F
4. `(x < z) && (x = y);` — T
5. `x > y || y >= z;` — T
6. `y == z && !(x < y);` — F
7. `!y == 0 || z != 3;` — T
8. `!(y == 0 || z == 3);` — F

Other Common Errors

```
int age = 27;


if( 13 <= age <= 19 ) {
    cout << "Teenager!" << endl;
} else {
    cout << "Not a teen" << endl;
}
```

~~Other Common Errors~~ FIXED

```
int age = 27;


if( 13 <= age && age <= 19 ) {
    cout << "Teenager!" << endl;
} else {
    cout << "Not a teen" << endl;
}
```

Other Common Errors

```
int a = 27, b = 27, c = 27;


if( a == b == c ) {
    cout << "True!" << endl;
} else {
    cout << "false" << endl;
}
```

~~Other Common Errors~~ FIXED

```
int a = 27, b = 27, c = 27;


if( a == b && a == c ) {
    cout << "True!" << endl;
} else {
    cout << "false" << endl;
}
```

Other Common Errors

```
float num1 = 1 / 3;
float num2 = 0.333;


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

Other Common Errors

```
float num1 = 1.0f / 3.0f;
float num2 = 0.333f;


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

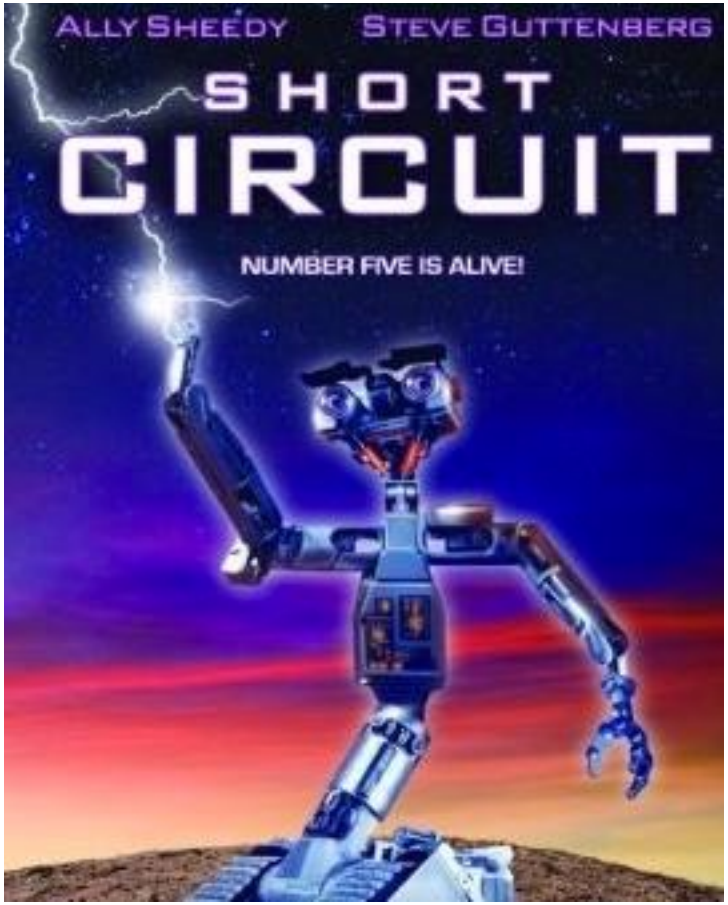
~~Other Common Errors~~ FIXED!

```
float num1 = 1.0f / 3.0f;
float num2 = 0.333f;
const float THRESHOLD = 0.000001f;

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

Short Circuiting

```
bool x = true;
bool y = false;


if( x || y ) {
   // something
}
```

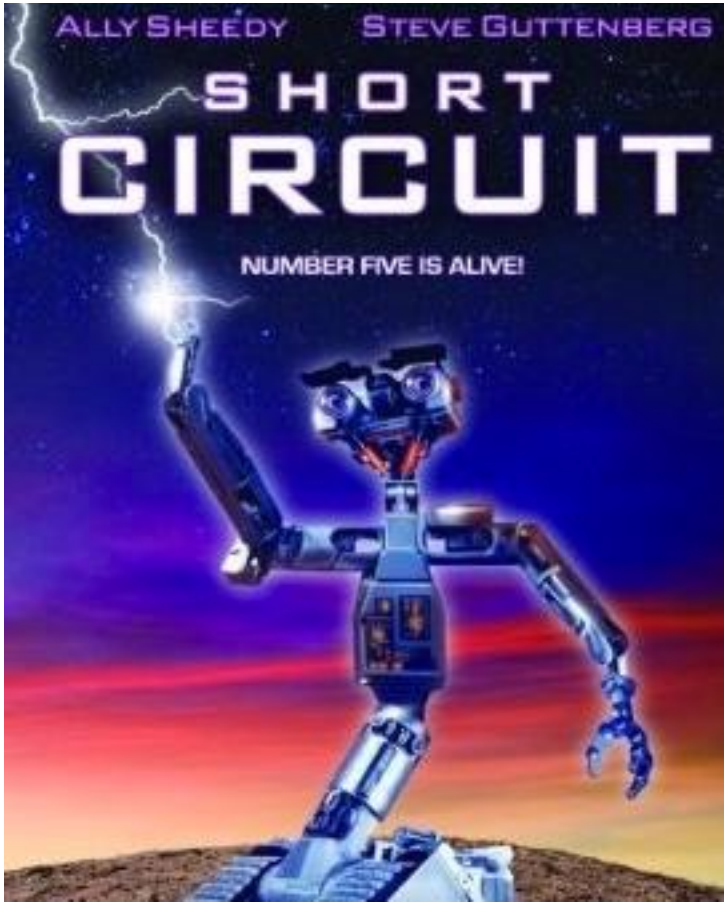
Won't be evaluated



Short Circuiting

```
bool x = true;
bool y = false;


if( y && x ) {
   // something
}
```

Won't be evaluated



Short Circuit Example

- (you wouldn't actually write this)

```
int x(5), y;

cout << "Enter 1 for True and 0 for False: ";
cin >> y;

cout << "\nx: " << x << "\ny: " << y << endl << endl ;

cout << "Testing ( y && (x+=3) ): ";
if( y && (x+=3) ) {
        cout << "evaluated true" << endl ;
} else {
        cout << "evaluated false" << endl ;
}

cout << "\nx: " << x << "\ny: " << y << endl << endl ;
```

On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

If / else if / else if / … / else

```
int menuChoice;
cout << "Select a menu option: ";
// print menu
cin >> menuChoice;
if( menuChoice == 1 ) { // do something
} else if( menuChoice == 2 ) { // do something else
} else if( menuChoice == 3 ) { // do something different
}
...
} else if( menuChoice == 67238 ) { // get Sonic drink combo #67238
} else {
    cout << "INVALID OPTION!" << endl;
}
```

switch

- Used for multiple-selection decision making
- Used to replace nested if-else

```
switch( expression ) {
case constantA:
     statement(s);
     break;              // break is optional
case constantB:
     statement(s);       // statements are optional
     break;
default:                 // default is optional
     statement(s);
}
```

Expression must be of type integer or character

Keyword ‘case’ must be followed by constant or literal

break needed or you will execute statements in the following case

switch Example

```
switch( code ) {
case 10:
    cout << "Turn on circulating fan." << endl;
    break;
case 11:
    cout << "Caution - recheck in 5 minutes." << endl;
    break;
case 12:
    cout << "Too hot - turn equipment off." << endl;
    break;
default:
    cout << "Normal temperature range." << endl;
}
```

Equivalent to

```
if( code == 10 ) {
    cout << "Turn on circulating fan." << endl;
} else if( code == 11 ) {
    cout << "Caution - recheck in 5 minutes." << endl;
} else if( code == 12 ) {
    cout << "Too hot - turn equipment off." << endl;
} else {
    cout << "Normal temperature range." << endl;
}
```

And also equivalent to

```
if( code == 10 ) {
    cout << "Turn on circulating fan." << endl;
} else {
    if( code == 11 ) {
        cout << "Caution - recheck in 5 minutes." << endl;
    } else {
        if( code == 12 ) {
            cout << "Too hot - turn equipment off." << endl;
        } else {
            cout << "Normal temperature range." << endl;
        }
    }
}
```

Switch Practice

```
int x;
x = 21 % 8;
switch( x ) {
case 2:
    cout << "You may say that I'm a dreamer" << endl;
case 5:
    cout << "But I'm not the only one" << endl;
case 8:
    cout << "I hope someday you'll join us" << endl;
    break;
case 11:
    cout << "And the world be as one" << endl;
    break;
default:
    cout << "by John Lennon" << endl;
}
```

What is the output?

Rewrite with a Partner

```
int input;
cout << "Enter a number between 1 and 5." << endl;
cin >> input;


if( (input >= 1) && (input <= 5) ) {
    if( input == 1 ) {
        cout << "The first number." << endl;
    } else if( (input == 2) || (input == 3) ) {
        cout << "The second or third number." << endl;
    }
} else {
    cout << "Input error." << endl;
}
```

Rewrite the if structure with a switch structure

Solution

```
int input;
cout << "Enter a number between 1 and 5." << endl;
cin >> input;
switch( input ) {
case 1:
    cout << "The first number." << endl;
    break;
case 2:
case 3:
    cout << "The second or third number." << endl;
    break;
case 4:
case 5:
    break;
default:
    cout << "Input error." << endl;
}
```

On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

Local Scope

- { } denote a code block
- Variables only exist within that code block
  - Concept of “scope”

Scope Notes

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

On Tap For Today

- Program Flow
- Conditionals
  - `if/else if/else`
  - Boolean Logic: Relational & Logic Operators
  - `switch`
- Scope
- Practice

To Do For Next Time

- Can start on Lab1B and A1
- zyBooks Chapter 4