# BOOLEAN EXPRESSIONS CONTROL FLOW (IF-ELSE) INPUT/OUTPUT

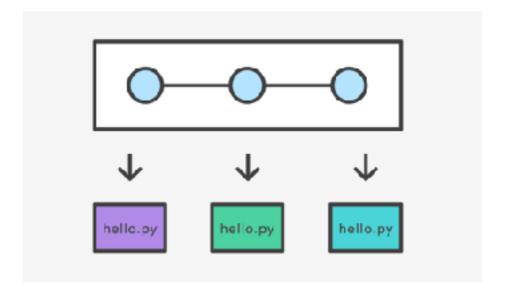Problem Solving with Computers-I

# What is git?

Git is a version control system (VCS).

A VCS allows you to keep track of changes in a file (or groups of files) over time

Git allows you to store code on different computers and keep all these different copies in sync

# Git Concepts

**repo** (short for repository): a place where all your code and its history is stored

Remote repo: A repo that exists on the web (in our case [github.com](https://github.com))

# In class demo

- **creating a repo on github.com**

- **adding collaborators to the repo**

- **adding files to the repo**

- **Updating files in a remote repo using a web browser**

- **Viewing the version history**

# Boolean Expressions

- An expression that evaluates to either true or false.
- You can build boolean expressions with relational operators comparing values:

```
==   // true if two values are equivalent
!=   // true if two values are not equivalent
<    // true if left value is less than the right value
<=   // true if left value is less than OR EQUAL to the right value
>    // true if left value is greater than the right value
>=   // true if left value is greater than OR EQUAL to the right value
```

# Boolean Expressions

- Integer values can be used as boolean values
- C++ will treat the number 0 as false and any non-zero number as true.

**bool x = 5 == 1;  // x = 0**

**bool x = 3 != 2;  // x = 1**

- Combine boolean expressions using Logical Operators

**!      // inverts true to false or false to true**

**&&  // boolean AND**

**||     // boolean OR**

- Example

**bool x = true;**

**bool y = true;**

**x = !x;            // x = false**

**x = x && y     // x = false**

**x = x || y        // x = true**

# Control flow: if statement

- The `condition` is a **Boolean expression**
- These can use relational operators

```
if ( Boolean expression) {
  // statement 1;
  // statement 2;
}
```

- In C++ 0 evaluates to a false
- Everything else evaluates to true

# Examples of if statements

- The `condition` is a **Boolean expression**
- These can use relational operators

```
if ( 1 < 2 ) {
  cout<< "foo" ;
}


if ( 2 == 3) {
  cout<<"foo" ;
}
```

Use the curly braces even if you have a single statement in your if

# Fill in the 'if' condition to detect numbers divisible by 3

A. `x/3 == 0`

B. `!(x%3)`

C. `x%3 == 0`

D. Either B or C

E. None of the above

```
if ( _____ )
  cout<< x << "is divisible by 3 \n" ;
}
```

# Control Flow: if-else

```
if (x > 0){
    pet = dog;
    count++;
}  else {
    pet = cat;
    count++;
}
```

- Can you write this code in a more compact way?

# Control Flow: Multiway if-else

```
if (x > 100){
    pet = dog;
    count++;
}  else if (x > 90){
    pet = cat;
    count++;
}  else {
    pet = owl;
    count++;
}
```

- Can you write this code in a more compact way?

# Input from user (using cin)

- Getting input from stdin (standard input)

```
int x;
cout<< "Enter a number"<<endl;
cin>>x;
```

# Input from user (via the command line)

- We can pass information into a C++ program through the command line when executing the program.
- The main function will need to have the following:

   int main(int argc, char *argv[])

- `int argc` is the number of "arguments" the program has, including the executable name.
- `char* argv[]` is the "list" of arguments passed into the program.

# Next time

- Loops