

# C++ DATA TYPES BASIC CONTROL FLOW

---

Problem Solving with Computers-I  
Chapter 1 and Chapter 2

C++

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

int main()
{
    cout << "Hola Facebook!";
    return 0;
}
```



# Review: Program compilation

What does it mean to “compile” a C++ program?

- A. Write the implementation of the program in a .cpp file
- B. Convert the program into a form understandable by the processor
- C. Execute the program to get an output
- D. None of the above

# Kinds of errors

Which of the following types of errors is produced if our program divides a number by 0?

- A. Compile-time error
- B. Run-time error
- C. Both A and B
- D. Neither A nor B

# Let's play Fizzbuzz

We'll play fizzbuzz and then code it up!

In the process we will learn about different ways of getting input into C++ programs:

Standard input cin

- Arguments to main
- Reading from files (a later lecture)

We will also learn about different ways of showing output to C++ programs:

## Let's code Fizzbuzz -1.0

**\$ Enter a number: 1**

**1**

**\$ Enter a number: 2**

**2**

**\$ Enter a number: 3**

**fizz**

**\$ Enter a number: 4**

**4**

**\$Enter a number: 5**

**5**

**\$Enter a number: 6**

**fizz**

**\$Enter a number: 7**

**7**

**\$Enter a number: 15**

**fizz**

# Review: C++ Variables and Datatypes

- **Variables** are containers to store data
- **C++** variables must be “declared” before they are used by specifying a datatype
  - `int`: Integers
  - `double`: floating point numbers
  - `char`: characters

# C++ Uninitialized Variables

- Value of uninitialized variables is “undefined”
- Undefined means “anything goes”
- Can be a source of tricky bugs
- What is the output of the code below?

```
int main() {  
    int a, b;  
    cout<<"The sum of "<< a << " and " << b<< " is:"<< a+b<<endl;  
}
```

# Variable Assignment

- The values of variables can be initialized...

```
int myVariable = 0;
```

**-or-**

```
int myVariable;  
myVariable = 0;
```

- ...or changed on the fly...

```
int myVariable = 0;  
myVariable = 5 + 2;
```



# Variable Assignment

- ...or even be used to update the same variable!

```
int myVariable = 0;  
myVariable = 5 + 2;  
myVariable = 10 - myVariable;  
myVariable = myVariable==0;
```

# 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 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;  
}
```

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

Let's code Fizzbuzz -2.0 (taking arguments from main)

```
$ ./fizzbuzz 1
```

```
1
```

```
$ ./fizzbuzz 9
```

```
Fizz
```

```
$ ./fizzbuzz 15
```

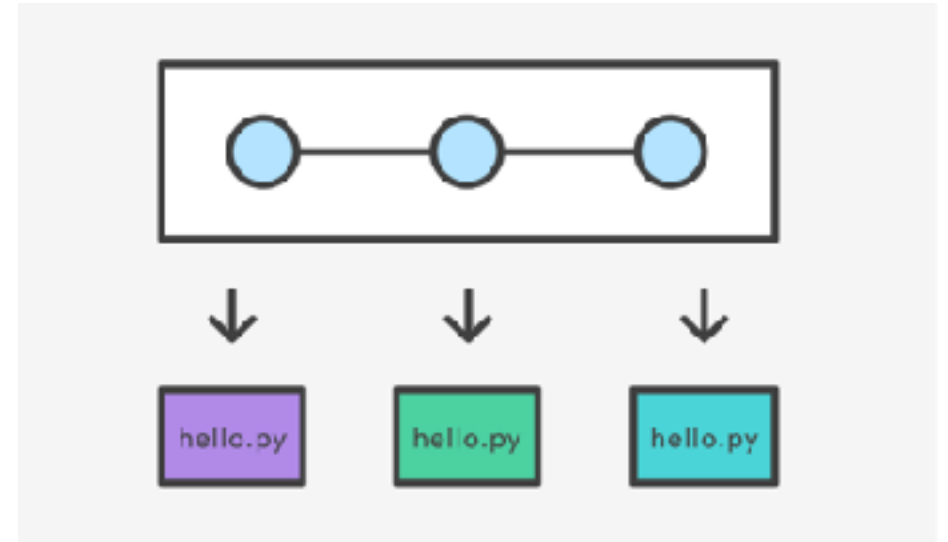
```
Fizzbuzz
```

# 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

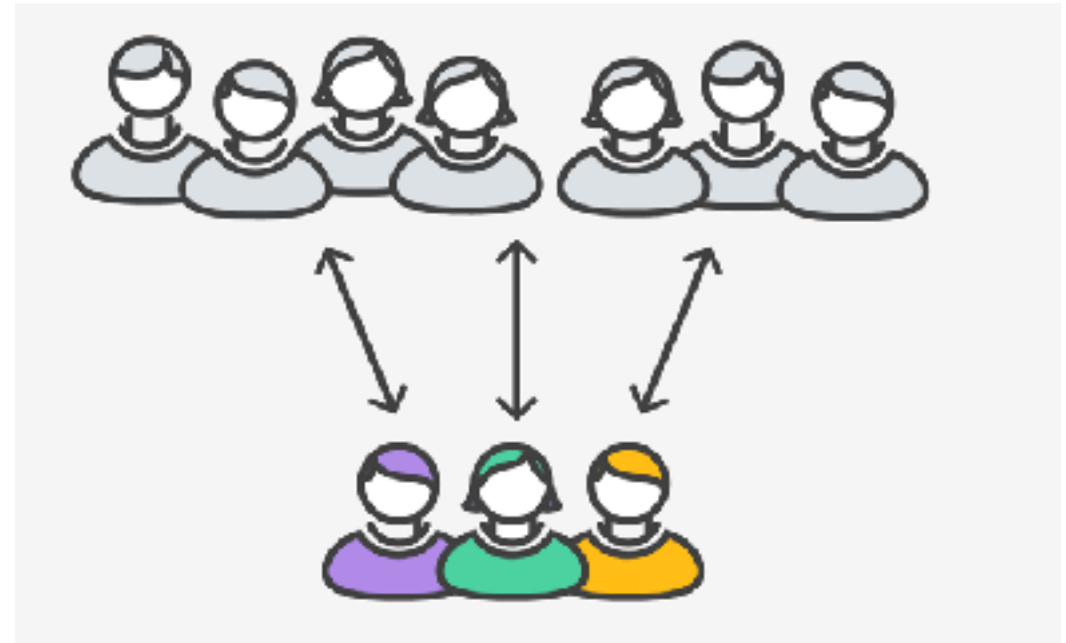
Git allows you to store code on different computers and keep all these different copies in sync





# Why are we learning git in this class?

- Collaborate
- Share code ownership
- Work on larger projects
- Provide feedback on work in progress
- Learn professional software development tools



# Git Concepts

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

# Creating a repo on the cloud ([www.github.com](https://www.github.com))

Navigate to [www.github.com](https://www.github.com) and create a repo on the internet

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

ucsb-cs24-s18 ▼

Repository name

lab00\_jgaucho\_alily

Great repository names are short and memorable. Need inspiration? How about **potential-lamp**.

Description (optional)

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☒ Initialize this repository with a README

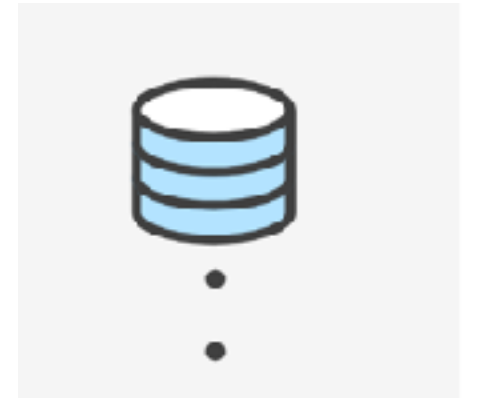
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: C++ ▼

Add a license: None ▼



Create repository



## Remote repo

Make sure the repo is inside our class organization! (Otherwise you won't be able to make a private one)

# Cloning a repo

```
git clone <repo>
```



# Different “states” of a file in a local repo

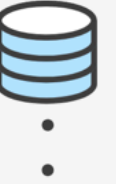


To inspect the state of a file use:  
**git status**

**Workspace**

**Staging area**

**Saved in local repo**



**Remote repo**

- Any file that is modified (in an editor) is saved in the **workspace**

# Saving a file (in the local repo)



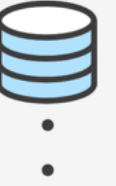
**Workspace**

```
git add <filename>  
git add .
```

```
git commit -m "message"
```

**Staging area**

**Saved in local repo**



**Remote repo**

# Syncing repos: pushing local updates to remote

**git push**



**Local repo**



**Remote repo**

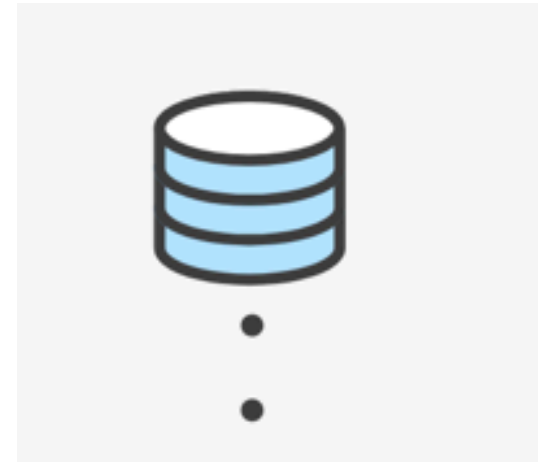


# Syncing repos: pulling the latest changes from remote

**git pull**



**Local repo**



**Remote repo**





# Numbers

**DISPLAY 2.2** Some Number Types

Type Name	Memory Used	Size Range	Precision
<i>short</i> (also called <i>short int</i> )	2 bytes	-32,768 to 32,767	(not applicable)
<i>int</i>	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
<i>long</i> (also called <i>long int</i> )	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
<i>float</i>	4 bytes	approximately $10^{-38}$ to $10^{38}$	7 digits
<i>double</i>	8 bytes	approximately $10^{-308}$ to $10^{308}$	15 digits
<i>long double</i>	10 bytes	approximately $10^{-4932}$ to $10^{4932}$	19 digits

These are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. **Precision** refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types **float**, **double**, and **long double** are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

# Comparison Operators

## DISPLAY 2.10 Comparison Operators

Math Symbol	English	C++ Notation	C++ Sample	Math Equivalent
=	equal to	==	<code>x + 7 == 2 * y</code>	$x + 7 = 2y$
≠	not equal to	!=	<code>ans != 'n'</code>	$\text{ans} \neq \text{'n'}$
<	less than	<	<code>count &lt; m + 3</code>	$\text{count} < m + 3$
≤	less than or equal to	<=	<code>time &lt;= limit</code>	$\text{time} \leq \text{limit}$
>	greater than	>	<code>time &gt; limit</code>	$\text{time} > \text{limit}$
≥	greater than or equal to	>=	<code>age &gt;= 21</code>	$\text{age} \geq 21$