

INTRO TO GIT FLOATS AND LOOPS

Problem Solving with Computers-I

C++

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

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



About you!

EE	35%
STSCI/ACTSC	49%
CMPEN	10%
CS	2%
OTHER	2%

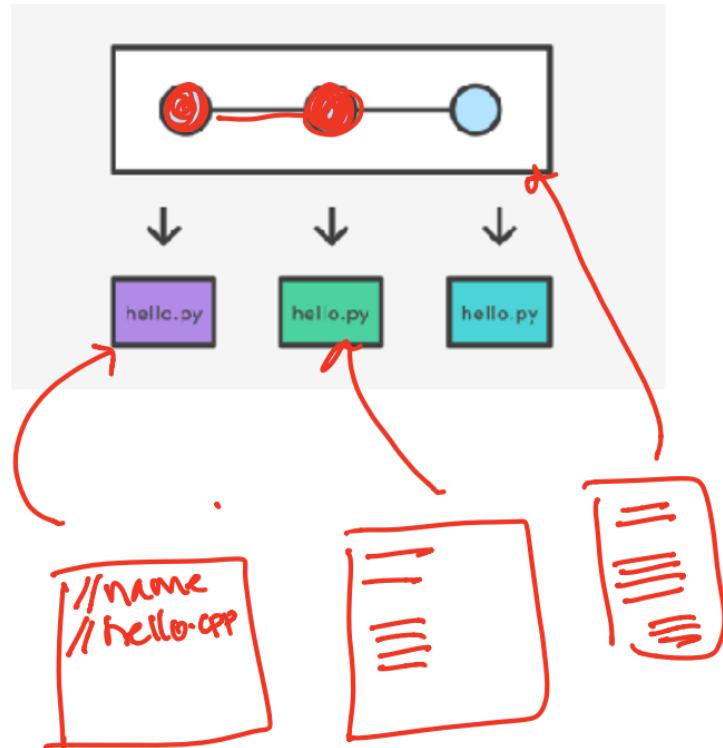
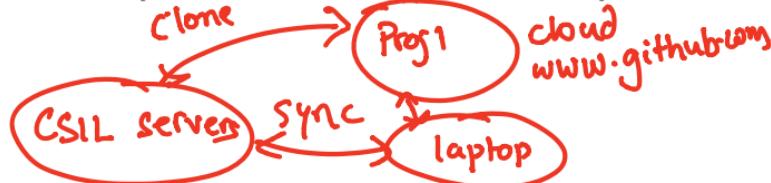
What is git?

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

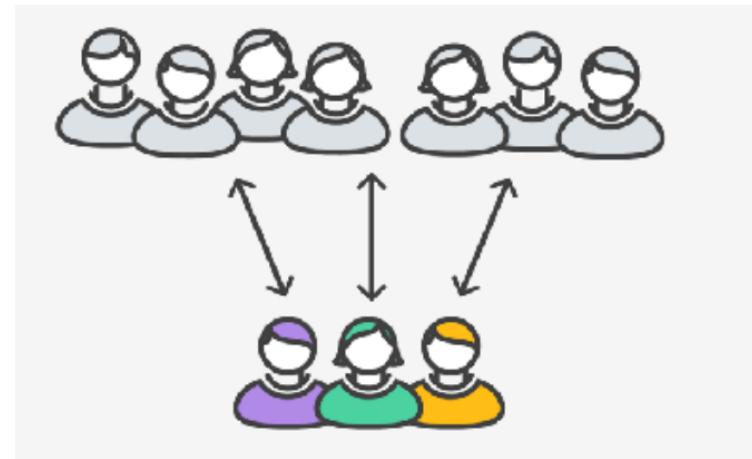
• shared ownership

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)

Navigate to 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 Repository name

 ucsb-cs24-s18 / lab00_jgauchao_ally

Great repository names are short and memorable. Need inspiration? How about [potential-lamp](#).

Description (optional)

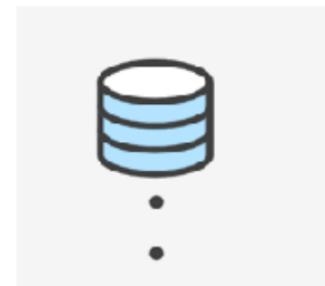
 Public
Anyone can see this repository. You choose who can commit.

 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

You may submit new versions of your code via www.github.com.

Git will remember all the different version

Cloning a repo

```
git clone <repo>
```

To get a copy of your code on your local machine, you may either download the zip file or clone the repo



Git Concepts: REPO

How is a directory different/similar to a git repository?

- A. Files are tracked in a directory but not in a repository
- B. Files are tracked in a repository but not in a directory
- C. Files are tracked in both a directory and repository



C++ types in expressions

```
int i = 10;
```

```
double sum = 1/i;
```

```
cout << sum ;
```

```
double sum; // Declaration  
sum = 1/i;
```

What is printed by the above code?

- A. 0
- B. 0.1
- C. 1
- D. None of the above

```
double sum = 1/i;  
// Declare and initialize
```

sum = $\frac{1}{i}$;

expression evaluates to 0 because 1 & i are both integers

Setting up output when printing doubles

See pages 91 and 190 of textbook

```
int i = 10;           explicit cast + type
double j = 1/static_cast<double>(i);
cout.setf(ios::fixed); // Using a fixed point representation
cout.setf(ios::showpoint); // Show the decimal point
cout.precision(3);
cout<<j;
```

number of digits after the decimal point
1.0/i;

double, i is automatically promoted to a double

What is printed by the above code?

- A. 0
- B. 0.1
- C. 0.10
- D. 0.100
- E. None of the above

x/y
Suppose x & y
are both integers
 x/y will be
rounded to an int
To avoid rounding
cast either x or y

C++ for loops

A for loop is used to repeat code (usually a fixed number of time)

C++ syntax:

```
for (int i=0; i<5; i++) {  
    cout << "Hello" << endl;  
}  
Hello
```

Diagram illustrating the execution flow of the for loop:

- Initial State:** The variable **i** is initialized to **0**.
- Iteration 1:** The condition **i < 5** is checked. It is true, so the loop body is executed.
- Body Execution:** The statement **cout << "Hello" << endl;** is executed, printing **Hello** to the console.
- Update:** The loop variable **i** is updated to **i + 1** (post-increment).
- Iteration 2:** The condition **i < 5** is checked again. It is still true, so the loop body is executed again.
- Body Execution (2nd iteration):** The statement **cout << "Hello" << endl;** is executed again, printing **Hello** to the console.
- Update (2nd iteration):** The loop variable **i** is updated to **i + 1** (post-increment).
- Iteration 3:** The condition **i < 5** is checked again. It is still true, so the loop body is executed again.
- Body Execution (3rd iteration):** The statement **cout << "Hello" << endl;** is executed again, printing **Hello** to the console.
- Update (3rd iteration):** The loop variable **i** is updated to **i + 1** (post-increment).
- Iteration 4:** The condition **i < 5** is checked again. It is still true, so the loop body is executed again.
- Body Execution (4th iteration):** The statement **cout << "Hello" << endl;** is executed again, printing **Hello** to the console.
- Update (4th iteration):** The loop variable **i** is updated to **i + 1** (post-increment).
- Iteration 5:** The condition **i < 5** is checked again. It is no longer true (**i** is now 5), so the loop exits.

Write a program that calculates the series:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n},$$

where `n` is specified by the user

While loops

A while loop is used to repeat code while some condition is true

C++ syntax:

```
int i=0      ↗ boolean expression  
while ( i<5 ) {  
    ↘ True  
    i++;  
}
```

do-while loops

A while loop is used to repeat code until some condition is no longer true

C++ syntax:

do {

 // code

} while (condition);

Nested for loops – ASCII art!

Write a program that draws a square of a given width

```
./drawSquare 5
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Draw a triangle

Which line of the drawSquare code
(show on the right) would you modify
to draw a right angled triangle

```
./drawTriangle 5
```

```
*
```



```
* *
```



```
* * *
```



```
* * * *
```



```
* * * * *
```

```
6  for(int j = 0; j < n; j++){ //A
7      for(int i=0; i < n; i++){ //B
8          cout<<"* "; //C
9      }
10     cout<<endl; //D
11 }
12 cout<<endl; //E
13
```

Infinite loops

```
for(int y=0;y<10;y--)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;;y++)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;y<10;);  
    y++;
```

```
int y=0;  
while(y<10)  
    cout<<"Print forever\n";
```

```
int y=0;  
while(y=2)  
    y++;
```

How is the pace of the class?

- A. Too fast
- B. Fast, but I am able to catch up once I do the labs
- C. Slow
- D. Too slow
- E. Its fine for me

Next time

- C++ functions and function call mechanics
- Passing parameters to programs