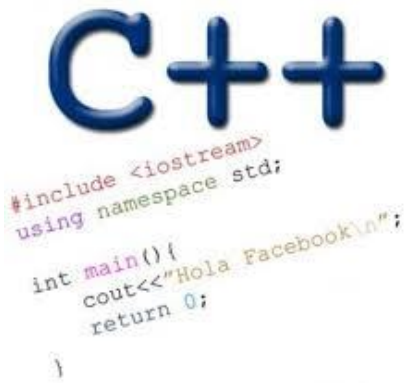
GIT AND GITHUB, C++ DATA TYPES BASIC CONTROL FLOW

Problem Solving with Computers-I

https://ucsb-cs16-wi17.github.io/



Announcements

- Submit your homework 1
- Homework 2 has been released
- Reading for homework is due before each class
- Reminder about our policy on electronic devices
- Cookies during office hours – come visit!!
- Some comments on labs:
  - Please make sure you read ALL the information in the lab write up prior to coming to section
  - Start looking for partners to pair with in your section (for lab01) – We recommend using Piazza

Which of the reasons best describes why you are taking this class?

A. You are a CS/CE major, and will be taking follow up classes
B. You are NOT a CS/CE major, but are contemplating on switching into CS or CE
C. Your major requires you to take the class. You are NOT a CS/CE major, are NOT contemplating on switching into CS or CE.
D. Other - Just curious

What is Git and Github?

- **Git** is an example of “version control”
- **Git** performs version control on “repositories”
- **Repository (repo)** is just a collection of files
- **Github** is a repository hosting service for Git

Q: Can you suggest ways to store the three different versions of hello.cpp shown on the right?

```
#include<iostream>

int main() {
}
```

```
#include<iostream>

int main() {
   cout<<“Hello World”;
}
```

```
#include<iostream>

int main(){
   cout<<“Hello World”<<endl;
   return 0;
}
```

Github Structure

- Unlimited public repositories and collaborators on all plans
- Limited private repositories
- Ability to add unlimited repository collaborators

- Organizations are great for projects that need multiple owners & admins.
- Private repositories
- Team-based access permissions

Demo: creating a new repo in our class organization!

Cloning Repos

REMOTE REPOSITORY

LOCAL REPOSITORY

Demo: Cloning a repo to CSIL servers and why that's useful

Clickers out – frequency AB

Which code produces a compile-time error?

A.

```
int main(){
    cout<<"Enter two numbers:";
    cin>>a >> b;
    cout<<"The sum of "<< a << " and " << b<< " is:"<< a+b<<endl;
}
```

B.

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

C. Both **A** and **B**

D. Neither **A** or **B**

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

```
int main() {

    cout<<"Enter two numbers:";
    cin>>a >> b;
    cout<<"The sum of "<< a << " and " << b<< " is:"<< a+b<<endl;
}
```

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;
```

Variable Assignment

- ...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;
```

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

- Find the main differences in each case
- Write the generalized if statement for each case

In Python	In C++
`if True:` `  itIsTrue()`	`if (true)` `  itIsTrue();`

`if True:`
`  itIsTrue()`
`  itIsAlsoTrue()`
	`if (true){`
	`  itIsTrue();`
	`  itIsAlsoTrue();`
	`}`

Generalized if statement

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

```
if ( condition ) {
   // statement 1;
   // statement 2;
}
```

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

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

Will both code instances give the same output?

A. Yes because they have equivalent logic
B. Yes, even though the logic is not equivalent
C. No because the logic is not equivalent
D. One will produce a compile-time error

```
int myVar =0;


if (myVar ==0)
    cout<<"inside if\n";

cout<<"outside if \n";
```

```
int myVar =0;


if (myVar=0)
    cout<<"inside if\n";

cout<<"outside if \n";
```

Fill in the ‘if’ condition to detect numbers divisible by 3

A. `x/3 == 0`

B. `x%3 = 0`

C. `x%3 == 0`

D. None of the above

```
if ( _________ )
   cout<< x << “is divisible by 3\n” ;
}
```

Control Flow: if-else

```
if (cond){
    doOnTrue()
    doThisAlso();
}  else {
    doOnFalse();
    doThisAlso();
}
```

- Can you write this code in a more compact way

Control Flow: for loops

```
int x;

for ( x = 0; x < 50; x++ )
  oneStatement( x );


for( x = 0; x < 50; x++ ) {
  statementOne();
  statementTwo();
}
```

Next time

- Basic File IO
- Number representation