

C++ VARIABLES, I/O, CONTROL FLOW

TOOLS: GIT

Problem Solving with Computers-I
Chapter 1 and Chapter 2

C++

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

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



CLICKERS OUT

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

Review: 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 or B

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
 - `string`: sequence of characters e.g. “apple”

Naming variables

- **Variable names must:**
 - Start with an alphabet (a-z, A-Z) or underscore(_)
 - Other characters can be alphanumeric and underscore characters
 - No spaces or other special characters.
- **C++ is case-sensitive**
 - 'x' and 'X' are considered different variables.

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;  
    return 0;  
}
```

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;
```


C++ types in expressions

```
int i =10;
```

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

What is printed by the above code?

A. 0

B. 0.1

C. 1

D. None of the above

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
- **In C++, 0 or empty string (“”) evaluate to a false**
- **Everything else evaluates to true**

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;  
}
```

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?

Input from user (using cin)

- Input streams: stdin (standard input)
- Output streams: stdout (standard output) and stderr (standard error)

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


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

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?

Let's code Fizzbuzz -2.0

\$ Enter a number: 1

1

\$ Enter a number: 2

2

\$ Enter a number: 3

fizz

\$ Enter a number: 4

4

\$Enter a number: 5

Buzz

\$Enter a number: 6

fizz

\$Enter a number: 7

7

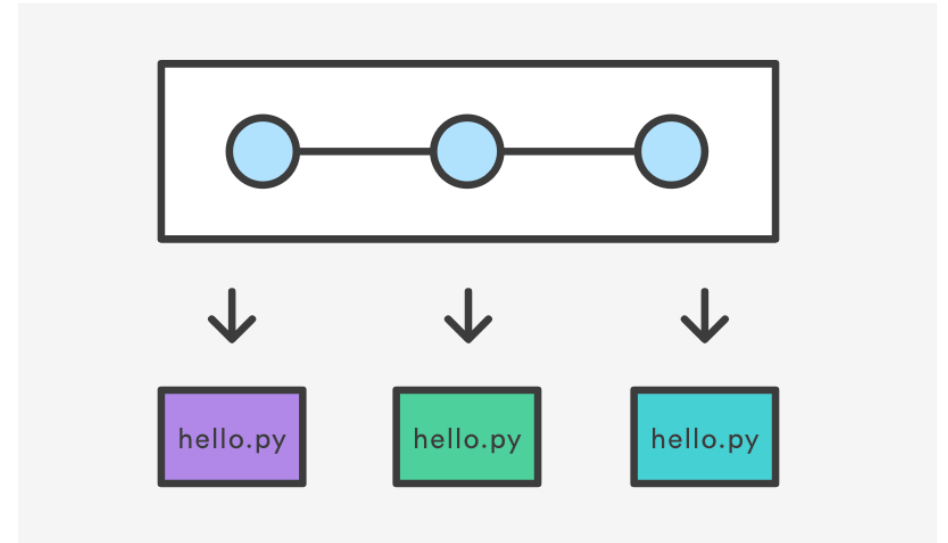
\$Enter a number: 15

fizz buzz

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)

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**

Vim survival skills

- Learn the basic 8: https://ucsb-cs16.github.io/topics/vim_basic_eight/
- Open a new file: `vim <filename>`
 1. Quit without saving
 2. Enter code
 3. Save, Save and quit
 4. Copy paste, cut and paste
 5. Search, Search and replace
 6. Show line numbers
 7. Go to a line number
 8. Save as

Next time

- Functions and loops