COP 3331
OBJECT ORIENTED DESIGN
SUMMER 2017

WEEK 1 (MONDAY, MAY 15TH) - INTRODUCTION

USF
UNIVERSITY OF SOUTH FLORIDA

WHAT IS EXPECTED

WHAT IS EXPECTED

- If you are enrolled in this course, it is because you have already taken:
  - COP 2510 (Programming Concepts)
  - COP 3514 (Program Design)

- You are familiar with
  - Java
  - C

WHAT TO EXPECT

WHAT TO EXPECT

- This course will cover Object Oriented Design using C++
  - C++: Object Oriented version of C
  - C: Procedural Programming Language
    - Consisted of series of carefully ordered structures and routines
    - Does not support objects and classes
- C++ does not retain complete source code compatibility
  - New syntax to explore!

WHAT TO EXPECT

- Programming Assignments: 40%
  - Assigned on Wednesday
  - Due the following Tuesday
  - Submit on Canvas

- **No** late assignments!

- Re-grade requests within **5** days of receipt

WHAT TO EXPECT

- Midterm: 25%
  - Closed Book/Closed Notes
  - Multiple Choice/Free Form
  - Tentatively scheduled for June 16th , In-class

- Final Exam: 35%
  - Same format as Midterm
  - Scheduled for **Wednesday July 19th , In-class**

WHAT TO EXPECT

- Text: Deitel and Deitel, *C++ How to Program*, 9th Edition

- Extra Credit?
  - Not guaranteed.

WHAT TO EXPECT

- Use of PCs encouraged in class
  - For programming/class related work only!
- Recommended IDE for course
  - Windows: Visual Studio
  - Mac: Xcode
  - Linux: GNU C++
- Text has great guides for installation and use

WHAT TO EXPECT

- Lectures: 9:30 am – 11:40 am
  - Short break (10-15 min) at/around 10:30 am
- Any cancellation will be announced on Canvas
- Grades placed on canvas to help you keep track
  - Final grade calculation based on metric described on syllabus
- Lowest passing grade is a **C**

DESIGN TASK: FIZZBUZZ

FIZZBUZZ?

- Fizz Buzz: common coding interview question

- Task:
  - Write a program that prints the numbers 1-100.
  - For multiples of 3, print “Fizz” instead of the number
  - For multiples of 5, print “Buzz” instead of the number
  - For multiples of 3 and 5, print “FizzBuzz” instead of the number
  - Example of output: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8…

FULL FIZZBUZZ OUTPUT

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
FizzBuzz
46
47
Fizz
49
Buzz
Fizz
52
53
Fizz
Buzz
56
Fizz
58
59
FizzBuzz
61
62
Fizz
64
Buzz
Fizz
67
68
Fizz
Buzz
71
Fizz
73
74
FizzBuzz
76
77
Fizz
79
Buzz
Fizz
82
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz

FIZZBUZZ!

- Pseudocode:

```
while number is between 1 and 100
    if number divisible by 3 and 5
        print “FizzBuzz”
    otherwise if number divisible by 3
        print “Fizz”
    otherwise if number divisible by 5
        print “Buzz”
    otherwise
        print the number
```

FIZZBUZZ – VERSION IN C

```
#include <stdio.h>
int main()
{
    int i;
    for(i=1; i<=100; i++)
    {
        if( ((i%3)== 0 && (i%5) == 0))
            printf("FizzBuzz\n");
        else if((i%3)==0)
            printf("Fizz\n");
        else if((i%5)==0)
            printf("Buzz\n");
        else
            printf("%d \n", i);
    }
    return 0;
}
```

FIZZBUZZ – VERSION IN C++

```
#include <iostream>
int main()
{
    int i;
    for(i=1; i<=100; i++)
    {
        if( ((i%3)== 0 && (i%5) == 0))
            std::cout << "FizzBuzz\n";
        else if((i%3)==0)
            std::cout << "Fizz\n";
        else if((i%5)==0)
            std::cout << "Buzz\n";
        else
            std::cout << i << std::endl;
    }
    return 0;
}
```

USING C++

USING C++

- Two parts to learning the C++ “world.”
  - The C++ language itself (the core language), and
  - How to use the classes and functions in the C++ Standard Library.

- Some concepts will be familiar, while others will be new to C++

- C++ (like C) upgrades their standards to be compatible with newer technology
  - C++ 14 current standard;

USING C++

- Familiar syntax:
  - // single line comments
  - /* */ multiline comments
  - # indicates line to be processed by the preprocessor
  - include directive

- #include<iostream> - tells preprocessor to include contents of the input/output stream header file

USING C++

- main is a part of every C++ program

- Output and input in C++ accomplished with streams of data
  - cout: output
  - cin: input
- std:: before cout is required when we use names brought into the program from iostream

USING C++

- Specifically, std::cout means that we are using a name, cout, that belongs to a “namespace” std

- << stream insertion operator
  - Value to the right is printed to the screen
  - Used as many times as needed for output

- Familiar: escape sequences, such as \n can be used to format output

- endl: end line – same effect as \n

USING C++

- Writing the std:: prefix every time we use cout, cin and other names can get *cumbersome*

- To eliminate the repetition, some programmers use the using declarations

- Example:
  - using std::cout; // program uses cout

C++ ORIGINAL FIZZBUZZ

```
#include <iostream>
int main()
{
    int i;
    for(i=1; i<=100; i++)
    {
        if( ((i%3)== 0 && (i%5) == 0))
            std::cout << "FizzBuzz\n";
        else if((i%3)==0)
            std::cout << "Fizz\n";
        else if((i%5)==0)
            std::cout << "Buzz\n";
        else
            std::cout << i << std::endl;
    }
    return 0;
}
```

C++ FIZZBUZZ WITH USING DECLARATION

```
#include <iostream>
using std::cout;        // program uses cout
using std::endl;        // program uses endl

int main()
{
    int i;
    for(i=1; i<=100; i++)
    {
        if( ((i%3)== 0 && (i%5) == 0))
            cout << "FizzBuzz\n";
        else if((i%3)==0)
            cout << "Fizz\n";
        else if((i%5)==0)
            cout << "Buzz\n";
        else
            cout << i << endl;
    }
    return 0;
}
```

USING C++

- An even better way is to use the using directive
  - This eliminates the need to specify the names you expect to use
  - It instead allows you to use *all* the the names in any C++ header

- In this case, we will insert the directive
  - using namespace std;

C++ FIZZBUZZ WITH USING DIRECTIVE

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

int main()
{
    int i;
    for(i=1; i<=100; i++)
    {
        if( ((i%3)== 0 && (i%5) == 0))
            cout << "FizzBuzz\n";
        else if((i%3)==0)
            cout << "Fizz\n";
        else if((i%5)==0)
            cout << "Buzz\n";
        else
            cout << i << endl;
    }
    return 0;
}
```

USING C++

- Recall:
  - To declare and initialize a variable you may write

```
int i = 0;
```

  - C++ allows you to initialize in this manner (thanks to C++ 11 standard):

```
int i{0};
```

  - Multiple initialization with a comma-separated list:

```
int i{0}, j{0}, k{0};
```

USING C++

- Newer features in the text will have 11 printed next to it to indicate new standards

- Embrace new syntax, and incorporate into familiar syntax and concepts!