Discussion Session
Week 2

Basics of Programming/Early Practice

Before we get started...

- We'll be coding today, so everyone should open up their chosen IDE

Learning To Code vs Learning To Program

- Anyone can learn to code
- While learning to program, you will learn to code
- **Programming** is the process of solving a problem that's been given to you
  - This is the key point to remember while working/learning
- **Coding** is writing the code to support your solution to a problem in a given syntax

The best way to learn to program is to do it….A lot

- https://open.kattis.com/problems?order=problem_difficulty
  - A site containing tons of problems to practice and learn with

C vs C++

- C++ is just “C with Libraries”
- C code can be used in C++ programs, C++ cannot be used in C programs
- C has 32 keywords, C++ has 52
  - You don’t need to memorize them
- C++ is more object oriented, C is function-driven
- In most of your classes, you’ll be working in C++ at URI

Refreshers

- C++ is strictly typed
  - You must explicitly declare the type of a variable(or function) when creating it
  - If a function does not return anything (yes this is possible), then it must be declared as a **void** function
- Comments are your friend
  - Get used to writing them and reading them
- All C/C++ programs must have a main function

Sample Program

```cpp
/**
 * Sample C++ program showing some syntax and what code should look like
 **/

/***********************
 *      Includes       *
 ***********************/
#include <iostream>
#include <stdio.h>

/***********************
 *      Defines        *
 ***********************/
#define MAX_INPUT 20

//Factorial function computes the factorial of a given number
int fact(int num);

int main(int argc, char* argv[])
{
    int input = 0;
    int sumOdds = 0;

    //Take in inputs from the user repeatedly until the user enters something greater than 20
    for(std::cin >> input; input < MAX_INPUT; std::cin >> input)
    {
        //Print factorial of input if even, add to sum otherwise
        (input % 2 == 0) ? printf("Factorial of %d is %d\n", input, fact(input)) : sumOdds += input;
    }

    std::cout << "Sum of odd inputs: " << sumOdds << std::endl;

    return 0;
}

int fact(int n)
{
    if(n <= 1) return n;
    return n * fact(n-1);
}
```

Classical Problems

- Sorting
- I/O
- Decision
- Search
- Counting
- Optimization

Classical Problems

- Sorting
- **I/O**
- **Decision**
- Search
- **Counting**
- Optimization

Functions

- In general, you want as little code in Main as possible
  - It should handle some basic I/O, and call helper functions to solve problems
- Functions are essentially blocks of code that can be called repeatedly
- Function syntax
  - functionReturnType functionName(param1Type param1Name, param2Type param2Name……)

    {

        Function body

        Return statement

    }

- Parameters are arguments to your functions
  - Think of parameters like x in the math expression f(x)

Best Practice

- Some things are mandatory, else we will receive compiler errors
  - For instance, if a function is called in Main before it is declared, that's an error
- Learn camelCase
  - Gives readability to your programs
- Common cpp file to get used to
  - Includes, defines, and other preprocessors are at the top
  - Local function declarations and contracts are written next
  - The main function follows those function declarations
  - The local functions are then defined below main

Sample

```cpp
#include <iostream>
#include <string>

/**
 * Function to print the length of a string
 * @Param str the string to find the length of
 * @Return an int, the length of the string
 * @Remark length includes whitespaces
 **/
int myStrLen(std::string str);

int main(int argc, char* argv[])
{
    //Prompt the user and take in input
    std::cout << "Please input a string: ";
    std::string temp;
    std::getline(std::cin, temp);

    //Output the result of the myStrLen call
    std::cout << myStrLen(temp) << std::endl;

    return 0;
}

int myStrLen(std::string buf)
{
    //Iterate through the input string
    int i = 0;
    while(buf[i++]){}
    --i;

    //Return the length
    return i;
}
```

Exercise One

- Write a program that does the following
  - Creates a function that does not return anything
    - In that function, print “Hello World”
  - Creates another function that adds two integers together and returns that sum
  - Calls both of these functions in main
    - Print out the result of your sum function

Solution

```cpp
#include <iostream>

void printHello()
{
    std::cout << "Hello World" << std::endl;
    return;
}

int sum(int a, int b)
{
    return a+b;
}

int main(int argc, char* argv[])
{
    printHello();

    std::cout << sum(5, 10) << std::endl;

    return 0;
}
```

Reading Documentation

- Cppreference is the goto resource for programming
- When you understand this site, you will outperform others who cannot understand it
- Look it up before asking

function template

std::min

<algorithm>

C++98 | C++11 | C++14

```
default (1)          template <class T> const T& min (const T& a, const T& b);
custom (2)           template <class T, class Compare>
                       const T& min (const T& a, const T& b, Compare comp);
initializer list (3) template <class T> T min (initializer_list<T> il);
                     template <class T, class Compare>
                       T min (initializer_list<T> il, Compare comp);
```

Return the smallest

Returns the smallest of *a* and *b*. If both are equivalent, *a* is returned.

The versions for *initializer lists (3)* return the smallest of all the elements in the list. Returning the first of them if these are more than one.

The function uses operator< (or *comp*, if provided) to compare the values.

The behavior of this function template (C++98) is equivalent to:

```
template <class T> const T& min (const T& a, const T& b) {
  return !(b<a)?a:b;     // or: return !comp(b,a)?a:b; for version (2)
}
```

Parameters

`a, b`
: Values to compare.

`comp`
: Binary function that accepts two values of type `T` as arguments, and returns a value convertible to `bool`. The value returned indicates whether the element passed as first argument is considered less than the second.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.

`il`
: An initializer_list object.
These objects are automatically constructed from *initializer list* declarators.

`T` shall support being compared with operator<.

C++98 | C++11

For *(3)*, `T` shall be *copy constructible*.

Return value

The lesser of the values passed as arguments.

Example

```
// min example
#include <iostream>     // std::cout
#include <algorithm>    // std::min

int main () {
  std::cout << "min(1,2)==" << std::min(1,2) << '\n';
  std::cout << "min(2,1)==" << std::min(2,1) << '\n';
  std::cout << "min('a','z')==" << std::min('a','z') << '\n';
  std::cout << "min(3.14,2.72)==" << std::min(3.14,2.72) << '\n';
  return 0;
}
```

Edit & Run

Preprocessors

#include

- Copy and paste a file at this location

#define

- Create a variable or function (different use cases)

#ifndef

#endif

- Basically, these expressions occur before compilation

Include

- This is where the “++” part of C++ comes in
- There are many functions, classes, structs, and other artifacts predefined for you
- If you want to use a function defined in a library, you must first include that library
  - #include <libraryName>
  - #include “localFileHeaderName”

Exercise Two

Using the math.h library, write a program that does the following

- Takes in two numbers on standard input
- Prints out the greater of the two numbers
- Prints out the result of raising the larger number to the power of 6

Caveats

- Although many functions are written for you in libraries, be careful when completing assignments
- The vast majority of the time, you will need to write your own functions
  - With some exceptions
- Many beginner level programs are solved widely online
  - Don’t copy and paste a solution….you’ll get caught

Compiling

- You're hopefully familiar with g++ by now
  - There are many other things that can be done besides simply compiling a program
- Default executable name is a.out
- -no-pie -g -O0 -Wall -Wextra -Werror -Wfatal-errors -std=c++11 -pedantic
  - And many, many other compiler flags
- Each flag has its own purpose
- Eventually (beyond scope of this course), you will learn about linking while compiling
  - Basically a means of including libraries that are not standard with C++
    - Libraries you have defined or downloaded locally
- Learning some flags and using them will make you a better programmer
  - Werror treats warnings as errors and forces you to be more careful
  - Wall, Wextra, Weverything adds more warnings to your code
  - std=c++11 specifies that this program uses C++11
  - etc