**CS 10A – Declarations and Definitions …C++ concepts to think about … (Information Sheet)**

**Why does C++ offer both declarations and definitions?** The declaration/definition distinction reflects the fundamental distinction between what we need to use something (an interface) and what we need for that something to do what it is supposed to (an implementation).

For a variable (introduced in Gaddis Chapter 2), a declaration supplies the type but only the definition supplies the object (the memory).

For a function (introduced later in Gaddis 6), a declaration again provides the type (argument types plus return type) but only the definition supplies the function body (the executable statements). Note that function bodies are stored in memory as part of the program, so it is fair to say that function and variable definitions consume memory, whereas declarations don’t.

The declaration/definition distinction allows us to separate a program into many parts that can be compiled separately. The declarations allow each part of a program to maintain a view of the rest of the program without bothering with the definitions in other parts. As all declarations (including the one definition) must be consistent, the use of names in the whole program will be consistent.

**Declaration Definition**

//a function defined

double sqrt(double d) // a function prototype double sqrt(double d)

{

// pow() is a predefined function in math.h header file

return pow(d, 0.5);

}

int value; value=10;

or int value=10;

//see function\_declaration\_definition.cpp

#include<iostream>

using namespace std;

int function(/\*in\*/int); //a value returning function prototype (aka. function declaration)with formal parameters

void anotherfunction(/\*inout\*/int&); //a non value returning function prototype (aka. function declaration)with formal parameters

int main()

{

int number=10; //a local variable

cout<<"The local variable number in main contains "<<number<<endl;

cout<<"The value returned in main after a call to int function "<<function(number)<<'\n'; //a value returning function call with actual argument passed into the function

anotherfunction(number); //a call to a non value returning function (aka. void function)with actual argument passed into the function

//\*note the statement by itself above..what was processed

cout<<"The local variable number in main contains "<<number<<endl;//what caused number to change here

}

//value return function defined with formal parameters...note dataflow comments

int function(/\*in\*/int someValue)  
{

someValue=someValue+100;

return someValue; //computer value returned to calling function

}

//non value returning function defined (aka. void function)with formal...note dataflow comments

void anotherfunction(/\*inout\*/int& someValue)  
{  
int newValue=someValue+200;

someValue=newValue\*2;

cout<<"The local variable newValue in anotherFunction contains "<<newValue<<endl;

//return someValue;

}

**Function specific Programming Challenges**

1. **Lowest Score Drop**

Write a program that calculates the average of a group of test scores, where the lowest score in the group is dropped. It should use the following functions:

* void getScore() should ask the user for a test score, store it in a reference parameter variable, and validate it. This function should be called by main once for each of the five scores to be entered.
* void calcAverage() should calculate and display the average of the four highest scores. This function should be called just once by main and should be passed the five scores.
* int findLowest() should find and return the lowest of the five scores passed to it. It should be called by calcAverage, which uses the function to determine which of the five scores to drop.

Input Validation: Do not accept test scores lower than 0 or higher than 100.

(Gaddis Chapter 6 Programming Challenge 11)

1. **High Scores**

Consider a text file named scores.txt that contains player scores for a game. A possible sample is shown below where Ronaldo's best score is 10400, Didier's best score is 9800, etc.

Ronaldo

10400

Didier

9800

Pele

12300

Kaka

8400

Cristiano

8000

Write a function named getHighScore that takes a string reference parameter and an integer reference parameter. The function should scan through the file and set the reference parameters to the name of the player with the highest score and the corresponding score.

1. **Reduce fractions**

Write a function named convertToLowestTerms that inputs two integer parameters by reference named numerator and denominator. The function should treat these variables as a fraction and reduce them to lowest terms. For example, if numerator is 20 and denominator is 60, then the function should change the variables to 1 and 3, respectively. This will require finding the greatest common divisor for the numerator and denominator then dividing both variables by that number. If the denominator is zero, the function should return false, otherwise the function should return true. Write a test program that uses convertToLowestTerms to reduce and output several fractions.

(Savitch Chapter 4 Programming Challenge 15)

1. **Sort numbers**

Write a function named sort that takes three integer parameters by reference. The function should rearrange the parameter values so that the first parameter gets set to the smallest value, the second parameter gets set to the second smallest value, and the third parameter gets set to the largest value. For example, given the variable assignments a = 30; b = 10; c = 20; then the function call sort(a,b,c) should result in a = 10, b = 20, and c = 30.

Note that the array construct will also give you a way to solve this problem for an arbitrary number of items instead of only for three items.(Savitch Chapter 4 Programming Challenge 17)