Functions: Declarations

Lab Sections

Declarations of Functions

1. **Objectives**

After you complete this experiment you will be able to:

   a. **declare a function**
   b. **understand the alternate syntax for a function declaration**
   c. **understand why function prototypes are important**

2. **Introduction**

Using functions in programs has several advantages which include: 1) allows for code reuse; 2) allows programs to be decomposed into smaller components (modularization); 3) they make debugging easier; 4) they make testing easier; 5) they allow for the implementation of recursion; 6) allow for generalizing a program; 7) allow information hiding.

3. **Definitions**

We will define several terms you need to know to really understand functions. They are as follows:

   a. Formal parameters reside inside a function header. They allow the actual arguments to be passed to a function. The scope of formal parameters is local to the function where they are declared.
   b. Actual arguments reside in the function that did the invocation (call). Their values are passed to the called function through the formal parameters.
   c. The header of a function includes the function return type, the function name, and the formal parameter list enclosed in parenthesis.
   d. The signature of a function includes the function name and the formal parameter list enclosed in parenthesis.
   e. The function prototype includes the function return type, the function name, and the formal parameter list enclosed in parenthesis. The prototype ends with a semi-colon.
   f. Function declarations (prototypes) provide information to the compiler to aid in the setup for a function call.

4. **Declaration Syntax**

There are two ways to code a function declaration:

```
Function_return_type   Function_name (Type1 FF1, Type2 FF2, …, TypeN FFN);
```

or the alternative syntax

```
Function_return_type   Function_name (Type1, Type2, …, TypeN);
```

Note:

   a. Type1, Type2, …, TypeN refer to the data types of the formal parameters.

b. FF1, FF2, …, FFN refer to the names of the formal parameters
c. All declarations end with a ";".

More information on functions can be found in your course textbook and on the web.

5. Experiments

Step 1: In this experiment you will learn how to declare functions.
Enter, save, compile and execute the following program in MSVS. Call the new project "FunctionsExp1" and the program "Functions1.cpp". Answer the questions below:

Question 1: What are the differences between the two function declarations shown above?

Question 2: Based on our previous lab, we learned about data types and the size of space allocated in memory [for instance an **int** may take up 4 bytes of space]. Based on this knowledge and different ways to write a prototype, what observations can you make regarding how a compiler "knows" how much space to allocate?

Question 3: What specific information does the compiler need to perform a successful function call?

Question 4: Write the header and signature for the following function. Also write the two forms of its prototype.

```
                double Print_Message(double balance, double limit)
                {
                      if (limit > 0)
                      {
                            cout<<"Your Balance is $"<<balance<<endl;
                            cout<<"Your have money.  Proceed with your transaction\n";
                      }
                      else
                      {
                            cout<<"Your Balance is $"<<balance<<endl;
                            cout<<"Your transaction has been cancelled\n";
                      }

                      return balance
                }
int main()
{
      double balance = 99.99, limit = 2000.22;

      cout<<"Balacne and Limt have the following values before Print_Message is 
called"<<endl;
      cout<<"balance= "<<balance<<" and limit = "<<limt<<endl;
      Print_Message(balance, limit);
      cout<<"a and b have the following values after swap is called"<<endl;
      cout<<"Balance = "<<balance<<" and Limit = "<<limit<<endl;
      return 0;
}
```