Lab 5: Simple Functions

Hello World in a Function

This is a hello world program using a function. The hello_world function will print "Hello World" when called to the console.

You have been using functions all along and may not have realized it, 'main' is also a function.

This is the syntax for a function definition (with no parameters).

```
type name ()
{
  statements
}
```

Where will this program start? At hello_world or main?

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

// this is a function definition
void hello_world ()
{
   cout << "Hello World";
}

int main()
{
   // this is a function call
   hello_world();

   return 0;
}
```

Function Prototypes

The below program is similar in output to the above program, but it uses a prototype. **A prototype tells the compiler the function definition can be found after the first function call. Remember, the compiler goes through a program sequentially, so if a function call does not have a prototype/definition, it will cause an error. The prototype simply lets the compiler know it will be defined later.**

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

// this is a prototype. Make sure to note the semicolon at the end.
// type name ( parameter1, parameter2, ...) ;
void hello_world();

int main()
{
   // Function call
   hello_world();

   return 0;
}

// Function definition
void hello_world ()
{
   cout << "Hello World";
}
```

Function Returns

C++ Syntax requires the function declaration to begin with a type. This is the data type to be returned by the function. If the function does not need to return a value 'void' can be used to indicate that the function will not return anything. The 'main' function has an 'int' type, while the hello_world functions have a 'void' type. At the end of the main function there is 'return 0'. This means that when the main function ends, it will return the value '0' to the function that called it. Our hello_world function has no 'return' because its type is void. Here are several more examples of functions with different return types and how they return values.

```
#include <iostream>
usingnamespace std;
// prototypes
int int_func();
double double_func();
boolbool_func();
void void_func();
int main()
{
   cout << "At top of main. calling functions: " <<endl;
   int i;
   double d;
   bool b;

   // the function's returned value can be stored in a variable
   i = int_func();
   d = double_func();
   b =bool_func();

  // Why will you get an error if you uncomment this code?
  //int v = void_func();

  void_func();

  cout << "At bottom of main:" <<endl;
  cout << "i is : " << i << " d is : " << d << " b is : " << b <<endl;

  return 0;
}
// int_func definition
int int_func()
{
   cout << "Inside int_func, will return value of 100"<<endl;
   return100;
}
// double_func definition
double double_func()
{
   cout << "Inside double_func, will return value of 2.414"<<endl;
   return 2.414;
}
//bool_func definition
boolbool_func()
{
   cout << "insidebool_func, will return 'true'"<<endl;
   return true;
}
void void_func()
{
   cout << "inside void_func, will not return anything" <<endl;

   // you can use a return with no value, or nothing at all.
   return;
}
```

Example

This program takes two numbers from the user and prints their sum.

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

// prototypes
int get_sum ();

int main()
{
   int sum;

   sum = get_sum();

   cout << "sum is :" << sum;
   return 0;
}

// get_sum function takes 2 numbers from a user and returns the sum
int get_sum ()
{
   // Note that we need to declare variables inside the function
   // because variables in main do not exist here. We will learn
   // more about this in the later lessons.
   int num1, num2, total;

   cout << "enter two numbers to be added" << endl;
   cin >> num1 >> num2;

   total = num1 + num2;

   return total;
}
```