A Technical Treatment of Functions and Parameters

A function is a reusable piece of code that can be *called* or invoked by other modules. In its simplest form, a function does not share any data with the function that called it. The technique of *parameter passing* allows a function to give information to another function as it calls it. In C++, there are two types of parameters: *value parameters* and *reference parameters*. When a value parameter is passed to a function, any changes made to the variable inside the function disappear as soon as the function ends. In other words, a value parameter is really a copy of the parameter, not the variable itself. In contrast, a function that receives a reference parameter receives the variable itself, and any changes made inside the function will remain once the function ends. Reference parameters are denoted with an ampersand (&).

A function's *declaration* (its first line) defines how many parameters it will expect, what their data types will be, and their parameter type (value or reference). The parameters are given as a *parameter list* in parentheses, listing expected parameters and their types. See the example below for an illustration of this concept.

Every function can optionally "return a value" to the calling function. The return type of a function is determined by the first word in the function's header—`int`, `double`, `char`, or `void` for no return type at all. The return value of a function is specified by the `return` keyword, followed by the return value. If a function's body has several `return` statements, the function is terminated the first time a `return` statement is encountered. Therefore, a function can be conditionally terminated in the middle of its body.

In C++, the header (declaration) of a function called by `main` is often placed before the `main` function as a *prototype.* Then, the header plus the code itself would be placed after `main`.

Only functions that are called by `main` will execute. Also, variables declared in `main` will not be known to other functions unless they are passed as parameters. Variables only exist in the function they are declared in.

On the next page is an example of all the topics discussed above.

Functions: Examples

A program that demonstrates different types of functions

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

//*****************Function prototypes************************
void title();                  //Takes no parameters and no return
int getOneNum();               //Returns a single integer
void getData(int &x, int &y); //Reference parameters and no return
double doMath(int x, int y);  //Value parameters and returns a decimal value
void displayResult(int s);    //Value parameter and no return

//*****************Function main*******************************
int main()
{
  title();
  int numOne, numTwo;
  getData(numOne, numTwo);
  double sum = doMath(numOne, numTwo);
  displayResult(sum);
  return 0;
}

//*****************Function definitions************************

void title()
{
  cout << "Want to add some numbers?" << endl;
}

int getOneNum()
{
  cout << "Enter a number: ";
  int n;
  cin >> n;
  return n;
}

void getData(int &x, int &y)
{
  x = getOneNum();
  y = getOneNum();
}

double doMath(int x, int y)
{
  return (x + y) * 2.4;
}

void displayResult(int s)
{
  cout << "The answer is " << s << "." << endl;
}
```