

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.

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`.

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

Functions: Examples

A program that demonstrates functions that are void and return values

```
#include <iostream>

using namespace std;

//*****Function prototypes*****
void title(); //Takes no parameters and no return
int getOneNum(); //Returns a single value
string getName(); //Returns a persons full name

//*****Function main*****
int main()
{
    title();
    int age;
    string fullName;
    age = getOneNum();
    fullName = getName();
    cout << "Your name is: " << fullName << " and you are " << age << " year(s) old.";
    return 0;
}

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

void title()
{
    cout << "Let's find out your name and age." << endl;
}

int getOneNum()
{
    cout << "Please enter your age: ";
    int n;
    cin >> n;
    return n;
}

string getName()
{
    cout << "Please enter your first name: ";
    string fName;
    cin >> fName;
    cout << "Please enter your last name: ";
    string lName;
    cin >> lName;
    return fName + " " + lName;
}
```

A program that demonstrates functions that are void and use parameters

```
#include <iostream>
```

```
using namespace std;

//*****Function prototypes*****
void title(string heading);           //Outputs the schedule heading
void getCourses(char block, string & class); //Returns a course name for the block
void output(char block, string class);  //Outputs a class and it's block

//*****Function main*****
int main()
{
    string heading = "Course Schedule", class1, class2, class3, class4;
    char block = 'D';
    getCourses('A', class1);
    getCourses('B', class2);
    getCourses('A', class3);
    getCourses(block, class4);
    title();
    output('A', class1);
    output('B', class2);
    output('C', class3);
    output(block, class4);
    return 0;
}

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

void title(string heading)
{
    cout << heading << endl;
    cout << "-----" << endl;
}

void getCourses(char block, string & class)
{
    cout << "Please enter your " << block << "- block class:" ;
    getline(cin, class);
}

void output(char block, string class)
{
    cout << block << "-block " << class << endl;
}
```

A program that demonstrates all the different types of functions

```
#include <iostream>

using namespace std;

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

//*****Function main*****
int main()
{
    title();
    int numOne, numTwo;
    getData(numOne, numTwo);
    int sum = sumData(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();
}

int sumData(int x, int y)
{
    return (x + y);
}

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