Introduction to Functions

This lesson will have to do with using functions and recursion to improve readability and efficiency of your code.

We'll cover the following

Functions

Simply put, a **function** is a segment of code that is isolated from the *main* code segment. A *function* is called from a *section* of code. When the function's code has been executed, it returns to the *calling code*.

The general form of a *function* is:

```
return_type function_name ([arg1_type arg1_name, ...]) { code }
```

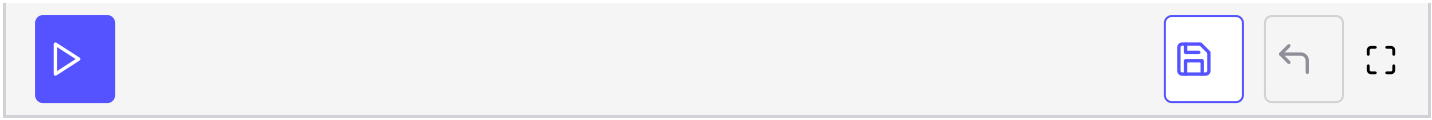
Example of Function

Here is an example of a *function*.

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

int addTwoInts(int arg1, int arg2) //function takes two integers arg1,arg2
{
   int sum = arg1 + arg2;    // adding both integers together to compute sum
   return sum;    // retuns the sum of type int which is same as function type
}

int main() {
  int answer = addTwoInts(2,4);   //calling our addTwoInts function in main()
  cout << "Answer is: " << answer << endl;
  return 0;
}
```



Explanation

- In line **4** the `int` at the starting of function is the *return type* of the function.
- `addTwoInts` is the *function's* **identifier**. It will be what is used to call the *function*.
- `int arg1` and `int arg2` are called **parameters**. They are defined in the form . These will be explained later.
- The code for the *function* is enclosed in a set of **curly brackets** `{ }`.
- Every function with a *return type* other than `void` **must** have a *return statement*. This is the data that the *function* will be sending back to the *calling code*.

Many conventions exist governing the form of *functions* in C++. Generally, whatever form is most readable to you is the one you should use. For example, some coders will leave the opening `{` on the same line as the *function* definition while others will give it its own line. Also, some coders would name the above function `add_two_ints`. This is mostly personal preference.

The main Function

The `main` function is a *special function*. Every C++ program must contain a function named `main`. **It serves as the entry point for the program**. The computer will start running the code from the beginning of the `main` function.

There are **two** main types of main functions.

Types of main Function

Down below is the first type which is a `main` function without parameters:

```
// Without Parameters
int main()
{
   ...
}
```

Now let's take a look at the second type which is a `main` function with parameters:

```
// With Parameters
```

```
int main(int argc, char * const argv[])
{


   ...
}
```

The reason for having the parameter option for the `main` function is to allow input from the *command line*.

When you use the `main` function with *parameters*, it saves every group of **characters** (separated by a space) after the *program name* as elements in an **array** named `argv`.

> **Note:** The *array* datatype may be beyond you at the moment, but don't fret. In time you will learn about it.

Here is an example of a *command line* statement with a program that uses a `main` function that accepts *arguments*.

```
C:\> program.exe text 234
```

This command would pass the following information to the main function:

- **argc = 3**
- **argv = {program.exe, text, 234}**

> **Note:** that the **'234'** is a `string`, not an *integer* value.

Since the `main` function has the *return type* of `int`, the programmer **must** always have a *return statement* in the code. The *number* that is returned is used to inform the *calling program* what the result of the program's *execution* was. Returning **0** signals that there were no problems.

Let's dig deeper into functions in C++ in the next lesson. Keep on reading to find out more!