Lecture 4: Functions

Curtin FIRST Robotics Club (FRC) Pre-season Training

Scott Day

265815F@curtin.edu.au

November 11, 2016

Curtin University

Insert Mandatory Programming Joke

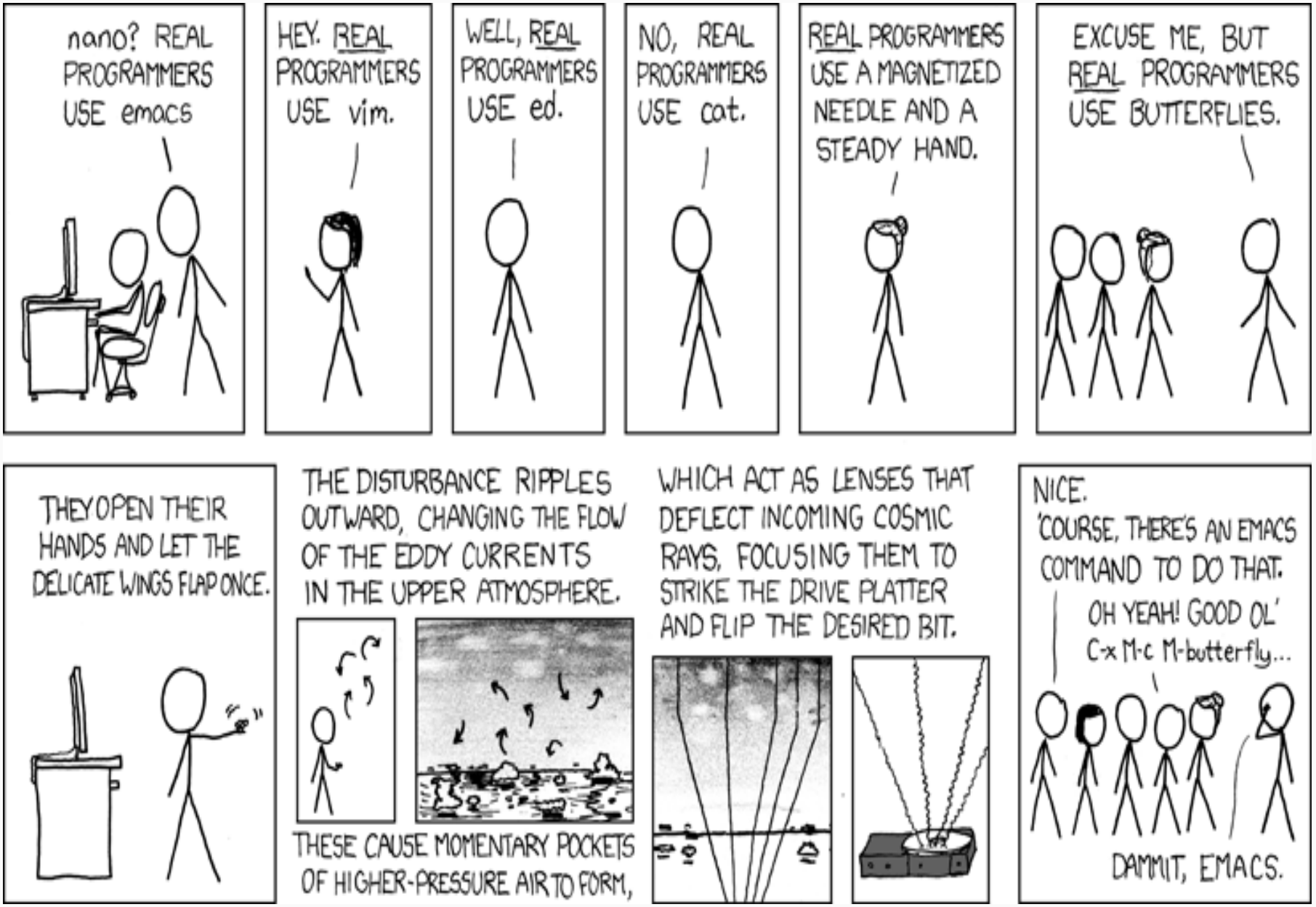

Table of contents

Function Justification

Functions

In C++ we can subdivide the functional features of a program into blocks of code known as functions. In effect these are subprograms that can be used to avoid the repetition of similar code and allow complicated tasks to be broken down into parts, making the program modular.

Until now you have encountered programs where all the code (statements) has been written inside a single function called main(). Every executable C++ program has at least this function. In the next sections we will learn how to write additional functions.

Function Definition

In C++, a function is a group of statements that is given a name, and which can be called from some point of the program. The most common syntax to define a function is:

```
returnValueType functionName(type parameter1, type parameter2, ...)
{
    statements;
}
```

Where:

return-value-type Is the type of the value returned by the function.

function-name Is the identifier by which the function can be called.

parameters Each parameter consists of a type followed by an identifier, with each parameter being separated from the next by a comma.

statements Is the function's body. It is a block of statements surrounded by braces { } that specify what the function actually does.

Cute Dog

Example of Function Definition, Declaration and Call

Example 1 - Addition

- A very basic example of using a function
- Note: Both examples do the exact same thing.

```cpp
// Function Example
#include <iostream>
using namespace std;

int addition(int a, int b)
{
    int result;

    result = a + b;

    return result;
}

int main()
{
    int temp;

    temp = addition(5, 3);
    cout << "The result is: " << temp << endl;

    return 0;
}

```

```cpp
// Function Example
#include <iostream>
using namespace std;

int addition(int a, int b)
{
    return a + b;
}

int main()
{
    cout << "The result is: " << addition(5, 3) <<
    ↪    endl;

    return 0;
}

```

Example 2 - Factorial

Let us first look at an example of a program writen entirely with the function main() and then we will modify it to use an additional **function call**.

We will illustrate this with a program to calculate the factoriall (n!) of an integer number (n) using a for loop to compute:

`n! = 1 x 2 x 3...(n-2) x (n-1) x n`

Example 2 - Factorial

```cpp
// Program to calculate factorial of a number
#include <iostream>
using namespace std;

int main()
{
    int ii, number = 0, factorial = 1;

    // User input must be an integer number between 1 and 10
    while(number < 1 || number > 10)
    {
        cout << "Enter integer number (1-10) = ";
        cin >> number;
    }

    // Calculate the factorial with a FOR loop
    for(ii = 1; ii <= number; ii++)
    {
        factorial = factorial * ii;
    }

    // Output result
    cout << "Factorial = " << factorial << endl;
    return 0;
}

```

Even though the program is very short, the code to calculate the factorial is best placed inside a function since it is likely to be executed many times in the same program.

Example 2 - Factorial

```cpp
// Program to calculate factorial of a number with function call
#include <iostream>
using namespace std;

// Function declaration (prototype)
int factorial(int number);

int main()
{
    int number = 0, result;

    // User input must be an integer number between 1 and 10
    while(number < 1 || number > 10)
    {
       cout << "Integer number = ";
       cin >> number;
    }

    // Function call and assignment of return value to result
    result = factorial(number);

    // Output result
    cout << "Factorial = " << result << endl;
    return 0;
}

// An integer, number, is passed from caller function.
int factorial(int number)
{
    int ii, factorial = 1;

    // Calculate the factorial with a FOR loop
    for(ii = 1; ii <= number; ii++)
    {
        factorial = factorial * ii;
    }

    return factorial; // This value is returned to caller
}

```

Example 2 - Factorial Discussion

Three modifications have been made to incorporate a function:

- The **declaration** of the function above `int main()`.
  The declaration (A.K.A the prototype) tells the compiler about the function and the type of data it requires and will return on completion.
- The **function call** in the main body of the program determines when to branch to the function and how to return the value of the data computed back to the main program.
- The **definition** of the function `int factorial(int number)` below the main program.
  The definition consists of a **header** which specifies how the function will interface with the main program and a **body** which lists the statements to be executed when the function is called.

Remember the Cuteness - A nose so sharp it could fly

Function Header and Body

t

t

t

Function Declaration

t

t

t

Function Call and Execution

t

t

t

Function Arguments

t

t

t

Passing by Value or Reference

t

t

t

Recursion - just to f*** with you

t

t

t

References I