Why Steve Says Const is Your New Best Friend

(no, really!)

Part 1: Const and Parameters

Say we have the following segment of code. What happens when this is run?

```
// This function does an operation on two critical
// data structures and returns the solution.
int addData ( int &a, int &b) {
        if (a = b) {
                return (2 * a);
        }
        else {
                return (a + b);
        {
}
```

What does this function actually do?

If this bug was buried deep in some code, it might take hours to find it. How could we avoid this frustration?

Try the following:

```
// This function does an operation on two critical
// data structures and returns the solution.
int addData ( const int &a, const int &b) {
        if (a = b) {
                return (2 * a);
        }
        else {
                return (a + b);
        {
}
```

Now what happens? Why?

Why Steve Says Const is Your New Best Friend, continued

Part 2: Const and Classes

The following is a short implementation of a familiar, simple class:

```
// FlyingMonkey.h

#ifndef FLYINGMONKEY_H
#define FLYINGMONKEY_H

// class definition
class FlyingMonkey {

        public:
        FlyingMonkey ( );
        // should just check status
        bool checkStatus ( );
        // should just check if its flying
        bool checkFlying ( );
        // should just check if its alive
        bool checkAlive ( );

        private:
        bool isAlive;
        bool isFlying;
};

#endif
```

```
// FlyingMonkey.cpp

#include "FlyingMonkey.h"

FlyingMonkey::FlyingMonkey ( ){
        isAlive = isFlying = false;
}

bool FlyingMonkey::checkStatus ( ){
        if (checkFlying() && checkAlive()){
                return true;
        }
        else return false;

bool FlyingMonkey::checkFlying( ){
        if (isFlying = true) return true;
        else return false;
}

bool FlyingMonkey::checkAlive( ){
        if (isAlive = true) return true;
        else return false;
}
```

Given the above, what happens in the following code?

```
// Main.cpp
#include "FlyingMonkey.h"

int main ( ) {
        FlyingMonkey Bob;
        Bob.checkStatus( );
        . . . .
        return 0;
}
```

Why Steve Says Const is Your New Best Friend, continued

Part 2: Const and Classes, continued

A bug like this could take a long time to fix. How should we fix it?

Now look at the following, fixed, implementation of the FlyingMonkey class:

```
// FlyingMonkey.h

#ifndef FLYINGMONKEY_H
#define FLYINGMONKEY_H

// class definition
class FlyingMonkey {

        public:
        FlyingMonkey ( );
        // should just check status
        bool checkStatus ( ) const;
        // should just check if its flying
        bool checkFlying ( ) const;
        // should just check if its alive
        bool checkAlive ( ) const;

        private:
        bool isAlive;
        bool isFlying;
};

#endif
```

```
// FlyingMonkey.cpp

#include "FlyingMonkey.h"

FlyingMonkey::FlyingMonkey ( ){
        isAlive = isFlying = false;
}

bool FlyingMonkey::checkStatus ( ) const{
        if (checkFlying() && checkAlive()){
                return true;
        }
        else return false;

bool FlyingMonkey::checkFlying( ) const{
        if (isFlying = true) return true;
        else return false;
}

bool FlyingMonkey::checkAlive( ) const{
        if (isAlive = true) return true;
        else return false;
}
```

How does this solve our problem?