Institut für Informatik — TUM
Scientific **C**omputing in **C**omputer **S**cience
R. Wittmann

WT 2015

Tutorial (Advanced Programming) Worksheet 5:

Class Assignment 1: Out-of-bounds access

In this exercise we will show how erroneous out-of-bounds accesses can change data in adjacent memory locations, even if they are marked as const. Please make sure that your compiler has C++0x or C++11 support enabled. Otherwise, it will complain about unknown classes, e.g. `array`, and invalid initialization schemes. Download the source file `reveal_secrets.cpp` from moodle and work through the following tasks:

1. Look at the first byte of our encoded message. What is it's purpose with respect to string operations?
2. Increase the variable `offset` in line 36 until the encoded message is shown.
3. Increase the variable `overwrite` in line 46 until the buffer shows the encoded message along with some extra characters.

Assignment 2: Greatest Common Divisor

In this task we consider the greatest common divisor algorithm which is defined as follows:

$$\begin{aligned} gcd(a,0) &= a \\ gcd(a,b) &= gcd(b, a \texttt{ mod } b) \end{aligned}$$

Implement this algorithm in three different variants:

1. Recursively. You are allowed to use temporary variables, but no static or global ones.
2. Iteratively. Same constraints as in the recursive approach.
   Hint: Apply your knowledge about "tail-recursion".
3. Iteratively, but this time without temporary variables.
   Hint: You can swap two variables using simple arithmetic (or logical) operations.

However, the function signature should remain the same, with the following properties: The function shall take two integer arguments as input and should return one integer. The latter should contain the greatest common divisor of the provided input arguments.

```
int gcd(int a, int b);
```

Homework Assignment 3: Fibonacci

Now, we will cover a very famous sequence, namely the Fibonacci sequence, which is defined by the following recurrence relation:

$$\begin{aligned} F(n) &= F(n-1) + F(n-2) \\ F(0) &= 0 \\ F(1) &= 1 \end{aligned}$$

Again, we want to implement different versions.

1. Implement a recursive version directly following the definition.

2. Implement a second recursive version requiring only one recursive call. Hint: In addition to the number n, use two formal parameters to cache already computed results.