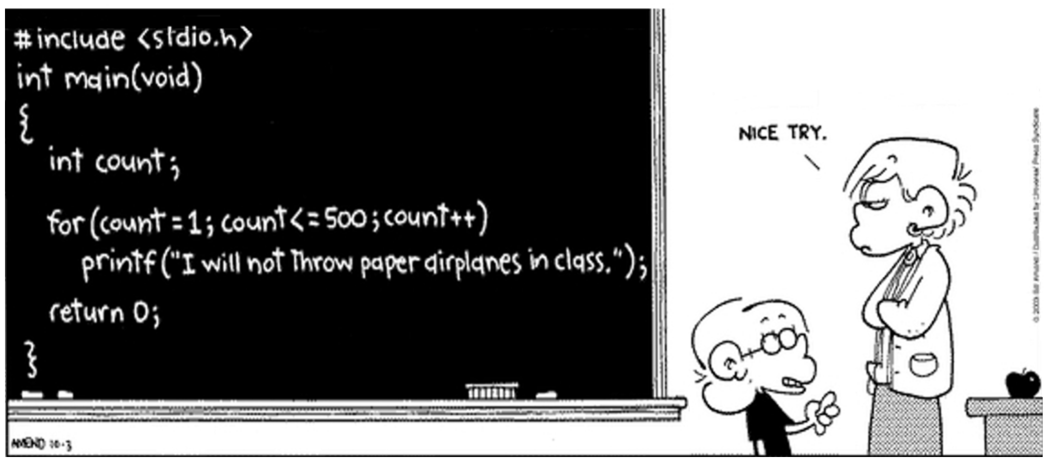
Wi4260: Math Programming on Linux, using C and Python

Kees Lemmens and Hai Xiang Lin, Faculty EEMCS,
Delft Institute of Applied Mathematics

Starts in 3th quarter (Feb 2017)



```
#include <stdio.h>
int main(void)
{
  int count;
  for(count=1;count<=500;count++)
    printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

Mathematicians and engineers sooner or later will have to convert their ideas and formulas into a computer code, for instance if they want to run a simulation. At that point some try to learn a little programming in a 30 minutes online crash course (preferably using Matlab), start to do their coding and are already quite happy if it seems to work!
But these people are a little naive (are you like them?) and while writing their code they often make several common errors.
So, what we actually want you to learn during this course is : "**Do not trust a computer blindly!**"
Equations that seem to be exactly the same for a mathematician (someone like you?) may give totally different answers if converted to a computer code. E.g. if you think that the product of 2 positive numbers always computes to a positive, look at the 2 lines in the code below and you may be in for a surprise :-)

```
int main()
{  int x = 2e9;
   printf("(2 *  %d) / 2  = %d\n", x, (2 *  x) / 2);     // result is -147483648
   printf(" 2 * (%d  / 2) = %d\n", x,  2 * (x  / 2));    // result is 2000000000
}
```

Computers introduce new types of errors to your math, like **overflows** (you probably heard about these!) but also **underflows** and **invalid** operations. How they can affect your results and how to deal with these in an elegant way is one of the topics in this course.
Other topics are the speed differences between some popular languages (if you think Matlab is fast then at least come to the first lab session!), deal with **'C' Pointers**, use Math Libraries like **Blas**, **FFTW** and **GSL**, Debug existing code without using lot's of **Paracetamol** and use **Profiling** to find performance bottlenecks.

We also put attention to that strange Operating System **Linux** and it's mysterious command line as Linux is very popular in the world of the Scientific Programming. If you are curious to learn how your computer internally deals with math and want to see how you can solve math problems using mainly **C** or **Python** then THIS is the course for you!

Study Goals Some of the goals of this course are that you should be able to :

- Convert a mathematical problem into a working C or Python program
- Write fault tolerant programs, i.e. they do not crash if the input or the results is different from what you expected
- Use debugging techniques and tools : you should be able to find out what could be wrong with a program if it doesn't work as expected, e.g. find and fix discretisation, round off and memory allocation errors, memory leaks, memory corruption etc.
- Analyse a program using timers and profilers in order to improve the performance
- Use some of the most popular scientific software libraries, like Blas, FFTW and GSL
- Use basic (Unix/Linux) commands to manipulate your source code and data files

Practical details

- Registration: Lectures and lab sessions start in February 2016. To enroll, please register on Blackboard or send an email to one of the instructors. Due to the capacity of the computer lab the maximum number of participants is 24.
- ECTS points: Succesful participation will be assessed by a 3-hours exam, with a theoretical part and a practical part where you have to fix problems in existing code or write small portions of a code by yourself. This course gives you 3 ECTS points.
- Schedule: The course consists of about 5-6 weekly lectures (Tuesday afternoon) and 8-9 mandatory weekly lab sessions (Building EECMS, Mekelweg4, lab room on floor 5 , 05.150).

  The first session will be in **week 5, February 2016** in labroom 05.150.
- Language: The course will be given in English.
- Study Materials: "Writing Scientific Software" by Suely Oliveira & David Stewart. Cambridge University Press, ISBN 0-521-67595-2

More information: use Blackboard or contact the instructors : C.W.J.Lemmens@tudelft.nl or H.X.Lin@tudelft.nl