Methods of Programming DV2

http://www.it.uu.se/edu/course/homepage/pm2/VT08/

Lars-Henrik Eriksson (lhe@it.uu.se)

Teaching assistant:
Manivasakan Sabesan (msabesan@it.uu.se)

Unofficial motto:
You can write good programs even if you have to use C!

Purpose of the course

"Techniques and methods for developing correct, stable, maintainable and efficient software."
(from the course plan)

You will not learn how to program (you should already know) but...
You will learn how to:
• write code which is easy to read and to maintain
• write code which is stable and works well
• write good documentation
• use program development tools
• profile and optimise code
...by carrying out a substantial programming project!

The project

Develop a ray tracer – a program which given a description of a three-dimensional scene, creates a two-dimensional picture by computing the path of the light rays from light sources to a "camera".

There is a detailed project specification on the course web site.
(Remember? http://www.it.uu.se/edu/course/homepage/pm2/VT08/)

What is "good code"?

• Understandable – good style, elegant, concise, good algorithms, good documentation
• Correct – gives the right results for all valid input
• Safe – rejects invalid input, doesn't silently do something wrong, resistant to faults in the surrounding environment
• Maintainable – adding/changing/removing functionality
• Efficient – acceptable CPU, memory, disk etc. usage, real-time response requirements
• Reusable – in new applications/contexts
• Portable – another CPU type, OS, window environment, etc.

How do we achieve this?
Particularly as we use a primitive language such as C...

Methods and techniques

• Good specifications (SE students will do most of it for you)
• Proper design (in advance, pseudocode, (data)flow charts)
• Coding standards (consistency, good practise, avoid pitfalls)
• Abstraction (a crucial issue!!!)
• Code review and analysis (better to find bugs before testing)
• Testing (code with testing in mind, plan your testing, save tests)
• Debugging (understand the bug, why was it there?)
• Rapid prototyping (early testing, partial functionality, code stubs...)
• Optimisation (get the code working first)
• Profiling (know what to optimise)
• Documentation (start early)
• Development/hazard log (future changes, problems, explanations)
• Software Engineering stuff in general.

Tools

Primarily traditional Unix development tools
• Source code management (CVS)
• Libraries (libxml...)
• Program build (compilers, make)
• Code analysis (compilers, lint...)
• Testing (scripts)
• Debugging (gdb...)
• Profiling (gprof...)
• Data preparation (XML editor)
Course setup

- Supporting lectures and literature (course compendium)
- Coding standards
- Work in groups of 3 – form groups now and e-mail the TA's.
- Effort: period 3, low speed (10%), period 4, normal speed (30%)
- Assignment 1 – Abstract data types (done by week 7)
- Assignment 2 – Basic ray tracer (done by week 14)
- Assignment 3 – More complete ray tracer (done by week 19)
- Assignment 4 – Optimise the program (done by week 22)
- Assignment 5 – Make a movie! (done by week 22)
- During assignments 2 and 3 students from the SE course will plan and carry out system tests and code inspections.

Deadlines and such...

- The deadlines are firm. If you need an extension, have a good reason and ask well in advance.
- The TA can give the following grades to each assignment:
  - Fail (U) – You flunk the course. Only happens if you don’t take the work seriously or lack basic qualifications.
  - Revise (K) – The work has serious shortcomings which must be corrected at once. Resubmit as soon as possible!
  - Pass and revise (GK) – The work has minor shortcomings which need not be corrected until the next assignment (but must be corrected by that time).
  - Pass (G) – The work is fine! (But the TA may still have given useful comments.)

Assessment

To pass the course (grade 3):
- Pass the project (i.e. all assignments)
- Pass a home exam (with questions about your particular project)

To pass the course with distinction (grade 4/5):
- Passing the project and home exam with at least grade 4 will earn you a 4 grade on the course.

You can additionally take an optional traditional exam (with questions about the course literature) in which case
- your grade on the course will be the average of the grade of the project and the two exams.

Prerequisites

- A total of 120 credits (new style, or ECTS credits)
- Mathematics 30 credits
- Computer Science 45 credits including
  - Methods of Programming DV1
  - Algorithms and Data Structures DV1
  - Algorithms and Data Structures DV2
  - Compiler Design MN1/DV1
- Computer Programming MN1+2 is considered equivalent to Methods of Programming DV1+Algorithms and Data Structures DV1.