**Algorithm Project :**

**Documentation & Report Writing**

**§****1.0 Introduction**

* Final element of software development strategy was
  + Presentation - including relevant report
  + Remember - reports are a *vital* part of whole process of software development!
* Reports matter
  + in university, for assessment
  + in research and development, for dissemination of ideas
  + in systems development
    - for your boss to convince their boss of your worth
    - to lower subsequent maintenance costs
    - Reports should be shorter rather than longer, but the shorter they are (if complete!) the longer they take to write
  + e.g. 3 weeks for 50 pages of final honours report!
* Specific tip for report writing
  + keep systematic note of all design decisions, however trivial   
    *(later, this will in any case be a required part of professional practice)*
  + subsequently, delete those decisions still obvious
  + rewrite the others in a suitable, cursive style

**§****1.1 Presentation Style**

* As with programs, read all the *good* examples you can find
* University reports are hard to find, so try (for style as well as content)
  + *Journal of the A.C.M*   
    ([American] Association for Computing Machinery)
  + *IEEE Software*   
    ([American] Institution of Electronic & Electrical Engineers)
* Literary Style?
  + anything you can write succinctly
  + need not now be written in third person, past tense, passive voice   
    "the program was written ... "
* But your reports - all reports - *must*
  + be typed
  + be grammatical and have correct spelling
* No educated person will have any sympathy with the following, all-too-common errors
  + misused apostrophes - they should be used
    - with nouns, to show possession
    - more generally, for ellipsis
  + misuse of "its" and "it's" or of the variations of "they're", "their", "there"

|  |  |  |
| --- | --- | --- |
|  | its | of it |
|  | it's | it is *or* it was |
|  | they're | they are *or* they were |
|  | their | of them |
|  | there | place, *or*,well, as in 'there is....' |

* + (whose and who's are respectively just like "its" and "it's")
  + non-matching number of subject and verb
* Spelling checkers are a boon, but are from from infallible ...
* Moral? Read and correct your text again, some hours after you'd finished it!

**§****1.2 Report Overview**

* **Part 1 - short!**

*Introduction*   
... what you're going to describe

* **Part 2 - the majority!**

The *description* itself

* **Part 3 - short!**

*Summary*   
... of what you've described

**§****1.3 The Report Itself**

* You will need chapters, and properly headed sections
* For this class, but not in general, some middle chapters can be combined
* Note that almost all your detailed work is consigned to appendices!
* Remember
  + particular projects may force a slightly different structure on you
  + as with programming style, you may have to apply a house style
* Format for Computer Science (and, indeed, most) reports -

|  |  |  |
| --- | --- | --- |
| Introductory Material | | [§1.3.1](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.1#4.3.1) |
| Chapter | 1. Problem Definition 2. Related Work | [§1.3.2](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.2#4.3.2) |
| 1. Outline Solution 2. Detailed Design | [§1.3.3](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.3#4.3.3) |
| 1. Testing & Validation | [§1.3.4](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.4#4.3.4) |
| 1. Future Development 2. Conclusion - Solution Summary | [§1.3.5](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.5#4.3.5) |
| Appendix | 1. References 2. Test Material 3. User Manual 4. Relevant Code | [§1.3.6](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.6#4.3.6) |

* For this class, any sensible, cheap binding will suffice - perhaps the plastic covered two prong files sold by the Students' Association?

**§****1.3.1 Introductory Material**

* Title Page

Here (other classes or projects may differ somewhat in detail)

* + Class Number and Name
  + Project Name
  + Your Name, Registration Number and Degree Course
  + Date of submission
  + A signed statement that it's all your work unless marked to the contrary
* Abstract

Typically, 200-300 words

Perhaps something like:

|  |
| --- |
| "This report outlines the design and development of a computer software system to ... The program was written in C++ to run under the Unix operating system on ... The design and ensuing program are modular in nature and make maximum use of abstract data types and of software re-use. Particular attention is paid to ... The report includes a full user manual, as well as the whole of the code that was written." |

Not very good (excepts perhaps for this class!); you will do better - but only with practice. But *think* about it carefully, to understand the difference between an abstract and the descriptive material which forms the introductory chapters of the text.

And, yes, the reference to C++ rather than Java is deliberate; otherwise the bit about the operating system would get lost, wouldn't it?

* Contents Page

- self explanatory!

**§****1.3.2 Problem Definition & Related Work**

* The Problem

- as defined in the real world

- working through to the specifications you were given and/or developed; perhaps these should form an appendix?

* It may be appropriate to mention languages and target machines (if they were laid down), but normally that will go into the chapter giving the outline solution.
* Related Work is probably irrelevant for this class

- but in future it will give you the chance to show, through references and a brief description, where you got your ideas. Remember the earlier comment on plagiarism?

**§****1.3.3 Outline Solution & Detailed Design**

* Outline Solution

- is an overview

- may develop particular points of importance or of difficulty

- covers general techniques, for example use of defensive programming

* Detailed Design is more awkward

- there are two schools of thought

* + explain everything slightly generally
  + skip the easy bits, but explain some parts right down to code level

- you pays your money and ...!

* What are the roles of structure charts, DFD's [Data Flow Diagrams] and the like?

**§****1.3.4 Testing & Validation**

* Explain how you know that your program works properly
  + Full validation is a lengthy business, concerned with verification, and testing, and with assessment of the software's user and maintenance profiles
  + In the context of this class, you will be concerned with testing
  + And, in this chapter, with explaining the [testing *strategies*](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_3.html#3.1) you followed and the sorts of data you used
  + Remember that efficient testing will almost certainly require a variety of approaches, each one chosen for its appropriateness for different sections of the code
* Note that your detailed test results - *a selection of the various test cases you applied* - normally belong in an appendix
  + Even in the appendix, probably only some of the test cases will be included
  + Be sure that you annotate your test runs to explain clearly what each test is doing!
* If something doesn't work properly, be honest about it
  + To do otherwise is simply evidence of poor testing ... or maybe of dishonesty
  + Use the next chapter to outline how, given the opportunity, you might have cured the problem

**§****1.3.5 Future Development & Conclusion**

* Future Development

- the chance to cover the bits you would like to do differently?

- the bits you didn't get right?

- genuine possibilities for extension?

* Seldom if ever should you extend the original specification to a slightly different problem!
* Conclusion

- "Iacta alea est"... *Suetonius, on Julius Caesar*

"The die is cast" - reportedly said two thousand years ago when Caesar   
the gambler crossed the River Rubicon to invade Rome and seize power.

- or, more prosaically,

|  |
| --- |
| "This report has described the successful design and development of ... " |
| See, here meaning that you should more or less repeat! the text of the [abstract](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.1#4.3.1). |

**§****1.3.6 Appendices**

* References
  + should be referenced from text!
  + Preferably using one of the two standard systems (by dereferencing consecutively numbered footnotes, or - more common in Computer Science - by an alphabertic list which dereferences author-date pairs in the text)
  + a bibliography is different, although in your reference appendix it may be appropriate to say (for example), "Extensive use was also made of the ADT material in ..."
  + Do ensure you understand the difference between "bibliography" and "list of references"!
* Test Output

- already dealt with, in [§1.3.4.](https://www.cis.strath.ac.uk/~paul/classes/archived/52.222/222_L_4.html#4.3.4#4.3.4)

* User Manual
  + include locations of relevant program files!
  + assume a competent machine user ... but do tell them how to run the program, as well as where to find it!
  + give illustrations of possible inputs and corresponding outputs
  + cover *all* possible program responses   
    ... although of course this principle becomes unachievable for bigger programs
* Code
  + should you include it all?   
    (Answer for this class : yes!)
  + it will, of course, be self-documenting, so that's no problem!
  + it may be preferable to photo-reduce it, if you can afford to access appropriate equipment
  + remember that in the report your code is merely an appendix, even though the report could not have been written without it!

**§****1.4 Concluding Comments on Reports**

* Yes, I know you will have covered this before

- for Highers projects  
- or in an HND  
- or in employment  
- or ...

* But good communication, both oral and written, is vital.
* Without good communication, your talents as a computer scientist - or as a person - are wholly squandered.



