Academic Writing in Computer Science

Hadrien Mélot, Alain Buys

Computer Science Department
Faculty of Science, UMONS

Hadrien.Melot@umons.ac.be
Alain.Buys@umons.ac.be

October 2020, Version 1

Foreword. The first versions of these academic writing guidelines were written by Prof. Hadrien Mélot (in French) under the title "Eléments de rédaction scientifique en informatique" and used by UMONS students, mostly for their Master's thesis and various Master's projects. Due to the growing tendency towards internationalizing the curriculum, an increasing number of classes are taught in English and assignments often consist of writing reports and giving presentations in English. In this context, it seemed extremely useful to adapt these existing guidelines into a version more specifically covering academic writing in English.

(Alain Buys)

Abstract. Writing a scientific or technical document, like a Master's thesis in computer science, requires specific skills. In this document, we describe the basic elements of scientific writing: bibliography, structure, contents, style and presentation. Our goal is to offer readers a method and guidelines to improve their writing. These guidelines come with counterexamples to help detect typical mistakes and give examples to fix them. We also cover plagiarism, a serious problem given the present accumulation of easily accessible digital resources.

The various steps of scientific writing are covered: organizing thoughts, writing and final reviewing/proofreading. A written report is often followed by an oral defense: therefore, we also offer some advice for such public presentations.

Faculté
des Sciences

UMONS
Université de Mons

Contents

1 Introduction

This document presents the basic elements related to the writing of scientific or technical documents. More precisely, we discuss the principles to follow when writing reports in a computer science curriculum.[1] It is of primary importance – for a computer science student – to be able to put the results of their work, research, modeling or implementations on paper. Very often, the written report is the main source of information provided to the teachers for their evaluation. The goal of the student is then to deliver a written report giving a precise description of their results and highlighting their personal work. A student will be assessed not only on the contents of their report but also on the efficiency of its writing, and this requires good quality writing.

Scientific writing is not an easy task. The main difficulties are: having a fairly good knowledge of the language used, having sufficient writing experience, knowing the basic principles of "good" writing and using an efficient working method. The knowledge of the language and the experience in writing are out of the scope of this document, since the student is supposed to learn and improve these on their own (or via language classes foreseen in their curriculum). Instead, the purpose is to provide a resource to help the student to improve their own writing and to design an efficient working method for their own use.

The writing advice given here is illustrated with examples of what you should not do and with examples of what you should do instead. These pieces of advice are marked with the symbols shown in Table 1.

Table 1 – Symbols used to highlight advice and (counter-)examples.

Symbol	*Meaning*	*Description*
💡	Advice	Specific advice or tips related to the topic
✗	To avoid . . .	Example of what you should not do (counterexample)
✓	. . . prefer	Hints on how to improve the previous counterexample
?	Questions	Questions that the previous counterexample could raise

The various writing steps are covered in Sections 2 to 5. The organization of ideas and the choice of page layout tools must take place before starting the actual writing: this is the topic covered in Section 2. Section 3 addresses the bibliography, to be built in parallel with the writing, and plagiarism. The actual writing is covered in Section 4, where its basic elements are presented: text structure, rigorous and precise scientific contents, appropriate style for a scientific report and graphical presentation. Section 5 gives some advice for the post-writing step.

A work must often be presented orally (*e.g.*, a Master's thesis presentation). Section 6 gives some specific advice related to oral presentations. Indeed, even if the purpose is the same as the written report – highlighting the personal work in a clear and structured way – the available tools are different. Usually, the jury expects a full written report, rigorous and precise. The oral presentation and the supporting slides must be more concise because the allotted time is limited. One must get

[1]The principles covered in this note are also valid for any other type of scientific writing, but the examples and guidelines were chosen assuming that the reader is a computer science student.

to the heart of the matter, without sending the audience to sleep with an overflow of information and details.

Note 1. *In the first sentence of the previous paragraph, we used the abbreviation "*e.g.*" which means "for example" (*exempli gratia*). Other common Latin abbreviations, and the related typographic rules, are given in Appendix A.*

2 Before writing

Writing takes *a lot* of time. You should not start the day before the deadline. Moreover, before even starting to write, some preparation must take place, including

a) creating a structure, organizing ideas;

b) setting up a writing plan;

c) laying down the domain specific vocabulary and notations;

d) taking into account the expectations of the future reader.

These steps are further explained in Sections 2.1 and 2.2.

Then, software tools must be selected to produce a digital version of the document (*cf.* Section 2.3).

2.1 Planning: Mapping out ideas and thoughts

A written document should always be structured following the pattern shown in Table 2.

Table 2 – Organization of a written document.

1. Cover Page and Title Page
2. Acknowledgments (optional)
3. Table of Contents
4. Introduction
5. Chapters (book, Master's thesis) or Sections (shorter document)
6. Conclusion
7. Bibliography
8. Appendices (optional)

Note 2. *The acknowledgments are sometimes placed at the end of the document, especially if it is a short one. In some cases, the table of contents is also at the end, but putting it at the beginning gives a better insight into the structure of the work.*

It is really important to stick the structure in Table 2 and the role of each part will be explored in more detail later in this chapter (*cf.* Section 4.1). The main objective at this stage – before writing – is to develop part 5 of this template. Which chapters and sections will form the base of the work?

Note 3. *A long report, like a Master's thesis, is usually broken down into chapters. Each chapter is divided into sections. A section can itself be split into subsections. The introduction and the conclusion are then considered as chapters.*

A shorter work, like an article, does not usually have chapters and is directly split into sections and possibly subsections, like this note. The introduction and the conclusion are then considered as sections.

The depth of levels when defining sections should not be too large (numberings like 1.3.2.1.5 should be avoided). It should be limited to subsections or sometimes, for a long document, to "subsubsections".

To be able to organize ideas, it matters to know where we are heading and raise the relevant issues. What is the "storyline" (the central idea) of the work? Which approaches are followed? How to present the ideas hierarchically? What are the personal contributions? The answers to these questions lead to the writing plan whose outcome is the future table of contents.

The chapters between the introduction and the conclusion are the body of the text and describe the object and the development of the study. They contain the following elements:

- subject or problem presentation;
- context of the problem (state of the art, known results);
- presentation of possible approaches;
- reasons for the choices between them;
- description of the work and the achievements:
  - presentation of the study key ideas;
  - explain each idea in a more in-depth way, but keep more technical parts in appendices (for example, present the main steps of an algorithm, but leave the detailed code in the appendix);
- comparison of the obtained results with previous results.

Do not organize the presentation of your work in a *chronological* way but use a *logical* and *hierarchical* structure. The reader does not need a journey through the maze of the problem solving process, nor through a piece of the story of your life, but an account of the results of a scientific work.

✗	✓
1. Study of method A	1. Existing methods
2. Move away from method A	2. Comparison of the various methods
3. What about method B?	2.1. Comparison objective criteria
4. Study method C	2.2. Pros and cons
5. Method C advantages	2.3. Choice of method
6. Method C implementation	3. Implementation

To set up your working plan, why not use a proven technique in programming, a *Top-Down* approach? Concepts are first split into "high-level" blocks which match the chapters and the process is repeated for each chapter to obtain sections, and possibly again later for subsections.

The writing plan is crucial: it is the skeleton of the report. It is not created magically overnight. With a first draft, ask yourself some questions: Can you tell which chapter or section each idea will fit in? Does the structure make sense? You should not hesitate to review your plan: if it is clear and logical, writing will be simpler.

Do not hesitate to have meetings with your project or thesis supervisor on a regular basis! Once you think that your writing plan is ready, discuss it with your supervisor and review it in light of their comments.

When reviewing the writing plan, consider the same issues and questions pertaining to programming with modularity mechanisms, line Object-Oriented (OO) programming. In OO programming, it is advisable [1] to respect the cohesion principle and to minimize the coupling between classes.

As a reminder, a class is *cohesive* if all its constructors' public methods are related to the (unique) concept that the class represents. In the same way, the contents of chapters and sections must be consistent and related to their title (matching what the reader expects to find in them).

A class depends on another class if it uses objects defined in this other class. In a written report, there are also – unavoidable – dependencies between sections. Their number should be minimized in order to save the reader from moving back and forth while reading the document.

2.2 Writing style consistency

Laying down the vocabulary and notations is another important step before starting to write to make sure that they will not change from the beginning to the end of the document. Choose your notations among those found most often in the field literature. This way, you will not have to search through the text for inconsistencies.

Finally you need to consider who you are writing for: most likely computer science teachers and possibly other students (if your work is very good, it might be proposed as an example for future students). Keep in mind that all your readers will not be specialists in the field: you should explain the context, define the basic concepts and describe the state of the art (what has already been done regarding the subject).

Put yourself in your fellow students' shoes: what do they know in the field? What is specific to your subject that they will not have studied in class?

2.3 Selecting page layout tools

From the beginning, you should choose the software tools that you will use to edit your work.

There are very popular *WYSIWYG* word processors (*What you see is what you get*), such as *OpenOffice*, *LibreOffice* and *Microsoft Word* which would seem to be very convenient because you can immediately see the result of your input. However, they are not always very appropriate for scientific and technical writing: the writer needs to both create the contents and manage the page layout at the same time. For a significant work, it then becomes challenging to obtain a final result with a "professional" look. While it is true that these tools offer automatic settings (*e.g.*, style sheets or autocorrection), these options may tend to lack flexibility. [2]

[2] Some irritating real-life examples: a text processor automatically capitalizes a word that you want to keep lowercase, changes your indentation or stubbornly adds a bullet list bullet at the beginning of your new paragraph.

An alternative is available: the LaTeX system[3] whose main goal is to provide an (almost) perfect typographical result. It is an *Open Source* system compatible with most operating systems. LaTeX is a high-quality page layout management system, which is particularly well-suited to writing scientific and technical documents. It has become *the* standard for article publishing in hard sciences, but can be beneficially used for any document type.

Unlike *WYSIWYG* word processors, LaTeX works on a plain text file (like a `.java` file) which contains text formatting commands. This file (source file, with a `.tex` extension) is then compiled using the `latex` command to produce a *DVI* file, which can be converted into *PostScript* or *PDF*. The *PDF* file can also be generated directly with the `pdflatex` command. This process is very similar to the production of an executable file in computer programming.

By taking this approach, the user does not need to worry about the page layout and can focus more on the content. For this reason, LaTeX can be called a *WYSIWYM – What you see is what you mean* – system. Learning LaTeX might take a while, but this effort is rewarded by a high quality result and provides more efficiency and productivity thereafter. Computer science students are encouraged to "boldly take the step"and to use LaTeX.

Here is an example of a small source file (that we called `gauss.tex`):

```
\documentclass{article}
\begin{document}
My \emph{first} experience with \LaTeX : $\sum_{i=1}^n i= \frac{n(n+1)}{2}$.
\end{document}
```

These instructions are understood as follows:

- this is the sort of document you intend to write: an "article" (of course, other pre-defined types are available for books, letters, *etc.*);
- the contents of the document are located between the `\begin{document}` and `\end{document}` instructions;
- the word `first` will appear in italics thanks to the `\emph` instruction (used to "emphasize" the word);
- the word `LaTeX` will be shown as the LaTeX logo;[4]
- the formula contains a sum and a fraction, which are easily encoded without requiring extra software to painfully "draw" them.

The command `pdflatex gauss.tex`, typed in a terminal will generate a *PDF* file with these contents:

$$\boxed{\text{My } \textit{first} \text{ experience with } \LaTeX\text{: } \sum_{i=1}^n i = \frac{n(n+1)}{2}.}$$

Some dedicated editors can also be used to encode the source file and generate the *PDF* output simply be clicking on a button or a menu item (*e.g.*, *emacs* and *Kile* for the usual three platforms, *TeXShop* for *Mac OS X*, *MikTeX* for *Windows*). Some even allow LaTeX files to be edited in a *WYSIWYG*-type environment, like *Lyx* [2].

LaTeX is much more powerful than you could understand from the above example. Among other things, it allows [3]:

[3]Pronounced "Latek". This document has been written using LaTeX.

[4]Note that LaTeX instructions are case-sensitive.

- the control of typographic elements (spacing after some characters, language-specific rules);
- long documents containing sections, cross-references (*cf.* Section 4.4), tables and figures to be edited;
- scientific articles, technical reports, books and presentation slides to be edited;
- complex mathematical formulae to be written;
- environments providing auto-formatting for theorems, definitions, algorithms, *etc.* (*cf.* Section 4.4) to be defined;
- automatic generation of the table of contents and the references.

Plenty of information on LaTeX is available in libraries and on the Internet [3]. A good introduction is given by *The not so short introduction to LaTeX2ε* [4] which is updated on a regular basis (Translations in more than 20 languages are also available on the same web site). For a more thorough document, the book *The LaTeX Companion* [5] is a well-known reference.

The reader will find more advice and examples about LaTeX, relevant to the elements addressed in this document, in Appendix B.

2.4 Version control systems

Version control systems are used to keep a record of the changes made on file sets and allow a rollback to a previous version. These are often used in programming, allowing members of a software development team to work simultaneously on a project. However, they can also be very useful to keep track of the various writing steps of a document.

A more detailed description is out of the scope of these guidelines, but the interested reader will easily find more information on the most used systems:

- *CVS*: `http://www.nongnu.org/cvs/` (the "historical" system, which is still frequently used)
- *subversion*: `http://subversion.apache.org/` (offers more functionalities than CVS)

These two systems use a "client-server" architecture, but a computer can play both roles at the same time. A distributed system is also available: *Bazaar*: `http://bazaar-vcs.org/`.

3 Bibliographic references

Bibliography is a key element of a scientific or technical work. The starting point of scientific reasoning is to build on existing work to propose personal contributions. Text and bibliographic references should make a clear distinction between personal contributions and results from other sources. Bibliographic references must be complete and accurate to allow them to be identified and retrieved ambiguity.

Section 3.1 explains how to obtain a book or an article used as a reference. Section 3.2 describes bibliography building and reference formatting. Plagiarism and simple means to avoid it are treated in Section 3.3.

3.1 Selecting bibliographic resources

There is a substantial amount of information on the Internet, but reliability is not always right around the corner. It can be challenging to decide whether a website is credible. Whenever possible, use published references (books, scientific articles from journals or conference proceedings) instead.

When building your bibliography, avoid relying on web pages only. They can be modified any time or even disappear. Aren't there any reference books? Scientific articles? The search for credible published references is part of the scientific writing work and tools exist to assist you in your search.

Generally speaking, avoid referring to *Wikipedia*, *quora.com*, *stackoverflow.com* or other popular sites if scientific publications are available. Nevertheless, when you refer to a web site (software home page, otherwise unpublished text), always mention the exact last visit date, since a change could occur at any time.

How to obtain a scientific article used as a reference? Before going to the library, check for an electronic version on a website such as the following:

- *Citeseer* (`citeseer.ist.psu.edu`),
- *Google Scholar* (`scholar.google.com`),
- *ScienceDirect* (`www.sciencedirect.com`),
- *ACM Digital Library* (`portal.acm.org`),
- *IEEE Digital Library* (`www.computer.org/portal/site/csdl/index.jsp`).

On the two first sites you will often find *preprints* (articles submitted for publication which might differ from the version eventually published). *ScienceDirect* gives access to the contents of hundreds of scientific journals, but you need to visit it from the University campus (or use a VPN connection). The *ACM* et *IEEE* digital libraries provide a large number of publications in computer science, but you need a personal authentication (check whether your supervisor has a subscription). Other computer science resources are available through the Science - Medicine and Pharmacy library web site.

For an article or a book which is difficult to find, check with your supervisor or check with the library desk. You may ask for an interlibrary loan if it is not listed in the UMONS catalog.[5]

3.2 Building your bibliography

The bibliography has to be built in parallel with the writing. You are strongly advised against adding extra references at the end of the process.

A bibliographic reference must be complete and accurate to allow the reader to easily retrieve it from the library or from the web sites mentioned in Section 3.1. The format of the references must be uniform. Examples:

- *for a scientific article published in a journal*: list of authors by name, article title, journal name, volume, issue number, year, pages (*e.g.*, references [6, 7] in this document);

[5]This catalog is available on the library web site: `http://biblio.umons.ac.be/webopac/Vubis.csp`.

- *for a book*: list of authors by name, book title, edition number, publisher, publisher address (city), year (*e.g.*, references [1,5,8–10]);
- *for an electronic resource (unpublished otherwise)*: list of authors by name, document title, URL, year, precise last visit date (*e.g.*, references [4,11]);
- *for software*: program name, title, home page URL, version, last visit date (*e.g.*, references [2,3]).

Do not list references not used in the text, or references to publications that you did not (at least) browse.

Reference a publication only once, even if it is mentioned several times in the text. For example, avoid multiple entries for different chapters of the same book. You can then refer to it as [4, Chap. 3], then [4, Chap. 7], or [4, pages 15–17] to indicate a more accurate location.

Throughout this document, bibliographic references are marked as $[x]$ where x is a number (a "label"). For a long document, like a Master's thesis, non-numerical labels could be a better option and allow the first author's surname and the publication year to be mentioned. For example, instead of [1], the label will appear as [Horstmann, 2006]. These labels appear then in the same format in the *Bibliography* or *References* [6] section.

If you choose numerical labels, it is worth mentioning the authors' names in the text. This way, the reader can remember the document and does not need to check the bibliography each time you refer to it. [7]

✗ "In [6], the authors suggest . . . "	✓ "Lindvall and Sandhal [6] suggest . . . " or "In Lindvall and Sandhal's article [6] . . . "

Numerical labels must appear between brackets [. . .] while parentheses are used to refer to equations.

3.3 Avoiding plagiarism

A scientific work is built on existing work to propose personal contributions. It makes sense to "use" the work of other people, but in a clear way, without making your own results from others (plagiarism). Plagarism is regarded as serious misconduct and real penalties can be imposed: from course failure to suspension or even expulsion from the University. Plagiarism can be detected using a variety of software tools [12].

There are several methods to mention the sources, depending on the way they are used.

a) If elements of your work are based on an existing work but you refer to it with your own words, a bibliographic reference must be mentioned.

   Example 1. *This section draws on a note written by Palme [11].*

b) However, if you want to *quote exactly* and use somebody else's words, this can be achieved either by using double quotes or by indenting a paragraph.

[6]The title *Bibliography* is often used for a long document while *References* is reserved for shorter reports.
[7]Note that the example below also sounds ugly when you read it aloud, being forced to say the number.

Example 2. *Valduriez highlights the importance of the presentation of a scientific article: "The presentation must ease the task of the reader (understanding the contribution) by relying on organization, brevity and illustration." [7, p. 373].*

Example 3. *Palme ends his note on plagiarism by [11]:*

> *Never use other people's text hoping you will get away with it. The risks for you are not worth it. Especially since it is so simple to avoid plagiarism by using the methods described above.*

Using a figure or data from external sources must also be acknowledged.[8] Finally, if you use a literal translation of a text, mention it and use the same rules as for an exact quote. In this case, you could quote the original text as well, so the reader can check that this is a faithful translation of the original.

Example 4. *Here is a famous quote from Albert Einstein [13]:*

> *Was man der eigenen Großmutter nicht erklären kann, hat man nicht wirklich verstanden. [You do not really understand something unless you can explain it to your grandmother.]*

Always mention your sources and use these rules when you use somebody else's words, to avoid plagiarism. When using a figure borrowed from another work, a reference is absolutely needed (usually in the caption). Example: "Source: Planet Software Evolution (www.planet-evolution.org)".

Another useful tool for references is the *DOI* system (for *Digital Object Indentifier*). A DOI is an alphanumeric key, stored on a permanent basis allowing the easy retrieval of a digital document. For example, the Lindvall and K. Sandahl article [6] is associated to the DOI *10.1016/S0164-1212(98)10019-5*. The link to the corresponding web page is built from `dx.doi.org/` followed by the key. This URL redirects to the hosting web site. In our example, the DOI redirecting web page is `dx.doi.org/10.1016/S0164-1212(98)10019-5`. Whenever the hosting page URL changes, this link is maintained and remains usable.

4 Writing

Besides a bibliography, writing builds on these core elements: structure, contents, style and presentation. Structure is addressed in Section 4.1. Section 4.2 deals with producing scientific and rigorous content. Finally, style and presentation advice is given in Sections 4.3 et 4.4.

Note 4. *A written work is organized into chapters, sections, subsections, etc. For the sake of simplicity and brevity, we will use the word* section *as a presentation* unit*, regardless of its level in the structure.*

4.1 Structure

The overall structure of a written work has already been considered in Section 2.1. At this point, a writing plan must be present, even if it is not finalized. Then comes the understanding of the purpose of the various parts listed in Table 2, p. 4 (introduction, conclusion, *etc.*).

[8]Usually in the figure caption for figures

Completing the general structure

The writing plan already lays down the section titles. These are important keys for the reader. When writing the contents of a section, it could make sense to reconsider its title, keeping it short but self-explanatory enough. Keep the structure skeleton intact and use the same grammatical format for all the titles.

✗	✓
1. To introduce the problem	1. Introduction
2. The existing solutions	2. State of the art
3. My solution	3. Results
4. A few comments as a conclusion	4. Conclusion

When using titled (sub-)subsections, unnumbered titles, absent from the table of contents, are a possible choice. This is the option used for this document (*e.g.*, the current section).

✗

2. Tarjan's Algorithm
   2.1. General principle
      2.1.1. DFS tree usage
         2.1.1.1. Reminders
            Text.
         2.1.1.2. Pseudocode
            Text.

✓

2. Tarjan's Algorithm
   2.1. General principle
      2.1.1. DFS tree usage
         Reminders
            Text.
         Pseudocode
            Text.

Local structure

Sections already present in the writing plan give the *global* structure of the work. Within each section, a *local* structure of the text must also be set.

The presentation units used to give a structure in a section are the *paragraphs*, appearing in a logical sequence. The paragraph *itself* must also be organized in a logical way. Its few sentences are linked together and are related to a point treated in the section.

✗ Like the overall structure, do not organize the paragraphs in a *chronological* way. Organize the results presentation and avoid giving a chronological list of mistakes and failed attempts. The first paragraph of a section should introduce its topic.

💡 Try to balance the paragraphs and make them not too short (one or two lines) and not too long (half a page). A very short paragraph can be used to highlight an important point, but should remain an exception.

Introduction

The introduction is usually organized as follows [7]: context, problem definition, presentation of possible already existing solutions and their limitations, work objectives and key ideas. It should then end with a short description of the following chapters (see for example the two last paragraphs of Section 1).

The introduction is an important section. It should convince the reader that the document is worth reading. It should attract an *a priori* not interested reader and should explain why the treated topic is important, what your contribution will be and why the proposed solutions are appropriate. Keep in mind that the reader still has to read your document, is not an expert in the field and does not know the subject.

In your introduction, explain the *purpose* of your work and not the fact that it is a computer science work (the reader should have guessed as much).

Valduriez [7] suggests *starting* to write a scientific article with the introduction to set the goals and the structure. If you have begun to write chapters (the state of the art, for example) but pieces of the whole picture are still missing, then it might make sense to keep the introduction for later (without rushing it before the deadline). Although writing a scientific article or a thesis are different things, both options have their pros and cons.

Chapters and Sections

Writing chapters and sections are at the heart of the writing process, describing the subject in further detail, developing the ideas following the established roadmap and presenting results and convincing the reader of their validity. The contents of the chapters must follow scientific reasoning. Some guidelines for producing precise and rigorous content are given in Section 4.2.

When writing the contents of a section, keep in mind that its title – used to give a structure to the text and introduce the section topic – might be skipped by the reader! Its first paragraph should introduce the section and explain its purpose. For high-level sections (like chapters), starting with a short description of its contents and its subsections is useful. Note that this principle has been respected in this document.

Conclusion

The conclusion is the last part of the written work (with bibliography and appendices not being considered as integral parts of the text). It is usually organized as follows [7]: work summary and contributions, reminder of the main results, potential applications of these results, limitations of the proposed solution and possible directions for further work.

The conclusion should remain fair but highlight the author's contribution to the subject.

The conclusion is also an important part of the document. It summarizes the personal contribution of the work and highlights the main results. Unlike in the introduction, we will assume that the reader has read the preceding chapters and learned about the subject. The conclusion should then allow the reader to confirm their views on the merits of the work.

The conclusion should logically be written last [7].

Other parts of the document

Besides the parts covered so far, a written work also consists of the following parts (*cf.* Table 2, p. 4):

- the *appendices* (optional) are placed at the end. They contain elements which do not significantly contribute to the understanding of the work and whose reading can be skipped by the reader. They are included for the sake of completeness. In the appendices you will find implementation details of an algorithm described in the main part of the document or a description of the syntax of programming languages;

  ✗ Do not include long full source codes in the appendices. A CD-ROM or a USB stick – containing the code, the data or even executable programs – is more practical and more environmentally friendly.

- the *cover page* including: department name and the university where the work was done; university and faculty logos; author's name; title; date (month, year); supervisor's name. For a Master's thesis, mention the related diploma and check for specific university/faculty conventions;

  ✗ The information listed above should be present on the cover page. Do not use "Master's thesis" or "Project report" as a title. Give a *real title*, properly describing the subject, like "Balanced search binary trees". You are of course allowed to mention on the cover that your work is a "project report" or another kind of work, but not as a title (rather a subtitle).

- the *title page* (sometimes preceded by a blank page) has the same contents as the cover page;

- the *table of contents*, essential for a large document, is usually located at the beginning;

- the *acknowledgments* (optional) mention the people who helped you (supervisor, readers, proofreaders, *etc.*). They can be at the beginning (*e.g.*, Master's thesis) or at the end (*e.g.*, article or short report).

4.2 Contents

To convince the reader of the merits of your work, a scientific approach must be followed (hypotheses, measurements, checking, proof, *etc.*). Any claim or figures must be supported by scientific evidence, or must be referenced when it does not come from a personal contribution. The reader should see a clear distinction between personal contributions and material from other sources (*cf.* Section 3.3).

Always discuss technological choices and possible alternatives. Why did you use this tool, language, algorithm, formalism? Define and use precise and objective criteria to support your choice (Big-O notation complexity, results of benchmarks and CPU time, functionalities, *etc.*).

✗ In software evolution, only 30 to 40% of the classes actually modified are predicted to be modified.	✓ On the basis of an empirical study, Lindvall and Sandahl [6] observe that only 30 to 40% of the classes actually modified were predicted to be modified.
✗ We chose to use the programming language X because we already used it in class before for the course A.	✓ Multiple inheritance is a feature present in programming language X, but not in languages Y and Z. This feature is very important in our case, because (. . .).

✗ I myself believe that TCL and TK are marvelous languages, allowing many things without requiring the student to read ten books and have 10 years of experience.	? Marvelous? Many things? ten books? 10 years of experience? Give objective criteria in favor of these languages. Be unbiased and give quantitative arguments.
✗ Among the two algorithms presented above, we selected algorithm A, for various reasons. First, it allows a simpler manipulation of the data structure. Then, algorithm A is more "objective" than B, which seemed to be necessary	✓ Time complexities in the *worst case* of both A and B algorithms are the same ($O(n^2)$). However, *average* execution times differ: $\Theta(n \log n)$ for A and $\Theta(n^2)$ for B. On random data, better execution times are then expected for algorithm A and this is indeed confirmed by the tests shown below $(\ldots)$

Computer benchmarks

For computer benchmarks or any tests performed on a computer, the hardware configuration should be specified, as well as the software used (origin of the programs – home web sites –, versions, *etc.*). This information is necessary to allow the reader to reproduce the tests.

✗ The execution time of our program on dataset X is 12 seconds and 23 seconds on dataset Y.	✓ Table x shows the execution times, expressed in CPU micro-seconds, on the different datasets. These tests were performed on a computer with the following specifications: Dell Dual Core, 2.66 GHz, 2 Gb RAM, system SuSE Linux 10.0 (kernel 2.4.2), java 1.5.0, *etc.* To compute the CPU time, the *ThreadMXBean* class was used.

Algorithms

To properly explain a complex algorithm, pseudocode is not enough. Key ideas should be disclosed first, before going progressively into detail. For an algorithm essential to understanding the work, one could:

1. explain the algorithm's purpose (goals, input, output);
2. describe the main ideas in plain English;
3. go into more detail for key ideas;
4. display the pseudocode (only main parts if it is too long);
5. apply the algorithm to a small example;
6. prove its correctness;
7. mention (and give a proof) of its time and memory complexities;
8. if necessary, give implementation details in an appendix.

To describe algorithms or data structures, consider computer science renowned books as models, such as those written by Aho and Ullman [8] and by Cormen *et al.* [9].

4.3 Style

Your purpose is not to write a novel but a scientific paper. Your style should be appropriate to the nature of the document. The contents should be intelligible for a non-expert. [9] Your text should not look like "shorthand notes" or sound like "spoken language", but consist of correct sentences. Again, take a renowned book as a model. A suitable style for a research paper should follow these rules [7, 10].

1. *Precision*. Concepts (*formalism*) must be defined with precision when they first appear and always referred to with the same expression (*consistency*). When introducing a concept, it should be emphasized with italics. The same rule applies to notations and symbols.

2. *Concision*. Get to the point and use short sentences. Avoid useless words. Limit a sentence to a single idea when possible. A complex sentence can be split into shorter ones.

3. *Neutrality*. Keep a detached and non-emotional style. Do not use the singular form of the first person "I", except in the acknowledgments.

4. *Verb tenses*. Active voice is more direct than passive voice. Use the present tense for a more dynamic style (except for the presentation of your results or for the conclusion where the past tense could make more sense). Sentences must have verbs.

5. *Correct spelling*. Check your spelling and grammar.

6. *Typography*. Follow the language-specific typography rules (*e.g.*, double quotes style – distinguish left/right quotes when available in the character font –, no spacing before a comma, colon, semicolon, exclamation or question mark in English, superscript mark to a footnote outside punctuation in English). [10] Use non-breaking spaces to avoid starting a line with a reference mark like $[x]$, with "…", or with anything which would give an ugly look.

7. *Examples*. Illustrate important or complex concepts with simple examples.

8. *Acronyms*. Avoid unnecessary abbreviations and acronyms. When you need them, indicate their meaning where they first appear.

To improve your writing style, do not hesitate to check for instance the book written by Strunk and White [10]. The guidelines given above also apply to titles and *a fortiori* to the title of your document. Below, you will find a few examples and counterexamples.

[Precision: formalism and consistency]

✗ The number of nodes of a graph is denoted as n. A graph is a set of vertices V and edges between nodes. $(\dots)$ A graph for which $|V| = 0$ is an empty graph.

✓ An *undirected graph* G is an ordered pair (V,E), where V is a finite set of elements called *vertices* and E is a set of unordered pairs of vertices distinct from V. Each element $\{s,t\} \in E$ is an *edge* joining vertices s and t. One denotes n the *number of vertices* of G. $(\dots)$ If $n = 0$, then the graph is *empty*.

[9] Although the reader is most likely a computer science professor, researcher or student.

[10] The `babel` package, available with LaTeX, can be used to automatically apply some language-specific typography rules.

[Precision: consistency]

✗ (. . .) software evolution (. . .) software change (. . .) software modification (. . .)	? Do these various expressions refer to the same concept? If only the first one was defined in the text, the reader could have some doubt. Avoid using synonyms for the same concept.

[Concision]

✗ Saving the data can be done using plain text files, representing the data in a readable format for a human being, with a sequence of characters; or using binary files representing the data with a sequence of bytes.	✓ There are two different ways to store the data: *plain text* or *binary* format. Data in plain text are a sequence of *characters* readable by a human being. Data in binary format are a sequence of *bytes*.

[Neutrality]

✗ I was starting to have second thoughts about programming the four operations, but I had set myself an ambitious target and I wanted *to run where the brave dare not go and reach the unreachable star*. In any case, this would make my day even more interesting and challenging!	? Who cares? Avoid this kind of demonstration of personal feelings. And next time, add a reference to Joe Darion and Mitchell Leigh.

[Concision, precision, neutrality]

✗ While learning by doing, one notes that colors can be obtained by mixing other colors. One reached pretty quickly the conclusion that there must exist basic colors from which every color could be obtained. These famous colors are called basic colors: red, green and finally blue. From these colors, it is then possible to obtain any color by a clever mix of these three colors.	✓ Any color can be obtained by mixing 3 basic colors: red, green and blue.

[Verb tenses]

✗ • The algorithm will be executed in $O(n)$ time. • We first tried method X but (. . .). • A value of 0 representing black and a value of 1 representing white.	✓ • The algorithm time complexity is $O(n)$. • Method X is not appropriate (. . .). • A value of 0 represents black and a value of 1 represents white.

[Acronyms, precision]

✗ The purpose of this work is to develop a *CMS*.	✓ The purpose of this work is to develop a *Contents Management System* (*CMS*). A *CMS* is (. . .).

When you are done writing the first pages of your report, submit them to your supervisor. Stylistic lapses (that you might not be aware of) are often recurring mistakes and correcting them early will save your and their time.

4.4 Presentation

To properly design the page layout of your work, use an adequate text processing software, like LaTeX (*cf.* Section 2.3). Choose the character fonts and sizes carefully and avoid mixing too many of them. Normally, text in a paragraph is *justified*.[11] Title hierarchy must be visually consistent (*e.g.*, title size).

Footnotes may be used (not too many) to give precisions which do not significantly contribute to the understanding of the work. They might be skipped by the reader. To make sure that a precision is read, it should be inserted in the text, between parentheses, instead. A footnote should *never* be used as an alternative to a bibliographic reference, even for mentionning a web site.

Other elements, described below, add visual comfort and improve presentation: environments, cross-references, tables and figures.

Environments

Use environments to highlight important text parts.

Definition 1. An *environment* is a text part with a well-defined role, like a theorem, a definition, an example, a proof, a piece of program code. This text part is formatted to allow direct identification of its role. Some environments are numbered to be easily referred to somewhere else in the text.

Example 5. *Definition 1 is displayed in an environment used for important definitions. This example itself is shown in an environment whose purpose is to illustrate a concept.*

✗ Algorithm A has a $O(n^2)$ complexity because (long list of arguments…). It produces the expect result because (another long list of arguments…).

✓

Proposition 1. *Algorithm A's complexity is $O(n^2)$ where n denotes the input array length.*

Proof. (long list of arguments…) □

Theorem 1 (Algorithm A correctness)**.** *When the execution of Algorithm A ends, the resulting array contains numbers sorted in ascending order.*

Proof. (other long list of arguments…) □

Cross-References

Definition 2. Any text element (section, equation, definition, bibliographic reference, figure, theorem, table, *etc.*) with a *label* (a number, a composite number or a group of numbers/letters) can be quoted in the text using that label. This is called *cross-referencing*.

Example 6. *We introduced the environment concept in Definition 1.*

The word used for a cross-reference to a numbered environment, a figure or a table is capitalized ("See Proposition x", "as shown in Table y").

Example 7. *This document does not contain many equations. Nevertheless, you will find Equation (1) in the Green theorem formulation (*cf.* Theorem 2, p. 24).*

[11] Text is aligned along both margins, this is achieved by adjusting word-spacing.

Figures and tables

Tables and figures enhance the readability and illustrate results or measurements. A figure can assist in the understanding but does not represent an argument in itself. Choices or results must still be substantiated in the text. A table summarizes facts and can be used, for example, to compare various solutions or to show benchmark results.

A figure or a table must *always* be numbered and come with a caption (above a table but under a figure).

Do not use unnecessary figures which are not referred to in the text. A cross-reference should appear in the text for every figure or table.

5 After writing

Writing is an iterative process. Each written part should be reviewed several times until the desired result is achieved. Do not hesitate to "discard" unnecessary text fragments, reorganize, read and read again, trying to consider the text from the reader's perspective.

Once you have a first (almost) complete draft of your document, use the available means to review it:

- *always* use a spell checker;
- use a good dictionary and a good English grammar book (or an online resource);
- ask various people to read your draft (acquaintances for clarity, grammar and spelling; your supervisor for the contents);
- review the introduction and conclusion sections very carefully;
- check the consistency of the vocabulary and notations;
- improve the document's graphical appearance, take a "step back" to check the page lay-out/look: figure arrangement, unwelcome large gaps within a section, *etc.*

It is sometimes difficult to have the necessary distance to objectively examine your own text after numerous hours of work. Do not fear criticism: proofreaders are there for your convenience and can allow you to improve your writing.

6 Oral presentation

Once your document is finally submitted, an oral presentation can also be on the menu. The principles and advice given so far for the writing also apply to an oral presentation:

- prepare a good presentation plan;
- be clear and structured;
- be aware of the audience's level of expertise;

- introduce the main objectives first;
- define concepts when they first appear;
- highlight personal contributions;
- end with a proper conclusion

However, there are differences between a written document and an oral presentation, both in form and in contents:

a) *Contents*. The written report must be complete and precise. The time allotted for an oral presentation is usually short and does not allow every detail of your work to be covered. Get to the point, deliver key ideas without going into technical details. If you cannot cover the contents of all the chapters, focus on the most important ones and only briefly mention the contents of the others.

   With different contents, structures may differ as well: do not necessarily try to follow the document writing plan. Build a new plan matching the presentation's key ideas.

b) *Form*. A presentation is an *oral* performance and slides are just intended to support your talk. In a written report, one expects proper sentences. A slide should not be packed with information, but should use keywords, diagrams, short sentences. Limit a slide contents to only one or a few ideas.

   As for a written report, it is worth giving the audience a glimpse of your results *at the beginning* of the presentation, a "teaser" to keep your audience interested and open to learning more. It is also important to remember the results when concluding to make sure that your message has been heard loud and clear.

   To better explain abstract concepts, use practical examples, which are simple but self-explanatory.

Do not read the text from your slides. The audience is able to read it without help and expects an explanation of your work. Keep eye-contact with the public rather than staring at the screen.

Take the time to explain the contents of each slide (usually one or two minutes). Therefore, slides should not be too numerous or too packed with information. You need to rehearse and time[12] your presentation: if it takes too long, do not be tempted to increase the speed but reconsider your choices. *Never* exceed the time limit: keeping to time shows respect for the audience.

7 Conclusion

A scientific or technical document allows the author to present their work, and highlight their results and their personal contributions. This requires good quality writing and the basic elements to achieve this have been presented in this note. The writing plan is organized logically, as the global and local structures of sections and paragraphs. The contents are rigorous and presented step by step (main ideas first, before going into details). The style is precise, concise, dynamic and neutral. The presentation is clear and good looking. Scientific writing should be seen as a part of scientific reasoning, building on existing work to propose personal contributions. Using the words or ideas of another author must respect simple but precise rules, to avoid plagiarism.

[12]Be aware that the durations of the rehearsal and the actual presention could differ slightly due to stress.

The writing approach proposed in these guidelines has two main stages: the organization of ideas and the writing itself. Both are iterative: the writing plan and the text must be reviewed until the desired result is achieved.

Some advice has also been given for the oral presentation of a scientific or technical work.

Examples given to help the reader to improve their writing were chosen from the field of computer science. In particular, we expect this document to help computer science students for their written assignments (*e.g.*, reports, Master's thesis) and to spare their teachers from having to repeat the same instructions again and again.

Acknowledgments

We would like to thank our colleagues V. Bruyère, J. Cardinal (ULB), A. Decan, O. Delgrange, T. Mens and J. Wijsen for their comments on previous versions. Hadrien Mélot also wishes to thank P. Hansen for his valuable advice while writing articles with him. Some real-life pieces of students' works (written in French) were used as the basis for some examples and freely adapted to English. We thank them for their involuntary and anonymous involvement. We also thank N. Dickson (CLV-UMONS) for her careful proofreading of this document.

References

[1] C. Horstmann. *Java Concepts*. Wiley, New Jersey, 4th edition, 2006.

[2] LyX. The Document Processor. URL: `www.lyx.org`, Version: 2.3.3. (last visited on December 5, 2019).

[3] LaTeX. A Document Preparation System. URL: `www.latex-projet.org`, Version: LaTeX2ε (last visited on December 5, 2019).

[4] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. The Not So Short Introduction to LaTeX2ε. URL: `ctan.tug.org/tex-archive/info/lshort/english/`, 2018 (last visited on December 5, 2019).

[5] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion (Tools and Techniques for Computer Typesetting)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 2004.

[6] M. Lindvall and K. Sandahl. How Well do Experienced Software Developers Predict Software Change? *J. Systems and Software*, 43(1):19–27, 1998.

[7] P. Valduriez. Some Hints to Improve Writing of Technical Papers. *Ingénierie des Systèmes d'Informations*, 2(3):371–375, 1994.

[8] A. Aho and J. Ullman. *Foundations of Computer Science*. Computer Science Press, Inc., New York, 1992.

[9] T.H Cormen, Ch.E Leiserson, R.L Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, London, 2nd edition, 2002.

[10] W. Jr Strunk and E.B White. *The Elements of Style*. Macmillan, New York, 4th edition, 1979.

[11] J. Palme. Plagiarism. URL: `people.dsv.su.se/~jpalme/plagiarism.html`, (last visited on December 7, 2019).

[12] Moodle Docs. Moodle plugin directory. URL: `moodle.org/plugins/browse.php?list=category&id=35`, (last visited on December 7, 2019).

[13] M.O. Scheele. Die kunst vom lesbaren source code. URL: `www.jobpushy.de/blog/24/die-kunst-vom-lesbaren-source-code`, (last visited on December 7, 2019).

Appendix A: Common Latin abbreviations

Latin abbreviations are often found in scientific articles. Table 3 lists the most common ones with their meanings and possible alternatives. Using abbreviations is of course optional, but helps the reader to focus on the more important elements.

Table 3 – Common Latin abbreviations

Abbreviation	**Latin words**	**Meaning**	**Alternative**
e.g.	*exempli gratia*	for example	
i.e.	*id est*	in other words	that is
cf.	*confer*	compare	see
etc.	*et cetera*	and other things	
et al.	*et alia*	and others	
Q.E.D.	*quod erat demonstrandum*	which was to be demonstrated	$\square$

Remarks:

1. When an alternative is available, choose your favorite abbreviation, but use it uniformly throughout your document.
2. Respect the exact typography (dots are important: they signal the fact that the word is an abbreviation).
3. Latin abbreviations are sometimes italicized, as are Latin words written in full (*e.g. a priori* or *per se*) but the rule is not always clear in English (when applicable, check the publisher's guidelines).
4. "*e.g.*" and "*i.e.*" should not be confused: the former is used to give examples, the latter to reformulate something with other words.
5. "*e.g.*" and "*i.e.*" are usually followed by a comma in U.S. English. If you chose to use a comma, be sure to use the same rule uniformly.
6. Do not use extra periods (omission marks) after "*etc.*"
7. A list of examples after "*e.g.*" should not end with "*etc.*" or extra periods.
8. "*et al.*" is often used in a reference when there are many authors (for instance: Cormen *et al.* [9]).

Appendix B: More tips on LaTeX

The purpose of this appendix is not to give a detailed documentation of the LaTeX system but to present a few examples and tips. For each of them, the relevant section of this document is mentioned. For a more general information, bibliographic references are given in Section 2.3.

Note 5. *Some examples require a* package *to be included. This is easily done by inserting the command* `\usepackage{PackageName}` *in the document preamble (*i.e.*, between the* `\documentclass` *and* `\begin{document}` *commands).*

Easily editing scientific contents (*cf.* Section 2.3, p. 7)

In the example below, the theorem and the equation are formatted and numbered automatically. *Math* mode, used to write mathematical expressions, is delimited by `$` signs. It is also implicitly present in the *equation* environment.

LaTeX CODE

```
\begin{thm}[Green] Let $C$ be a positively oriented,
piecewise $C^1$, simple closed curve in a plane,
$D$ the domain bounded by $C$ and $Pdx + Qdy$ a differential $1$-form
on $\mathbb{R}^2$. If $P$ and $Q$ have continuous partial derivatives
on an open region containing $D$, then:
\begin{equation}
\int_{C} P dx + Q dy = \iint_{D} \frac{\partial Q}{\partial x}
- \frac{\partial P}{\partial y}\  dx\ dy
\end{equation}
\end{thm}
```

OUTPUT

Theorem 2 (Green)**.** *Let C be a positively oriented, piecewise C^1, simple closed curve in a plane, D the domain bounded by C and $Pdx + Qdy$ a differential* 1*-form on* $\mathbb{R}^2$*. If P and Q have continuous partial derivatives on an open region containing D, then:*

$$\int_C Pdx + Qdy = \iint_D \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\ dx\ dy \tag{1}$$

Managing bibliographic references (*cf.* Section 3.2, p. 9)

The BibTeX [5] tool comes with most LaTeX distributions. With this reference management software, bibliographic references stored in a `.bib` file are automatically formatted. Here is an example of a BibTeX entry:

```
@book{Goossens04,
  author = {Goossens, M. and Mittelbach, F. and Samarin, A.},
  title = {{The LaTeX Companion}},
  publisher = {Addison-Wesley},
  address = {Boston},
  year = {2004}
}
```

This reference can be encoded in a `.tex` file as `\cite{Goossens04}` where it needs to be referred to. Such pre-written BibTeX entries can be found online for computer science publications (*e.g.*, on the *DBLP* web site, `dblp.uni-trier.de`).

BibTeX also allows easy modifications of the way labels are displayed: simple number or authors' names and date, *etc*. Another convenient feature of this tool is that only references actually used in the text appear in the bibliography section.

Merging bibliographic references (*cf.* Section 3.2, p. 9)

The `cite` package allows successive references to be grouped where they are mentioned and displayed as a range.

LaTeX CODE

```
Here is a list of references~\cite{lyx, latexweb, Strunk79,
Lindvall98,notShort}.
```

OUTPUT WITHOUT THE `cite` PACKAGE

Here is a list of references [2, 3, 10, 6, 4].

OUTPUT WITH THE `cite` PACKAGE

Here is a list of references [2–4, 6, 10].

Direct quotations (*cf.* Section 3.3, p. 10)

To quote someone's exact words by indenting the paragraph, use the `quote` environment.

LaTeX CODE

```
Regarding illustrations, Valduriez specifies that it may be of any kind like drawings,
examples or algorithms and gives the following advice~\cite[p. 374]{Valduriez94}:
\begin{quote}
In all cases, avoid illustrations that are either too simple, \emph{e.g.}, a 2-line
algorithm, or two complex, \emph{e.g.}, a 50-line algorithm. Like text, drawings
should be brief and precise, avoiding unnecessary coloring and details.
\end{quote}
```

OUTPUT

Regarding illustrations, Valduriez specifies that it may be of any kind like drawings, examples or algorithms and gives the following advice [7, p. 374]:

> In all cases, avoid illustrations that are either too simple, *e.g.*, a 2-line algorithm, or two complex, *e.g.*, a 50-line algorithm. Like text, drawings should be brief and precise, avoiding unnecessary coloring and details.

Structure and table of contents (*cf.* Section 4.1, p. 11)

The various sections are defined in the text using the `\chapter`, `\section`, `\subsection` and `\subsubsection` commands. Their titles are then formatted and numbered by LaTeX. To insert a table of contents in the document, one simply uses the `\tableofcontents` command.

Note 6. *The* `.tex` *file must be compiled twice to obtain a correct result. It first creates (or updates) a* `.toc` *file with section names and page numbers. Then, it uses this file to actually update the table of contents. Sometimes (a table of contents of several pages), a third compilation could be needed (this information appears in the compiler messages). A similar procedure takes place when compiling the bibliography (*`.bbl` *file) or using cross-references (*`.aux` *file).*

Explaining algorithms (*cf.* Section 4.2, p. 15)

Pseudocode algorithms can be nicely displayed using the `algorithm` and `algorithmic` packages.

LaTeX CODE

```
\begin{algorithm}
\caption{Linear search for a  maximum}
\begin{algorithmic}[1]
\REQUIRE an array of integers $A$
\ENSURE the value of the largest integer in $A$
\STATE $max \leftarrow -\infty$
\FOR{$i \leftarrow 1$ to $length[A]$}
\IF{$max < A[i]$}
\STATE $max \leftarrow A[i]$
\ENDIF
\ENDFOR
\RETURN $max$
\end{algorithmic}
\end{algorithm}
```

OUTPUT

Algorithm 1. Linear search for a maximum

```
Input:  an array of integers $A$
Output:  the value of the largest integer in $A$
1: $max \leftarrow -\infty$
2: for $i \leftarrow 1$ to $length[A]$ do
3:     if $max < A[i]$ then
4:         $max \leftarrow A[i]$
5:     end if
6: end for
7: return $max$
```

Note that to obtain the result above, we redefined a couple of commands in the preamble:

```
\renewcommand{\algorithmicrequire}{\textbf{Input:}}
\renewcommand{\algorithmicensure}{\textbf{Output:}}
```

Footnotes (*cf.* Section 4.4, p. 18)

Inserting a footnote is simply done with the `\footnote` command.

LaTeX CODE

```
Alan Turing\footnote{June 23rd, 1912 -- June 7th, 1954.}
was an English mathematician, logician and cryptographer.
```

OUTPUT

Alan Turing[13] was an English mathematician, logician and cryptographer.

[13] June 23rd, 1912 – June 7th, 1954.

Environments (*cf.* Section 4.4, p. 18)

You can easily define your own LaTeX environments (see [4] for example), automatically numbered and formatted. Other pre-defined environments are available in the `amsmath` and `amsthm` packages, such as `proof` which adds a square (□) to indicate the end of a proof (instead of *Q.E.D.*).

LaTeX CODE

```
\begin{proof}
By induction. (\ldots)
\end{proof}
```

OUTPUT

Proof. By induction. (. . .) □

Cross-References (*cf.* Section 4.4, p. 18)

LaTeX manages cross-references automatically: if a new definition is inserted before Definition 1, its label and those that follow are automatically incremented by 1 in the document. To refer to an existing object, one must define a `label` and create a cross-reference using the `ref` command. The page number is also available via the `pageref` command.

LaTeX CODE

```
\begin{defn} \label{def_algo}
An \emph{algorithm} is a finite sequence of non-ambiguous steps
allowing a problem to be solved.
\end{defn}

In Definition \ref{def_algo} (\emph{cf.} p. \pageref{def_algo}),
an important element is missing: the concepts of algorithm
input and output.
```

OUTPUT

Definition 3. An *algorithm* is a finite sequence of non-ambiguous steps allowing a problem to be solved.

In Definition 3 (*cf.* p. 27), an important element is missing: the concepts of algorithm input and output.

Note 7. *The* `.tex` *file must be compiled twice when using cross-references. It first creates (or updates) a* `.aux` *file with label names and page numbers. Then, it uses this file to actually update the cross-references.*

Spell-checking (*cf.* Section 5, p. 19)

There are several spell-checkers that you can use to check your `.tex` files, depending on your setup (operating system, text editor). For example, a spell-checker is included in *TeXShop* (*Mac OS X*). Under *Linux*, the `ispell` program is available.

Editing presentation slides with LaTeX (*cf.* Section 6, p. 19)

If you enjoyed writing your report with LaTeX, why would you not use it for your presentation slides as well? There are packages and document classes available for this purpose, such as *Beamer* or *Prosper*.

You will need to install these packages, as they are not included in standard LaTeX distributions. They can be found at the following addresses:

- *Beamer*: `sourceforge.net/projects/latex-beamer/`
- *Prosper*: `sourceforge.net/projects/prosper/`