Trusted Component
Exercise Session I

Manuel Oriol
November 8th 2006
Reminder

- No one metric is adapted to every project to evaluate the quality of a project
- A metric that apply to a project will apply to next iterations
Exercise description

• For each of the forthcoming guidelines describe the technique for checking it
• Discuss the validity of the guideline
Do not use gotos!

The go to statement as it stands is just too primitive; it is too much an invitation to make a mess of one's program. One can regard and appreciate the clauses considered as bridling its use. I do not claim that the clauses mentioned are exhaustive in the sense that they will satisfy all needs, but whatever clauses are suggested (e.g. abortion clauses) they should satisfy the requirement that a programmer independent coordinate system can be maintained to describe the process in a helpful and manageable way.

Dijkstra: http://www.acm.org/classics/oct95/
Write Clearly

Don't be overly clever. Just because you can do something in a single line of code doesn't mean you should. Do not write code just for the compiler; write the code for you and your fellow programmers. Remember it may be you who returns to this code days, months, or years later and can't remember what that complex statement does. Never sacrifice clear code for "efficient" code.

http://www.codecogs.com/pages/standards/programming.htm
Overloading functions can be a powerful tool for creating a family of related functions that only differ in the type of data provided as arguments. If not used properly (such as using functions with the same name for different purposes) they can, however, cause considerable confusion. When overloading functions all variations should have the same semantics (be used for the same purpose).

http://www.codecogs.com/pages/standards/programming.htm
Parenthesis

It is generally a good idea to use parentheses liberally in expressions involving mixed operators to avoid operator precedence problems. Even if the operator precedence seems clear to you, it might not be to others - you should not assume that other programmers know precedence as well as you do.

http://www.codecogs.com/pages/standards/programming.htm
assert and verify

An assertion statement specifies a condition that you expect to hold true at some particular point in your program. If that condition does not hold true, the assertion fails, execution of your program is interrupted, and the Assertion Failed dialog box appears.

The key feature of the assert statement is that it is only included in 'Debug' executable code, with these statements being automatically removed for Release code. As such, assert only slows down Debug executables, but has no ill effect (either speed or size) on the Release code.

http://www.codecogs.com/pages/standards/programming.htm
Templates - Generics

Template are a powerful feature of C++ that allow a function to take on many different forms during compilation. We strongly encouraged their usage, since they can allow functions to become significantly more flexible, while remaining type safe.

http://www.codecogs.com/pages/standards/programming.htm
Command-Query separation

It states that every method should either be a command that performs an action, or a query that returns data to the caller, but not both. More formally, methods should return a value only if they are referentially transparent and hence possess no side effects.

Multiple Inheritance

Don't use multiple inheritance! Multiple inheritance appears to be relatively harmless at first but can lead to maintenance nightmares after a time. It also requires more intervention from the compiler and thus introduces a small performance penalty. Some ways to avoid multiple inheritance are; to move the required functionality up the class hierarchy or to use intermediate objects that refer to the original objects.

http://www.xaraxtreme.org/developers/documentation/top_20_guidelines.html
Indentation Style

For core GNOME code we prefer the Linux kernel indentation style. Use 8-space tabs for indentation.

Using 8-space tabs for indentation provides a number of benefits. It makes the code easier to read, since the indentation is clearly marked. It also helps you keep your code honest by forcing you to split functions into more modular and well-defined chunks - if your indentation goes too far to the right, then it means your function is designed badly and you should split it to make it more modular or re-think it.

The Standard Template Library establishes uniform standards for the application of data containers, and a small selection of algorithms. Iterators are provided to traverse the data containers, and all the algorithms are fully compatible with each container.
Variables

In the use of variables in a program the following guidelines should be followed:

• Declare variables when they are needed and not before. Try to restrict the scope of a variable by keeping it as local as possible. However, avoid defining local variables that hide declarations at higher levels.

• Consider carefully the scope of the variable – how long is the value this variable holds needed. Despite popular misconceptions, it is not inefficient to redeclare many times simple variables, such as int, double, char, etc.

• Try to initialise variables once they have been declared. If you don’t have enough information to initialise the variable then it shouldn’t have been declared. Never initialise it will an empty value only to be assigned a new value later.

• Variables should be declared in a separate declaration statement in order to ease commenting.

• Avoid global variables

http://www.codecogs.com/pages/standards/programming.htm
Write Modular Code

Code should be broken down into smaller pieces in order to make testing easier and to facilitate re-use of code. Functions that span several pages of printed text are hard to follow, harder to debug and are less likely to be reusable in alternative applications. As a general rule, a function should not span more than 2 pages (or 100 lines). Furthermore, two or more functions can be adapted by others for a wider range of uses than one single function.

http://www.codecogs.com/pages/standards/programming.htm