Ada Programming/All Chapters

From Wikibooks, the open-content textbooks collection
< Ada Programming

This Wikibook was voted **Book of the Month** for **September 2005**!

Augusta Ada King, Countess of Lovelace.

1 Preface

Welcome to the **Ada Programming** tutorial at Wikibooks. This is the first Ada tutorial covering the imminent Ada 2005 standard. If you are a beginner you will learn the future standard - if you are a seasoned Ada user you can see what's new.

Current Development Stage for **Ada Programming** is "⊞ (Jul 27, 2005)". At this date, there are more than 200 pages in this book, which makes **Ada Programming** the largest of the programming wikibooks (see wikistats (http://en.wikipedia.org/wikistats/wikibooks/EN/Wikibooks_EN.htm) , Category:Ada Programming or /All Chapters).

But still there is always room for improvement — do help us to expand **Ada Programming**. Even beginners will find areas to participate.

This book is currently being worked on by the Wikipublish Wikiproject. It has been judged to be of high enough quality to be published and distributed. All authors of this book are welcome to help with the publishing effort.

1.1 About Ada

Ada is a programming language named after Augusta Ada King, Countess of Lovelace, which is suitable for all development needs.

Ada has built-in features that directly support structured, object-oriented, generic, distributed and concurrent programming.

Ada is a good choice for Rapid Application Development, Extreme Programming (XP), and Free Software development.

1.1.1 Programming in the large

Ada puts unique emphasis on, and provides strong support for, good software engineering practices that scale well to very large software systems (millions of lines of code, and very larg development teams). The following language features are particularly relevant in this respect:

- An extremely strong, static and safe **type system**, which allows the programmer to construct powerful abstractions that reflect the real world, and allows the compiler to detect many logic errors before they become bugs.
- **Modularity**, whereby the compiler directly manages the construction of very large software systems from sources.
- **Information hiding**; the language separates interfaces from implementation, and provides fine-grained control over visibility.
- **Readability**, which helps programmers review and verify code. Ada favours the reader of the program over the writer, because a program is written once but read many times. For example, the syntax bans all ambiguous constructs, so there are no surprises, in accordance with the Tao of Programming's Law of Least Astonishment (http://www.brandwand.com/tao/tao_4.html) . (Some Ada programmers are reluctant to tal about source code which is often cryptic; they prefer program text which is close to Englis prose.)
- **Portability**: the language definition allows compilers to differ only in a few controlled ways, and otherwise defines the semantics of programs very precisely; as a result, Ada source text is very portable across compilers and across target hardware platforms. Most often, the program can be recompiled without any changes (see this paper by Eurocontrol (PDF, 160 kB) (http://portal.acm.org/affiliated/ft_gateway.cfm?id=958426&type=pdf&coll=portal&dl=ac).
- **Standardisation**: standards have been a goal and a prominent feature ever since the design of the language in the late 1970's. The first standard was published in 1980, just 3 years after design commenced; and a new revision of the standard is being finalised in 2006. Ada compilers all support the exact same language; there are no dialects.

Consequences of these qualities are superior **reliability**, **reusability** and **maintainability**. Fo example, compared to programs written in C, programs written in Ada 83 contain ten times fewer bugs, and cost half as much to develop in the first place (see this study by Stephen F. Zeigler (http://www.adaic.com/whyada/ada-vs-c/cada_art.html)). Ada shines even more in software maintenance, which often accounts for 80% of the total cost of development. With support for object-oriented programming, Ada95 brings even more cost benefits, although no serious study comparable to Zeigler's has been published.

1.1.2 Programming in the small

In addition to its support for good software engineering practices, which are applicable to general-purpose programming, Ada has powerful specialised features supporting low-level programming for real-time, safety-critical and embedded systems. Such features include, among others, machine code insertions, address arithmetic, low-level access to memory, contro over bitwise representation of data, bit manipulations, and a well-defined, statically provable concurrent computing model called the Ravenscar Profile.

Other features include restrictions (it is possible to restrict which language features are accepted in a program) and features that help review and certify the object code generated by the compiler.

Several vendors provide Ada compilers accompanied by minimal run-time kernels suitable for use in certified, life-critical applications. It is also possible to write Ada programs which requir

no run-time kernel at all.

It should come as no surprise that Ada is heavily used in the aerospace, defense, medical, railroad, and nuclear industries.

1.1.3 The Language Reference Manual

The Ada Reference Manual (RM) —full name Ada Reference Manual, ISO/IEC 8652:1995(E)— i the official language definition. If you have a question no one can answer for you, you will find something in the RM (albeit often a bit cryptic for non-language-lawyers). For this reason, all complete (not draft) pages in **Ada Programming** contain links into the appropriate pages in the RM.

- You can browse the complete RM for Ada 95 in one of the following sites:
  - http://www.adaic.org/standards/95lrm/html/RM-TTL.html
  - http://www.adapower.com/rm95.php
- There are two documents related with the RM:
  - The Annotated Reference Manual (http://www.adaic.org/standards/95aarm/html/AA-TTL.html) , an extended version of the RM aimed at compiler writers or other persons who want to know the fine details of the language.
  - The Reference Manual Rationale (http://www.adaic.org/standards/95rat/RAThtml/rat95-contents.html) , an explanation of the features of the language.
- The draft for the upcoming Ada 2005 can be found here. Beware: it's not finished yet.
  - http://www.adaic.org/standards/ada05.html

1.1.4 Ada Conformity Assessment Test Suite

Unlike other programming languages, Ada compilers are officially tested, and only those which pass this test are accepted, for military and commercial work. This means that all Ada compiler behave (almost) the same, so you do not have to learn any dialects. But because the Ada standard allows the compiler writers to include some additions, you could learn a cool new feature only to find out that your favorite compiler does not support it....

1.2 Programming in Ada

1.2.1 Getting Started

Where to get a compiler, how to compile the source, all answered here:

- Basic Ada — ⊞ (Sep 22, 2005), **Read Me First!**
- Finding and Installing Ada — ⊞ (Sep 22, 2005)
- Building an Ada program — ⊞ (Sep 22, 2005)

1.2.2 Language Features

These chapters look at the broader picture. They are more tutorial like and show how the keywords, operators and so forth work together.

- Control Structures — ⊞ (Dec 14, 2005)
- Type system — ⊞ (Jan 5, 2006)
- Strings — ⊞ (Jan 5, 2006)