ADA USER JOURNAL

Volume 46
Number 3
September 2025

Contents

Page

Editorial Policy for Ada User Journal 138

Editorial 139

Quarterly News Digest 140

Conference Calendar 163

Forthcoming Events 170

Articles from the AEiC 2025 Work-in-Progress Session

D. Pedro, J. Cecílio, P. M. Ferreira, A. Oliveira de Sá
"*Automated Execution of Attack Chains to Train Data-Driven IDS in Cyber-physical Systems*" 172

M. Samadi, T. Carvalho, L. M. Pinho, S. Royuela
"*Task-to-Accelerator Mapping for Heterogeneous Systems Using Heuristics*" 176

Articles from the AEiC 2025 Industrial Track

C. Dross, J. Huguet, J. Kanig
"*Reasoning about Subprogram Termination in SPARK*" 180

Articles from the Ada Developer Room at FOSDEM 2025

D. Craeynest, F. Oleo Blanco "*12th Ada Developer Room at FOSDEM 2025*" 185

F. Oleo Blanco "*Updates on the Ada Ecosystem from the Past Few Years*" 186

A. J. Ianozi "*Get Started with Ada in 2 Minutes or Less*" 188

G. de Montmollin "*Advent of Compression: Writing a Working BZip2 Encoder in Ada from Scratch in a Few Days*" 190

G. de Montmollin "*Ada and Mini-Ada: a Solution to the Two-Language Problem*" 191

F. Oleo Blanco "*Understanding Liquid Types, Contracts and Formal Verification with Ada/SPARK*" 192

O. Scherer "*The State of Rust Trying to Catch Up with Ada*" 195

C. Sagaert "*Big Integers for Cryptography in SPARK*" 196

J. Verschelde "*Multiword Arithmetic and Parallel Computing*" 198

C. Simon "*Developing Device Drivers for Ironclad Using Ada*" 200

T. McGlinn "*Kids Learn to Code with AdaBots*" 202

Ada-Europe Associate Members (Ada Organizations) 204

Ada-Europe Sponsors Inside Back Cover

Quarterly News Digest

Alejandro R. Mosteo

Centro Universitario de la Defensa de Zaragoza, 50090, Zaragoza, Spain; Instituto de Investigación en Ingeniería de Aragón, Mariano Esquillor s/n, 50018, Zaragoza, Spain; email: amosteo@unizar.es

Contents

Preface by the News Editor	140
Ada-related Events	140
Ada-related Resources	142
Ada-related Tools	144
Ada and Operating Systems	149
Ada Inside	149
Ada Practice	151
Ada Frontiers	161
SPARK/Ada	162

[Messages without subject/newsgroups are replies from the same thread. Messages may have been edited for minor proofreading fixes. Quotations are trimmed where deemed too broad. Sender's signatures are omitted as a general rule. —arm]

Preface by the News Editor

Dear Reader,

Given the lively discussion in the ada-lang forum, it is becoming difficult to pick just a couple of threads to highlight. I will start with a thread [1] about the benefits of using Ada in the large. It is a bit of preaching to the choir, but interesting nonetheless.

Then I would like to point out some efforts to expand the reach of Ada: a new full toolchain for FreeBSD [2], and two projects related to embedded development: a bare-board runtime generator [3] and configurable runtimes for STM32 [4].

A rather unique announcement has also recently been made by the Indian government: its first 32-bit chip for space applications is being produced, with high-level support for Ada [5]. I could not find exactly what this support entails, so let us keep on the lookout!

Finally, do not miss the various announcements for the Crate of the Year Award, monthly meet-ups, FOSDEM 2026 and AEiC 2026 in the Ada-related Events section.

Sincerely,
Alejandro R. Mosteo

[1] "Experiences with Large Projects", in Ada Practice

[2] "Full GNAT Ada 2022 Toolchain for FreeBSD", in Ada-related Tools

[3] "Bare Board Runtime Generator 0.0.1", in Ada-related Tools

[4] "New Configurable STM32 Bareboard Runtimes", in Ada-related Tools

[5] "Vikram-3201: India's Custom 32-bit Processor Built for Ada", in Ada Inside

Ada-related Events

Info about Ada Monthly Meetup, Ada Developers Workshop 2025 and Some More

From: Irvise
Subject: Info about Ada Monthly Meetup, Ada Developers Workshop 2025 and Some More
Date: Wed, 2 Jul 2025 11:11:32 +0000
Newsgroups: forum.ada-lang.io

I hope you are doing well and that you are getting ready for holidays. I would like to give you all a small update regarding a few things I am involved in and that I know some people have some expectations.

- Ada Monthly Meetup will not be taking place in July nor August. This is due to these months being holidays for many people (including me) and that the Developers Workshop took place a couple of weeks ago. Also, I am quite busy.
- The publishing of the videos for the Ada Developers Workshop are going to take much longer than expected. This is due to my life being extremely busy and things just keep popping up… Sorry for that, but this is just how things are.
- I would also like to apologise again due to the technical issues that took place during the Workshop… Some recordings have had parts of them lost and the people online could not see the webcam feed for most of the day… I should have tested the setup more thoroughly… Either way, it seems there was a bug with OBS and the drivers, so I could not have fixed it, but still…
- The weekend after the Workshop, 20-22 of June, I was in OpenSouthCode [1]. It was my first time there. I gave a 2h long Introduction to Ada in Spanish. Sadly, due to some conditions, not many people attended… Nevertheless, the presentation was recorded. The slides and recording will be made available in the talk's website entry [2]. Sadly, the webcam recording was extremely blurry as I forgot to disable autofocus and the camera did a terrible job during most of the presentation…
- The conference itself was extremely good! I met a ton of people and wonderful communities. I knew there were a lot of cool open source developers in Spain, but I was very impressed by the number and quality of them! It was also great to see the amount of foreigners that attended the conference (50% of the talks were in English). The organisation of the conference was also outstanding and they really focused on networking and people getting together. It was truly lovely and I hope to go back next year.
- I am looking into ways that some things in the Ada community could get some funding… I would like to see the *BSD patches for GNAT upstreamed and improved upon… I would also like to see someone take over supporting GNAT on Apple systems, Zephyr support for Ada (see @jgrivera67 topic in the Workshop), Ada support for RTEMS new build system, GTK 4 support, etc.

Well, that was a bit of talking… Anyhow, I wish you all a great summer!

[1] https://www.opensouthcode.org/conferences/opensouthcode2025

[2] https://www.opensouthcode.org/conferences/opensouthcode2025/program/proposals/851

2025 Crate of the Year Awards

From: Fabien.C
Subject: 2025 Crate of the Year Awards!
Date: Thu, 10 Jul 2025 08:16:09 +0000
Newsgroups: forum.ada-lang.io

Hello here, AdaCore just announced the 4th edition of the Crate of the Year Awards [1]. No need to register, winning crates will be selected from all the crates in the Alire ecosystem.

I am opening this thread as a place for everyone to discuss which crate they think would be a good candidate or even promote their crate.

Have fun and happy hacking!

[1] https://blog.adacore.com/announcing-the-2025-ada-spark-crate-of-the-year-award

Ada Monthly Meetup, 13th of September

From: Irvise
Subject: Ada Monthly Meetup, 13th of September
Date: Wed, 20 Aug 2025 14:17:36 +0000
Newsgroups: forum.ada-lang.io

I would like to announce the September (2025) Ada Monthly Meetup which will be taking place on the 13th of September at 14:00 UTC time (16:00 CEST). As always, the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to YouTube.

Important: due to some personal scheduling issues, this meetup will be taking place the second week of the month, and one hour later than usual. This should be a one-off, but still, be aware.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no proposals. Nevertheless, we will talk about the AEiC conference, a bit of preparation for FOSDEM and then some news, regarding Alire, increase of popularity of Ada in some programming language indexes and other topics that the community may want.

Here are the connection details from previous posts:

The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is Jitsi Meet [1] The room name is “AdaMonthlyMeetup” and in case it asks for a password, it will be set to “AdaRules”.

I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

Best regards and see you soon!
Fer

P.S: I still have not started video editing the talks of AEiC… I have been very busy… Let's see if I can get some time and energies in the following weeks.

P.S: feel free to repost this in other forums or chats, such as Reddit!

See related post of the same name for more info

[1] https://meet.jit.si/AdaMonthlyMeetup

From: Irvise
Date: Sat, 13 Sep 2025 16:40:54 +0000

The meeting just finished and here are the Minutes of the Meeting:

- @Irvise gave his perspective and feelings about the Ada Developers Workshop which took place during this year’s Ada-Europe international Conference. He has received pretty good feedback and people seemed to be quite happy with it. The recordings of the videos still need to be processed, he apologises for the substantial delay.
- @DirkCraeynest then shared with us the slides that were used during AEiC to present the 2026 edition of the conference! It will be taking place 9-12 June 2026, Västerås, Sweden [1]. The call for papers/presentations in the different tracks is already open. We will try to repeat again the Ada Developers Workshop, as it had quite a lot of attendance.
- @Irvise told us about his experience giving an introductory workshop about Ada. It was given in OpenSouthCode [2]. There was not much attendance but there was good reception. It was an incredible experience and a great conference overall.
- Also, from that meetup, he learnt about SysML v2 [3]. It may be very interesting to see an Ada code generator for it in the future… who knows!
- FOSDEM [4] 2026 has started preparations and we hope to see the official dates and the Call for Devrooms quite soon. Of course, we plan on applying again this year.
- India has developed their own CPU and Ada will be a first class citizen in their tech stack! [5]
- @Irvise pointed out that India, South America and China barely have any representation in the wider Ada community and that they are growing markets that could substantially benefit from it. We may need to have a push or showcase that targets those communities

NVIDIA:

- They published their work on ISO-26262 certification [6] and published a guide on how they did it!
- They certified to ASIL-D their DriveOS, a 7 MLOC OS written in SPARK [7]
- Their White Hat Hackers showcased in Defcon their work on securing over a billion RISC-V chips with the use of Ada! [8] (related article [9]). Slides are here [10] and the live-recording [11]. I sadly cannot find the specific video of their talk, but I know it exists. Please, feel free to link it here.

- Ada keeps on climbing in popularity rankings! [12] We need to keep on sharing the word, teaching and showcasing our passion for it!
- REMINDER: The Ada Crate of the Year Award [13] is still ongoing. It is until the end of the year, but please, don’t leave your projects till the very end!
- @godunko did a PR to get GNAT-LLVM ready for Alire [14] quite a while ago! That is great news, let's see if we can get it accepted!
- Reminder that @JeremyGrosser @kevlar700 and @Irvise are now moderators, so now it is not only Jeremy dealing with the spam. If you have any questions, issues, etc; feel free to contact us or create a forum thread with the category of “site feedback”.
- There are several job postings by Raytheon [15]
- The Ada User Society [16] has improved its website and it will hold the General Assembly on the 7th of October. If you are a member, do not forget to attend!
- @Irvise has started discussions with them about creating a fund to provide prizes, money or incentives for people to support and work on relevant Ada programs and improve the wider Ada experience. Expect more news in the future, but for the time being, this is all I can share.
- Luke shared a method for funding which was used for the Amiga [17]
- @mosteo recently became a father and his work on Alire has been paused for the time being. So no need to panic over fewer commits being done there! And let this serve as a reminder that anybody can contribute to Alire!

EDIT: I forgot to link the video recording/livestream https://www.youtube.com/watch?v=qnir_hk1ZC4

Thank you all that attended!

Best regards,
Fer

[1] https://www.ada-europe.org/conference2026/

[2] https://www.opensouthcode.org/conferences/opensouthcode2025/program/proposals/851

[3] https://www.omg.org/sysml/sysmlv2/

[4] https://fosdem.org/2025/

[5] https://forum.ada-lang.io/t/vikram-3201-india-s-custom-32-bit-processor-built-for-ada/3793

[6] https://forum.ada-lang.io/t/nvidia-publishes-spark-process-to-meet-iso-26262-requirements/2114

[7] https://forum.ada-lang.io/t/article-nvidia-drives-ada-and-spark-into-driverless-cars/2117

[8] https://defcon.org/html/defcon-33/dc-33-speakers.html#content_60337

[9] https://riscv.org/blog/2025/02/how-nvidia-shipped-one-billion-risc-v-cores-in-2024/

[10] https://media.defcon.org/DEF CON 33/DEF CON 33 presentations/Adam Zabrocki Marko Mitic – How to

secure unique ecosystem shipping 1 billion%2B cores.pdf

[11] https://www.youtube.com/live/MEPF0dEAjPg

[12] https://forum.ada-lang.io/t/ada-on-the-pypl-index/425/11

[13] https://forum.ada-lang.io/t/2025-crate-of-the-year-awards/2203

[14] https://github.com/alire-project/GNAT-FSF-builds/pull/83

[15] https://forum.ada-lang.io/t/several-ada-jobs-at-raytheon/3798

[16] https://www.ada-user.org/

[17] https://will-templates.com/amigabounty/

From: DirkCraeynest
Date: Tue, 16 Sep 2025 19:07:05 +0000

> the slides that were used during AEiC to present the 2026 edition of the conference

They were also posted on the AEiC 2026 page in Ada-Belgium's calendar of Conferences and events for the international Ada community [1]. See the invitation to Ada-Europe 2026 at MDU [2] that was presented during the AEiC 2025 closing session.

[1] https://people.cs.kuleuven.be/~dirk.craeynest/ada-belgium/events/list.html

[2] https://people.cs.kuleuven.be/~dirk.craeynest/ada-belgium/events/26/260609-aeic-mdu.pdf

FOSDEM 2026, Call for Participation

From: Irvise
Subject: FOSDEM 2026, Call for Participation
Date: Sun, 21 Sep 2025 20:29:19 +0000
Newsgroups: forum.ada-lang.io

The CfP has been issued! [1]

I am creating a thread quickly to track our submission and submission message

IMPORTANT EDIT: if you would like to participate as a co-organiser, please, please, please, let us know ASAP!!!

[1] https://fosdem.org/2026/news/2025-09-21-call-for-participation/

CfC 30th Ada-Europe Int. Conf. Reliable Software Technologies

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: CfC 30th Ada-Europe Int. Conf. Reliable Software Technologies
Date: Wed, 24 Sep 2025 15:51:11 +0000
Newsgroups: comp.lang.ada

[CfC is included in the Forthcoming Events Section. —arm]

Ada-related Resources

[Delta counts are interpolated from July 1st to October 1st. Tabulated raw data is available at https://bit.ly/auj-signals —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 13 Oct 2025 23:36 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 2_680* members [1]
- LinkedIn: 3_704 (+46) members [2]
- Stack Overflow: 2_443 (=) questions [3]
- Ada-lang.io: 404 (+20) users [4]
- Gitter: 293 (+4) people [5]
- Telegram: 226 (-1) users [6]
- Discord: 197 (+5) users [7]
- Libera.Chat: 75 (=) concurrent users [8]

[*] Reddit has changed how it measures engagement. Rather than showing all subscribers since the Epoch, it now shows a running average of active users during the last month. Delta count is therefore not representative of any real changes in membership.

[1] https://old.reddit.com/r/ada/

[2] https://www.linkedin.com/groups/114211/

[3] https://stackoverflow.com/questions/tagged/ada

[4] https://forum.ada-lang.io/u

[5] https://app.gitter.im/#/room/#ada-lang_Lobby:gitter.im

[6] https://t.me/ada_lang

[7] https://discord.gg/fEmGe87c

[8] https://netsplit.de/channels/details.php?room=%23ada&net=Libera.Chat

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 13 Oct 2025 23:36 CET
To: Ada User Journal readership

GitHub: >7_800* (=) repositories [1]
>1_000* (=) developers [1]
Alire: 1_432 (+6) releases [2]
585 (+26) crates [3]
Rosetta Code: 1_051 (+25) examples [4]
43 (=) developers [5]
SourceForge: 241 (-1) projects [6]
Open Hub: 214 (=) projects [7]
Codelabs: 64 (+1) repositories [8]
Bitbucket: 36* (-1) repositories [9]

*This number is a lower bound due to site search limitations.

[1] https://github.com/search?q=language%3AAda&type=Users

[2] https://alire.ada.dev/crates.html

[3] `alr search --list --full`

[4] https://rosettacode.org/wiki/Category:Ada

[5] https://rosettacode.org/wiki/Category:Ada_User

[6] https://sourceforge.net/directory/language:ada/

[7] https://www.openhub.net/tags?names=ada

[8] https://git.codelabs.ch/?a=project_index

[9] https://bitbucket.org/repo/all?name=ada&language=ada

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date: 13 Oct 2025 23:36 CET
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. —arm]

- PYPL Index: 11 (+4) 2.17% (+0.65%) [2]
- TIOBE Index: 18 (-6) 1.06% (-0.59%) [1]
- Stack Overflow Survey: 40 (=) 0.9% (=) [3]
- IEEE Spectrum (trending): 43 (+3) Score: 0.0038 (+0.0015) [4]
- IEEE Spectrum (general): 46 (+4) Score: 0.0027 (+0.0012) [4]
- IEEE Spectrum (jobs): 40 (+14) Score: 0.0017 (+0.0016) [4]
- Languish Trends: 165 (-11) 0.01% (=) [5]

[1] https://www.tiobe.com/tiobe-index/

[2] http://pypl.github.io/PYPL.html

[3] https://survey.stackoverflow.co/2024/

[4] https://spectrum.ieee.org/top-programming-languages/

[5] https://tjpalmer.github.io/languish/

Ada on the PYPL Index

From: zertovitch
Subject: Ada on the Pypl Index
Date: Tue, 1 Jul 2025 23:04:51 +0000
Newsgroups: forum.ada-lang.io

1.65%

Through the roof

From: pyj
Date: Wed, 2 Jul 2025 01:09:14 +0000

Needs more up arrows

From: gast
Date: Thu, 3 Jul 2025 21:08:26 +0000

What is the main reason for Ada's growing popularity this year?

From: charlie5
Date: Fri, 4 Jul 2025 00:54:23 +0000

Reached critical mass? …

From: F-Loyer
Date: Sun, 6 Jul 2025 11:53:25 +0000

Programmers usually don't choose (only) a language, but an ecosystem. If I want to program the computation of a filter response and the drawing a bode plot, I guess I would choose Python because of the matplotlib (even if Ada would be nice with its included complex computation). It is also a strength of Java: many things available out of the box (GUI binding, Web, etc.).

I guess Alire will have an important value for many programmers who are considering Ada to develop something. Unfortunately, not every library can be easily installed with Alire.

From: zertovitch
Date: Thu, 4 Sep 2025 15:17:27 +0000

September 2025: 2.11%

From: krischik
Date: Thu, 4 Sep 2025 16:31:45 +0000

Yes, Ada at place 12

Ada almost caught up with Rust

And at TIOBE it's similar, place 13

From: zertovitch
Date: Fri, 5 Sep 2025 00:26:42 +0000

Note, regarding TIOBE, that the keyword "nvidia" is excluded from the Web searches. It is explained in "This Month's Changes in the Index" (August 2025):

> W. H. told us that programming language Ada might have been boosted in the TIOBE index because of the Ada Lovelace architecture of NVIDIA. For this reason we have added "-NVIDIA" to Ada's search term. As a consequence, Ada dropped from position #9 last month to position #13 this month.

This is unfortunate because the exclusion discards lots of articles about NVIDIA's own usage of Ada.

From: DirkCraeynest
Date: Fri, 5 Sep 2025 13:28:13 +0000

Hence, an additional correction of Ada's search term is required. +"NVIDIA Ada programming" maybe?

From: zertovitch
Date: Fri, 5 Sep 2025 18:19:12 +0000

> +"NVIDIA Ada programming" maybe?

With that query, only that exact sequence (theoretically) would be searched, and Ada would land at place +/- #513.

But I have searched a few weeks ago with Google, with a 1-year filter (as the TIOBE guy does) +"ada programming" +"NVIDIA" to see the impact of doing +"ada programming" -"NVIDIA", as the TIOBE guy does currently.

37 hits were shown.

33 hits mentioned the Ada language (so, should be counted).

1 hit mentioned the "Ada Lovelace" GPU (so, should not be counted).

3 were about other topics or some chat in Indonesian (this noise is everywhere, is counted for all programming languages, so there is a mutual neutralization).

So I have suggested to the author just to remove the -"NVIDIA" filter.

From: waleedmebane
Date: Fri, 5 Sep 2025 20:31:19 +0000

How about remove "-NVIDIA" and add '"-Lovelace"?

From: kevlar700
Date: Sat, 6 Sep 2025 00:33:24 +0000

This might be new too and Ada Lovelace is championed in some pro Women establishments of late.

Ada Computer Science

https://adacomputerscience.org/

From: pmnw
Date: Wed, 24 Sep 2025 14:03:17 +0000

Those interested in rankings may find the following interesting.

IEEE Spectrum – 23 Sep 25 [1]

AI Is Redefining the Concept of a Programming Language's Popularity [1]

Python reigns supreme again, but is AI changing the game for programming languages? Find out how coding is transforming.

[1] https://spectrum.ieee.org/top-programming-languages-2025

From: cantanima
Date: Wed, 24 Sep 2025 23:30:32 +0000

> Those interested in rankings may find the following interesting.

TL;DR:

- the good news is that, before long, we won't be worried / squabbling about programming language popularity
- the bad news is that, before long, we'll be worried / squabbling about Large Language Model popularity
- the worse news is that, not much longer after that, people will look at the world of software developers / engineers / etc. much the same way they look at the world of 8-bit cpu hobbyists: "how charmingly quaint!"

NB:

- i said that was the TL;DR, /not/ that it was my personal belief
- i'm being tongue in cheek (hence the)…
- …mostly

Ada Back in the TIOBE Top 20

From: F-Loyer
Subject: Ada Back in the TIOBE Top 20 (march 2025)
Date: Tue, 15 Jul 2025 22:26:56 +0000
Newsgroups: forum.ada-lang.io

In the top 10… (July 2025):

- Python (26,98 %);
- C++ (9,8 %) ;
- C (9,65 %) ;
- Java (8,76 %) ;
- C# (4,87 %) ;
- JavaScript (3,36 %) ;
- Go (Golang) (2, 04%);
- Visual Basic (1,94%);
- Ada (1,77 %);
- Delphi/Object Pascal (1,77%).

Curious Rust is not there.

From: mgrojo
Date: Thu, 31 Jul 2025 18:30:26 +0000

Developer Tech News – 14 Jul 25

Why is 40-year-old programming language Ada hot again? [1]

Ada, a programming language born in the late 70s, has managed to break into the top 10 of the TIOBE Index for July 2025.

Estimated reading time: 4 minutes

[1] https://www.developer-tech.com/news/why-is-40-year-old-programming-language-ada-hot-again/

Old but Cool: Ada Mapping to CORBA

From: Irvise
Subject: Old but Cool: Ada Mapping to CORBA
Date: Sun, 28 Sep 2025 12:39:29 +0000
Newsgroups: forum.ada-lang.io

I recently found that the OMG (Object Management Group) had available the "standard" on how to map Ada to CORBA [1]. It is freely available here for anybody that may be interested About the Ada Language Mapping Specification Version 1.3 [2]

[1] https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture

[2] https://www.omg.org/spec/ADA

From: charlie5
Date: Mon, 29 Sep 2025 04:24:38 +0000

Hi Fer … Not sure if it's relevant but CORBA is one of the 'personalities' available in PolyORB.

3. Overview of PolyORB personalities — PolyORB 26 User's Guide

https://docs.adacore.com/live/wave/polyorb/html/polyorb_ug/Overview_of_PolyORB_personalities.html#corba

From: cup
Date: Tue, 30 Sep 2025 06:27:27 +0000

If you're translating any Java/C++ CORBA code, the common coding convention was that references to CORBA objects were suffixed _ptr. Just remember that these are CORBA references and have to be deleted the CORBA way: not using the language's native delete methods. This was a very common problem for newbies - caused no end of crashes.

Writing CORBA code in Java was more elegant than C++. Hopefully writing CORBA in Ada isn't as messy as C++.

Ada-related Tools

Bare Board Runtime Generator 0.0.1

From: godunko
Subject: Announce: Bare Board Runtime Generator Version 0.0.1
Date: Sat, 5 Jul 2025 07:39:14 +0000
Newsgroups: forum.ada-lang.io

While Alire provides ready for use runtimes for some boards, it is not trivial to run an application on other boards, and even harder to use Ada tasking. I've developed a runtime generator to create custom runtimes for bare board applications. Runtime can be fine-tuned for a particular application, it is generated near to application's code and does not need to be distributed.

https://github.com/godunko/a0b-tools

I've been able to run a blink led example on STM32F401/411 and STM32G431/474 boards, and on Arduino Due.

From: kevlar700
Date: Sun, 24 Aug 2025 10:37:57 +0000

I would never normally recommend using C however for clock setup and PWR setup then they are probably very well tested by ST so using their SDK code might not be so bad? (caveat: I have shaken my head at their code before, for example their clock setup code uses a global variable that makes it fragile in relation to procedure execution order). It would mean that porting the Ada runtime might be a drop in. I guess it depends if finding the right C #includes are less or more of an issue. Certainly gprbuild makes it easy to do it once you do. The trickier issue is low power parts and making the runtime be compatible with user executed clock changes which is an exercise I shall have to hope to have the time for one day.

From: godunko
Date: Mon, 25 Aug 2025 07:03:59 +0000

Time and tasking support in GNAT RTL should be improved to support flexible CPU frequencies. Right now, CPU frequency is hardcoded.

Do you have experience with changing power modes? For STM changing of CPU clocks configuration requires change of clock prescalers for peripherals. How is it handled?

From: kevlar700
Date: Mon, 25 Aug 2025 10:03:20 +0000

I have only done it with the low power parts L and U which are designed for it but yes some drivers like timers have to get the current speeds and if I recall correctly there are differences in the code generated by STM32Cube IDE. So my Ada code to get the clock speeds had to be adapted between L and U. To save time I currently have bindings to SystemClock_Config, HAL_RCC_GetPCLK1Freq, HAL_RCC_GetSysClockFreq and HAL_RCC_GetHCLKFreq pulling in around 40 .c and .h files. I have a script to compare them to upstream which is not ideal but I guess it's easier to copy in those 40 files from any new chip than compare for changes in the generated code and you can just copy the clock init code (SystemClock_Config) after any changes made in the STM32Cube IDE. The others are in the HAL package with just an example of binding SystemClock_Config as that needs to be added to main once generated based on STs IDEs clock configuration tooling.

From: Lucretia
Date: Sun, 24 Aug 2025 11:58:16 +0000

Can it be used for osdev (not embedded)?

From: godunko
Date: Mon, 25 Aug 2025 07:01:51 +0000

No, sources of the native RTL are not available in bb-runtimes

Why might it be useful?

New Configurable STM32 Bareboard Runtimes

From: damaki
Subject: New Configurable STM32 Bareboard Runtimes
Date: Tue, 8 Jul 2025 10:45:35 +0000
Newsgroups: forum.ada-lang.io

I've recently released some configurable bareboard runtime crates for several STM32 families, with support for the light, light-tasking, and embedded runtime profiles:

STM32F0xx crates [1]:

• light_stm32f0xx [2]

• light_tasking_stm32f0xx [3]

• embedded_stm32f0xx [4]

STM32G0xx crates [5]:

• light_stm32g0xx [6]

• light_tasking_stm32g0xx [7]

• embedded_stm32g0xx [8]

STM32G4xx crates [9]:

• light_stm32g4xx [10]

• light_tasking_stm32g4xx [11]

• embedded_stm32g4xx [12]

The runtimes are based on GNAT FSF 15, but I could do releases for older versions of GNAT if there's a demand for it.

Each runtime supports every device in the family. The runtime is configured for a specific device via crate configuration variables. For example, to configure the light_tasking_stm32g4xx runtime for the STM32G431K6 device, add the following to your project's alire.toml:

```
[configuration.values]
light_tasking_stm32g4xx.
MCU_Sub_Family = "G431"
light_tasking_stm32g4xx.
MCU_Flash_Memory_Size = "6"
```

The clock tree is also configurable. By default the runtime sets up the system clock from the PLL and the high-speed internal (HSI) oscillator, but you can also configure it for the HSE if your board has one (some Nucleo boards seem to have one) or configure a different clock speed if you want. Here's an example of configuring the light_tasking_stm32g4xx runtime to generate a 170 MHz system clock from a 24 MHz HSE oscillator in alire.toml:

```
[configuration.values]

# Configure a 24 MHz HSE crystal oscillator
light_tasking_stm32g4xx.
HSE_Clock_Frequency = 24000000
light_tasking_stm32g4xx.
HSE_Bypass = false

# Select PLLRCLK as the SYSCLK source
light_tasking_stm32g4xx.
SYSCLK_Src = "PLLRCLK"

# Configure the PLL VCO to run at 340 MHz
from the 24 MHz HSE (fVCO = fHSE * (N/M))
light_tasking_stm32g4xx.
PLL_Src = "HSE"
light_tasking_stm32g4xx.
PLL_N_Mul = 85
light_tasking_stm32g4xx.
PLL_M_Div = 6

# Configure the PLLRCLK to run at 170 MHz
from the 340 MHz VCO.
light_tasking_stm32g4xx.PLL_R_Div = 2

# Configure the AHB and APB to also run at
170 MHz
```

```
light_tasking_stm32g4xx.AHB_Pre = "DIV1"
light_tasking_stm32g4xx.APB1_Pre = "DIV1"
light_tasking_stm32g4xx.APB2_Pre = "DIV1"
```

See the projects' GitHub pages (links below) for full details of what is configurable.

The hope is that these runtimes are flexible and easy-to-use enough that they help people get started with a new STM32 project quickly, without needing to build their own runtimes for their specific device/board since they can use and configure these runtimes instead.

[1] https://github.com/damaki/stm32f0xx-runtimes/

[2] https://alire.ada.dev/crates/light_stm32f0xx

[3] https://alire.ada.dev/crates/light_tasking_stm32f0xx

[4] https://alire.ada.dev/crates/embedded_stm32f0xx

[5] https://github.com/damaki/stm32g0xx-runtimes/

[6] https://alire.ada.dev/crates/light_stm32g0xx

[7] https://alire.ada.dev/crates/light_tasking_stm32g0xx

[8] https://alire.ada.dev/crates/embedded_stm32g0xx

[9] https://github.com/damaki/stm32g4xx-runtimes/

[10] https://alire.ada.dev/crates/light_stm32g4xx

[11] https://alire.ada.dev/crates/light_tasking_stm32g4xx

[12] https://alire.ada.dev/crates/embedded_stm32g4xx

Zip-Ada 61

From: zertovitch
Subject: Zip-ada Version 61
Date: Tue, 8 Jul 2025 16:44:39 +0000
Newsgroups: forum.ada-lang.io

Here are blog posts diving into what is "under the hood" in the new BZip2 encoder.

Writing a BZip2 encoder in Ada from scratch in a few days - part 1 [1]

Writing a BZip2 encoder in Ada from scratch in a few days - part 2 [2]

Spoiler: it is also likely already the best BZip2 encoder for a given block size.

[1] https://gautiersblog.blogspot.com/2024/11/writing-bzip2-encoder-in-ada-from.html

[2] https://gautiersblog.blogspot.com/2025/07/writing-bzip2-encoder-in-ada-from.html

Strings Edit 3.9

From: dmitry-kazakov
Subject: Strings Edit 3.9
Date: Sat, 12 Jul 2025 12:56:19 +0000
Newsgroups: forum.ada-lang.io

String processing for Ada [1]

Changes to the previous version:

- Bug fix in Strings_Edit.UTF8.Maps.Is_Prefix;
- Unread was added to Strings_Edit.Streams to return the contents available to read;
- Get function was added to.Strings_Edit.UTF8.Maps.

[1] https://www.dmitry-kazakov.de/ada/strings_edit.htm

Simple Components 4.75

From: dmitry-kazakov
Subject: Simple Components 4.75
Date: Sat, 12 Jul 2025 13:04:41 +0000
Newsgroups: forum.ada-lang.io

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), arbitrary precision arithmetic, pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, persistent storage, multiple connections server/client designing tools and protocols implementations.

Simple components for Ada [1]

The new version provides an implementation of SNOBOL-like patterns. The patterns are integrated into the parsing framework. They can be constructed using expressions and then matched against the generic source.

SNOBOL patterns are more powerful than regular expressions. A BNF grammar can be directly translated into pattern. Therefore they can be recursive. Features like immediate assignment and printout are fully supported. Patterns can be extended by user-defined matching functions. Unicode is fully supported. In particular, matching letters involve Unicode categorization.

Changes to the previous version:

- The package Parsers.Generic_Source.Patterns was added to implement SNOBOL-like patterns;
- The package Parsers.Generic_Source.Patterns.Generic_User_Pattern was added to provide user-defined patterns;
- The package Parsers.Generic_Source.Patterns.Generic_Parametrized_User_Pattern was added to provide user-defined patterns with parameter;
- The function Top added to the package Stack_Storage;
- The interface procedure Set_Pointer was modified in the generic package Parsers.Generic_Source.

[1] https://www.dmitry-kazakov.de/ada/components.htm

Ada Version of Curl?

From: zertovitch
Subject: Ada Version of Curl?
Date: Wed, 16 Jul 2025 19:18:15 +0000
Newsgroups: forum.ada-lang.io

I have just read this post: Death by a thousand slops [1]

daniel.haxx.se – 14 Jul 25 [1]

I have previously blogged about the relatively new trend of AI slop in vulnerability reports submitted to curl and how it hurts and exhausts us. This trend does not seem to slow down. [...] Even though the topic is more about fake bugs (as I understand it), the kind of vulnerabilities mentioned (mostly buffer overflows) hint at (or even scream for) a nice project idea: an Ada version of curl:

- Name: Aurel
- Primary API: in Ada
- C API for compatibility and replacing curl

The most possible parts would be made with SPARK. It could be a nice advertisement for Ada/SPARK (my impression is that curl is a very "visible" / pivotal software). But perhaps such a project already exists?

[1] https://daniel.haxx.se/blog/2025/07/14/death-by-a-thousand-slops/

From: OneWingedShark
Date: Wed, 16 Jul 2025 21:29:53 +0000

This is misguided: do /everything/ without respect to C /*first*/… then shim in a C compatibility module for export.

From: dmitry-kazakov
Date: Thu, 17 Jul 2025 08:29:15 +0000

> then shim in a C compatibility module for export.

Well, if the Ada API relies on indefinite types (unbounded arrays, tagged types) and/or generics that would be very difficult. You better plan C API in advance, if you want them.

Furthermore, C will be involved anyway in the form of OpenSSL and/or GNUTLS. I doubt the author plans to reimplement them. Which I would welcome, of course.

To the point. There was an Ada Crypto Library [1] project, which unfortunately is no longer supported.

I studied it a lot with regard to primality and arbitrary precision numbers, when I implemented the Simple Components. It is well written, understandable and contains many encryption algorithms which might be useful. I too implemented ChaCha20 and AEAD and have an Ada ASN.1 implementation (you need it for keys and certificates).

Anyway if anybody is brave (or stupid) enough to give an Ada TLS a try, let it be known!

[1] https://github.com/cforler/Ada-Crypto-Library

From: waleedmebane
Date: Thu, 17 Jul 2025 08:38:48 +0000

How about SparkNaCl?

I have never used it, but the README [1] contains this text:

> 22nd June 2022

> Jon Andrew [2] has kindly contributed implementations of the ChaCha20 stream cipher (in SPARKNaCl.Stream), SHA-256 (in SPARKNaCl.Hashing), SHA-256-based HMAC and HKDF algorithms (in SPARKNaCl.MAC and SPARKNaCl.HKDF respectively) and the an AEAD algorithm (RFC 8439) using ChaCha20 and Poly1305 (in SPARKNaCl.SecretBox), plus additional test cases and clean proofs. These additions take the library well beyond the original specification of NaCl, and move towards supporting TLS 1.3.

[1] https://github.com/rod-chapman/SPARKNaCl/

[2] https://github.com/docandrew

From: docandrew
Date: Fri, 18 Jul 2025 21:54:38 +0000

Yes, it's certainly possible and a verified TLS library in SPARK is a long-term goal (maybe dream) of mine.

SPARKNaCl has all the primitives you'd need, it recently added more SHA functions to make the other TLS 1.3 cipher suites possible.

I have a proof-of-concept for the handshakes here: [1]

I started on my own x509 library [2] but it's pretty rough, Dmitry's might be a better starting point. I think RecordFlux's verified record parsers would also be a really great tool to try to apply to the project.

The building blocks are all there! It just needs a hero with more mental horsepower and/or free time than I have

[1] https://github.com/docandrew/SPARKTLS/blob/fd58359697d4d086738f0e1d2bf058027f7819ff/src/tls.adb

[2] https://github.com/docandrew/SPARKx509

Full GNAT Ada 2022 Toolchain for FreeBSD

From: captain-haddock17
Subject: Full Gnat Ada 2022 Toolchain for Freebsd
Date: Wed, 16 Jul 2025 20:54:47 +0000
Newsgroups: forum.ada-lang.io

I'm pleased to announce the availability of the full GNAT Ada 2022 toolchain for FreeBSD.

- GNAT latest Ada commits on 2025-07-04, with GCC 13, 14, 15.1.1 and 16-devel
- GPRBUILD, latest commits on 2025-03-12
- Alire, 2.1.0 from branch

All the binaries are on AdaForge's GitLab [1] in their "Package registry".

[...]

ACATS-4.2.1 [2]

- is on its way (68 Class L programs built)

Side Note:

There is already a first port of gnat13 done by FreeBSD GCC port maintainer Thierry with whom I had a nice chat former friday, but as I had some issues to build it on my rig, and already had a working gnat12 built mid-2022, I took the challenge to set-up a full CI-CD for our Ada toolchain on our FreeBSD server with build system poudriere.

Next step: PR to FreeBSD maintainer to have it direct in the FreeBSD Port & Pkg eco-system, ready to be downloaded.

William J. F.

[1] https://gitlab.com/adaforge

[2] https://gitlab.com/adaforge/devtools/ada-compilers/Ada_AcaaACATS

Added Gentoo Support to Alire

From: Lucretia
Subject: Added Gentoo Support to Alire
Date: Thu, 31 Jul 2025 22:03:19 +0000
Newsgroups: forum.ada-lang.io

I have a branch [1] which adds in support for portage and emerge into Alire. I cannot debug this as it raises an exception reading the settings TOML on a newline, 0x0a. I recreated my settings from scratch with alr and it still happens.

Also, when you ctrl-c the emerge I get an unhandled exception [2] even though all are handled inside the Install subprogram like all the other package managers.

So, if anyone on gentoo wants to try it out, you'll need to modify an external package in the following way [3].

[1] https://github.com/Lucretia/alire/tree/add-gentoo

[2] https://bpa.st/EB6Q

[3] https://github.com/alire-project/alire-index/compare/stable-1.4.0...Lucretia:alire-index:add-gentoo-sdl2-pkgs

ALI_Parse 1.0

From: zertovitch
Subject: Ali_Parse, V1.0
Date: Fri, 1 Aug 2025 08:34:01 +0000
Newsgroups: forum.ada-lang.io

ALI_Parse is a parser for the .ali files generated by the GNAT Ada compiler.

Alire crate:
https://alire.ada.dev/crates/ali_parse

Project site 1:
https://github.com/zertovitch/ali_parse/

Project site 2:
https://sourceforge.net/projects/ali-parse/

ALI means Ada Library Information. You find the .ali files in the same directories as the object files (.o) containing the machine code produced by GNAT.

Currently, ALI_Parse is focused at cross-references within a set of Ada source files. Two command-line tools using the parser are provided:

- GNATHTML, which generates a set of Web pages from Ada sources
- ALI_Stats, which shows the list of entities, the list of cross-references, the count of references to each entity and the list of files.

ALI_Parse is pure Ada 2012 and doesn't depend on any other resource.

From: zertovitch
Date: Sat, 16 Aug 2025 14:43:19 +0000

An example of a Web view generated by ALI_Parse with the Zip-Ada sources:

https://unzip-ada.sourceforge.io/za_html/index.html

From: Blady
Date: Sun, 17 Aug 2025 15:28:46 +0000

Thanks for reviving GNAThtml

Game Made in Ada

From: SabeDoesThings
Subject: Game Made in Ada
Date: Sun, 3 Aug 2025 22:09:23 +0000
Newsgroups: forum.ada-lang.io

I just made a game in Ada for the GMTK [1] game jam! It's probably not the greatest game ever but it was the first thing I actually really worked on aside from little things. I thought this was a great way to test my knowledge from learning the language and I had a great time with it. If you wanna check it out and maybe rate it for the game jam you are more than welcome. Though I've only checked it on Windows so I don't know how or if it will work on other systems so be aware of that.

Loop of Life by SabeDoesThings

https://sabedoesthings.itch.io/loop-of-life

[1] https://itch.io/jam/gmtk-2025

From: mgrojo
Date: Mon, 4 Aug 2025 10:35:11 +0000

We like to see the Ada source code. I've found it here: https://github.com/SabeDoesThings/Loop-of-Life-src

I tried to compile it using the sdlada crate, but it seems to require another binding or another version of the same binding:

loop_of_life.adb:4:06: error: file "sdl-mixer.ads" not found

From: SabeDoesThings
Date: Mon, 4 Aug 2025 11:51:23 +0000

yeah sdl-mixer is a separate binding so you would have to add it. So that could be the problem.

https://github.com/ada-game-framework/sdlada_mixer

Simple Raycaster in Ada

From: SabeDoesThings
Subject: Simple Raycaster in Ada
Date: Thu, 7 Aug 2025 12:53:08 +0000
Newsgroups: forum.ada-lang.io

Made a simple raycaster in sdlada based off this article [1] took about an hour or so. The source code and footage can be found here [2].

[1] https://lodev.org/cgtutor/raycasting.html

[2] https://github.com/SabeDoesThings/Simple_Ada_Raycaster

Simple Components 4.76

From: dmitry-kazakov
Subject: Simple Components 4.76
Date: Sun, 10 Aug 2025 13:49:47 +0000
Newsgroups: forum.ada-lang.io

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), arbitrary precision arithmetic, pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, persistent storage, multiple connections server/client designing tools and protocols implementations.

Simple components for Ada [1]

Changes to the previous version:

- The parsing example was upgraded from Ada 95 to Ada 2022. The changes include:
- Unicode identifiers;
- new () syntax;
- Container aggregate;
- for expressions in container aggregate;
- declare expressions;
- delta aggregates;
- if expressions;
- case expressions;
- raise expressions;
- The package Parsers.Generic_Ada_Parser.Generic_Dot was added for writing Ada syntax trees in the DOT format of Graphviz;
- The package Parsers.Generic_Ada_Parser.Generic_Text_IO was added for printing Ada syntax trees;
- The package Parsers.Generic_Source.UTF8_Keywords was added.

[...]

[1] https://www.dmitry-kazakov.de/ada/components.htm

GNAT Studio 26.0 for macOS Ventura

From: Blady
Subject: Gnat Studio 26.0 for Macos Ventura
Date: Thu, 14 Aug 2025 19:29:17 +0000
Newsgroups: forum.ada-lang.io

Here is a very preliminary version of GNAT Studio 26.0wa [1] as a standalone app for macOS.

Introduced for the first time last year, the GNATStudio launcher looks for a gnatstudio_launcher.rc file in .gnatstudio folder from either $HOME or $GNATSTUDIO_HOME locations. If it exists, we can define some environment variables with the standard syntax VAR=VALUE. If the VAR exists then VALUE is appended to it. If not, VAR is created with VALUE. Thus, it permits to set extra PATH to GNAT compiler and builder folders or GPR_PROJECT_PATH. If a line begins with '#' then it is not considered. An example file of gnatstudio_launcher.rc is provided in the archive. Modify the content and put it in your .gnatstudio folder.

See readme [1] for details.

Limitation: Ada Language Server has some latencies and doesn't respond when parsing source code with more than 1000 lines. It may be due to some compilation options I missed.

There could be some other limitations that you might meet.

Feel free to report them here.

Any help will be really appreciated to fix these limitations.

[1] https://sourceforge.net/projects/gnuada/files/GNAT_GPL%2520Mac%2520OS%2520X/2025-ventura/

Ahven 2.9

From: Tero Koskinen <tero.koskinen@iki.fi>
Subject: ANN: Ahven 2.9
Date: Mon, 1 Sep 2025 18:32:46 +0000
Newsgroups: comp.lang.ada

Ahven is a simple unit test library for Ada programming language.

Release 2.9 is long overdue maintenance release with following notable changes:

- Alire support
- SPDX identifiers added to all source code files
- Support for custom Set_Up and Tear_Down procedures in test suites

Direct links to the source code distributions:

https://www.ahven-framework.com/releases/ahven-2.9.tar.gz

https://www.ahven-framework.com/releases/ahven-2.9.zip

[...]

Ahven homepage: https://www.ahven-framework.com/

Mercurial source control: https://hg.sr.ht/~tkoskine/ahven

Git mirror: https://git.sr.ht/~tkoskine/ahven/

AdaCL 6.3.0 Release Announcement

From: krischik
Subject: Adacl 6.3.0 Release Announcement
Date: Thu, 4 Sep 2025 12:19:17 +0000
Newsgroups: forum.ada-lang.io

Excited to share the AdaCL 6.3.0 release! This suite of Ada 2022 libraries brings robust tools for app development with wide character support. Check the crates below:

adacl

- Offers strings, tracing, AUnit, smart pointers, and Getopt with wide character support for robust apps.
- Explore: https://adacl.sourceforge.net/adacl/

adacl_eastrings

- Developed by Björn Persson, provides encoding-aware strings for i18n, supporting UCS-4, UTF-8, and more.
- Ideal for multilingual apps. See: https://adacl.sourceforge.net/adacl_eastrings/

adacl_regexp

- Features Ada 2022 regex and SPITBOL patterns with wide char support, enhanced from GNAT.
- Perfect for pattern matching. Check: https://adacl.sourceforge.net/adacl_regexp/adacl_sar

- A powerful search/replace library by Martin Krischik, with wide/wide-wide support for text pipelines.
- View: https://adacl.sourceforge.net/adacl_sar/

hp41cx_tools 1.7.0

From: krischik
Subject: Hp41cx_tool 1.7.0 Release Announcement
Date: Sat, 6 Sep 2025 18:22:50 +0000
Newsgroups: forum.ada-lang.io

hp41cx_tools: Showcasing Ada 2022 and AdaCL with Wide_Character Power

New release of hp41cx_tools crate, a cross-platform suite for decoding and converting HP-41CX emulator FOCAL code, demonstrating the strength of *Ada 2022* and the AdaCL library. Available on Alire [1], this project runs on macOS, Linux, and Windows, and is a stellar showcase for AdaCL's Wide_Character support.

Why It's a Great Ada Showcase

- AdaCL with Wide_Character: Leverages AdaCL.Wide_Strings.Regexp and AdaCL.Wide_Strings.Spitbol.Patterns for context-sensitive conversions of special characters (e.g., Σ, ↑, ≠) in FOCAL code. The adacl-regex crate's Wide_Character support ensures precise handling of Unicode, surpassing standard regex for ALPHA strings and commands.
- Robust AUnit Testing: Comprehensive unit tests, executed via make test, ensure rock-solid reliability, covering decoding and conversion edge cases.
- Ada 2022 Power: Built with modern Ada features for maintainable, high-performance code, handling .px41 and .dm41 memory dumps and conversions to .utf8.focal, .dm41.focal, and .px41.focal formats.

Features

- Decode: Transform HP-41CX emulator hex dumps into FOCAL source code with automatic input detection.
- Convert: Seamlessly convert between Unicode, DM41, and PX-41CX FOCAL formats, handling quote styles and XROM key assignments.
- Pipe-Friendly: Four converters use stdin/stdout for scripting ease.

Note: Encoding is planned for a future release.

[1] https://alire.ada.dev/crates/hp41cx_tools

[2] https://sourceforge.net/p/calculator-scripts/code/ci/master/tree/Tools/hp41cx_tools/src/

Demo of 'arcana' Game

From: charlie5
Subject: Demo of 'arcana' Game Written Ada
Date: Thu, 11 Sep 2025 02:29:44 +0000
Newsgroups: forum.ada-lang.io

Here is a video of a hobby game I've been playing with lately.

The game is written in Ada (of course) and is very much a WIP. The game can be built in single player mode when built with gprbuild or in multi-player mode when built with PolyORB's po_gnatdist, using exactly the same code base. It's only 2d and there is much to do.

Just thought it may be of some interest.

From: mgrojo
Date: Thu, 11 Sep 2025 18:13:30 +0000

Nice! We like source code too. I think it's here: https://codeberg.org/charlie5/arcana

From: charlie5
Date: Thu, 11 Sep 2025 20:22:55 +0000

Yes, that's the arcana git repo.

The game engine library (GEL) is here: https://github.com/charlie5/lace

Tbh, I'm not overly proud of a lot of the code (in both). I overuse access types and there are a lot of TODO's. Frankly the whole thing could do with a re-write, if I had the time.

The game engine also handles 3d: https://www.youtube.com/watch?v=f2Ot8jFjuDw

AWS End of Life

From: heharkon
Subject: Aws End of Life
Date: Tue, 23 Sep 2025 07:28:46 +0000
Newsgroups: forum.ada-lang.io

So ummm… what does this mean in practice?

> AWS is a deprecated product. It will be baselined with the GNAT Pro release 28. After this release, there will be no new versions of this product. Contact your sales representative or send a message to sales@adacore.com to get recommendations for replacements.

From: Heziode
Date: Tue, 23 Sep 2025 13:13:10 +0000

Hum… they will soon sell their families in GNAT Pro…

AdaCore and the Apple Syndrome. Nightmare.

From: AJ-Ianozi
Date: Tue, 23 Sep 2025 14:43:50 +0000

As someone who just recently built a free html5 multiplayer online game using websockets and AWS… and is currently working on an MMORPG that leverages AWS as its login server… oh crap!

Also YASS [1] is using AWS for its templating engine, and YASS is gpl3. Any other open source projects relying on AWS?

EDIT: So if AWS is no longer supported, this may mean the community needs to step up and take over the project. Anyone interested in helping?

[1] https://github.com/yet-another-static-site-generator/yass

From: Heziode
Date: Tue, 23 Sep 2025 18:14:41 +0000

Ping @stcarrez, I think you will also be impacted with AWA [1]

[1] https://github.com/stcarrez/ada-awa/tree/master

From: stcarrez
Date: Tue, 23 Sep 2025 21:33:25 +0000

Yes, I was aware since Fabien talked about it offline at Ada Europe in June.

AWA [1] is using AWS but can also be based on EWS [2]. I also have a prototype server based on Simple Components [3] from Dimitry but it has some issues.

AWS is still a good choice for me as long as it is open source.

The question will be whether AdaCore allows the open source community to take over the project and continue if some developers are interested. Personally, I already have some specific branches that solve build issues on NetBSD or FreeBSD (by getting rid of GNATColl dependency which brings lots of complexity and build issues).

[1] https://github.com/stcarrez/ada-awa

[2] https://github.com/simonjwright/ews

[3] https://dmitry-kazakov.de/ada/components.htm

From: stcarrez
Date: Wed, 24 Sep 2025 21:30:25 +0000

I've seen some companies make changes on the license of projects over time. The copyright holder is AdaCore (if I'm not wrong) which means that they can change the license at any time. [...]

If AdaCore transfers the copyright to some Foundation (Apache?), or some non-profit organization (Ada User Society?) this creates a more positive path for the future.

From: robdaemon
Date: Wed, 24 Sep 2025 21:47:50 +0000

"Take over" isn't quite the right way to look at it. This can take a few forms:

- Asking AdaCore to put the repository under community ownership. It would likely have to move GitHub organizations, and needs maintainers.
- Anyone can fork it, since it's GPL, but you can't call it AWS anymore.

Those are the two main ways of "taking over", but the latter removes the AWS name, since AdaCore would retain the copyright.

Ada and Operating Systems

Ada.Text_IO with -mwindows Switch (MSYS2)

From: jere
Subject: Ada.text_io Paired with Mwindows Switch (msys2)
Date: Fri, 1 Aug 2025 14:54:55 +0000
Newsgroups: forum.ada-lang.io

Using the -mwindows switch with an application that uses Ada.Text_IO seemingly causes the application to either exit / exception out. I was wondering if anyone knew of another switch that I could pair with this to allow the program to run. The text_io stuff is just for debug, so I only specify the -mwindows switch in release mode, but I do want it to spawn the console and print stuff in debug mode.

From: godunko
Date: Fri, 1 Aug 2025 17:12:56 +0000

You can use VSS.Text_Streams.Standards.Output_Stream instead, it outputs to stdout when it is available and does nothing otherwise.

From: jere
Date: Fri, 1 Aug 2025 18:21:59 +0000

Thanks. That's kind of the way I am currently going. I can't get VSS, but I'm emulating the same thing using pragma Debug and wrapping that in a function for now.

Ada Inside

Vikram-3201: India's Custom 32-bit Processor Built for Ada

From: dragon-spirit-wtp
Subject: Vikram-3201: India's Custom 32-bit Processor Built for Ada
Date: Wed, 3 Sep 2025 06:36:59 +0000
Newsgroups: forum.ada-lang.io

India's first 32-bit microprocessor, Vikram-3201, was entirely designed and developed by the country. It has its own custom instruction set and was specifically designed to work seamlessly with the Ada programming language, due to being in widespread use in mission-critical systems, particularly in aerospace and defence applications. Even a custom compiler and other tools were developed for it.

"What Is Vikram-3201 Chip? India's Leap In Indigenous Semiconductor Technology | Indian Defence News"

https://www.indiandefensenews.in/2025/09/what-is-vikram-3201-chip-indias-leap-in.html?m=1

From: JeremyGrosser
Date: Wed, 3 Sep 2025 09:36:02 +0000

ISRO & SCL develops 32-bit Microprocessors for space applications [1]

https://scl.gov.in/products_pdf/SC1130-0_Information_sheet.pdf

It's a 32-bit extension of the 16-bit VIKRAM1601. Both the 16 and 32 bit chips are fabbed on 180nm.

[1] https://www.isro.gov.in/vikram3201.html

Ada for Xtensa ESP32 (LX6/LX7) [draft]

From: alexispaez
Subject: Ada for Xtensa Esp32 (lx6/lx7) [draft!]
Date: Fri, 19 Sep 2025 09:23:19 +0000
Newsgroups: forum.ada-lang.io

Will this work with an ESP32-WROOM-32? I just saw it on sale at AZ-Delivery

From: Irvise
Date: Fri, 19 Sep 2025 15:31:27 +0000

ESPRESSIF does not recommend it for new projects https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

It is a rather old CPU by now. However, it has an Xtensa CPU, so it should work with the toolchain. It is probably under discount in order to get rid of extra inventory.

Newer CPUs/Boards include the ESP32-C6, C3, S3, etc

From: Max
Date: Fri, 19 Sep 2025 19:59:14 +0000

It should work in theory, but right now nobody ports Ada runtime for esp32. We only have a compiler.

From: alexispaez
Date: Mon, 22 Sep 2025 08:09:38 +0000

Please excuse my ignorance (I'm trying to learn) …

What does not have an available runtime mean and does it apply to all ESP32 boards?

Does it mean you can't use any Ada library code and you need to do programming on pure bare metal?

What is required to port an Ada runtime for ESP32? Just wondering if it's something I could take on, for didactic reasons, or if it's totally over my head for now

From: Max
Date: Mon, 22 Sep 2025 16:36:58 +0000

> What does not have an available runtime mean and does it apply to all ESP32 boards?

It means that our toolchain for ESP32 is incomplete. We have only a compiler, but not runtime (for any board). So with the compiler one could starting porting Ada runtime and build it: get libgnat.a and a bunch of *.ali files from runtime source files. Then having a compiler and runtime anyone could build and link an Ada application.

> Does it mean you can't use any Ada library code and you need to do programming on pure bare metal?

Without runtime you can't compile any user's compilation unit (package/subprogram), because it depends on the Standard package that is part of runtime. During compilation the compiler reads the System package to find target properties. In turn, runtime comes in several types. The light runtime is most thin and most easy to port. It provides only basic support, you won't be able to use tasks, protected objects, but you can build an application. light-tasking runtime has support for tasks/protected objects with some restrictions (See Ravenscar profile). To get full Ada you need full Ada runtime, that is very big and probably requires real OS underneath.

> What is required to port an Ada runtime for ESP32? Just wondering if it's something I could take on, for didactic reasons, or if it's totally over my head for now

Porting runtime could be interesting and you could find out much about how it works. From my point of view it is not hard, at least for light runtime. You will need to find out

- compiler flags for your particular chip
- flash layout to create a linker script
- number and names of interrupts to write interrupt vector (in assembly)
- create a starter assembly chunk to initialize stack, clean bss memory section

Some links:

- GitHub - Fabien-Chouteau/bare_runtime: Minimal Ada/SPARK run-time for embedded or other restricted targets [1]
- 5. Predefined GNAT Pro Run-Times — GNAT User's Guide Supplement for Cross Platforms 26.0w documentation [2]
- Embedded Systems | ada-lang.io, an Ada community site [3]
- GitHub - AdaCore/startup-gen: A startup code generator for embedded projects [4]
- Startup-gen User's Guide — startup-gen documentation [5]

Don't hesitate to ask questions!

[1] https://github.com/Fabien-Chouteau/bare_runtime

[2] https://docs.adacore.com/gnat_ugx-docs/html/gnat_ugx/gnat_ugx/gnat_runtimes.html

[3] https://ada-lang.io/docs/learn/getting-started/embedded

[4] https://github.com/AdaCore/startup-gen

[5] https://docs.adacore.com/live/wave/startup-gen/html/startup_gen_ug/index.html

From: Irvise
Date: Mon, 22 Sep 2025 19:39:41 +0000

Adding info to what Maxim just linked, the AdaCore blog has quite a few posts about this topic:

> Porting the Ada Runtime to a new ARM board [1]

> by Jérôme Lambourg – Sep 03, 2018. A step by step tutorial to adapt the ARM runtime to new MCUs/boards.

> Ada on FPGAs with PicoRV32 [2]

> by Fabien Chouteau – Oct 08, 2018. When I bought the TinyFPGA-BX board, I thought it would be an opportunity to play a little bit with an FPGA, learn some Verilog or VHDL. But when I discovered that it was possible to have a RISC-V CPU on it, I knew I… [...]

> Starting micro-controller Ada drivers in the Alire ecosystem [3]

> Open-Source Ada: From Gateware to Application [4]

> by Olivier Henley – Feb 10, 2025. The Neorv32 BIOS project demonstrates how Ada can serve as a powerful alternative to C in open-source embedded development.

[1] https://blog.adacore.com/porting-the-ada-runtime-to-a-new-arm-board

[2] https://blog.adacore.com/ada-on-fpgas-with-picorv32

[3] https://blog.adacore.com/starting-micro-controller-ada-drivers-in-the-alire-ecosystem

[4] https://blog.adacore.com/open-source-ada-from-gateware-to-application

Using Both Cores of RP2040

From: RREE
Subject: Using Both Cores of Rp2040
Date: Sun, 21 Sep 2025 20:13:28 +0000
Newsgroups: forum.ada-lang.io

I am trying to use both cores of a RPi Pico RP2040. Unfortunately the example multicore program in Pico_Examples does not blink. It gets stuck when starting a procedure in the second core. When I stop the program in GDB, it stops in RP.Multicore.Launch_Core1.

```
Starting program: /home/re/Devel/rp2040/pico_examples_2.2.0_ba306c6f/multicore/obj/main
^C
Program received signal SIGINT, Interrupt.
0x10003d3e in rp.multicore.launch_core1 ()
(gdb)
```

I am now on the way to use a tasking runtime. That was discussed before here [1]. Unfortunately I cannot find an example program (simple or not). Do I have to declare a task for core0? If not, how would I communicate between the two tasks? Can I still use the internal FIFO queue or is that excluded in the tasking runtime?

[1] https://forum.ada-lang.io/t/ravenscar-profile-tasking-on-rp2040/744

From: Fabien.C
Date: Mon, 22 Sep 2025 08:51:14 +0000

Which run-time are you using?

From: RREE
Date: Mon, 22 Sep 2025 11:23:43 +0000

The original problem (program stuck in launch_core1) arises in the unmodified pico_examples crate. It uses light-cortex-m0p for all examples, including the multicore. I tried with gcc-14 and gcc-15.

The questions in the last paragraph were a preparation for switching to light-tasking-rpi-pico (or light-tasking-rpi-pico-smp). Perhaps that works better, but I am missing some examples. So I haven't tried yet.

From: damaki
Date: Mon, 22 Sep 2025 13:10:47 +0000

I don't have an example to hand right now for using the tasking runtime on the RP2040, but to answer your questions about tasking:

> Do I have to declare a task for core0?

It's not mandatory as the environment task (which calls your program's main) runs on core0. You can also create multiple tasks on core0 if you wish; the runtime will use priority-based scheduling to manage task execution on the same core.

> how would I communicate between the two tasks?

You can use standard Ada tasking mechanisms for inter-task communication, i.e. protected objects [1].

> Can I still use the internal FIFO queue or is that excluded in the tasking runtime?

The SIO FIFO is reserved for use by the runtime only (the runtime uses this to "poke" the other core to wake it up when it might need to (re)schedule a task).

Note that if you want to use both cores with the tasking runtime then you'll need to use the light-tasking-rpi-pico-smp runtime (the other runtime only runs on core0). To declare a task that runs on the second core you can declare the task at package level like this:

```
task My_Task with CPU => 2;
```

[1] https://learn.adacore.com/courses/intro-to-ada/chapters/tasking.html#protected-objects

From: damaki
Date: Thu, 25 Sep 2025 07:53:35 +0000

It looks like both rp2040_hal and the tasking runtime are both trying to use the same alarm interrupt (ALARM2). The runtime reserves this peripheral and interrupt to implement the semantics of Ada's delay and delay until statement. The last chance handler is being reached because RP.Timer.Interrupts also tries to register an interrupt handler for ALARM2, which is rejected by the runtime because that is reserved.

As a workaround I recommend avoiding the use of RP.Timer.Interrupts (don't with it at all in your code) and use Ada's delay and delay until statements instead when using the tasking runtime.

The lack of documentation/examples about the tasking runtimes is certainly something that could be improved.

From: RREE
Date: Wed, 24 Sep 2025 18:22:10 +0000

OK, something strange. It works and the program runs fine if I remove the debugger and cycle power. It does NOT blink if the debugger is attached or if power was not cycled (i.e. debugger removed but power left on)

From: damaki
Date: Thu, 25 Sep 2025 07:58:13 +0000

> OK, something strange. It works and the program runs fine if I remove the debugger and cycle power. It does NOT blink if the debugger is attached or if power was not cycled (i.e. debugger removed but power left on)

I notice that rp2040_hal doesn't provide a way to reset core1 before launching it. I wonder if that could be contributing to the problem.

In the tasking runtimes, I made it so that it hard-resets core1 before launching it to ensure it is in a known state (i.e. the core1 bootrom is waiting for the handshake protocol).

You could try copying this code to reset core1 before launching it and see if that helps:

https://github.com/AdaCore/bb-runtimes/blob/2ebde6e89a7fe60f38faa4af4671278f59ae6e86/src/s-bbsumu__rp2040.adb#L204-L219

From: RREE
Date: Thu, 25 Sep 2025 19:37:29 +0000

Thanks very much. Adding your reset routine actually solved the problem. Now the code starts on both cores, even without detaching the debugger and without power cycling. I only had to replace the 0 and 1 in your code by False and True. I created a corresponding PR for rp2040_hal.

Ada Practice

A Gem Amongst Gems

From: Fabien.C
Subject: A Gem Amongst Gems
Date: Fri, 4 Jul 2025 13:38:24 +0000
Newsgroups: forum.ada-lang.io

In the process of reviewing the Ada Gems, I was amazed by this feature:

Gem #20: Using pragma Shared_Passive for data persistence

https://www.adacore.com/gems/ada-gem-20

[This gem describes how a pragma Shared_Passive in a package makes global variables persistent across program runs, with the help of an auxiliary file that is transparently written and read by the runtime. –arm]

Who knew about this one? (Honestly)

From: dmitry-kazakov
Date: Fri, 4 Jul 2025 14:27:37 +0000

I did not know it, though I worked on persistence a lot.

However there are issues with that:

- In practice you want to use some other backend than arbitrary files, typically a database or a configuration file in a certain format like registry or XML;
- Portability is a big issue. You want to access storage from different platforms and move it across the platforms;
- You need a lot of work for linked structures, like graphs, links etc;
- You also need to handle tagged types, i.e. you need some means to store and restore tags and then construct objects from;
- External references are another problem. You might have not only internal links that can be stored as offsets to the first allocated element, but also a closure of objects your persistent object refers to. Since objects can be shared you need to handle multiple references and maintain a catalogue of objects in the persistent storage. Then you need some policy of updating shared objects.

From: jere
Date: Fri, 4 Jul 2025 18:06:40 +0000

I really like the Ada gems series for things just like this. I hope they never take the archive down as it is a really valuable resource.

From: OneWingedShark
Date: Fri, 4 Jul 2025 18:54:58 +0000

> Who knew about this one?

I did.

I used it to implement configuration for a bit of custom logging. — Basically a store of Pascal-style strings (256-bytes: 1 length byte, followed by the string data), plus the integral data.

> In practice you want to use some other backend than arbitrary files

I don't know that I agree; for many things, yes… but for small things with very few parameters?

> Portability is a big issue. You want to access storage from different platforms and move it across the platforms;

I mean, you're using Shared_Passive, which means that you should be able to hook it into a DSA program easily. (Though, TBF, I think the LRM stance on DSA programs are that it's implementation-defined; which makes sense: incorporating a truly portable RPC/data-communication protocol would be (a) constraining all DSA programs to those protocols, and (b) would entail incorporation-by-reference to something that would likely be ASN.1 or something of similar capability.)

> You need a lot of work for linked structures, like graphs, links etc

This is perhaps the place that current Ada disappoints the most: it's rather obvious that the original (Ada83) design was done with the idea of being able to incorporate self-referential data structures like the aforementioned, when "compiler/language technology caught up", such that you could make lists/graphs/links. (Actually, I suspect that the access type was /always/ meant to be key: that the Cursor-type in the Ada.Containers.* packages /should/ be access types, discriminated by the container.)

> External references are another problem. You might have not only internal links that can be stored as offsets to the first allocated element, but also a closure of objects your persistent object refers to. Since objects can be shared you need to handle multiple references and maintain a catalogue of objects in the persistent storage. Then you need some policy of updating shared objects.

Honestly, at /that/ point what would be useful is a common Ada+PL/SQL runtime, with the runtime containing a full database-engine.

From: bjorn-lundin
Date: Sat, 5 Jul 2025 10:02:12 +0000

I did. I saw the gem many years ago, and used it to keep track of a sequence number in a communication process. The protocol wants the telegrams ordered by sequence num, which increases by 1 for each tgm. To make the other side happy I saved the sequence num on disk through this pragma, so I did not need to serialize it myself. It is read upon restart which was the point.

That protocol was later changed to accept 0 as a restart marker.

I also customized the storage format. At first it was just 4 integers (32 bit num) but I made it human readable.

It is still running on that site.

Scalar Object 'Valid Attribute

From: klnelson
Subject: Scalar Object 'valid Attribute
Date: Sat, 5 Jul 2025 03:33:45 +0000
Newsgroups: forum.ada-lang.io

I would like to better understand the rationale for the 'Valid attribute for scalar objects as it seems backward to me. Take the case of performing an Unchecked_Conversion to assign an integer value to an enumeration object whose type has a representation clause assigning each member an integer value. The unchecked conversion is performed, assigning the object a (potentially erroneous) value then 'Valid is applied to check if the object is erroneous and if so, the error is handled/corrected.

It seems it would be easier if 'Valid could be applied to the enumeration type and the integer could be checked by doing My_Enum'Valid (My_Int) before doing the assignment and erroneous objects could be avoided all together. This could be wrapped in a simple function:

```
function To_Protocol_ID
  (id : C.unsigned_short)
  return Protocol_ID is
    (if Protocol_ID'Valid (id)
       then Protocol_ID'enum_val (id)
       else Protocol_None);
```

The best general solution I came up with is a Checked_Conversion generic, wrapping Unchecked_Conversion where an enumeration default value is returned in the case where 'Valid returns false. The conversion default is specified at instantiation or as a parameter replacing the instantiated default at checked conversion time.

```
generic
    type Source is ();
    type Target is ();
    Target_Default : Target;
function Checked_Conversion (S : Source;
  Default : Target := Target_Default)
return Target;

function Checked_Conversion (S : Source;
   Default : Target := Target_Default)
return Target is
    function To_Target is new
    Ada.Unchecked_Conversion
    (Source, Target);
    Obj : constant Target := To_Target (S);
begin
    return (if Obj'Valid then Obj
       else Default);
end Checked_Conversion;
```

Is there a better way? If not, why doesn't Ada have Ada.Checked_Conversion?

[Code example omitted. –arm]

From: OneWingedShark
Date: Sat, 5 Jul 2025 04:13:07 +0000

> It seems it would be easier if 'Valid could be applied to the enumeration type and the integer could be checked by doing My_Enum'Valid (My_Int) before doing the assignment and erroneous objects could be avoided all together.

I tend to agree with you, but there are some reasons it is how it is.

IIUC, the reason that it is on the value, rather than the type/subtype, is for things like (a) memory-overlay, and (b) memory-mapped I/O. — I believe the rationale was that you could define your [sub]type with all the correct values and use overlay, mapped-I/O, or unchecked conversion on (e.g.) incoming data, then use Val'Valid on that object, rather than something like:

```
declare
  Raw_Data : Integer with Import,
    Address => IO_Address;
begin
  if Target_Type'Valid( Raw_Data ) then
  -- …
end;
```

As you can see, the problem with a Type'Valid attribute is that it would be hiding a conversion, then testing against the validation, /*then*/ returning a Boolean… meaning that you just lost the passing value if it *is* valid, forcing a reconversion. — we could have had it work on the type, if we'd had attributes that reference functions with an out-parameter, but 'Valid was prior Ada2012.

@sttaft would have more details on it.

From: klnelson
Date: Sat, 5 Jul 2025 16:36:28 +0000

> As you can see, the problem with a Type'Valid attribute is that it would be hiding a conversion, then testing against the validation, /*then*/ returning a Boolean… meaning that you just lost the passing value if it *is* valid, forcing a reconversion. — we could have had it work on the type, if we'd had attributes that reference functions with an out-parameter, but 'Valid was prior Ada2012.

I am not familiar with these type conversion mechanics, though this check was pretty simple. In the case of Protocol_ID, just take the candidate value and look it up in a table containing the valid values of 10, 20, and 30 and return a boolean, avoiding creating an erroneous object and/or raising Constraint_Error. A valid object of type Protocol_ID can then be created without an exception occurring.

IMO, enumeration types have been broken since the language inception as you can't (easily) model values of this nature and it looked like 'Valid, 'Enum_Rep, and 'Enum_Val and resolved this. If someone would be kind enough to point me to the correct section of ARM / rationale document(s), I can just read about it. I've been away from Ada since 2005 and have lost track of the specification documents. A quick google did not provide any results and will try again.

How to 'pass a Function' to a Procedure in Ada?

From: Ret_Build_Engineer
Subject: Very Dumb Review Question: How to 'pass a Function' to a Procedure in Ada?
Date: Mon, 7 Jul 2025 23:38:48 +0000
Newsgroups: forum.ada-lang.io

Please pardon the elementary nature of the question.

I'm trying to remember (I know I did something like this previously), but I can't remember when/where; also I really don't feel that 'pass a function in Ada' is a sufficiently well phrased search question. Meanwhile I guess I'll use it as I can't think of anything better yet.

The end result would be a library of Ada code where someone wants to explore specific results (numerical integration in this case) from an arbitrary function, where the code being exercised has already been provided as a library.

So, how are functions passed as parameters in Ada? Or how the heck does a library know what function it is going to be executing in the future?

Right now, I'm just going to hard-code a function inside the integration procedure.

From: OneWingedShark
Date: Tue, 8 Jul 2025 02:42:43 +0000

> So, how are functions passed as parameters in Ada?

As generic formal parameters.

/Technically/ you can use an access to a subprogram, but generics ought to be preferred over mere accesses.

> Or how the heck does I library know what function it is going to be executing in the future?

For a generic, let me present the problem where you want to insert a print/logging of a value and return that value:

```
generic
   type Value is limited private;
   with Procedure Print( Object : Value)
     is null;
function Debug_Value( X: Value )
   return Value;
function Debug_Value( X: Value )
   return Value is
begin
  return Result : constant Value := X do
    Print( X );
  end return;
end Debug_Value;
```

Here, we are providing a Print function (as well as the type) as a formal parameter, this allows us to instantiate Debug_Value with something like, say, a function of Write_Line(Object'Image).

Note: It may be prudent to have a polynomial type, perhaps implemented as Array (Natural range) of Real, where the elements thereof are the [coefficients of the] exponents.

From: OneWingedShark
Date: Tue, 8 Jul 2025 03:38:06 +0000

Sorry, I meant *"coefficients of the exponents"*.

EG: (3 => 2, 2 => -4, 1 => 7, 0 => 11) would be the array for $2x^3-4x^2+7x^1+11x^0$.

From: Ret_Build_Engineer
Date: Tue, 8 Jul 2025 04:06:54 +0000

Thanks for the clarification regarding the coefficients

Let's say the function in question was in terms of more than one variable…or was a trig function?

I was thinking about defining the function as something like (typing on the fly here):

```
function f (x : float) return float is
begin
  return sin (x); -- potentially much more
                  -- complicated
end f;
```

then the function which is in the library (not yet) which does the math (a numerical integration method in this case) would be given the parameter of 'f', the function.

Again, I really do not feel that I am expressing myself fluently here

From: OneWingedShark
Date: Tue, 8 Jul 2025 06:06:33 +0000

You're getting into potentially tricky territory.

While it is easy to see that integrating sin(x) gives you -cos(x)[+C], the thing that you have to remember is that the computer /isn't/ doing symbol manipulation (well, it /can/, but then you're in for doing the whole of the engine itself, in which case you're probably better off using LISP) — there are ways to tackle the problems, but full-blown numerics is…difficult.

Consider:

```
generic
  with function F( X : Real ) return Real;
package Numeric_Stuff is
  -- Note that by instantiating this, you get a
  -- function which returns a single value, and
  -- is without parameters.
```

```
   generic
      Start, Stop : Real;
   function Generic_Evaluate return Real;
   --…
   Function Generic_Evaluate return Real
     is (F(Stop) - F(Start));
   --…
```

As you can see, there are major consequences of how to use a library based on the choices in design. The above is /probably/ less useful than some other approach, but might be well suited for the particular application.

From: dmitry-kazakov
Date: Tue, 8 Jul 2025 09:11:27 +0000

This is the case when you do not want generics. Normally generics are always the last design choice because of their static nature. You cannot do anything that requires late bindings with them. As it was already pointed out you can use access to subprogram and thus pass any subprogram from anywhere, e.g. from a library loaded /*after*/ your program was compiled.

A more complicated but also more flexible way is to use methods. This approach is actively deployed in various software patterns, e.g. in the visitor pattern when you need to call something when iterating something while maintaining some state external to the iterator itself. In its simple form you pass an object of a type that has a primitive operation which is then called back. The primitive operation is your function. Now this is also static as you need to override the operation in order to put your function body in there. But you can make it dynamic using an access discriminant. So you pass your function as a discriminant of:

```
type Functor
   (F : access function (X : Float)
      return Float
   ) is tagged null record;
function Call_Me (Object : Functor;
   X : Float) return Float;
```

Now let your integrator declared as:

```
function Integrate (What : Functor;
   From, To : Float);
```

Then you can use it as:

```
   Integrate (Functor (Sin'Access), 0.1, 0.5);
```

From: jere
Date: Tue, 8 Jul 2025 20:14:07 +0000

Everyone has already discussed pros and cons. Here are a few examples in compilable code if you need.

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use
   Ada.Integer_Text_IO;
procedure Test is
   -- WAY #1
   -- Named subprogram access type
   -- parameter is used when you want to
   -- "save" the passed in function for later
   type Function_Access is access
      function(Item : Integer) return Integer;
   function f1(f : Function_Access)
      return Integer;

   -- WAY #2
   -- Anonymous access parameter is used
   -- when you just want to execute
   -- it as soon as possible
   function f2(f : not null access
      function(Item : Integer) return Integer)
      return Integer;

   -- WAY #3
   -- Generic is another option
   generic
      with function f(Item : Integer)
         return Integer;
   function f3_generic return Integer;

   -- Junk implementations just for testing
   function f1(f : Function_Access)
      return Integer is (f(100));
   function f2(f : not null access
      function(Item : Integer) return Integer)
      return Integer is (f(100));
   function f3_generic
      return Integer is (f(100));

   -- Here's the function to pass in
   function Test(Item : Integer)
      return Integer is (2*Item);

   -- Junk variables to hold results
   v1 : Integer := f1(Test'Access);
   v2 : Integer := f2(Test'Access);

   function f3 is new f3_generic(Test);

   v3 : Integer := f3;

begin
   null;
end Test;
```

Finalize/Adjust Raised Exception

From: MarioBlunk
Subject: Raised Program_error : Finalize/Adjust Raised Exception
Date: Thu, 10 Jul 2025 08:28:41 +0000
Newsgroups: forum.ada-lang.io

After updating to GNAT 15.1.1 20250626 I get an exception in connection with controlled types. I have a function that returns a controlled type. Before the return statement I get the above mentioned exception.

So before providing more details, my question is whether anyone here is experiencing the same trouble?

From: dmitry-kazakov
Date: Thu, 10 Jul 2025 08:53:37 +0000

This one of the most common bugs Ada programmer faces. Controlled types are all OK, you need to check your implementations of Finalize and Adjust for unhandled exceptions.

See also GNAT.Exception_Actions for tracing exceptions. Program_Error just an indication of some chain of exceptions prior to it.

From: jere
Date: Thu, 10 Jul 2025 13:13:39 +0000

Also keep in mind that GNAT itself has some bugs with that specific scenario. I've had standard Ada containers raise that exception (with no overrides from me). Recently I had to wrap an indefinite holder instance in a separate controlled type and catch the exception in a custom finalize operation to avoid it.

I haven't tried v15 yet though, this is all pre 15 experience on my part.

GUI Libraries - Suggestions

From: vjalmr
Subject: Gui Libraries - Suggestions
Date: Sat, 12 Jul 2025 17:36:38 +0000
Newsgroups: forum.ada-lang.io

My current scenario: I need to write an application with a GUI that works on *nix, win, and osx (or whatever apple calls it these days).

What is my best option for this, in regards to GUI? Could I achieve this by using GTK? I have never seen GTK running on win or osx, so any and all comments would be appreciated.

Please, /no/ web tech.

From: dmitry-kazakov
Date: Sat, 12 Jul 2025 19:31:48 +0000

In case you didn't know GNAT Studio (GPS) is GTK3. So yes, Windows and BSD (Mac stuff) are supported.

The choice is usually Qt or GTK. GTK tries to use look-and-feel of the host OS. Qt is available on a wider range of OSes. Another difference is that Qt is C++ and GTK is C. C++ is far more difficult to communicate.

In any case both have Ada bindings.

From: jere
Date: Sat, 12 Jul 2025 21:28:35 +0000

I use Gnoga [1] a lot. It's pretty handy at work for making some quick simulators and things when I want a GUI. It's browser based, so that helps with cross platform scenarios. I only use Windows and Linux, but so far all of my stuff has compiled for both pretty easily.

[1] https://github.com/Blady-Com/gnoga

From: Fabien.C
Date: Tue, 15 Jul 2025 08:39:14 +0000

I am eagerly waiting for the release of LibreFrame: https://www.libreframe.com/

I would love to see if we can make a "native" rendering front-end for it.

Starting with something like RayGUI and maybe move to a full Ada implementation afterwards.

(I don't know if the Sowebio people are hanging out around here)

From: dmitry-kazakov
Date: Tue, 15 Jul 2025 09:19:20 +0000

Interesting, but does not look like a GUI at all.

To me GUI is all about widgets and handling events/messages. The problem with existing GUI like Qt, GTK, Win32 etc is abstraction inversion since handling events happens in wrong places. E.g. button press is handled by button widget rather than by a container of.

The event processing is unstructured and requires to be in front while the problem space logic is pushed back into callbacks.

Then of course, there is a push vs. pull problem. GUIs are built event-driven. This architecture is totally unsuitable for real-time, high-frequency and massive amounts of data applications where you would rather use polling. You can of course create an oscilloscope widget. I did it in GTK and Win32, but it requires a lot of acrobatics.

From: NumberSix
Date: Tue, 15 Jul 2025 14:57:54 +0000

We're trying a different approach. Give us a chance to show you something consistent. Time doesn't respect what's done without it. And there are, for the moment, only two of us.

For now, we're taking care of the foundations by rewriting the websocket interface and using epoll. We're now up to 1.5 M req/s on average hardware. But we can do even better. And we'll do it if it makes sense. Like automatic reconnection, since websocket is obviously sensitive in this respect.

As this is still being settled, we also have a schedule to complete. Building - LibreFrame [1]

Thanks for waiting a few months, but we'll already have things ready between September and October this year.

We are, of course, always open to comments. We're really counting on it.

All the best from Oleron Island.
Stéphane

[1] https://www.libreframe.com/building

From: dmitry-kazakov
Date: Tue, 15 Jul 2025 15:36:04 +0000

Ada GUI project might be very interesting. Though it is a mammoth task I am not even sure how to approach it avoiding typical errors of GUI architectures.

Mentioning WebSockets in that context rings alarm bells to me. Anyway, good luck.

From: NumberSix
Date: Thu, 17 Jul 2025 15:58:16 +0000

Yes, websocket is natively very different from what we're used to on the web.

However, the use of websockets is a key feature of Web GUI LibreFrame architecture.

In the age of fast, universal networks, permanent bidirectional communication (with the necessary precautions) could greatly enhance the user experience, compared with the classic transactional HTTP, with tons of JS…

To discuss this in more detail, I'd have to talk about LibreFrame's complete architecture, which can be summed up as "everything from the server". Indeed, in our concept, the amount of client-side JS is negligible.

To be user friendly, LibreFrame retains the entire user session, even across multiple tabs. LibreFrame emulates HTTP navigation. Arrows to return to a previous page or move forward are supported.

Similarly, logging out from a session with 10 tabs automatically closes the 10 views. It's a security enhancement. Logging again automatically restores them to the page they were on. We've already demonstrated this at AEiC 2025.

As the icing on the cake, LibreFrame's websockets communication protocol is designed to be very lightweight, and even works on 64 Kbps links (i.e. 0.064 Mbps).

So, may be WebSockets, properly managed, could ring you some pleasant melody

And, yes, you're absolutely right. LibreFrame, as a whole, is a mammoth task. But we're tough. This project originated in 2020 with v20 library. By 2023-2024, we've reached a lot of milestones with v22 framework, checked our assumptions, drawn up a schedule, and we're moving on.

LibreFrame 1.0 will be on show at FOSDEM 2026, with applications already built using it.

We're really looking forward to this first version to gather feedback.

From: elimliadam
Date: Thu, 17 Jul 2025 17:52:40 +0000

FLTKAda [1]

[1] http://www.jedbarber.id.au/cgi-bin/cgit.cgi/fltkada/about/

From: docandrew
Date: Fri, 18 Jul 2025 21:40:18 +0000

I did a little experiment with a native Ada IMGUI concept here:

GitHub - docandrew/DAGBuild: Modern Build Tool

https://github.com/docandrew/dagbuild

It ended up just being a science project but might be interesting to folks here.

Uses the Ada SDL backend.

Hosting an Ada Web App

From: patrickjh
Subject: Hosting an Ada Web App
Date: Sun, 13 Jul 2025 07:05:54 +0000
Newsgroups: forum.ada-lang.io

I am exploring the idea of using Ada to write a web backend mainly because it seems like a nice language. I am just curious - do you have any thoughts about the best way to write and host Ada code for the cloud? I can think of some possibilities and am looking for feedback on which of these might be a good idea and which are a bad idea. Any other ideas are welcome too:

Idea 1: Write the Ada code as CGI scripts running on a Linux cloud virtual machine using something like Lighttpd. Seems simple enough, but I would have to sys admin a Linux system and CGI is kinda low performance.

Idea 2: Use an Ada web framework and run the Ada code as a process on a Linux cloud virtual machine. Also seems simple enough but I would have to sys admin a Linux system.

Idea 3: Write some serverless functions for AWS Lambda in Ada. Similar to the CGI idea the code would be simple but I would not have to sys admin anything. Has anyone done this before? Seems a bit tricky as Ada is not one of the officially supported languages and apparently you have to create some kind of container image.

Idea 4: Since Ada can be used for embedded use cases, is there maybe a way to create a VM image of Ada code that can run as a web server on something like EC2 without any operating system? It would probably have challenges but I imagine that if Ada can run on baremetal hardware without an OS, there might be a way to run in the cloud without an OS?

Idea 5: Create a Docker Container Image with the Ada binary and use that with some cloud service like Kubernetes. There do seem to be some Docker containers for Ada, like this container, → https://hub.docker.com/r/esolang/ada. Has anyone used anything like that?

From: NumberSix
Date: Mon, 14 Jul 2025 09:38:53 +0000

You can give a try to:

Transactional (http)

AWA: Ada Web Application [1] from Stéphane Carrez. It's based on Ada Web Server, comes with an ORM and a lot of goodies.

Bi-directional (ws)

Now (which is the past somewhere), our POC called v22, with real industrial apps created with it, running 365/214: https:/v22.soweb.io

It's based on Gnoga and comes with a highly effective overlay to boost productivity and everything you need to make a complete application.

The future (coming quickly): https://www.libreframe.com

Based on v22 but replacing Gnoga with something completely new. After a presentation at AEIC2025 in Paris, which used an existing Ada websocket layer, we are now recoding this part, for greater simplicity and performance. Our initial feedback has been excellent (speed gains of over x200 and 1,5M req/s on an average server!), and we are now recoding the http web server (websocket starts from http 1.1). When that's finished, we'll extend the available widgets and theming. Expect all this by the end of the year.

As far as CGI is concerned, Matreska has a Fast CGI implementation which, when coupled with a Fast_CGI cache via Nginx, delivers extraordinary performance, but this only applies to static pages.

On the production side, we're not seduced by docker (old technology with a new look) and kubernetes (too complex and irrelevant in our case). We don't use microservices (inefficient and difficult to debug) and we don't use the cloud (too expensive and not so reliable).

We host our own infrastructure on real bare-metal servers, using the Xen hypervisor in PVH mode (no qemu, full para-virtualization and bare metal speed), with Debian everywhere…

There are other ways of doing things, it's just ours. If you feel more comfortable with Docker, that's not a problem.

Administering a Linux server is no more difficult than administering the cloud. On the other hand, a Linux server is much cheaper to run, especially if you've virtualized it so you can have lots of instances. A little effort up front for a big financial gain and greater autonomy.

[1] https://ada-awa.readthedocs.io/en/latest/

From: stcarrez
Date: Mon, 14 Jul 2025 18:52:36 +0000

Some information for you:

- The Ada France [1] web site is running on a (tiny) bare-metal Linux and the web server is based on AWA+AWS. The database used is MySQL. Sources are available at [2]
- My web site https://blog.vacs.fr is also running on bare-metal Linux with AWA+AWS. The Ada binary behind it contains 3 AWA applications that are separate to serve different web sites (virtual hosts). The database used is MySQL.
- The Porion build manager [3] is another AWA+AWS server but it runs within a virtual machine. The database used is SQLite (it could be PostgreSQL or MySQL). Sources are available at [4]
- HTML/Wiki to Wiki Converter [5] is another AWS+Ada Servlet server (a small subset of AWA). It runs within a Docker container. Sources are available at: GitHub - stcarrez/wi2wic: Wiki 2 Wiki Converter [6]
- To show some features of AWA, I created Atlas [7] and you can run it from a Docker container. You'll find some information there on how to start/stop the container.

Docker and Kubernetes bring another level of complexity and you can probably leave that later in the deployment phase.

[1] https://www.ada-france.org/

[2] https://github.com/Ada-France/ada-france

[3] https://porion.vacs.fr

[4] https://gitlab.com/stcarrez/porion

[5] https://wi2wic.vacs.fr/wi2wic/index.html

[6] https://github.com/stcarrez/wi2wic

[7] https://github.com/stcarrez/atlas

Experiences with Large Projects

From: cowile
Subject: Experiences with Large Projects
Date: Tue, 22 Jul 2025 04:20:13 +0000
Newsgroups: forum.ada-lang.io

Ada is often represented as a language for programming in the large. Can someone here with experience in large Ada projects talk about what the experience was working on large Ada projects. What was different from large projects in other languages? What were the primary pains?

Large may be defined generously however you choose, but I'm imagining projects with one or more of:

- Dozens of contributors
- Hundreds of thousands to millions of lines of code
- Production critical for years
- Expansive and changing scope
- Stringent performance requirements such as latency or throughput goals

From: Irvise
Date: Tue, 22 Jul 2025 17:48:38 +0000

I cannot answer this question from first hand experience, however, I do know quite a few Ada programmers working in very large codebases (+2 million lines).

TL;DR: it is just great!

About your points:

- Dozens of contributors: I guess that having a good SCM would help manage all the people. Nevertheless, I assume you are referring to the typical case of "Bob changed X things without notifying Alice, will things break?".
- Ada's very strong typing system, the Compilers sanity checks, extra features (such as liquid types and contracts [1]) and the obvious use of SPARK; will ensure that either random wrong changes do not compile, get caught very early (runtime checks during testing or production) or simply are not allowed by SPARK (the prover).
- A lot of SLOCs: Ada was not a pioneer of the module system, but back in the 80s it was one of the first and it was quite useful and natural. Segmenting large projects by modules/functionality is very easy, recommended and scalable. Newer PLs (read, after 2010s) have realised how important this is and they are also using modules (see Rust). Even C++ finally got some basic module functionality! Just 40 years after Ada… Maybe using C as the baseline design was not a good idea… For Rust, maybe creating an improved version of C++ was also not a good idea…
- Ada was created specifically to deal with very large projects. Eurocontrol [2] deploys a 2.2 million lines program written in Ada (here you can see some insights on its extreme performance tracking [3]). NVIDIA just recently announced that they got ASIL-D certification for their 7 million OS written in Ada/SPARK [4]
- Production critical: pleaaaaase, is this even a question?? See the above answer and also take a look at the Ariane rocket program, Airbus, Boeing, Paris Metro, etc.
- The modular system of Ada allows changes to be contained. Breaking changes (read requirements that change scope) can be detected early by the compiler if they are trivial. The techniques mentioned in the bullet point of point 1. should provide even more assurance and early detection of clashes between changes. If you use generics, type reflection/attributes, etc; a simple change in a source file will automatically expand throughout the rest of your program without you having to do anything!
- Ada generates incredibly performant code. full-stop. Really! Take string (base type, so stl:: bullshit), they are not null/\0 terminated, their length is part of the type itself. The algorithms on structures whose size is always know are incredibly performant. Use Godbolt/Comiler Explorer [5] to see how Ada code compares to other languages. Also, take a look at the bullet point for the second point of yours, that should give you a really good idea on how deep you can go with Ada.
- Just a word of caution, use user defined types for your data, this may help the compiler generate substantially more performant code. If you see that the performance/assembly of Ada is substantially worse than that of C/C++/Rust, it may be that these

languages allow for undefined behaviour and they are taking advantage of it while Ada may be forbidding it!

I hope this answers your points. Please, post any questions, issues or anecdotes that you may have while learning Ada. A great place to get started learning it is https://learn.adacore.com/

[1] https://fosdem.org/2025/schedule/event/fosdem-2025-4879-understanding-liquid-types-contracts-and-formal-verification-with-ada-spark/

[2] https://www.eurocontrol.int/

[3] https://archive.fosdem.org/2020/schedule/event/ada_performance/

[4] https://www.eenewseurope.com/en/nvidia-drives-ada-and-spark-into-driverless-cars/

[5] https://godbolt.org/

From: cowile
Date: Thu, 24 Jul 2025 04:26:57 +0000

Thanks for the answers!

> Ada was created specifically to deal with very large projects. Eurocontrol [1] deploys a 2.2 million lines program written in Ada (here you can see some insights on its extreme performance tracking [2]). NVIDIA just recently announced that they got ASIL-D certification for their 7 million OS written in Ada/SPARK [3]

Would you happen to know anything like how fast those projects build or how much unit testing there is compared to relying on the compiler and formal methods?

[...]

> Another general difference is that you need to be much less defensive (it is due to Ada's type system but also myriads of details like the syntax of the *for* loop, where you can't do dumb things like writing the exit condition wrong, having jump steps of 2 or tampering the loop parameter)

This was something I definitely appreciated about Ada. Some categories of mistake are eliminated by thoughtful syntax, like you have to explicitly specify what block you are closing with end if; or end loop;. That's a big improvement over C's "brackets optional for one line rule". And the additional characters don't bother me because it's more comfortable to type than bracket langs.

> The downside of this is that sometimes you'll see code generators to get around a lot of boilerplate (like in AUnit for unit testing).

> General nestability of packages/functions can lead to insanity

Thanks. These were the kind of things I was looking for when asking about pain points, though the latter seems like a problem in any language.

From: Irvise
Date: Thu, 24 Jul 2025 09:34:27 +0000

> Would you happen to know anything like how fast those projects build or how much unit testing there is compared to relying on the compiler and formal methods?

Ada is quite modular, so once you build the entire project, the rest should just be incremental builds. There should be little to no cascade effects.

NVIDIA used the formal methods for their OS extensively. Eurocontrol, AFAIK, uses ASIS [1]. Sadly, ASIS does not support newer Ada versions and was removed from GCC in v10 (though they still get support from AdaCore as they pay for their tools). ASIS was an extended set of checks done by the compiler and behaved like an extended target/language. AdaCore would like people to start using LibAdalang [2] instead. Also, keep in mind that Ada, when compiled with GCC or LLVM (oh, yes, we have an LLVM backend! [3]) it can use the sanity checks and tooling that they offer, on top of the language spec. You can also use any other analysis tools, such as Valgrind, etc.

Additionally, when dealing with memory, you may not be aware that Ada supports "Memory Pools" (known as arenas in other langs) since 2012. They can also serve as an advanced tool to deal with safety, increase performance and debugability. You can learn much more about the Ada memory model in this outstanding presentation [4] by Jean-Pierre. Also, keep in mind that GNAT offers finalizable types (not yet standard) since V15, which are basically simple controlled types that are compatible with embedded systems, have a much better defined behaviour and can be analyzed by SPARK. Also, SPARK can now prove complete memory correctness ala borrow checker from Rust (including leaks and lifetimes, unlike Rust).

> But since I don't see a lot of opportunities to work on big projects with other people, I created this thread to ask about people that had. I contribute to the Linux kernel as part of my day job and am very interested in OS dev [...]

There are quite a few interesting OS projects in Ada. The biggest and probably the one you would like to dive deep into is Ironclad [5] with its Gloire distribution [6]. Ironclad is a *NIX kernel written in Ada/SPARK, similar to Linux and it can run now a ton of typical *NIX programs, such as videogames . It is not huge, it is only about 40k lines, but it can already do a lot. It also implements its own RTS (RunTime System). You can see an introduction to it here [7]. Oh, and this is mostly written by a couple of 25(±) year olds!

Another interesting OS is HiRTOS [8], it is a hard RT kernel written in SPARK and fully formally verifiable for ARM/RISC-V devices. This one is about the same size as Ironclad, but completely different in nature, targets and design.

Finally, a commercial open source Ada/SPARK micro/separation kernel is Muen [9], we know from disclosed information that it is being used in encryption devices and telecommunications/telephone systems. After a clone of the kernel with all of its submodules, there are 158 kLOCs of Ada (the kernel is only 6200 LOCs).

> I remember trying something like that about ten years ago during a previous spark (lol) of Ada interest, but ran into the problem of not knowing how to cut down the language runtime to work on bare metal and gave up for lack of finding good resources for how to do it.

Wait, what? The runtimes can be zero/light (no runtime whatsoever), extremely basic (allowing for secondary stack management, memory features…), light-tasking (think of the restricted tasking profiles Ravenscar and Jorvik [10], see section D.13) and full-tasking (Windblows, Macs, Linux *BSDs…). You do not have to cut the Runtime, you only have to use the features that can be used with the runtime you are targeting.

For example, if you want to use a light runtime, you must simply not use any tasking or advanced memory thingies. You can take a look at the impressive, great, wonderful and amazing SweetAda [11] project. In total it has 100 kLOC lines. It is a framework to create bare-metal applications in a f*ck ton of boards and architectures. It truly is an amazing project and it is done by a single incredible person. Runtime-wise, it only supports zero/light and secondary stack (selected arches) runtimes, though the goal is to support tasking. It also has some drivers for common peripherals and general virtual thingies, such as FAT support.

Another great project is bb-runtimes [12], this is a collection of arches/boards/fpgas with several runtimes. There are several that have light-tasking or even full-tasking runtimes. It also brings in some peripherals to the mix. This is probably the most complete Runtime-filled repo out there.

You can also check the runtimes for Ironclad, HiRTOS and Muen if you are interested. Also, a nice bit of trivia/updates, the Ravenscar and Jorvik profiles can be verified by SPARK, meaning that if you stick to them, SPARK can prove the correct behaviour of your multitasking applications [13]!

[1] https://www.adacore.com/asis

[2] https://github.com/AdaCore/libadalang

[3] https://github.com/AdaCore/gnat-llvm

[4] https://archive.fosdem.org/2016/schedule/event/ada_memory/

[5] https://ironclad-os.org/

[6] https://codeberg.org/Ironclad/Gloire

[7] https://fosdem.org/2025/schedule/event/fosdem-2025-4871-developing-device-drivers-for-ironclad-using-ada/

[8] https://github.com/jgrivera67/HiRTOS

[9] https://muen.codelabs.ch/

[10] https://www.adaic.org/resources/add_content/standards/22rm/rm-final.pdf

[11] https://github.com/gabriele-galeotti/SweetAda

[12] https://github.com/AdaCore/bb-runtimes

[13] https://docs.adacore.com/spark2014-docs/html/ug/en/source/concurrency.html

From: cowile
Date: Fri, 25 Jul 2025 06:08:52 +0000

[...]

> There's a killer feature I never see anyone talk about: You can specify dependencies between Ada packages for initialization during the environment task prior to the entry procedure, to avoid the static initialization order fiasco [1].

Interesting. Where can I read more about this feature?

> I'm wondering if the era of programming languages focusing on "expressiveness" is over, and those focusing on semantic correctness (like Ada) is now here.

Even before LLMs I noticed a (re)surging interest in strong, static type systems. Haskell gained some popularity in the mid-2010s before falling off (presumably happy to stay an ivory-tower language avoiding success at all costs) and Rust keeps growing. I think everyone is running from the horrors of the scripting language dominated web era. My view from inside Big Tech is the dominant dynamic languages JavaScript and Python are banned in their naked form and must use typed variants.

Ada has benefited from this trend and IMO ends up looking like a language ahead of its time. And the over-complexity criticisms don't stick anymore because if anything Ada feels simpler than a lot of more modern languages.

There will absolutely be synergy between LLMs and languages with strong correctness focus. Like you say it is easier to section off the vibe coded bits and good compiler feedback helps keep the LLMs on track for longer tasks. With LLms, the thing most pushing people towards "expressive" languages, the extra typing, is a non issue.

[1] https://en.cppreference.com/w/cpp/language/siof.html

From: Irvise
Date: Fri, 25 Jul 2025 10:19:27 +0000

> I'm also curious about any support that makes debugging easier. Does it play well with GDB and are there other more useful debugging tools out there?

GDB has built in support for Ada OOTB and it works wonders. I always launch GDB with catch assertions and catch exceptions and just let my Ada program (built with checks enabled, which is the default in GCC) and let it tell me where things fail. It literally is the most pleasurable experience I have ever had with debugging. Additionally, it supports all the features of GDB, so you can also do remote debugging with OpenOCD in embedded boards. I have used it with great success. SweetAda has built in support for it for its targets. Also, obviously, you can debug QEMU binaries, etc. Ada is no different than C/C++ and I would dare to say it has even better features than them because of the language design and because of the extra small features in GDB. Also, you can deploy an Ada program in Renode [1] if you wanna give them a test without the real hardware. Here is a small blog detailing Ada and Renode [2].

> Interesting. Where can I read more about this feature?

This is a good place [3]

> Ada has benefited from this trend and IMO ends up looking like a language ahead of its time. And the over-complexity criticisms don't stick anymore because if anything Ada feels simpler than a lot of more modern languages.

110% right!

[1] https://renode.io/

[2] https://blog.adacore.com/getting-started-with-renode-simulating-an-ada-stm32f429disco-blinky-firmware

[3] https://gcc.gnu.org/onlinedocs/gnat_ugn/Controlling-the-Elaboration-Order-in-Ada.html

From: kevlar700
Date: Fri, 25 Jul 2025 10:33:19 +0000

> There is no documentation for the "zero"/light runtimes because… that is what they represent, the absolute lack of any runtime x)

There is some runtime support in the light runtimes and even zero. It is just all independent of clocks etc. and so can be made to run on any chip much more easily and on chip families. For example array bounds, 'Image, 'Value etc.

5. Predefined GNAT Pro Run-Times [1]

Hopefully these may be of some use for Linux but I find Linux drivers a pain to read with all the abstractions never mind trying to integrate Ada code for many targets so good luck.

Hacking the Linux Kernel in Ada - Part 1 - Linux.com [2]

For this three part series, we implemented a 'pedal to the metal' GPIO driven, flashing of a LED, in the context of a Linux kernel module for the NVIDIA Jetson Nano development board (kernel-based v4.9.294, arm64) in my favorite programming language...

Ada on any ARM Cortex-M device, in just a couple minutes [3]

GitHub - alkhimey/Ada_Kernel_Module_Framework: Framework for writing Linux kernel modules in Ada [4]

GitHub - AdaCore/bb-runtimes: Source repository for the GNAT Bare Metal BSPs [5]

[1] https://docs.adacore.com/gnat_ugx-docs/html/gnat_ugx/gnat_ugx/gnat_runtimes.html

[2] https://www.linux.com/audience/developers/hacking-the-linux-kernel-in-ada-part-1/

[3] https://blog.adacore.com/ada-on-any-arm-cortex-m-device-in-just-a-couple-minutes

[4] https://github.com/alkhimey/Ada_Kernel_Module_Framework

[5] https://github.com/AdaCore/bb-runtimes

From: pyj
Date: Thu, 24 Jul 2025 11:59:47 +0000

[...]

> how fast those projects build or how much unit testing there is compared to relying on the compiler and formal methods?

IME, just the baseline Ada experience prevents you from shooting yourself in the foot 95% of the time. e.g. you cannot return a reference to an in and out param, or a stack variable unless you explicitly mark them as aliased.

Build times are pretty good compared to "I changed a low level header file and had to wait 30 minutes for a build." I have heard extensive formal methods proofs can take a long time (hours), though GNAT studio has ways to target analyses. My verified postfix calculator [2] takes about a minute or so on a 5950x, but proofs seem to parallelize really well. Trendy Test runs all tests randomized and in parallel, the problem being if you have unit tests depending on module state, the

result isn't well defined unless you mark those tests as "run in serial."

> contributing an Alire package called format_strings (AI-assisted)

I've been playing around more with LLMs and vibe coded decent Ada versions of Rust's Rc and Box equivalents in about 10 minutes. A big reason I still do Ada hobby dev is because the spec/implementation firewall gives a place to cordon off vibe-coded elements until you have time to go through them. I'm wondering if the era of programming languages focusing on "expressiveness" is over, and those focusing on semantic correctness (like Ada) is now here.

[1] https://en.cppreference.com/w/cpp/language/siof.html

[2] https://pyjarrett.github.io/2025/06/10/postfix-calculator.html

From: zertovitch
Date: Sun, 27 Jul 2025 16:20:38 +0000

> Would you happen to know anything like how fast those projects build

On a "normal" PC a ~3 million LOC project takes a few seconds to build when you are back after, say, an absence of one week and catch up with other people's changes. For a full build (usually unnecessary), it depends on the PC: a 4-core takes 7 minutes, a 24-core takes 45 seconds.

From: zertovitch
Date: Tue, 22 Jul 2025 19:32:11 +0000

The *big* difference with language X (choose one with conditional compilation, "include"'s, global scope by default, pointers everywhere, soft typing, …) is *Modularity*!

The bigger the teams and project size, the bigger the difference. While you get the whole modularity integrated with Ada and have minimal recompilation at each build, the X guys wait for whole rebuilds each time they change a curly bracket. Plus using, configuring, scripting external tools that try to figure out dependencies - it's a lot of work for them!

Then, Ada's waterproof type system and the low usage of pointers. The X guys worry about getting memory dumps, doing forensic about that, etc.

Another general difference is that you need to be much less defensive (it is due to Ada's type system but also myriads of details like the syntax of the *for* loop, where you can't do dumb things like writing the exit condition wrong, having jump steps of 2 or tampering the loop parameter).

This confidence is both a blessing and (potentially) a curse: with Ada you are much more relaxed thanks to the early bug detection. But you (individually or collectively) may even become /too/ relaxed and a bit spoiled. That's the potential curse.

From: JC001
Date: Wed, 23 Jul 2025 06:32:04 +0000

> But you … may even become /too/ relaxed and a bit spoiled.

> Ada has made you lazy and careless. You can write programs in C that are just as safe by the simple application of super-human diligence.

E. Robert Tisdale

From: JC001
Date: Wed, 23 Jul 2025 06:29:13 +0000

My experience is similar to zertovich's. Starting with the architecture phase, and continuing throughout the design phase, design decisions are formalized in package specifications. These are then enforced by the compiler when the project gets to code generation. In effect, there was no software integration effort needed, since it had been done by the compiler during implementation. Compared to other languages, especially those that would happily compile obsolete calls to subprograms whose specifications had changed, this represented significant savings.

Of course, I've also seen projects that treated package specifications like C header files: an inconvenience required by the compiler, but not something that anyone ever read. They also tended not to use user-defined numeric types, and made heavy use of access types. They did not see as much advantage from Ada.

From: pyj
Date: Wed, 23 Jul 2025 12:07:40 +0000

My normal day job is jumping into large codebases (millions of lines, usually C++) and helping out. [...]

> What was different from large projects in other languages?

1. A lot of problem detail is better hidden in packages due to the language structure.

Ada splits code into specification and body implementations, when written for GNAT each is in a separate file (.ads and .adb respectively). This, combined with encapsulation in Ada being at the package level (types are opaque or non-opaque, visibility is based on package relation), not the class level (C++, C#) means a lot of implementation details stay in the related implementation files. It also means types related to the same subproblem will be in the same file, and you don't need to hop between multiple files to see the entire picture. 99% of the time, just seeing the spec file is all you need, and that spec file is much shorter.

In C++, C#, or Java, you start peeling back classes and interfaces and it takes a while to get to an executable line of code, because you need a constellation of objects to do anything meaningful. In Ada, I find that subproblems end up as distinct packages and the focus is on functional behavior, not on playing connect the dots with types.

In C++ or Rust, you'll often deal with related types (types within types). In Ada, types cannot declare additional types inside themselves, everything is declared at the package level, so I often find an entire subproblem solved in the implementation with all the types used there, rather than jumping five files to understand a high level flow path.

2. A lack of macros and Turing-complete templates means less "magic code."

I'm confident I could take almost any programmer and toss them into an Ada codebase and they'd be able to read almost any part of the entire codebase in a few weeks. There's only a few secret handshakes in generics code [...] that you have to look up.

C++ and Rust code lets you really shoot yourself in the foot with complicated macros. Ruby on Rails is built on this sort of magic with metaprogramming as well. Rails, C++ and Rust do a lot of cool stuff with the macros and metaprogramming, but it can hinder understanding what is actually going on.

Ada doesn't have macros, so what you read is what you get. There's no wild __VA_ARGS__ usage in macros or std::enable_if SFINAE or macro_rules! code with wild and different rules. Ada generics occur at the package and function (or procedure) level, so the little bit of "this is what we're generic over" is the weird part and the rest is "just code." You don't end in angle-bracket hell, instead you get a different type of hell where you have to explicitly instantiate templates, but then you use them just like non-generic code.

The downside of this is that sometimes you'll see code generators to get around a lot of boilerplate (like in AUnit for unit testing).

Another downside is that straightforwardness sometimes just isn't convenient. In Trendy Test, I abuse the exception mechanism for flow control in various ways to handle test registration and running. If I had the macro power of C++ or Ruby, or the metaprogramming of Ruby, there would be much better ergonomics.

3. Constraints help

You can embed a lot of information in Ada directly. Two different ints might have different semantic meaning and shouldn't be mixed and Ada lets you do that, like a built-in Rust newtype or a Go type-definition. Ada depends on function overloading, so you can use that system to only allow meaningful operations: it makes sense to multiply Meters_Per_Second by Seconds to get Meters. If you screw something up with a derived type, the compiler has your back

and it embeds the meaning into the program.

This also goes for pre/post conditions which are on the specification, not an assert hidden inside an implementation. Often I didn't have to bother looking at the implementation of something, which saved time.

4. General nestability of packages/functions can lead to insanity

The one big issue I've seen in Ada is the ability to nest functions (using that term here instead of "subprogram") inside of functions combined with the lack of closures in the language. One project I was looking to contribute to had multiply-nested local functions, with a gigantic function with most contents starting halfway across my editor.

This is exaggerated, but something like this:

```
procedure Foo is
  procedure Foo1 is
    declare
      procedure Foo2 is
        procedure Foo3 is
          -- ...
    begin
      -- ... some actual executable code
```

[1] https://ada-lang.io/docs/arm/

From: zertovitch
Date: Wed, 23 Jul 2025 18:25:17 +0000

> In Ada, types cannot declare additional types inside themselves, everything is declared at the package level

It depends on what you mean by "/the/ package level": every declaration can be local (to packages or subprograms): it includes types, generics, …, and that, within nested packages, functions, etc.

You can have an Ada program without packages but with custom types:

```
with Ada.Text_IO;
procedure Hello is
  type ABC is (a, b, c);

  procedure Nested (p : ABC) is

    type HW is (hello, world);

  begin
    for i in HW loop
      Ada.Text_IO.Put (p'Image & ' ' & i'Image
& ' ');
    end loop;
  end;

begin
  for p in ABC loop
    Nested (p);
  end loop;
end Hello;
```

> General nestability of packages/functions can lead to insanity

But on the bright side, nesting is very practical. Some subprograms make sense only within a certain context. You also save tons of parameters - an advantage in readability /and/ performance.

Of course each feature can be misued.

Note that AdaControl has rules about limiting the nesting:

> Max_Nesting (x4) Controls scopes nested more deeply than a given limit.

From: pyj
Date: Thu, 24 Jul 2025 00:40:38 +0000

> It depends on what you mean by "/the/ package level": every declaration can be local (to packages or subprograms): it includes types, generics, …, and that, within nested packages, functions, etc.

You're exactly right.

What I was trying to do was contrast to nested types like in C++ (struct Foo { struct Bar {}; };). You can declare types inside declare or function or procedure or task bodies, but only those inside packages are visible from outside their containing elements AFAIK, and hence usable when from outside and hence part of the programming boundary interface. They are also visible inside nested subprograms.

C++ and Rust really do seem to love their related types (like $whatever_container::iterator, $whatever_type::reference) whereas in Ada these types would likely be declared at the same level (Ada.Containers.Vectors.Iterator, Ada.Containers.Vectors.Reference_Type)

> Some subprograms make sense only within a certain context. You also save tons of parameters - an advantage in readability /and/ performance. Of course each feature can be misused.

Helper subprograms definitely make sense in some circumstances. I find it hard to read more than one set of lexically nested subprograms. For efficiency, it depends on how many parameters, how often you're calling them, the weight of your parameters, if the compiler will inline and what the compiler actually outputs. In this case I'm not sure, I'd have to find or write something with deep nesting subprograms, measure and look at some assembly.

From: kevlar700
Date: Fri, 25 Jul 2025 10:42:33 +0000

Ada was designed with reducing project lifetime costs of large projects as a primary goal. I'm not sure if any other language was actually even designed as such but certainly not so carefully.

Comparing the development costs and other benefits of Ada or SPARK vs other languages

https://forum.ada-lang.io/t/comparing-the-development-costs-and-other-benefits-of-ada-or-spark-vs-other-languages/681

I can't even find a date on it [1] but it seems some were against Ada quite early on. I guess they liked getting monopolies by using application specific languages. Wow were they grasping at straws if this is the best they could come up with. Some features should only be used sparingly to alleviate a particular pain point and some have been improved upon since then too.

[1] https://dl.acm.org/doi/pdf/10.1145/1041326.1041327

From: pyj
Date: Fri, 25 Jul 2025 23:35:08 +0000

> Ada was designed with reducing project lifetime costs of large projects as a primary goal.

> Perhaps the most astounding example is the YF-22 development: here are the [HTML] slides [1] detailing how effective Ada is in large projects

There's also this one from AdaCore. [2]

Looking at these things from an outside point of view, they look like language proselytization propaganda.

However, after writing a whole bunch of Ada code and then coming back to it multiple years later and seeing how easy things are to change, my skepticism has /considerably/ waned.

> language ahead of its time

It doesn't matter if you're sitting on a gold mine if no one knows about it. A massive problem is that Ada's vernacular predates or conflicts with a lot of modern terminology (storage_element, task, protected object, 'class isn't a "class", limited type, tagged instead of class, entry, "entry families", "controlled type" instead of "RAII", access instead of "handle" or "pointer" – yes, I know they're not exactly the same, "generic packages" – the closest thing might be an OCaml signature?). I tried to address some of these in my FOSDEM 2022 talk. [3]

[1] http://archive.adaic.com/docs/present/engle/whyada/tsld043.htm

[2] https://www.adacore.com/uploads/techPapers/Controlling-Costs-with-Software-Language-Choice-AdaCore-VDC-WP.PDF

[3] https://archive.fosdem.org/2022/schedule/event/ada_outsiders_guide/

Why Do You Keep Using Ada?

From: SabeDoesThings
Subject: Why Do You Keep Using Ada?
Date: Fri, 5 Sep 2025 12:01:05 +0000
Newsgroups: forum.ada-lang.io

I've been using Ada for over a month now and I've been really enjoying it and I wanted to connect more with the Ada community so I wanted to ask. How long have YOU been using Ada and what do

you like about it? I'm curious what people who have been using it way longer than me have to say about this awesome (and VERY underrated) language.

From: Nordic_Dogsledding
Date: Fri, 5 Sep 2025 15:20:08 +0000

I wrote avionics software for two helicopters in a big team.

Later I ported the original avionics (big endian, Ada 83) to a new compiler of a different vendor on different hardware (little endian, up to Ada 2012) for a real time flight training simulator. It was amazing how fast this was.

From: Irvise
Date: Sun, 7 Sep 2025 18:38:47 +0000

Disclaimer, I am not a professional programmer.

- Because it is fun to do things with it. I do not write code so that I can write code. I write code to do fun stuff and Ada just gets out of the way. Other languages are complex, require me to reason quite heavily, force me to think in the language-space rather than in my-problem-space.
- Because it is brutally easy to debug. I once found a stupid error in a library (un-initialised variable) in a matter of 20 mins. This is huge. The library was several thousands lines of code I had never seen. I only had to give it input and it gave me outputs. But with a specific, correct input, it crashed. I found the offending line in 15 mins thanks to Ada's check and arithmetic. Had it been any other language, it would have produced a garbage output and I would not have noticed it until several hours or days into my project (the output was not meant to be read by humans, it was automatically generated code).
- Because it made me learn a lot. I learnt more about low level code with Ada than C. I learnt more about types than any other languages.
- Because the community is super cool and knowledgeable.
- Because it is rock solid and well designed.
- Because it has features other langs can only dream about, such as type predicates, contracts and SPARK

I hope this answers your question.

From: jere
Date: Sun, 7 Sep 2025 19:42:56 +0000

I don't get to use Ada in my professional life as much (most things are done in C, but I try to sneak in some Ada when I can), so most of my Ada use these days is fun stuff at home.

For me it's just a fun language. I like most things about it (there's definitely some pain points for me, but they are minor), but I like how a well constructed package looks and feels. I like the type system, how enums are utilized, and the unique tasking system among other things.

From: pyj
Date: Sun, 7 Sep 2025 20:50:32 +0000

This is my personal list of why I learned the language in 2021ish and keep coming back to the language.

- Simple rules, clear syntax (easy to learn, and I've come back years later to a project and was able to pick it right up again).
- Ranges on types ('first, 'last, 'pred, 'succ, 'range).
- Super-powered enums ('first, 'last', 'pred, 'succ, 'value, 'image). Built-in to-string and parsing of enums, and you can use them as array indices.
- Pre/post conditions.
- Package instead of class-level encapsulation. Packages cut up subproblems at the domain level, not at some arbitrary software architecture level, so I get fewer getters/setters in tightly related code, fewer "God classes" [1], and keeping related types/functions together prevents ravioli code [2].
- Spec/implementation separation emphasizes "Why am I writing this?". Specs prevent the need to code fold and combine the public interface of a module and make it explicit.
- Good low level control (bit representation, compiler intrinsics, ASM, easy import from C)
- Ease of creating semantically different primitives (e.g. type Foo is new Integer). Helps with units and preventing data from flowing between systems without being cleaned.
- Named parameters and aggregates.
- Access types and aliased make "pointer" semantics much more direct.
- Operator overloading when I want it (i.e. math code).
- Explicit generics show the cost and slow down "generify all of the things" you see in other languages.
- Scope-based types (controlled types) for auto cleanup.
- Alire (95% of the time, I don't need to mess with the build).
- SPARK (opt-in verification).
- Mostly easy-to-use standard library.
- Environment startup task lets you explicitly control system startup before the program enters the entry point.

tldr;

Ada makes it easy to write the right thing the first time, even for low level code.

[1] https://en.wikipedia.org/wiki/God_object

[2] https://en.wikipedia.org/wiki/Spaghetti_code#Ravioli_code

From: pmnw
Date: Mon, 8 Sep 2025 09:11:17 +0000

We can enumerate the many benefits all we want, but I think it all boils down to Ada simply being an excellent tool of engineering.

From: kevlar700
Date: Thu, 11 Sep 2025 23:58:31 +0000

> We can enumerate the many benefits all we want, but I think it all boils down to

> Ada simply being an excellent tool of engineering.

And excellently engineered. It may have done some things differently with hind sight or maybe just today (UTF-8 dominance) but the language is all exhaustively well considered and engineered, which is rare and perhaps unique.

From: kevlar700
Date: Fri, 12 Sep 2025 00:15:39 +0000

I believe Dewar called it a "real marvel of engineering".

They set out to create the best language ever deserving to replace all other languages. Personally I think they achieved it. I guess the fact that the D.O.D. isn't saving as much money from using Ada as they should be is largely a social problem and not a technical one. Perhaps the NSA didn't actually want everyone to use a secure language too like they didn't want people using cryptography (Bernstein vs United States).

From: cup
Date: Mon, 8 Sep 2025 10:36:31 +0000

Because I'm a masochist. It takes me 3-5 times longer to write and debug an Ada program as it does C++/Fortran/Python/Java/C#/TCL/VBA. Sometimes, one syntax error can get me bogged down for half a day. Basically, I haven't rearranged the problem in my design to do it the Ada way. I'm still in C++/Python mode.

I keep using it because I'd like to think [in] Ada and speed up the development time instead of thinking in another language and then translating and having to rethink the problem and recode it the Ada way. It gets easier once I've worked out the whys and the hows.

From: JC001
Date: Tue, 9 Sep 2025 08:39:31 +0000

I like Ada because it supports the way I think as a software engineer.

I think all software should be correct. I use Ada because it is the best language for creating software that is correct, as shown by its long record of use in domains where the software must be correct.

My experience is that creating correct software with Ada takes less effort than creating incorrect software in other languages.

From: F-Loyer
Date: Tue, 9 Sep 2025 12:05:16 +0000

> My experience is that creating correct software with Ada takes less effort than creating incorrect software in other languages.

I feel the same. Strongly and statically typed language makes it harder to have a program which compiles… but once it compiles, it is often OK.

From: kevlar700
Date: Wed, 10 Sep 2025 08:30:56 +0000

Because it's the best language, better than C and Rust for embedded, better than Go for memory control and language design quality and reliability (Go shines at throwing people into short projects). Better than Rust at basically everything (I'm only not sure about concurrency performance but certainly concurrency safety on a standard runtime). I'm also now somewhat invested and whilst Zig looks interesting, I haven't actually investigated and I doubt it comes even close to what Ada SPARK offers.

From: tcoram
Date: Thu, 11 Sep 2025 17:50:57 +0000

I do a lot of embedded development, from real time industrial devices to 8-bit microcontrollers. It's the only language (outside of Forth on MCUs) that I don't have to *convince* I want to do low level embedded stuff.

And the tasking capabilities, warts et al, are still heads above any other programming languages.

Default_Component_Value with Subtype?

From: Blady
Subject: Default_component_value with Subtype?
Date: Sun, 28 Sep 2025 17:37:30 +0000
Newsgroups: forum.ada-lang.io

ARM 3.6 Array Types specifies: "Default_Component_Value shall be specified only on a full_type_declaration."

Thus, default initial values can only be given for types and not for subtypes. Is there any raison? This prevents the user from specifying a user specific default value on an extern library type which can't be changed. This also prevents specifying different default values for several subtypes.

From: jere
Date: Sun, 28 Sep 2025 23:02:29 +0000

The discussion the ARG had on it makes it sound like there was some sort of standard rule of thumb they followed in that decision, but I haven't looked through it long enough to see if I can find out why that rule of thumb existed

REF: http://www.ada-auth.org/cgi-bin/cvsweb.cgi/ai05s/ai05-0228-1.txt?rev=Top1.13

From: sttaft
Date: Mon, 29 Sep 2025 15:30:29 +0000

> The discussion the ARG had on it makes it sound like there was some sort of standard rule of thumb they followed in that decision

Actually, the AI explains its reasoning:

> We considered making these subtype aspects rather than type aspects. That would allow giving different default values to different subtypes. However, given the existence of many ways to create anonymous subtypes it would be very hard to guarantee that all values of a type are initialized. Without that guarantee, these aspects lose a lot of their purpose.

More generally, subtypes are somewhat "soft" in Ada, and particularly for scalar types, exactly which subtype is relevant in a given context can be somewhat unclear. As suggested, a derived type gives you much more control, and the /type/ of a scalar object is never in doubt.

Be that as it may, the aspect could probably be made to work with subtypes, but it would definitely add complexity to the typical Ada implementation.

From: Blady
Date: Mon, 29 Sep 2025 19:44:02 +0000

[...] It is surprising that I can give a Dynamic_Predicate aspect to a subtype but not Default_Component_Value!

From: sttaft
Date: Mon, 29 Sep 2025 21:17:02 +0000

> It is surprising that I can give a Dynamic_Predicate aspect to a subtype but not Default_Component_Value!

Good point. But predicates don't in general work very well if you have to put them on the "first" subtype (aka the type as a whole).

In fact, this combination was anticipated, but it seems that whatever compiler you are using didn't follow the rules stated in RM 3.2.4(31/5), which indicate a predicate is only checked on a default-initialized object if in fact the object has some explicitly specified default initialization:

> For an object created by an object_declaration [1] with no explicit initialization expression [2], or by an uninitialized allocator [3], if the types of any parts have specified Default_Value or Default_Component_Value aspects, or any subcomponents have default_expression [4]s, a check is performed that the value of the created object satisfies the predicates of the nominal subtype.

So you might want to file a bug report for this, if you have a small reproducer for this problem.

[1] https://ada-rapporteur-group.github.io/ARM/Ada_202Y/AA-3-3-1.html#S0032

[2] https://ada-rapporteur-group.github.io/ARM/Ada_202Y/AA-4-4.html#S0132

[3] https://ada-rapporteur-group.github.io/ARM/Ada_202Y/AA-4-8.html#S0164

[4] https://ada-rapporteur-group.github.io/ARM/Ada_202Y/AA-3-7.html#S0063

Ada Frontiers

Suggestions for Next Version of Ada Standard

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Suggestions for next version of Ada standard
Date: Mon, 21 Jul 2025 20:19:28 +0000
Newsgroups: comp.lang.ada

[Distributed for the ARG —dc]

The Ada Rapporteur Group (ARG) of the ISO Working Group responsible for the Ada programming language standard (WG9) has a public GitHub repository for suggesting changes to the Ada standard:

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/134

You can get there by starting at the ARG website: https://arg.adaic.org

We have also advertised on the Ada Forum: https://forum.ada-lang.io/

https://forum.ada-lang.io/t/plan-for-next-version-of-international-ada-standard/2007/1

Tucker Taft, on behalf of the Ada Rapporteur Group

New `finally` Keyword in GCC 15.1

From: Irvise
Subject: New `finally` Keyword in Gcc 15.1
Date: Mon, 28 Jul 2025 21:09:30 +0000
Newsgroups: forum.ada-lang.io

I failed to document (or maybe it was added a bit later than when I finished the changelog for GCC 15) that there is a new GNAT extension available in GNAT/GCC 15, it is the `finally` keyword. You can read about it in the GNAT Reference Manual [1]. There is not much documentation about it though nor examples. However, I came across this SPARK commit [2] which provided some examples. I tested the first one with a quick gnatmake -gnatX0 XXX.adb and it compiled without much fuss.

Sooo… how could this be used? I am asking as a complete newbie, not as a

critique. It is also nice to see that SPARK now supports it too!

[1] https://gcc.gnu.org/onlinedocs/gnat_rm/Finally-construct.html

[2] https://github.com/AdaCore/spark2014/commit/f2aaedcadeb71db2c61ba6706d5abcd50bbd6d6a

From: jere
Date: Tue, 29 Jul 2025 00:22:06 +0000

The first thought that comes to mind is something like files that you want to close if you successfully open them but later encounter an exception while parsing them. You can use the finally section to close the file regardless of whether the full process completed or exceptioned out.

From: dmitry-kazakov
Date: Wed, 30 Jul 2025 10:24:39 +0000

> how could this be used? I am asking as a complete newbie, not as a critique

It is a feature complement to object finalization. Finalization is an OO approach to the general problem of safe finalization. Finally is a procedural counterpart.

For example, let you have a mutex you seize to protect a critical code section. To ensure that the mutex gets released you can create a holder object that seizes the mutex on initialization and releases it on finalization. Then you place the holder in a scope like this:

```
declare
   Lock : Holder (Mutex'Access);
begin
   ... -- Critical section
end;
```

Now the procedural alternative would be:

```
Mutex.Seize;
begin
   ... -- Critical section
finally
   Mutex.Release;
end;
```

You can use it in smallish designs when procedural decomposition is sufficient.

Note also. You need to be very careful with the scope where you place /finalize/. In the example above it would be a bug to place Mutex.Seize after the /begin/. The compiler cannot protect you here.

From: sidisyom
Date: Sun, 3 Aug 2025 12:50:18 +0000

As mentioned, this may come in handy in cases where any type of resources acquired during the main sequence_of_statements need to be freed when the block exits (i.e. semaphores, files, memory etc).

One interesting point, as mentioned above, might be that the finally block gets executed whether the main sequence_of_statements terminates normally or with an exception. The RM mentions that: /*after the main sequence_of_statements is executed, and after any potential exception_handler is executed*/ but it's not explicitly mentioned whether that's the case for /unhandled/ exceptions too where no handlers exist in the exception block? Assume that's the case?

It also seems to be an /abort-deferred/ region when applying ATC.

From: Nordic_Dogsledding
Date: Sun, 3 Aug 2025 16:06:09 +0000

> The RM

Note: This is *not* the Ada RM. AdaCore are leaving the standard and beginning to develop an Ada derivation of their own.

> mentions that: /*after the main sequence_of_statements is executed, and after any potential exception_handler is executed*/

Now, what happens if the `finally` part raises an exception?

From: dmitry-kazakov
Date: Sun, 3 Aug 2025 16:14:17 +0000

> Now, what happens if the finally part raises an exception?

I guess the same when the exception part raises. It will propagate out of the block. This is sort of inconsistent with the claimed purpose of /finally/ should an unhandled exception happen in the middle of it. But what would you expect from Java-esque stuff? Alternatively one could nuke the program with Program_Error…

SPARK/Ada

[AdaCore Webinar] Introduction to Formal Verification with SPARK

From: Irvise
Subject: [adacore Webinar] Introduction to Formal Verification with Spark
Date: Mon, 14 Jul 2025 17:13:17 +0000
Newsgroups: forum.ada-lang.io

AdaCore just uploaded the recording of one of their webinars. I really liked it. It is a great summary and showcase of SPARK with real world examples and problems. Also, for those that were in AEiC 2025, this is closely related to Fabien's presentation (which I hope to edit and publish… soon?).

Anyway, here is the YT link for the video: https://youtu.be/tYAod_61ZuQ

And if someone has a good reputation/points/whatever_kids_have_nowadays in HackerNews and similar avenues, feel free to do so! A bunch of very interesting information is shared, specially regarding real world companies using Ada/SPARK for real world projects!

SPARK - Disabling Check after Proofs Are Ok

From: sbenitezb
Subject: Spark - Disabling Check after Proofs Are Ok
Date: Fri, 26 Sep 2025 17:13:26 +0000
Newsgroups: forum.ada-lang.io

After SPARK proofs pass, does the prover store this information and then the compiler use it to disable the runtime checks that are no longer needed? I couldn't find this information anywhere.

From: streaksu
Date: Fri, 26 Sep 2025 17:30:15 +0000

Nope, you need to disable it yourself, there is no GNAT gnatprove communication. Even if there was, you probably want to disable them anyways to support compilation without verification.

From: Irvise
Date: Sat, 27 Sep 2025 07:03:30 +0000

One way to potentially automate this is to create a shell script or Makefile which calls GNATprove && GNATmake without checks. If GNATprove errors out, GNATmake won't be called without checks. You can expand this idea so that if GNATprove fails, then GNATmake is called with checks, debug info, etc

From: JC001
Date: Sat, 27 Sep 2025 07:37:47 +0000

How would the ObjectAda compiler know about the results from the prover?

From: kevlar700
Date: Sat, 27 Sep 2025 17:30:02 +0000

There are also various pragmas such as suppress as some will want performance and some will want defense in depth or both in different places.

https://learn.adacore.com/courses/Ada_For_The_Embedded_C_Developer/chapters/05_SPARK.html