ADA USER JOURNAL

Volume 46
Number 2
June 2025

Contents

Page

Editorial Policy for Ada User Journal 74

Editorial 75

Quarterly News Digest 76

Conference Calendar 99

Forthcoming Events 106

Articles from the AEiC 2025 Work-in-Progress Session

N. Costa, J. Cecílio, R. Salgueiro, M. Rosa, D. Domingos
"*LLM-based Framework for Email Classification and Phishing Detection*" 108

G. Calliet
"*Archaeology for the Future - Ada on OpenVMS*" 112

T. Carvalho, L. M. Pinho, P. Paiva
"*Development and Management of Digital Twins for Energy Monitoring Systems*" 116

A. Geraldo, J. Cecílio, P. M. Ferreira, A. Casimiro, A. Oliveira de Sá
"*A Comparative Study of Reliable Predictive Models for Energy-Efficient MPC in Buildings*" 120

J. Kohl, G. Muck, G. Jäger, S. Zug
"*Dynamic System Model Generation for Online Fault Detection and Diagnosis of Robotics Systems*" 124

J. K. Fichte, T. Philipp
"*Verified SAT Redundancy Checking in SPARK*" 129

Ada-Europe Associate Members (Ada Organizations) 136

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	76
Obituaries	76
Ada-related Events	77
Ada-related Resources	81
Ada-related Tools	83
Ada-related Products	89
References to Publications	89
Ada and Other Languages	89
Ada Practice	90
Ada Frontiers	95
SPARK/Ada	97

[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,

I am saddened to have to report the passing of esteemed Ada personality Simon J. Wright. Widely regarded as the go-to person for anything Ada-on-macOS, he will be sorely missed. Expressions of appreciation can be found in threads both in comp.lang.ada [1] and in forum.ada-lang.io [2].

Not to dwell solely on sad news, there is a fascinating thread on Ada 83 compilers and the possibility of reviving some of them [3]. Also, the next Ada iteration is in the works and preliminary ideas have been gathered, which are discussed here [4].

Sincerely,
Alejandro R. Mosteo

[1] "Simon J. Wright", in Obituaries

[2] "Simon J. Wright - RIP", in Obituaries

[3] "Open Source Ada 83 Compilers", in Ada-related Tools

[4] "Plan for Next Version of International Ada Standard", in Ada Frontiers

Obituaries

Simon J. Wright - RIP

From: Jesper-Quorning
Subject: Simon J Wright - Rip
Date: Thu, 29 May 2025 17:16:21 +0000
Newsgroups: forum.ada-lang.io

Saddened to read this:
Memorial to Simon J Wright [1]

[1] https://github.com/R34P3R-H09/RIP-SIMON-WRIGHT

From: OneWingedShark
Date: Thu, 29 May 2025 17:47:08 +0000

Yes, this is sad news.

From: Irvise
Date: Thu, 29 May 2025 21:03:51 +0000

He was a joyous, prolific Ada programmer, he was the soul of GNAT in Mac/Apple devices… I am glad that we could have some chats over at the monthly meetup… This is very sad. Posthumous thank you for your outstanding work.

RIP, you will be missed.

From: AJ-Ianozi
Date: Thu, 29 May 2025 21:17:57 +0000

This is heartbreaking news. Simon has been a vital and instrumental member of the Ada community for years, and someone who I respect and look up to highly. I would never have been able to use GNAT on my Mac without him, and all of my interactions with him have been so positive over the years. Rest in Peace.

From: khv2online
Date: Thu, 29 May 2025 21:28:44 +0000

Rest in Peace Mr. Wright

From: Lucretia
Date: Thu, 29 May 2025 21:35:57 +0000

R.i.P. fellow Englander.

From: gast
Date: Thu, 29 May 2025 21:38:10 +0000

Rest in Peace, Mr. Wright.

From: kevlar700
Date: Thu, 29 May 2025 21:45:06 +0000

Certainly one of the best. Rest in peace.

From: jere
Date: Thu, 29 May 2025 22:08:26 +0000

He was one of my favorite folks in the community. So nice, helpful, cared about others. I'll miss him.

From: streaksu
Date: Thu, 29 May 2025 23:00:01 +0000

All interactions I've had with Simon were amazing, and he always was willing to put his time and soul into helping people new and old in the community. Condolences to his family and close ones. Thanks for everything.

From: Micronian2
Date: Thu, 29 May 2025 23:37:27 +0000

I am so shocked and saddened by the news of Simon's passing. He was an invaluable member of our community, offering guidance and support for many years. His loss leaves a significant void. If he has family, I wish them well during this difficult time.

From: brownts
Date: Fri, 30 May 2025 01:08:53 +0000

This is very sad news indeed. Simon, you will be missed. He was always so enthusiastic and helpful. A big loss for the community. Rest in peace.

From: Ret_Build_Engineer
Date: Fri, 30 May 2025 02:36:02 +0000

I too appreciate greatly the positive impact Simon had on me. Very patient, pleasant, positive, extremely knowledgeable and helpful.

Is there a widow and/or family whom we could inform of how much he will be missed? That might be comforting to them.

On a technical note, is there someone who can at least partially fill his shoes with regards to Ada compiler support for Apple Silicon?

From: docandrew
Date: Fri, 30 May 2025 04:24:02 +0000

This is really shocking and heartbreaking. Simon was really a gentleman, eager to help and full of wisdom and knowledge. He'll be missed. Rest in peace Simon!

From: heharkon
Date: Fri, 30 May 2025 05:39:19 +0000

Ohh… Really sad news. I've had some small interactions with him over the years, while learning Ada, and he was always the most kind and knowledgeable person and a helping hand.

From: captain-haddock17
Date: Fri, 30 May 2025 07:17:46 +0000

Oh This is heartbreaking news! Simon showed me how to build GNAT on my Mac (and also could do it on FreeBSD).

I owe him a lot for his regular GNAT builds deliveries for macOS!

Thank you Simon for your writings and answers in the Ada discussion forums, with always a relevant and kind response.

He was certainly a good person.

To his family,

Rest in Peace

William

From: DirkCraeynest
Date: Fri, 30 May 2025 09:51:27 +0000

So sorry to read this.

Simon really was very important for the Ada community, not in the least for his work supporting Ada on Apple systems. A very friendly and helpful person too, and active in many Ada-related forums.

I was lucky to have met Simon IRL, though it was a long time ago: at the Ada-Europe conferences in Vienna (2002) and York (2005); we shared a cab to the airport in Vienna after the conference…

Reminds us once more that we're not immortal…

From: mosteo
Date: Fri, 30 May 2025 10:34:13 +0000

He was a great guy and a great programmer, a true guru in the best sense of the word. He always went beyond expectations in all my interactions with him, generally due to some quirk of Alire on macOS.

He showed us that you can be on top of your game until the very end, which I find comforting. Godspeed, Simon.

From: NumberSix
Date: Fri, 30 May 2025 15:22:22 +0000

I had used its simple and straightforward EWS web server as the basis for a telemetry system. Good, readable code, easily tweakable, accompanied by good documentation.

Perhaps this is a good time to say that, just as Mr. Wright was a great and helpful man, the Ada community he had served is also a great and helpful community, led by good people.

That, too, is a great merit.

Rest in Peace, Mr. Wright.

From: zmower
Date: Fri, 30 May 2025 21:20:56 +0000

I used to work with Simon before he retired. Lovely man, always helpful. Carried on in the same vein after he retired. I'll miss him.

Chris

Simon J. Wright

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Simon J. Wright
Date: Sat, 31 May 2025 08:34:23 +0000
Newsgroups: comp.lang.ada

Simon J. Wright passed away recently.

Simon is well known through his work supporting GNAT on the Mac, his participation in many Ada forums, and by being a very active and helpful member of the Ada community until the end.

https://forum.ada-lang.io/t/simon-j-wright-rip

He will be missed...

Dirk

From: Moi <findlaybill@blueyonder.co.uk>
Date: Sat, 31 May 2025 23:08:02 +0000

This is very sad news indeed.

Bill F.

From: Niocláisín Cóilín De Ghlostéir <spamassassin@irrt.de>
Date: Sun, 1 Jun 2025 10:50:00 +0000

Many Adaists began during this week to express sadness over Simon J. Wright's death but following hyperlinks via a first hyperlink on that Discourse Ada-forum webpage from May 29th, 2025 gives the notion that he died before April 3rd, 2025.

I noticed no alert on Team Ada and USENET and that Discourse Ada forum that Ms. Charlene Roberts-Hayden died on May 11th, 2025. Cf. Bryan Marquard, "Charlene Roberts-Hayden, pioneering Black woman in computer programming, dies at 86", "The Boston Globe",

https://www.BostonGlobe.com/2025/05/18/metro/charlene-roberts-hayden-computer-programming-pioneer-passes-away

which David Emery hyperlinks to from the LinkedIn group that is called Ada Programming Language since circa May 31st, 2025.

She wrote to Hal Hart via Team Ada in 1997:

"Hal,
I strongly agree with your comments. I know here in the Boston area, on various committees I participate on, I have heard nothing but distress and regre from the proponents of Ada who know the worth of the only highly reliable HOL in existence today and sarcasm from Ada's opponents, "even the DoD has given up on Ada". I think that without the Ada mandate, given the popular, existing languages unreliability and the software explosion far surpassing that of the 1980's the DoD is headed for an even more catastrophic software revolution than that which preceded and was the impetus for the Ada language."

From: Stéphane Rivière <stef@genesix.org>
Date: Sun, 1 Jun 2025 12:55:59 +0000

Niocláisín, thanks for your comments

Some related links (no paywall)

https://www.blackcatholicmessenger.org/charlene-roberts-hayden-obit/

https://hart-site.net/Purdue-CS-Panel-Reunion/WhateverHappenedTo/HalHart.html

> She wrote to Hal Hart via Team Ada in 1997: [...]

This was very true and has been verified.

Ada-related Events

Ada Monthly Meetup, 5th of April 2025

From: Irvise
Subject: Ada Monthly Meetup, 5th of April 2025
Date: Wed, 2 Apr 2025 21:36:12 +0000
Newsgroups: forum.ada-lang.io

Are there any topics that someone would like to share? I have received no requests.

Also, this meetup will be the last in a long time. In may I will not be available and June's meetup will be cancelled in favour of the AEiC 2025 Ada Developers Workshop, just saying

From: Irvise
Date: Sat, 5 Apr 2025 14:57:38 +0000

We just had the meetup and here are the topics that were discussed:

- @Max presented the following topics:
  - He has been working with https://notebooklm.google/ in order to create Podcast style conversations that cover Ada topics. He has already uploaded one chapter/sample and he would like to have feedback on it! https://www.youtube.com/watch?v=mxlTFRkgRro
  - He has also restarted work on Ada support for the Yocto Project https://www.yoctoproject.org/ A Linux build system geared towards small custom images for embedded and small devices. His work is present in GitHub - reznikmm/meta-ada at topic/hack [1]
  - He also pointed out the webinar from AdaCore, where they present the new improvements to their lineup of products and expected language improvements https://www.youtube.com/watch?v=mu4DEAHZeko
- Additionally to the video, anybody can have discussions regarding improvements to the language as proposed by AdaCore in their RFC GitHub repo GitHub - AdaCore/ada-spark-rfcs: Platform to submit RFCs for the Ada & SPARK languages [2] If you would prefer to see what the ARG is up-to and partake in the discussions, please, refer to Ada Rapporteur Group [3] and the Community Input tab.
- Fernando showcased the changelog for GCC/GNAT 15. It will be made public next week.

- A reminder on the Ada Developer Workshop call for proposals was made. It is still open for the next few days! Remember you can make a proposal for an online talk! https://www.ada-europe.org/conference2025/index.html
  - The cost of the Workshop is still not public pending some discussions for some sponsorships.

Talk to you soon.

[1] https://github.com/reznikmm/meta-ada/tree/topic/hack

[2] https://github.com/AdaCore/ada-spark-rfcs

[3] https://arg.adaic.org/home

Ada Developers Workshop @ AEiC, Speakers, Topics and Price

From: Irvise
Subject: Ada Developers Workshop @ AEiC, Speakers, Topics and Price
Date: Thu, 8 May 2025 20:40:49 +0000
Newsgroups: forum.ada-lang.io

Dear community,
It pleases us to announce the list of speakers and topics for the upcoming Ada Developers Workshop, which will be taking place on June 13th, in Paris, as part of the Ada-Europe international Conference on Reliable Software Technologies [1]. The list below does not represent the schedule, which will be published in the main website when ready:

- @jgrivera67: Writing Embedded Ada Applications on Zephyr
- @stcarrez: Automating License Identification with SPDX-Tool in Ada
- Pascal Pignard UXStrings: a Unicode and dynamic length string library for Ada
- Xavier Petit, Stèphane Rivière: A KISS Ada GUI
- @LionelDraghi: Using natural language for test specification, is that really wise?
- @zertovitch: Writing a competitive BZip2 encoder in Ada from scratch in a few days
- @Fabien.C: Property Based Testing in Ada: the missing 10%
- @Fabien.C: Let's Write a Safety Monitor for a Mars Rover!

Also, we are happy to announce that thanks to the generous support from AdaCore, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to 20€ for early registration and 35€ for late ones (up-to 20 people). As a reminder, online participation will be possible at no cost, however, we ask you to register, in order to keep some traceability of the number of participants and have clean communications.

The registration page [2] will open soon. There is also already a list of available accommodation places [3] for the conference. Of course, you are all more than welcome to join us however is best to you!

The Ada Developers Workshop team.

[1] https://www.ada-europe.org/conference2025/workshop_adadev.html

[2] https://www.ada-europe.org/conference2025/registration.html

[3] https://www.ada-europe.org/conference2025/accommodation.html

From: DirkCraeynest
Date: Fri, 9 May 2025 06:15:03 +0000

Thanks for distributing this info, Fer.

For the record. You wrote:

> Thanks to the generous support from AdaCore, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to 20€ for early registration and 35€ for late ones (up-to 20 people).

This should be (diffs emphasized): thanks to the generous support from AdaCore, /Ada-Europe/, Ada-France and Ada-Belgium, we have been able to reduce the cost of attendance in person to /25€/ for early registration and 35€ for late ones (up-to 20 people).

From: DirkCraeynest
Date: Sat, 14 Jun 2025 12:05:47 +0000

Members of the #AdaProgramming language community met yesterday, in a "tropical" Paris and online, at the 2nd Ada Developers Workshop #AdaDevWS held at the 29th #AdaEurope International Conference on Reliable Software Technologies #AEiC2025. "And there was much rejoicing!"

Slides and video recordings of all technical presentations will be uploaded to the #AdaDevWS webpage at

https://www.ada-europe.org/conference2025/workshop_adadev.html

Ada-Europe Int. Conf. Reliable Software Technologies, AEiC 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Int.Conf. Reliable Software Technologies, AEiC 2025
Date: Mon, 12 May 2025 20:12:34 +0000
Newsgroups: comp.lang.ada

Call for Participation

29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025)

10-13 June 2025, Paris, France

www.ada-europe.org/conference2025

*** Early registration DEADLINE May 27 ***

** Extensive info and registration online ***

*** Add tutorials and/or a workshop to your conference registration ***

#AEiC2025 #AdaEurope #AdaProgramming

Organized by Ada-Europe and Ada-France, in cooperation with ACM SIGPLAN, supported and sponsored by Embedded France, AdaCore, Emenda, Systerel, Scitools, Pacte Novation, Institut Frederik Bull, and Mines Paris

---

Preparations for AEiC 2025, the 29th Ada-Europe International Conference on Software Reliable Technologies, are well underway!

Come to the Ada-Europe conference in Paris, experience a packed program in an exciting town, benefit from tutorials on Tuesday, join a workshop on Friday, enjoy the social events and some sightseeing!

Register now: discounted fees until May 27!

<http://www.ada-europe.org/conference2025/registration.html>

Extra conference sponsorship allows for an extremely low 25 EUR fee for the Ada Developers Workshop on Friday!

See below for an overview, and visit our website for more details about the conference program, registration fees, social events and many more.

General Information

The 29th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2025) returns after 11 years to Paris, France. The conference schedule comprises keynote and invited talks, a journal track, an industrial track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops. There will be time for networking during breaks and lunches, as well as various social events.

AEiC 2025 is the latest in a series of annual international conferences started in the early 80's, under the auspices of Ada-Europe, the international organization that promotes knowledge and use of the Ada programming language and reliable software in general, into academic education and research, and industrial practice.

The Ada-Europe series of conferences has over the years become a leading international forum for providers, practitioners and researchers in reliable software technologies. These events highlight the increased relevance of Ada in general and in safety- and security-critical systems in particular, and provide

a unique opportunity for interaction and collaboration between academics and industrial practitioners.

The 2025 edition of the conference continues a number of important innovations started in previous years:

- reduced conference registration fee for all authors;
- extra low registration fee Ada Developers workshop;
- journal-based open-access publication model for peer-reviewed papers;
- compact program with two core days (Wednesday & Thursday);
- tutorials on Tuesday, followed by welcome event for all participants;
- workshops on Friday, optional chill event on Thursday evening.

Overview of the Week

- Tue 10: 4 half-day tutorials, 1 full-day tutorial, welcome reception;
- Wed 11: core technical program, conference banquet;
- Thu 12: core technical program, post conference chill-out;
- Fri 13: 3 full-day workshops.

Extensive information on AEiC 2025 will be on the conference website, such as an overview of the program, the list of accepted papers and presentations, and descriptions of workshops, tutorials, keynote and invited presentations, and social events. Also check the conference site for registration, accommodation and travel information. The Final Program brochure will be available there as well.

Venue

<http://www.ada-europe.org/conference2025/venue.html>

The conference will take place at Mines Paris, a founding member of Université PSL, a leading French engineering school and the French leader institution in research partnerships. It is located along the Luxembourg gardens, next to the Quartier Latin, and is close to public transportation, including line B of the RER to Charles De Gaulle airport (CDG).

Paris, the capital city of France, is renowned for its rich history, stunning architecture, and vibrant culture. Often referred to as "The City of Light," Paris is home to iconic landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral. If you can stay over before or after the conference, there's a lot to see. Check the Practical Information section of the conference website for more info.

Invited Speakers

<http://www.ada-europe.org/conference2025/keynotes.html>

This year the conference will once more feature keynotes and invited talks. All will address topics of relevance in the conference scope, with time for questions and answers.

- On Wed morning, June 11, a keynote talk by Stefano Zacchiroli, of Télécom Paris, France, on "Building a Safer Open Source Supply Chain with Software Heritage, the Great Library of Source Code".
- On Wed afternoon, June 11, an invited talk by Tullio Vardanega, from University of Padua, Italy, on "Language Ownership: Key Moments in the Lifetime of Ada".

Conference Core Composition

The core conference program features three distinct types of technical presentations, with different duration, in addition to the keynote and invited talks: journal-track talks (25 minutes), industrial-track talks (20 minutes), vendor presentations (15 minutes), work-in-progress-track talks (10 minutes).

All papers presented in the journal track, the industrial track and the work-in-progress track have undergone peer review. Presentations are combined into by-theme and not by-track sessions, in order that authors and participants alike enjoy all flavors of the program in a mixed as opposed to segregated combination.

The list of accepted papers and presentations, and the detailed schedule, will be announced shortly.

Tutorials

<http://www.ada-europe.org/conference2025/tutorials.html>

A full-day tutorial is offered on Tuesday 10th:

- "Developing Bare-metal Embedded Software in SPARK Ada for 64-bit ARM Platforms", by J. Germán Rivera, Tesla, USA.

In parallel, four half-day tutorials are scheduled:

- "Multiple Double and Multiword Arithmetic", by Jan Verschelde, University of Illinois at Chicago, USA;
- "Unleash the Power of Ada Generics", by Gautier de Montmollin, Ada-Switzerland, Switzerland;
- "Turning your Programming Language into a Modeling Language", by S. Tucker Taft, AdaCore, USA;
- "Introduction to the Alire Package Manager", by Quentin Dauprat, Université de Caen, Normandie, France.

Satellite Events

Three workshops are held on Friday 13th:

- 4th ADEPT workshop "AADL by its practitioners".

<http://www.ada-europe.org/conference2025/workshop_adept.html>

- 10th International Workshop on "Challenges and New Approaches for Dependable and Cyber-Physical System Engineering" (DeCPS 2025).

<http://www.ada-europe.org/conference2025/decps.html>

- 2nd "Ada Developers Workshop", an informal yet dynamic gathering for developers in the Ada community to meet, share insights, and present their latest projects or project updates; a full "Ada day" with 8 technical presentations on various Ada-related topics and 10 authors from 5 countries: Belgium, France, Spain, Switzerland, and USA.

<http://www.ada-europe.org/conference2025/workshop_adadev.html>

Social Program

<http://www.ada-europe.org/conference2025/social_program.html>

The conference provides several opportunities to socialize:

- Each day: coffee breaks and lunches offer ample time for interaction and networking with participants and vendors.
- Tuesday early evening: Welcome Reception (details to be announced).
- On conference days: Mineralogy Museum. The École des Mines features a splendid Mineralogy museum. During the conference, you can access it free of charge by showing your name badge. Do not miss this opportunity!
- Wednesday evening: Conference Banquet at the "Bouillon Racine", a typical Parisian restaurant built in 1906 and classified as "historical monument". The restaurant is located within walking distance from the conference location. We will have dinner in the beautiful dining room on the first floor, an authentic turn of the 20th century decoration.
- Thursday evening: Chill Event at the Café Latin, a casual typical café, to have a little chat with colleagues during an informal dinner (separate registration). For those who want to get a feeling of the Latin Quarter, we prepared an itinerary that will show you the main milestones of the neighborhood while walking to the Café Latin.

Further Information

Registration:

- registration information is provided at <http://www.ada-europe.org/conference2025/registration.html>
- early registration discount until Tuesday May 27, 2025

- payment possible by credit card or bank transfer
- special low conference fee for authors
- discount for Ada-Europe and ACM SIGPLAN members
- registration includes coffee breaks, lunches and social events
- strong discount on conference fees for students
- minimal fee for Ada Developers Workshop
- see registration page for all details

Promotion:

- recommended hashtags: #AEiC2025 #AdaEurope #AdaProgramming

The conference is organized by:

- Ada-Europe <http://www.ada-europe.org/>
- Ada-France <https://www.ada-france.org/>

in cooperation with:

- ACM SIGPLAN <http://www.sigplan.org/>

supported and sponsored by:

- Embedded France <https://www.embedded-france.org/>
- AdaCore <https://www.adacore.com/>
- Emenda <https://emenda.com/>
- Systerel <https://www.systerel.fr/>
- Scitools <https://scitools.com/>
- Pacte Novation <https://www.pactenovation.fr/index.php/logiciel-critique-ada-fondements-et-enjeux/>
- Institut Frederik Bull <https://institutbull.fr/>
- Mines Paris <https://www.minesparis.psl.eu/>

Please make sure you book accommodation as soon as possible.

<https://www.ada-europe.org/conference2025/accommodation.html>

For more info and latest updates see the conference website at <http://www.ada-europe.org/conference2025>.

We look forward to seeing you in Paris in June 2025!

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2025 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be
Dirk.Craeynest@kuleuven.be

* 29th Ada-Europe Int.Conf. Reliable Software Technologies (AEiC 2025)

(V5.1)

Ada-Belgium Spring 2025 Event, Sun 1 June 2025

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Belgium Spring 2025 Event, Sun 1 June 2025
Date: Mon, 19 May 2025 16:09:39 +0000
Newsgroups: comp.lang.ada

Ada-Belgium Spring 2025 Event

Sunday, June 1, 2025, 12:00-19:00

Leuven, Belgium

including at 15:00

2025 Ada-Belgium General Assembly

and at 16:00

Ada Round-Table Discussion

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250601-abga.html

---

Announcement

The next Ada-Belgium event will take place on Sunday, June 1, 2025 in Leuven.

For the 14th time, Ada-Belgium organizes their "Spring Event", which starts at noon, runs until 7pm, and includes an informal lunch, the 32nd General Assembly of the organization, and a round-table discussion on Ada-related topics the participants would like to bring up.

Schedule

* 12:00 welcome and getting started (please be there!)
* 12:30 informal lunch
* 15:00 Ada-Belgium General Assembly
* 16:00 Ada round-table + informal discussions
* 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Wednesday, May 28, 18:00, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2025 Event", so you can get precise directions to the place of the meeting. Even if you already responded to the preliminary announcement, please reconfirm your participation ASAP.

If you are interested to join Ada-Belgium, please register by filling out the 2025 membership application form [1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2025 with all member benefits [2]. Early enrollment ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events, as well as some back issues of the Ada User Journal [3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members. (Please indicate in the above-mentioned registration e-mail that you're interested, so we can bring enough copies.)

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form25.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/auj/home/

Informal lunch

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the lunch are also welcome: they can choose to join the organization or pay the sum of 25 Euros per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board [4]. See the separate official convocation for all details.

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/board/

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/25/250601-abga-conv.html

Ada Round-Table Discussion

As in recent years, we plan to keep the technical part of the Spring event informal as well. We will have a round-table discussion on Ada-related topics the participants would like to bring up. We invite everyone to briefly mention how they are using Ada in their work or non-work environment, and/or what kind of Ada-related activities they would like to embark on. We hope this might spark some concrete ideas for new activities and collaborations.

Directions

To permit this more interactive and social format, the event takes place at private premises in Leuven. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you!

Dirk Craeynest
President Ada-Belgium
Dirk.Craeynest@cs.kuleuven.be

Acknowledgements

We would like to thank our sponsors for their continued support of our activities: AdaCore, and KU Leuven (University of Leuven).

If you would also like to support Ada-Belgium, find out about the extra Ada-Belgium sponsorship benefits: http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html #sponsor

(V20250519.1)

Ada-related Resources

[Delta counts are interpolated* from April 1st to July 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
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 8_860** (-412) members [1]
- LinkedIn: 3_658 (+46) members [2]
- Stack Overflow: 2_443 (+5) questions [3]
- Ada-lang.io: 384 (+37) users [4]
- Gitter: 289 (+7) people [5]
- Telegram: 227 (+7) users [6]
- Discord: 192 (+1) users [7]
- Libera.Chat: 75 (=) concurrent users [8]

[*] This interpolation is intended to offer more representative and uniform delta counts from number to number of the Journal. The raw numbers at the actual date when they were taken can be found following the aforementioned link.

[**] 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

[7] 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
To: Ada User Journal readership

GitHub: >7_800* (=) repositories [1]
>1_000* (=) developers [1]

Alire: 1_426 (+64) releases [2]
559 (+23) crates [3]

Rosetta Code: 1_026 (+11) examples [4]
43 (+1) developers [5]

Sourceforge: 242 (=) projects [6]

Open Hub: 214 (=) projects [7]

Codelabs: 63 (+1) repositories [8]

Bitbucket: 37* (=) 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
To: Ada User Journal readership

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

- TIOBE Index: 12 (+8) 1.65% (40.00%) [1]
- PYPL Index: 15 (=) 1.52% (=) [2]
- Stack Overflow Survey: 40 (=) 0.9% (=) [3]
- IEEE Spectrum (trending): 46 (=) Score: 0.0023 (=) [4]
- IEEE Spectrum (general): 50 (=) Score: 0.0015 (=) [4]
- IEEE Spectrum (jobs): 54 (+1) Score: 0.0001 (1.82%) [4]
- Languish Trends: 154 (-8) 0.01% (-5.48%) [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/

Birth of Wiki Site "Ada 83 Memory"

From: VMo
Subject: Birth of Wiki Site "ada 83 Memory" at Http://www.ada83.org/
Date: Sat, 5 Apr 2025 10:36:38 +0000
Newsgroups: forum.ada-lang.io

```
with TEXT_IO;
procedure BIRTH_NOTIFICATION is
begin
  TEXT_IO.PUT_LINE("Ada 83 Memory
   wiki is started at http://www.ada83.org/");
end;
```

/Jelle Hermsen/ and I (/Vincent Morin/) found useful to create a wiki dedicated to Ada 83 to collect stories, technical elements about the ANSI/MIL-STD-1815 language (compilers, programming environments…) and its machine supports as iAPX or Rational.

Vintage atmosphere from the two rich decades from 1975 to 1995 can be evoked, as well as today's still relevance of the language in 83 standard.

This is a call to veterans of the language to be witnesses of the Ada language of the origins.

Feel free to have a look at the site embryo at http://www.ada83.org/
your contributions will be welcome!

Looking forward to read your stories and reflections.

GitHub-hosted Ada Reference Manual

From: sttaft
Subject: Github-hosted Ada Reference Manual
Date: Tue, 8 Apr 2025 16:03:39 +0000
Newsgroups: forum.ada-lang.io

Some folks have had trouble reaching ada-auth.org [1] on occasion. There is now a "mirror" of some of the Ada Reference Manuals on a GitHub-hosted site:

https://ada-rapporteur-group.github.io/ [2]

It is very reliable and very fast, so probably the better place to use for daily access.

[1] http://ada-auth.org

[2] https://ada-rapporteur-group.github.io

Ada Badge

From: LionelDraghi
Subject: Is There an Ada Badge like the Alire One?
Date: Wed, 16 Apr 2025 14:43:14 +0000
Newsgroups: forum.ada-lang.io

It seems to me that there is no badge to display that we are proud to use Ada on our Git [Hub|Lab] pages, like there is for Alire.

I found a collection for other languages here: Badges4-README.md-Profile [1], but nothing with our logo!

(That could be based on the AdaCore excellent and flexible proposal (Our Contribution to the Ada Logo Discussion | The AdaCore Blog [2])).

Did I miss something? Any suggestion?

BTW, for the "Ada crate of the year" awarded projects, feel free to reuse this one [3] with the right title and link.

[1] https://github.com/alexandresanlim/Badges4-README.md-Profile?tab=readme-ov-file#-languages-

[2] https://blog.adacore.com/our-contribution-to-the-ada-logo-discussion

[3] https://blog.adacore.com/ada-spark-crate-of-the-year-2024-winners-announced

From: mgrojo
Date: Thu, 17 Apr 2025 11:10:41 +0000

Good idea.

According to Logos | Shields.io [1], to add a logo to the badge, it seems the first step is adding it to https://simpleicons.org/ following Sign in to GitHub · GitHub [2]

I've created the request as a first step, but maybe it doesn't get solved unless someone contributes a pull request.

For the moment, we can experiment with the inline image: https://img.shields.io/badge/inside-green?style=for-the-badge&logo=[...]

[Removed large binary base64-encoded data –arm.]

I don't know how to get a bigger logo.

[1] https://shields.io/docs/logos

[2] https://github.com/simple-icons/simple-icons/issues/new?assignees=&labels=new+icon&template=icon_request.yml

[3] https://github.com/simple-icons/simple-icons/issues/13071

[GSOC] Working on Open Source Ada Projects and Getting Paid?

From: Irvise
Subject: [gsoc & Ada] Working on Open Source Ada Projects and Getting Paid?
Date: Sun, 11 May 2025 18:52:47 +0000
Newsgroups: forum.ada-lang.io

I am very happy to announce that @sttaft proposal to add the Ada 2022 parallel keyword to the GNAT frontend has been approved and will be funded if all the milestones are met!

Google Summer of Code [1] is a global program focused on bringing more developers into open source software development.

Congratulations to the participants!

[1] https://summerofcode.withgoogle.com/programs/2025/projects/Ex0T0sK7

From: sttaft
Date: Sun, 11 May 2025 19:59:18 +0000

Yes, thanks. Ethan Luis McDonough will be doing the work, and Richard Wai and I will be his mentors. It will be great to have these parallel features of Ada 2022 available in the FSF version of GNAT.

Books by Prof. Burns and Prof. Wellings

From: ornithophile
Subject: Books by Prof. Burns and Prof. Wellings
Date: Thu, 15 May 2025 18:34:28 +0000
Newsgroups: forum.ada-lang.io

Hi all, experienced programmer here trying to build my Ada knowledge. I just bought /Concurrent and Real-Time Programming in Ada/ (3rd edition) and then noticed they have a newer book, /Analysable Real-Time Systems: Programmed in Ada/. Has anyone here read both? Is it worth it to get the new book as well? I'd imagine the older book might be missing some Ada 2012 bits.

From: sidisyom
Date: Sun, 18 May 2025 21:40:28 +0000

I've read both and found them excellent sources with numerous code examples (though not always fully compilable if I remember right) of concurrency patterns and other things like fully functional servers (Polling/Defferable).

I think the difference is mostly that the latter covers a lot more thoroughly some aspects of scheduling theory and actually the first part of the book deals exclusively with that whilst the second part provides many code examples which are often similar to the ones found in the first book. Also, like you mention, the first one is Ada 2005 whilst the second one is 2012, so there may be some variation in the syntax.

So yeah, if you want to have more code examples, I'd go with the first one initially and then maybe move on to the second one for a more thorough treatment of real-time programming.

Ada Developing Environment

From: ocyou
Subject: Ada Environment
Date: Mon, 19 May 2025 13:19:40 +0000
Newsgroups: forum.ada-lang.io

From my understanding, Ada is like Prolog in the sense that it's an ISO standard, with multiple implementations/compilers.

Where can I find the list of Ada implementations and their build tools, package managers, etc?

I'm using for now Alire, which I find straightforward in building/running/managing crates, etc. I tried GNAT, but it seems more complex (make files, etc)

I'm just looking for the most efficient learning path. I want tools that spare me the pain of all what happens under the hood.

Also, in terms of docs, I'm confused: should I check *adacore docs or ada-lang or ada-auth*..? (normally, they should be ISO-compliant, but I just want to confirm…)

From: OneWingedShark
Date: Mon, 19 May 2025 15:53:10 +0000

ARG is the Ada Rapportaur Group, which produces/develops the standard.

The standards made available on Ada-auth and on AdaIC *are* the ISO standards, minus some formatting/styling imposed by the ISO.

Ada-Auth is the ARG's site.

AdaIC is the Ada Information Clearinghouse.

So both of these are official Ada.

AdaCore is a company, not official Ada.

From: DirkCraeynest
Date: Tue, 20 May 2025 07:27:18 +0000

> The standards made available on Ada-auth and on AdaIC *are* the ISO standards, minus some formatting/styling imposed by the ISO.

So strictly speaking what is on the Ada-auth and AdaIC sites are not the ISO standards, but correspond to the sources from which the successive ISO standards were derived.

FYI, for those who like to have a hard copy of Ada RMs: all successive versions have been printed by Springer in their LNCS series. The consolidated Ada 2022 RM is currently being prepared (2022 RM + Corrigendum 1), once again an initiative of Ada-Europe.

From: jere
Date: Mon, 19 May 2025 15:39:43 +0000

Alire is a package manager that wraps around GNAT and a program called GPRBuild, so if you are using GNAT you have 3 separate ways to build, with varying levels of ease and customization:

- gnatmake command - very bare bones, command line with switches
- gprbuild command - a build system that uses a special *.gpr file to tell gnatmake how to build with all of your options/switches but in a much more readable way that can be customized much more
- alr build command - Alire specific command that wraps around gprbuild that uses special *.toml files to tell Alire how to manage gprbuild files and options but also adds even more customization options on top of what gprbuild allows. It also gives you access to a plethora of libraries and examples.

Note that gnatmake and gprbuild are both products of a specific company that makes GNAT called AdaCore. Alire is separately maintained by members of the community.

All of that is for one Ada compiler. There are others out there. Here is a list of some options:

[1] https://github.com/ohenley/awesome-ada?tab=readme-ov-file#compilers

From: pmnw
Date: Mon, 19 May 2025 16:02:11 +0000

Yes, Ada is a standardised language and has been implemented by multiple vendors.

Nowadays, only GNAT seems to implement the latest Ada specification. Beyond GNAT, there are few closed-source compilers that provide older versions of Ada.

If you're interested in what is still available, search for PTC or Janus/Ada or Greenhills. There's also the free and amicable HAC.

Alire is the leading toolchain wrapper. For example, it uses the GNAT compiler and the GPRbuild build tool.

There's also AURA although I haven't seen it used.

ocyou:

> I'm using for now Alire…

This happens to be the simplest way to develop in Ada right now and was recommended by AdaCore (the corporate weight behind Ada) after they phased out the community version of their IDE called GNATStudio.

Please also check out ada_language_server.

[...]

Nvidia Publishes SPARK Process to Meet ISO-26262 Requirements

From: Irvise
Subject: Nvidia Publishes Spark Process to Meet Iso-26262 Requirements
Date: Wed, 4 Jun 2025 20:55:02 +0000
Newsgroups: forum.ada-lang.io

NVIDIA and AdaCore [1] have jointly worked on publishing the process to get an Ada/SPARK autonomous driving system to be ISO-26262 compliant.

Best part is, *NVIDIA open sourced their documents* for everybody to see (see link above)! The page with the rendered documentation can be found here [2].

Also, I would like to bring your attention to the fact that this is not the only area NVIDIA is working with Ada/SPARK. Cybersecurity is also one of their main focus points [3].

I believe it is very important to communicate and showcase these results/choices. People think that Ada is not used anymore; if it is used, it is in legacy systems; if it is used in legacy systems it is being replaced, etc. Well, here is a fresh, fresh, fresh, out of the oven, ripe and ready to be digested bit of news coming from the currently most valuable company (depending on market conditions) and it is not related to AI.

It would be nice if some of you reposted this info in HackerNews/Reddit/Phoronix/etc and proactively and kindly answered people's questions about Ada, its usage, etc. Also, it is very important to note that SPARK, GNAT, GNAT-LLVM and SPARKlib are all open source programs/libs under rather liberal licenses and that people can just get a hold of them and use them right now!

[1] https://www.adacore.com/press/ada-and-spark-enter-the-automotive-iso-26262-market-with-nvidia

[2] https://nvidia.github.io/spark-process/

[3] https://www.adacore.com/papers/nvidia-adoption-of-spark-new-era-in-security-critical-software-development

From: mgrojo
Date: Wed, 4 Jun 2025 21:47:46 +0000

Very good news. Til now, the fact that NVIDIA was using SPARK lacked a direct, undoubted reference from NVIDIA itself.

It's already on HackerNews.

[1] https://news.ycombinator.com/item?id=44184861

Ada-related Tools

Gprdeps: a New Tool for GPR Project Files Linting

From: ebriot
Subject: Gprdeps: a New Tool for Gpr Project Files Linting
Date: Tue, 1 Apr 2025 05:57:23 +0000
Newsgroups: forum.ada-lang.io

This tool is how I always thought gprbuild should have been implemented. I was part of the design discussions when AdaCore initially developed the GPR files, but we unfortunately chose to reuse the GNAT front-end for the parsing (for reasons still unclear to me, as the parser is really the simple part here). Having all dependency information in memory from the get-go, and not relying on external tools (the compiler) to discover the dependencies as we go opens the door for quite a number of things.

I remember we had an issue in the compiler at some point with the order of -l (libraries) on the linker command line. Having a graph of dependencies would have let us find a proper ordering with a simple topological sort for instance. One other idea is to easy add extra steps for generating sources (this is just going from source A to source B via code generation, then source B to object C via compilation for instance). Instead, gprbuild uses ad-hoc algorithm every time.

I have written the tool in Rust because I wanted to learn that language. It would just as easily have been written in Ada I am sure. I must say I was very surprised by the speed of the executable, which I attribute to compiler optimizations that are possible because of the lack of aliasing. But Ada would likely have been pretty close too.

Parallel For-loops in Ada

From: leo-brewin
Subject: Parallel For-loops in Ada
Date: Tue, 1 Apr 2025 08:27:15 +0000
Newsgroups: forum.ada-lang.io

I've created a GitHub repository describing my experiments in using Ada's tasking tools to run for-loops across multiple cpu cores. There are a handful of example codes using rendezvous calls and protected objects. I'm posting the link here as it might be useful for others wanting to play this game.

You can find the repository at github.com/leo-brewin/ada-parallel-for-loops [1].

You might also like to look at LWT, the Light Weight Threading library, from the ParaSail project. This is a much more serious attempt at this game than my little effort.

The LWT sources can be found here [2].

See also the announcement of LWT on the Ada-forum here:

Lightweight Parallelism library based on Ada 2022 features [3].

A copy of the LWT sources can also be found in my repo (under examples/lwt).

[1] https://github.com/leo-brewin/ada-parallel-for-loops

[2] https://github.com/parasail-lang/parasail/tree/main/lwt#light-weight-threading-library-for-ada-2022

[3] https://forum.ada-lang.io/t/lightweight-parallelism-library-based-on-ada-2022-features/516

From: sttaft
Date: Tue, 1 Apr 2025 12:47:36 +0000

Thanks for posting this! It's great to see this sort of use of parallelism.

Also thanks for the nice references to the LWT library. One goal of the LWT library is to avoid creating "heavy weight" tasks more frequently than necessary, but instead reusing them. Your timings show that creating and finalizing tasks is probably not a significant time sink, so long as you do enough work with them. The other goal of the LWT library was to support the parallel execution of a heterogeneous collection of light-weight threads. It sounds like that is not too important in your application area, since you are mainly focused on parallel loops, rather than something that might need a more general divide-and-conquer approach.

From: Irvise
Date: Tue, 1 Apr 2025 20:40:27 +0000

THANK YOU!

I have always been missing some benchmarks on Ada's tasking system and its (performance) behaviour! And also how it compares to other common parallelism models. LWT, afaik, uses OpenMP behind the scenes, which to me, is the parallelism standard (for a single CPU chip). It is great to see that tasks perform quite well! So thank you very very much for the tests

P.S.: as a LaTeX aficionado and I see you also use Cadabra, you just got a new follower.

From: sttaft
Date: Thu, 10 Apr 2025 01:20:56 +0000

Actually, LWT uses either OpenMP or a home-grown work-stealing scheduler (which is often faster than OpenMP). LWT has a "plug-in" architecture so you can insert other light-weight-thread schedulers underneath, by "with"ing the package that defines the scheduler. If you don't plug in any of them, LWT just runs the threads sequentially.

Archiving of GNATdashboard on GitHub

From: ch13
Subject: Question about the Archiving of the Gnatdashboard Project on Github
Date: Tue, 1 Apr 2025 16:15:23 +0000
Newsgroups: forum.ada-lang.io

I noticed that the gnatdashboard project [1] has been archived on GitHub since version 23 in April 2023. However, in my company, we continue to use version 25 of this product.

My question is: was this archiving in April 2023 due to a change in AdaCore's strategy? What are the reasons behind this decision?

Thank you in advance for your responses.

[1] https://github.com/AdaCore/gnatdashboard

From: Irvise
Date: Tue, 1 Apr 2025 20:28:32 +0000

Hi @ch13 and welcome to the forums!

@Fabien.C can probably provide a much nicer answer than I will, but still, I'll try.

From what I have talked with AdaCore people, they are focusing the open source efforts to what mostly benefits the community and does not interfere with their commercial offering (creating unwarranted expectations for example). From what I know, they have stopped releasing open updates to some products that are mainly only useful for commercial customers and products they would like to phase off in favour of others. Since your company probably has a GNATPro license, they will still get updates on the products, even if they are not publicly released.

I hope this helps!

P.S.: in case some people think AdaCore does not care about open source, read their blog and see how they changed their licenses from the GPL family to Apache and how their best products, such as GNAT and SPARK are fully open. Also, we the community get the features first, and no, that is not because we are used as guinea pigs. The new updates tend to be quite robust from the very beginning

From: ch13
Date: Wed, 2 Apr 2025 07:37:09 +0000

Hi @Irvise,

Thank you for your response. When you say that a product could be phased out in favor of others, do you mean that some tools will gradually be used less?

Currently, I am doing code quality analysis using GNAThub via the CodePeer plugin, which runs GNATSAS. Do you think this is still relevant? Or in the future, will we have to use GNATSAS directly or another tool?

That's a lot of questions

Thank you in advance.

From: Irvise
Date: Wed, 2 Apr 2025 21:34:47 +0000

An example of product transition is GNATpp (pretty print) being phased out in favour of GNATformat.

Regarding the other products you mention, they are for customers. I am not an AdaCore customer, so I cannot comment on them… Nonetheless, AdaCore makes their product line roadmap public, so you may find some more information there: https://docs.adacore.com/live/wave/roadmap/html/roadmap/index.html

LEA 0.92

From: zertovitch
Subject: Lea Version 0.92
Date: Wed, 2 Apr 2025 02:28:28 +0000
Newsgroups: forum.ada-lang.io

LEA is a Lightweight Editor for Ada

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

Source repository #1: https://github.com/zertovitch/lea

Source repository #2: https://sf.net/p/l-e-a/code/HEAD/tree/

Web site: http://l-e-a.sf.net/

Changes since last announcement here:

- Option box for HAC: you can choose the warnings & notes level
- Recognizes source path comments in HAC programs (source path is appended during build)
- Added 5 HAC examples (available through menu: Action / Code sample)

The LEA executable is standalone, holds in 4 MB and doesn't require an installation procedure.

Features:

- Multi-document
- Multiple undo's & redo's
- Multi-line & multi-point edit, rectangular selections
- Color themes, easy to switch
- Duplication of lines and selections
- Syntax highlighting
- Parenthesis matching
- Bookmarks
- Includes HAC, the HAC Ada Compiler
- Smart editor features (auto-completion, navigation, mouse-hover infos)
- Includes numerous examples of Ada programs, ready to be run
- Single executable, runs without installation
- Free, Open-Source

LEA is programmed in Ada, as well as HAC.

AZip 2.7.1

From: zertovitch
Subject: Azip Version 2.7.1
Date: Sat, 5 Apr 2025 05:46:28 +0000
Newsgroups: forum.ada-lang.io

AZip is a Zip archive manager.

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

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

Project site 2:
http://sourceforge.net/projects/azip

Home page: http://azip.sf.net/

Latest changes:

- Added option to make a backup for Update or Recompression operations
- Added brute-force Recompression mode

Some features of AZip:

- Aimed at simplicity, minimal option set
- Usual archive management: Add / Remove / Extract files
- Original, useful built-in tools:
- Text & name search function through an archive (no file is written during search)
- Archive updater
- Archive recompression, using brute force or an algorithm-picking approach for improving a Zip archive's compression.
- Flat view / Tree view toggle
- Multi-document
- Encryption
- Zip compression formats supported: Reduce, Shrink, Implode, Deflate, Deflate64, BZip2, LZMA
- Portable (no installation needed, no DLL); can operate with a config file instead of the registry (stealth mode)
- Installable if desired (application is its own installer)
- Free, open-source

The AZip executable is standalone, holds in 3 MB and doesn't require an installation procedure.

However, you can install AZip: the executable is its own installer and will offer to install itself.

"Under the hood" features:

- /AZip/ is from /A/ to /Z/ programmed in Ada
- Uses the highly portable Zip-Ada library - all in Ada.
- (regarding Windows "skin") Uses the GWindows library - all in Ada.

AdaStudio 2025.2 Free Edition

From: AdaStudio
Subject: Announce: Adastudio-2025.2 Release 02/04/2025 Free Edition
Date: Thu, 10 Apr 2025 10:32:03 +0000
Newsgroups: forum.ada-lang.io

AdaStudio-2025.2 release 02/04/2025 free edition based on Qt-6.9.0-everywhere opensource (expanded with modules from Qt-5.15: qtgamepad, qtx11extras, qtwinextras), VTK-9.4.1, FFMPEG-7.1, OpenCV-4.11.0, Whisper-1.7.4, SDL2-2.24.0, QtAV-1.13 MDK-SDK(wang-bin)

Qt6ada version 6.9.0 open source and qt6base.dll, qt6ext.dll (win64), libqt6base.so, libqt6txt.so(x86-64) built with Microsoft Visual Studio 2023 x64 Windows, GCC amd64 in Linux.

Package tested with GNAT gpl 2020 Ada compiler in Windows 64bit, Linux amd64 Debian 12.2.

AdaStudio-2025.2 includes next modules: qt6ada, vtkada, qt6mdkada, qt6cvada (face recognition, face detection, face identification, objects detection, QRcode detector, BARcode detection and others) and voice recognizer.

Qt6Ada is built under GNU LGPLv3 license.

Qt6Ada modules for Windows, Linux (Unix) are available from:
https://drive.google.com/drive/folders/0B2QuZLoe-yiPbmNQRl83M1dTRVE?resourcekey=0-b-M35gZhynB6-LOQww33Tg&usp=sharing

WebPage is
https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/adastudio.html

GCC v15.1 Ada Changelog

From: Irvise
Subject: Gcc v15.1 Ada Changelog
Date: Fri, 11 Apr 2025 12:55:37 +0000
Newsgroups: forum.ada-lang.io

GCC 15.1 should be released in the next few weeks once all the main bugs have been squashed. Nonetheless, the changes that have taken place for GNAT are already available in the Changelog page for GCC. There are quite a few nice things that I am personally very excited about!

https://gcc.gnu.org/gcc-15/changes.html

From: Lucretia
Date: Fri, 25 Apr 2025 13:18:49 +0000

Interesting…they added gcobol.

SweetAda Implements Some of Its Libc in Ada

From: Irvise
Subject: Sweetada Just Started to Implement Some of Its Libc in Ada
Date: Sun, 13 Apr 2025 20:31:14 +0000
Newsgroups: forum.ada-lang.io

I was informed recently that the SweetAda project [1], whose sole author is @SweetAda, has started to port some of its homegrown LibC implementation to Ada! You can find the c_wrapper.ad[s | b] files in the source tree and take a look at them [2]!

Remember, if you would like to help, you can always contribute by testing the code, making patches and fixes, writing documentation or supporting the authors!

My personal congrats to Gabriele! Best regards,

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

[2] https://github.com/gabriele-galeotti/SweetAda/blob/d0d9da3605e269df817d7ded415422c190769648/clibrary/c_wrappers.ads

Ada Bootstrap Compiler Funding Thanks to Nlnet!

From: Irvise
Subject: Ada Bootstrap Compiler Funding Thanks to Nlnet!
Date: Wed, 23 Apr 2025 16:09:02 +0000
Newsgroups: forum.ada-lang.io

The nlnet foundation has announced the funding grant for an Ada Bootstrap compiler [1]!

I am a bit surprised by this news as I was involved in some behind the scenes conversations about it but I was not expecting it to pop up as soon as this! I will try to gather more information and see if there is a candidate selected or if it is open to the public to select a candidate.

[1] https://nlnet.nl/news/2025/20250422-announcement-grants-CommonsFund.html

From: OneWingedShark
Date: Wed, 23 Apr 2025 18:48:56 +0000

> Ada is an important computer language with a long history, with the compilers being built for new architectures in an ad-hoc basis based on previously existing Ada compilers from other architectures. *This project aims to create a bootstrap path from the C language to an Ada compiler without relying on an existing Ada compiler binary.* This will allow us to have a fully auditable trail from C to a working Ada compiler, removing concerns about hidden backdoors or other issues that may arise from using a compiler without a clear bootstrap path.

I have three reactions:

- A bootstrap implementation, yay!
- Not using existing tech to leverage the bootstrap process; disappointing, but understandable.
- Using C… **sigh** (though it is auditable, yay-ish… that should be a self-hosted feature.)

IMO, using self-hosted SPARK-verified Ada (employing Annex H), to self-host the compiler, having a modular backend/codegen would be a better route: aside from the compiler, create two subprojects: (1) a SeedForth [small-wordset, tokenized Forth] interpreter, SPARK verified; and (2) a SPARK-verified SeedForth emitting backend. — You can then feed things into each other so that you get the ability to interpret Ada on a new architecture implementing about 30 Forth-words, with provably correct interpreter and codegen. Native-code generation then becomes implementing a new backend, and you can use the previous results to generate a native-code compiler.

From: Irvise
Date: Sat, 26 Apr 2025 14:13:48 +0000

I will try to shed some light on some of the topics raised.

> Not using existing tech to leverage the bootstrap process; disappointing, but understandable.

It is using the available tech… in the constrained environment in which this is taking place

The thing is, this is no normal bootstrap compiler, where one can pick and use any available technology to carry such a task. The framework for this project is the live-bootstrap [1] system. This is a project that is trying as hard as possible to create a root-of-trust from where the world can be rebuilt in a bit-by-bit reproducible and transparent manner, in a way that a human being could verify the process and each and every single line of code. And this is done with the absolute bare minimum, which means that they are starting from a tiny binary seed to bootstrap all the way to modern Linux, GCC and Guile.

> Using C… /*sigh*/ (though it is auditable, yay-ish… that should be a self-hosted feature.)

C and Scheme are some of the first languages bootstrapped in the current live-bootstrap system, C being the more complete and capable of the set (the Scheme they implemented is incredibly small and it is only used to create a basic LibC and barebones C compiler). We would like to have Ada bootstrapped as early as possible, so C is the ideal system. However, other ideas have been tossed around, such as Go, a more advanced Scheme (TR7 or Guile), Python or Rust for example.

[...]

[1] https://github.com/fosslinux/live-bootstrap/blob/2057d551e0d072f85dd3c8b046e90e6be81a3604/parts.rst

Open Source Ada 83 Compilers

From: VMo
Subject: Discussions about Open Source Ada Compilers
Date: Sat, 10 May 2025 12:04:38 +0000
Newsgroups: forum.ada-lang.io

Rather than diverging on the thread on GNAT bootstrapping nlnet project when answering OneWingedShark message, I prefer opening an appropriate new thread.

[...]

From: VMo
Date: Sat, 10 May 2025 15:20:01 +0000

More directly in line with the topic title:

What do we have at hand in relation with Ada open source compiler and what can we do which would be a useful community achievement.

First, I set GNAT apart. GNAT is a wonderfully useful tool, up to date with recent Ada revisions, its source code can be looked at, but it is only manageable with a team of first class engineers. AdaCore does an admirable job, this is not our play court.

Briefly, what are the other compilers whose source code is available, what are their characteristics what can we do with them.

Ada/Ed

Ada/Ed as everybody knows is an early demonstrator Ada 83 compiler written in C, targets an interpreter and lacks very useful features as record representation clauses which are basic to microcontroller Ada programming (for example the AVR map is conveniently represented with such a record).

Working with Ada/Ed source code is a pain for non-C programmers. The high-level representation of Ada/Ed is not DIANA (which is not that easy to use, but is relatively clear, documented and used both in the Ada 83 Peregrine source code and the Pascal IIPS module of M.Ciernak). Ada/Ed can surely be recompiled with modern C compilers with some modest editing. I have done it for the interpreter being curious of its byte code and runtime.

But I doubt that Ada/Ed will be really useful outside of its initial vocation which was research and education being the SETL specification descendant.

SmallAda and its descendant HAC

Gautier de Montmollin has done an admirable job over a decade and intensely over the five last years, when recovering the SmallAda system written in Pascal. HAC is an operational system able to execute some serious programs, chapeau for this work. Nonetheless some choices have been made which will hinder or preclude some desirable features. HAC is not written in Ada 83, but at least Ada 95 using child packages. I don't know enough of the HAC source to tell if some Ada 2X features are used. On the other hand, HAC compiles a subset of Ada 83, SmallAda was a vehicle for tasking research and not destined to be a full Ada 83 compiler.

That means that HAC cannot compile itself, and should be augmented as far as being able to compile its Ada 9X or later 2X if necessary. This is probably not the goal of the author.

The Peregrine Ada 83 / DIANA and its descendant (TLALOC ?)

From the Walnut distribution, I extracted an Ada 83 system written for DEC Ada by Peregrine Systems directed by Bill Easton. As it was somewhat strangely structured (there was an unclear mix of three usage of IDL for IDL itself, LALR and DIANA management) I did my best to clarify the affair and modified the system's structure, virtual pages pointer management and many other things. Thanks to GNAT, that must be said.

This system takes full Ada 83 source text and produces a DIANA high level intermediate representation of it.

As far as I tested it, it works well but for some rudimentary library management and care to manually compile modules in the correct order.

This system also swallowed the Kalinda OS code, and if somebody tries other Ada 83 source text (which should be done), I am confident it will produce the corresponding DIANA.

Then I decided to follow the polish IIPS project and began to browse the DIANA IR in the code_gen procedure to produce low level IR and final code. I think I'll call this whole system TLALOC.

Being written itself in rigorous Ada 83, TLALOC compiles itself without error and so produces all the DIANA structure of the compiler, which is already something. If a successful DIANA to low level IR code generator can be produced, this compilation system can become the only Ada 83 open source compiler which compiles itself (Ada/Ed cannot, being in C, HAC either being in 9X).

Ada 83 being a fixed standard, there is minimal risk of sideslipping from version to version as happens in GNAT. Only avoid inventing pragmas… Stick to the 83-LRM.

I rewrote the readme and markdown introductions in English on the Framagit with some clickable diagrams to access source code. Hoping it will encourage reading.

recall, the thing is here: Vincent Morin / Ada-83-compiler-tools · GitLab [1]

Is there anything else?

[1] https://framagit.org/VMo/ada-83-compiler-tools

From: sttaft
Date: Sat, 10 May 2025 18:37:31 +0000

The open-source ParaSail interpreter/compiler (all written in Ada) has a partial Ada 2022 front end. See parasail/ada202x_parser at [1] for the parser/lexer.

Internal tree representation, and static and dynamic analysis phases, are all in parasail/semantics at [2]

Interpreter/run-time is in parasail/interpreter at [3]

Translator from ParaSail VM (PSVM) to LLVM is in parasail/lib [4] in (ParaSail) source files such as compiler.{psi,psl}

[1] https://github.com/parasail-lang/parasail/tree/main/ada202x_parser

[2] https://github.com/parasail-lang/parasail/tree/main/semantics

[3] https://github.com/parasail-lang/parasail/tree/main/interpreter

[4] https://github.com/parasail-lang/parasail/tree/main/lib

From: waleedmebane
Date: Sun, 11 May 2025 02:47:47 +0000

Wow, that's impressive! (I read parasail_intro.pdf.) I wasn't aware of ParaSail as a new programming language though I've seen you mention ParaSail in posts in the context of OpenMP support and parallel for loops.

From: VMo
Date: Sun, 11 May 2025 07:19:23 +0000

Well, extremely interesting but somewhat stunning, I have no voice when considering the amount of work… The GitHub mentions only 3 contributors, I suppose there is some team behind. It is the work of how many people?

[...]

From: sttaft
Date: Sun, 11 May 2025 12:15:59 +0000

If you want to read most of the story of how ParaSail was designed, I would encourage you to take a look at the Blog which was started in 2009 and traced the week-by-week progress (Designing ParaSail, a new programming language: Why design a new programming language? [1]).

Having spent most of my career involved with the design and implementation of Ada, in 2009 I decided to start from scratch and design a stripped-down language focused on supporting multicore hardware, with high integrity. There was no "team" behind ParaSail, though it is based on things that were learned from collaborative work done over 40 years.

[1] https://parasail-programming-language.blogspot.com/2009/09/why-design-new-programming-language.html

From: VMo
Date: Mon, 12 May 2025 08:58:13 +0000

Thank you! I read the blog integrally. It is full of very interesting thoughts. Parasail is a leading edge research vehicle, somewhat far from my Ada 83 preservation and revival occupations.

But the modern and the ancient have some continuity.

For example I wondered why Ada 83 appeared to me to be such a peculiar software development when I was a young student and researcher; and why Ada 95 conveyed a more conventional feeling. I realized later that Ada 83 is a structure forcing language (to the point of being sometimes heavy on the software developer) and in fact discourages the use of pointers; of course there are access types, but much can be done without. On the contrary Ada 95 led the programmer to instinctively use aliased and access things and was much more conducive to pointers use. I'll say a horrible thing, but Ada 95 had a kind of C facility for the developer. So the "no pointer" policy of ParaSail echoes the "pointer set aside" encouragement of Ada 83.

Also, the "growing/shrinking" objects idea echoes the Ada 83 memory manager of my Kalinda OS trials. Kalinda's memory package is an Ada 83 (and at first Ada 95, after Pascal and shortly C, nobody's perfect) adaptation of the Macintosh memory manager and its double indirection handles. Ada forced me to implement a concept of dynamic size array with header to replace traditional pointer access to handles' contents. Pointers disappear, arrays are used. The same idea appears in ParaSail.

So thank you for all those intellectually vibrant ideas, and hoping good wind for ParaSail! (we are on the shore here in Brest, sailing evocates familiar pictures)

From: LionelDraghi
Date: Sun, 11 May 2025 20:05:12 +0000

This weekend, while clearing out the house where I grew up, I came across my papers from my final year at university. I think it was in 1989. I found articles and general documentation.

For example, the documentation for the Meridian compiler that we used at the university. Articles about the Alsys compiler (which I later used extensively in the industry), or about the amazing Rational system.

These Ada 83 systems were good and mature.

But they no longer exist, and haven't for years (even decades), and there's probably no more economic interest behind them.

I don't know who owns those software, but maybe we could ask for the current owners to release the sources under a free software licence, and give those marvelous pieces of code a second life. It would be a much easier starting point.

I regularly use FreeCAD. Its geometric kernel comes from Euclid, a software acquired by its competitor CATIA, and whose development ceased in 1998. Then, after some twists and turns, it was "liberated" and is now used in many open-source software and by many research projects.

Open Cascade Technology (OCCT, formerly named CAS.CADE) is an object-oriented C++ class library for 3D computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), etc. It is developed and supported by Open Cascade SAS company. It is free and open-source software released under the GNU Lesser General Public License (LGPL), version 2.1 only, which permits open source and proprietary uses. OCCT is a full-scale boundary representation (B-rep) modeling toolkit...

It would be great if the Alsys compiler could experience a similar resurrection.

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

From: VMo
Date: Mon, 12 May 2025 08:22:11 +0000

This is obviously a good idea. The difficulty will be to find contacts, time has already passed, some people are retired or even passed away, the history of firms like Alsys is complicated between Thomson, Aonix, PCT, Atego. Are there any Ada 83 compiler sources left? Where? Who could have a hand on them?

If some of us have relations or contact indications, it would be welcome.

I contacted RRSoftware asking if they would be interested in participating in an Ada 83 technology preservation action. We'll see if there is an answer.

I also contacted the Karlsruhe university (KIT) Institut für Programstrukturen und Datenorganisation (IPD), I remember some articles from Rudolf Landwehr who is at Atos and affiliated to KIT. They have done very interesting work, could it be that some source be disclosed?

Let's try…

From: zertovitch
Date: Mon, 12 May 2025 09:18:40 +0000

> It would be great if the Alsys compiler could experience a similar resurrection.

The Alsys compiler is not dead, it changed its name to ObjectAda and is doing well.

You will notice .prj files in some open-source projects, including mine, which are project files for the ObjectAda IDE.

From: LionelDraghi
Date: Mon, 12 May 2025 10:21:29 +0000

As far as I remember, when switching to Ada 95, Alsys bought the technology, and dropped the in house technology.

I remember the ObjectAda interface being very different from the old Adaworld interface. I'm not sure if there is a single line of code shared between the two families (maybe for some backend?).

This is why I think the Ada83 code could be released.

But I may be wrong!

From: VMo
Date: Mon, 12 May 2025 11:15:42 +0000

I should have been more precise in the topic title and state that the interest is on open source *pure Ada 83* compilers.

Ada 9X and 2X is still sensitive technology (so more as we get close to our epoch) and having access to those compiler source code is improbable. It should not be the case for pure Ada 83 which is normally considered as obsolete technology.

If there was an Alsys pure Ada 83 compiler, it would be interesting to have access to its source code. I always used Object Ada in the past, and never used a pure Ada 83 Alsys compiler.

Anyway, I am convinced that it is time and perhaps urgent to preserve Ada 83 technology artifacts both for computer history and give the occasion to the young to have a hand on rather remarkable software engineering achievements.

I am reading the documents on the Rational R1000s400 restarting at DataMuseum.dk. They have produced an emulator starting from schematics… It is extremely interesting, and from some points of view still modern. Reading the patents is also instructive even in 2025.

For now I see three possible paths to Ada 83 compiler source code:

- Janus Ada 83 which was for i386 DOS and Unix
- Karlsruhe compiler which probably targeted a Siemens machine and M68K (to verify)
- and Alsys (targets?).

If somebody has other ideas or personal contacts, feel free to participate in the context of *Ada 83 Memory* wiki (ada83.org [1]).

The link [2] has a section "ancient".

It mentions Alsys AdaWorld, Verdix VADS, Meridian.

That is where we should try to convince some people to disclose and save old Ada 83 only compilers. There should not be any technological or commercial risk for them to do so.

[1] http://ada83.org

[2] https://www.adaforge.org/DevTools/DevToolsBuild/Compilers/

CoAP-SPARK 0.9.0

From: mgrojo
Subject: Coap-spark 0.9.0 Released
Date: Tue, 20 May 2025 21:31:28 +0000
Newsgroups: forum.ada-lang.io

CoAP-SPARK [1] is a library implementing the Constrained Application Protocol (CoAP) as defined in RFC 7252 [2], developed in SPARK and verified to be free of run-time errors (silver level).

This version implements the client side of the protocol with some limitations:

- It does not support block-wise transfers (extension to RFC 7252).
- It does not support retransmission of messages.
- It only supports NoSec and PreSharedKey security modes.

An example and tool implementing a command-line CoAP client can be found in the coap_client [3] crate.

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

[2] https://www.rfc-editor.org/rfc/rfc7252

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

From: mgrojo
Date: Tue, 20 May 2025 21:39:18 +0000

Documentation of the project (my Master's Thesis draft) can be found in https://github.com/mgrojo/coap_spark/releases/download/v0.9.0/Memoria_TFM_CoAP-SPARK-Spanish.pdf (sorry, but only in Spanish). Any comment or feedback is appreciated.

Globe_3D 2025-04-18

From: zertovitch
Subject: Globe_3D 2025-04-18
Date: Thu, 22 May 2025 01:26:29 +0000
Newsgroups: forum.ada-lang.io

GLOBE_3D is a real-time GL Object Based 3D Engine written in Ada.

Alire crate:
https://alire.ada.dev/crates/globe_3d

Project site 1:
https://github.com/zertovitch/globe-3d/

Project site 2:
https://sourceforge.net/projects/globe3d/

Home page:
https://globe3d.sourceforge.io/

Latest changes:

- Code cleanup
- Visualisation of portals (on show_portals = True)

Ironclad Kernel 0.7.0

From: dragon-spirit-wtp
Subject: New 0.7.0 Release of Ironclad Kernel
Date: Sat, 31 May 2025 06:05:33 +0000
Newsgroups: forum.ada-lang.io

https://codeberg.org/Ironclad/Ironclad/releases/tag/v0.7.0

Non-breaking changes

- Greatly improved block device (SATA, NVMe) caching. Leading to big performance improvements.
- Added a basic NVMe driver.
- Implement support for x86's SMAP, and expose it in an architecture-independent way.
- Harden...

GNAT FSF 15

From: csagaert
Subject: Gnat Fsf 15 Released
Date: Thu, 5 Jun 2025 09:13:55 +0000
Newsgroups: forum.ada-lang.io

We released GNAT FSF 15 on Alire yesterday. This release is based on GCC 15.1.0; the release notes for Ada can be found here [1].

There are prebuilt binaries for Linux (x86_64 and aarch64), Windows (x86_64) and macOS (x86_64 and aarch64). Please note that the binaries for macOS aarch64 are based on an unofficial branch of GCC that has not yet released a final version for GCC 15, so there may be some issues that are absent from other platforms.

Cross compilers are also available on all platforms for the following targets: arm-elf, riscv64-elf, avr-elf and xtensa-esp32-elf.

You can report issues with the released compilers on this repository [2].

We want to thank @simonjwright for his work on building GNAT for macOS. His work helped us greatly to provide functional compilers for the platform. He will be sorely missed.

[1] https://gcc.gnu.org/gcc-15/changes.html#ada

[2] https://github.com/alire-project/GNAT-FSF-builds

Filoject, a Dependency Injection Framework for Ada

From: rehartmann
Subject: Filoject, a Dependency Injection Framework for Ada
Date: Sun, 22 Jun 2025 10:32:24 +0000
Newsgroups: forum.ada-lang.io

Filoject [1], a dependency injection framework for Ada, is now available.

What is this good for, you may ask?

I'm not claiming that Ada needs a DI framework, nor that it may even be a good idea to use one in Ada projects.

But given that practically all programming languages competing with Ada have DI frameworks, this may be something which is worth experimenting

with. Filoject is meant to provide a basis for this experimentation.

Filoject uses a code generator, a custom pragma and a custom aspect.

[1] https://github.com/rehartmann/filoject

Simple Components 4.74

From: dmitry-kazakov
Subject: Simple Components 4.74
Date: Mon, 30 Jun 2025 10:48:45 +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.

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

The focus of this release is an implementation of arbitrary precision rational numbers. The difference to standard library Ada.Numerics.Big_Numbers.Big_Reals:

- No limits, except for the pool size;
- String edit packages representing a number in a usual form (with the specified accuracy) rather than as a numerator/denominator ratio;
- An equivalent of Ada.Numerics.Elementary_Functions with rational approximations;
- Simple continued fractions support.

Changes to the previous version:

- The package Unbounded_Rationals provides an implementation of arbitrary precision rational numbers arithmetic;
- The package Unbounded_Rationals.Elementary_Functions provides approximations in rational numbers of some elementary, trigonometric and hyperbolic functions;
- The package Strings_Edit.Unbounded_Rational_Edit provides string editing for arbitrary precision rational numbers;
- The package Strings_Edit.Unbounded_Unsigned_Edit was optimized for better performance;
- Mixed Unsigned_Integer to Integer operations were added to the package Unsigned_Integers;
- The package Unbounded_Rationals.Continued_Fractions provides an implementation of simple continued fractions;
- The package Strings_Edit.Continued_Fraction_Edit provides string editing for simple continued fractions;
- Bug fixes in List_SetItem Python bindings;
- Check_Error message in Python binding was improved, trace back output added;
- Added higher level operations to create Python modules after initialization of the Python interpreter;
- Added higher level operations to create Python classes AKA heap types after initialization;
- An example of higher level module creation was provided;
- An example of higher level class creation was provided;
- IsInitialized function was added to Python bindings;
- OSX project settings fixed.

Ada-related Products

ObjectAda 10.6

From: JC001
Subject: Objectada 10.6 Released
Date: Wed, 21 May 2025 16:49:03 +0000
Newsgroups: forum.ada-lang.io

PTC announced the release of ObjectAda 10.6 [1] on 2025-03-03.

Those with a documented history (such as on-line repositories) of developing Ada open-source projects may obtain a free copy of ObjectAda by writing directly to Shawn Fanning, Software Development Director, user name sfanning at the domain ptc.com.

[1] https://ptc-p-001.sitecorecontenthub.cloud/api/public/content/f00b267f47f8426c9e9e41b01d5c4bbf

References to Publications

Article: Nvidia Drives Ada and SPARK into Driverless Cars

From: dragon-spirit-wtp
Subject: Article: Nvidia Drives Ada and Spark into Driverless Cars
Date: Fri, 6 Jun 2025 06:50:49 +0000
Newsgroups: forum.ada-lang.io

This is a great article that highlights various virtues of developing with Ada and SPARK that contributed to NVIDIA's recent achievement of certifying their DriveOS automotive operating system to the /highest/ automotive level of safety, ASIL-D. The first ever!

eeNews Europe – 4 Jun 25 [1]

Nvidia drives Ada and SPARK into driverless cars [1]

AdaCore and Nvidia have developed an open source reference flow for the Ada and SPARK formal languages in driverless cars.

Est. reading time: 5 minutes

Here is a paragraph from the article:

"Adopting a new programming language involves deploying a new environment, training teams to a new formalism, adapting programming patterns and many other issues. However, from a process standpoint, programming languages are vastly interchangeable, but Ada and SPARK is a different story."

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

Ada and Other Languages

Unexpected Ada Advocacy

From: LionelDraghi
Subject: Unexpected Ada Advocacy
Date: Tue, 1 Apr 2025 20:22:30 +0000
Newsgroups: forum.ada-lang.io

I was watching a video of a PyCon 2018 conference talk by Hillel Wayne titled Beyond Unit Tests: Taking Your Testing to the Next Level [1]

He was starting a "Design by contract" part of the speech (about min 18), and said:

> It turns out this is actually one of the most powerful verification techniques we know about.
> We've seen it work empirically at scale in all sorts of places but these places have mostly been avionics critical systems defense academia so nobody's heard of this thing which honestly I find completely crazy.
> I mean here's a question how many of you've heard of Ada?
> How many of you actually looked into what Ada can do?
> OK I've actually done some experimentation: you can do incredible things with Ada, but because it's Ada nobody cares.

Couldn't summarize the situation any better

And then later (around min 28), about Formal Method:

> This is the study of how we can prove software is correct at a level not seen in any kind of modern-day production language.

> Nothing really approach this level [of what] we can do with formal methods

not actually yet Ada can do it never mind!

Kudos to Hillel Wayne, someone who talks about Ada after actually trying it.

Useless to say I clicked the thumbs up!

[1] https://www.youtube.com/watch?=MYucYon2-lk

Ada Back in the TIOBE Top 20 (March 2025)

From: karlnyberg
Subject: Ada Back in the TIOBE Top 20 (march 2025)
Date: Wed, 2 Apr 2025 19:07:57 +0000
Newsgroups: forum.ada-lang.io

Dinosaurs indeed…

There be DRAGONS!

https://www.tiobe.com/tiobe-index/

From: zertovitch
Date: Wed, 2 Apr 2025 21:56:32 +0000

…and in the Top 15 in the PYPL index [1], possibly for the first time.

What's happening?

What is Bill Gates doing?!

[1] https://pypl.github.io/PYPL.html

From: pyj
Date: Sat, 5 Apr 2025 01:12:02 +0000

It does seem like the outreach has improved, but TIOBE doesn't make sense to me. I have a hard time believing C++ has a rating 3-4 times higher than JavaScript and 50 times higher than TypeScript. If you look at job boards or GitHub that seems inverted.

Languish [1] looks at the number of GitHub issues and GitHub pull requests which makes more sense, and Ada is 155 (it was 163 in 2022).

Looking at GitHub's recently updated [2] to see the number of projects:

• Zig: 5,600

• Pascal: 47,900

• Rust: 396,000

• Go: 1,100,000

• C++: 3,600,000

• Typescript: 4,200,000

• Javascript: 22,300,000

• Ada: 4,600

Don't worry about the charts and just have a good time writing cool projects.

[1] https://tjpalmer.github.io/languish/#y=mean&weights=issues%3D1%26pulls%3D1%26stars%3D0%26soQuestions%3D0&names=ada

[2] https://github.com/search?q=language%3AAda+&type=repositories&s=updated&o=desc

Ada Practice

Dimensional Analysis

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Subject: Dimensional Analysis
Date: Sat, 5 Apr 2025 07:17:50 +0000
Newsgroups: comp.lang.ada

From time to time, I come across ideas for introducing additional checking of operands in expressions that go beyond mere types, that try to take into account the actual dimensions of the units involved. The intention is that this will reduce the likelihood of expressions that, while they may be correct in conventional type-compatibility terms (all the numbers might be of compatible floating types), can be flagged if they fail a dimensional-analysis check.

I don't think it's practical to introduce new types for every single unit you want to distinguish, since that would lead to a combinatorial explosion in the necessary conversion operators. It's far more natural to build up the units in terms of basic primitives like the SI base units, and allow derived units to be expressed in terms of multiplication and division of (powers of) the base units.

Some examples, in pseudocode:

```
type velocity is units(metres / seconds);
type energy is units(kilograms * metres *
    metres * seconds / seconds);
kinetic_energy : energy;
particle_velocity : velocity;
particle_mass : units(kilograms);
```

Now suppose you have an expression like

```
kinetic_energy := 0.5 * particle_mass *
particle_velocity ** 3;
```

you can immediately see that is wrong, because the dimensions in the source and destination of the assignment don't match up. That power 3 is probably a typo, and should be a 2:

```
kinetic_energy := 0.5 * particle_mass *
particle_velocity ** 2;
```

Of course, this won't catch errors like multiplication/division by incorrect constant factors. Dimensional analysis is necessary for an expression to make physical sense, but it is not sufficient.

At first sight, this can seem useful. But you have to be careful not to carry it too far. For example, one scheme I saw wanted to specify separate units for angles so that the arguments and results of trig functions would be expressed in incompatible units. That I didn't think was a good idea.

So what do you think of this idea? Is it already in use in a significant way in any Ada code? I figured if any language would be looking at new ways of writing safer code, it would be the Ada crowd.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 5 Apr 2025 08:20:14 +0000

> So what do you think of this idea?

Static dimensionality checks are implemented by the GNAT compiler, see

https://gcc.gnu.org/onlinedocs/gnat_ugn/Performing-Dimensionality-Analysis-in-GNAT.html

For engineering purposes static checks are worthless. So in practice constrained types and dynamic checks are used instead. See

https://www.dmitry-kazakov.de/ada/units.htm

Atomic Increment Benchmarks

From: dmitry-kazakov
Subject: Atomic Increment Benchmarks
Date: Thu, 17 Apr 2025 09:03:03 +0000
Newsgroups: forum.ada-lang.io

I wrote a small benchmark program to compare three methods of implementing atomic increment in Ada:

• Classic Ada protected object

• System.Atomic_Operations.Integer_Arithmetic (since Ada 2022)

• GCC built-in

Here is the code:

```
pragma Ada_2022;

with Ada.Calendar; use Ada.Calendar;
with Ada.Text_IO;  use Ada.Text_IO;
with Interfaces.C; use Interfaces.C;

with System.Atomic_Operations.
Integer_Arithmetic;

procedure Lock_Free_Benchmark is

   function Add
         (  Ptr : access Integer;
            Val : Integer;
            Memorder : int := 0
         ) return Integer;
   pragma Import (Intrinsic, Add,
   "__atomic_add_fetch_4");

   protected Protected_Integer is
      procedure Increment;
      function Get return Integer;
   private
      Value : Integer := 0;
   end Protected_Integer;

   protected body Protected_Integer is
      procedure Increment is
      begin
         Value := Value + 1;
      end Increment;
      function Get return Integer is
      begin
         return Value;
      end Get;
   end Protected_Integer;

   type Atomic_Integer is new Integer with
   Atomic;
```

```
   package Atomic_Integers is
      new System.Atomic_Operations.
      Integer_Arithmetic (Atomic_Integer);
   use Atomic_Integers;

   Start : Time;
   T1, T2, T3 : Duration;
   X : aliased Atomic_Integer := 0;
   Y, Z : aliased Integer := 0;
   Times : constant := 100_000;

   procedure Report (T : Duration;
   Text : String) is
   begin
      Put (Text);
      Put (Duration'Image (T));
      Put (Integer'Image (Integer
      ((T * 1000_000_000) / Times)));
      Put ("ns ");
      New_Line;
   end Report;

begin
   Start := Clock;
   for I in 1..Times loop
      Protected_Integer.Increment;
   end loop;
   T1 := Clock - Start;

   Start := Clock;
   for I in 1..Times loop
      Atomic_Add (X, 1);
   end loop;
   T2 := Clock - Start;

   Start := Clock;
   for I in 1..Times loop
      Z := Add (Y'Access, 1);
   end loop;
   T3 := Clock - Start;

   Put_Line ("Method          Total       Time");
   Report (T1, "protected object: ");
   Report (T2, "atomic integer:   ");
   Report (T3, "gcc built-in:     ");

end Lock_Free_Benchmark;
```

As expected the second two are almost the same. Protected object is 2/3 times slower, which is quite good in my view.

Here are the measurements.

Debian:

Method	Total	Time
protected object:	0.000938000	9ns
atomic integer:	0.000536000	5ns
gcc built-in:	0.000382000	4ns

Windows:

Method	Total	Time
protected object:	0.001584700	16ns
atomic integer:	0.000534000	5ns
gcc built-in:	0.000384800	4ns

Interestingly, Linux is faster, maybe because it uses simpler locks than Windows. Remember discussion on external protected action which blocks under Linux but does not under Windows? It seems that Windows is slower for that same reason.

From: sidisyom
Date: Sun, 20 Apr 2025 21:13:22 +0000

Thanks for sharing, out of curiosity what compiler flags did you use for the executables? (assume the same in both machines?)

From: dmitry-kazakov
Date: Mon, 21 Apr 2025 08:58:06 +0000

-O2, however in this case it seems to have no visible effect.

From: Fabien.C
Date: Thu, 24 Apr 2025 14:51:27 +0000

Did you check the generated assembly code? GNAT will use atomic operations for protected objects when possible.

From: dmitry-kazakov
Date: Thu, 24 Apr 2025 15:50:23 +0000

Good point! It indeed does this when /Lock_Free/ aspect is used. If I change the line 18 to:

```
   protected Protected_Integer with
   Lock_Free is
```

Then the results look like this under Windows:

Method	Total	Time
protected object:	0.000658100	7ns
atomic integer:	0.000531200	5ns
gcc built-in:	0.000382000	4ns

Of course, in real applications a lock-free implementation would be almost never possible, especially on lousy RISC hardware. But it is great to have this option.

Generalized Private Extension ("is Record with Private")

From: LionelDraghi
Subject: Generalized Private Extension ("is Record with Private")
Date: Fri, 18 Apr 2025 22:37:32 +0000
Newsgroups: forum.ada-lang.io

There was here an interesting discussion [1] about setters and getters, and choosing between hiding or exposing data at record type level.

I sometimes have the in-between situation.

If I want to create a type Person, exposing all fields except the age, I have to either make a bunch of setters and getters, or make two types, one being private (with other annoying constraints if they are in the same package due to the “premature” use of the type).

Let’s give an example in a language that have made the (wrong) choice “Everything is a class”:

```
class Person {
  public name: string;
  // The name is publicly accessible
  private age: number;
  // The age is accessible only within the
  // Person class
constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

  public isAdult(): boolean {
    return this.age >= 18;
  }
}
```

What would be nice in Ada to have the ability to complete a type in the private part, something like:

```
package Person is

   type Person is record with private
      Name : Unbounded_String;
   end record;
   -- "record with private" is clear enough

   -- or
   type Person is record
      Name : Unbounded_String;
   with private;
   -- More natural syntax to me, but possible
   -- confusion with predicates,
   -- and taboo "end record" dropping :-)

   function Is_Adult (P : Person) return
   boolean is (P.Age >= 18);

private
   type Person private record is
      Age : Natural;
   end record;
   -- It would be strange to have here a
   -- normal record type declaration
   -- that wouldn't tell to the reader that other
   -- fields are already declared.
   -- This is why there is an explicit "private
   -- record is",
   -- close enough to what it actually means
   -- ("private additional part of the record is").
   -- But you could object that it is already the
   -- case (not having all the fields at hand)
   -- with tagged type, and that it not worth
   -- creating a new syntax for that reason.

end Persons;
```

From the semantic point of view, it would be a normal private type, with the same constraints applying on the partial view, except that public fields are in direct access, like if Setters and Getters where provided.

And for the sake of unification, this could also apply to tagged type private extension.

Does this proposal make sense?

[1] https://forum.ada-lang.io/t/moving-a-record-to-private/1590/3

From: pyj
Date: Sat, 19 Apr 2025 00:50:45 +0000

Yes, it makes sense conceptually, but no, I wouldn’t want it and don’t ever remember a case where I’ve done something like what you’ve shown.

Forcing types to be fully exposed or private forces a distinct separation between data-only types without behavior and opaque types which maintain

invariants. Ada conceptually makes this simpler by making it such that you can tell from a single line of declaration whether a type is data or an opaque type only manipulated through subprograms.

From: Nordic_Dogsledding
Date: Sat, 19 Apr 2025 07:04:54 +0000

Is this what you're looking for? What is it that you don't like in this design and want to improve?

```
with Ada.Strings.Unbounded;
use Ada.Strings.Unbounded;

package Mixture is
  type Hidden is private;

  type Person is record
    Name: Unbounded_String;
    More: Hidden;
  end record;

  function is_Adult (P: Person) return
  Boolean;

private

  type Hidden is record
    Age: Natural;
  end record;

  function is_Adult (P: Person) return
  Boolean is (P.More.Age >= 18);

end Mixture;
```

From: LionelDraghi
Date: Sat, 19 Apr 2025 16:27:15 +0000

> Forcing types to be fully exposed or private forces a distinct separation between data-only types without behavior and opaque types which maintain invariants. Ada conceptually makes this simpler by making it such that you can tell from a single line of declaration whether a type is data or an opaque type only manipulated through subprograms.

OK, I agree.

> Is this what you're looking for?

This is clear, short, almost perfect

I have oversimplified my case, and today I can't remember how this proposal could have helped with my problem!

I have a self referential structure, fully exposed in a spec (bbt/src/bbt-documents.ads at [1]).

Jeffrey [Carter] wrote (Papers/Self_Ref_No_Access.pdf at [2]) and talked about that problem.

The solution seems to be:

- Status Quo, I let the details exposed, no private part;
- Using Vector of access type instead of a Vector of the type works: I wrote the code, it works, I was able to split this big package in a child hierarchy by using limited and private with.

But I'm not a big fan because of the access type;

- Using an Ada.Containers.Indefinite_Holders instance to hold the Vectors is more complex than needed;
- Using tagged record instead of record for the nodes is OK (and actually make sense in my case, because all nodes share some common properties), but, first, having Vectors of Node'Class instead of Vector of Node make the code kind of lying on my design intents, and I have the impression of inviting possible errors…

None of the solutions is really nice, but I'm now experimenting with the last one.

Anyway, the night has gone, and I can't remember how this proposal could have helped with my problem.

[1] https://github.com/LionelDraghi/bbt/blob/1061e62626caeeb5511e7643360c580927083ffb/src/bbt-documents.ads

[2] https://github.com/jrcarter/Papers/blob/b579a6550bdbaa30bd3b7fe008f6c485dfa9f9dc/Self_Ref_No_Access.pdf

Generics and Compilation Time

From: Shuihuzhuan
Subject: Generics and Compilation Time
Date: Sun, 1 Jun 2025 10:06:38 +0000
Newsgroups: forum.ada-lang.io

I have a really simple generic package like that (spec only here):

```
pragma Ada_2022;

with Ada.Containers.Indefinite_Vectors;
with Ada.Strings.Text_Buffers;

generic
   type Element () is tagged private;
   with procedure Element_Image (Output:
   in out Ada.Strings.Text_Buffers.
   Root_Buffer_Type'Class;
   Item : Element) is <>;
package Gir_Reader.Generic_Lists is

   use type Element;

   package Element_Lists is new
   Ada.Containers.Indefinite_Vectors
     (Index_Type   => Positive,
      Element_Type => Element);

   type Element_List is new
   Element_Lists.Vector with null record
     with Put_Image => Image;

   procedure Image
     (Output : in out Ada.Strings.Text_Buffers.
        Root_Buffer_Type'Class;
      Item : Element_List);

end Gir_Reader.Generic_Lists;
```

Without using it, compilation of my whole program takes around 11s. Each time I instantiate this package, I get 3s more compilation time. Now, I'm around 25s with 5 instantiations.

If instead of using a generic I just repeat the code, the compilation time penalty is negligible.

Are there reasons for such compilation penalty using generics?

From: Irvise
Date: Sun, 1 Jun 2025 11:56:06 +0000

Generics are known to not be that efficient with GNAT. However, the time penalty that you are getting is way too large to be acceptable.

Could you provide information about what compiler version you are using and which flags are enabled?

From: Shuihuzhuan
Date: Sun, 1 Jun 2025 13:39:41 +0000

Thanks Irvise.

Without my own generic package, the time penalty is not as negligible as I thought. I had just tried with 2-3 copy-pastes. With the whole use cases in my code, I get around the same compile time penalty.

So this compilation time is from new Ada.Containers.Indefinite_Vectors.

My computer is quite old (AMD Phenom II X3 from 2009). That may explain part of the problem. But well, 3s for that :->

I'm on archlinux using Alire with standard flags.

```
Ada_Compiler_Switches :=
External_As_List ("ADAFLAGS", " ");
Ada_Compiler_Switches :=
Ada_Compiler_Switches &
 (
  "-Og" -- Optimize for debug
 ,"-ffunction-sections"
  -- Separate ELF section for each function
 ,"-fdata-sections"
  -- Separate ELF section for each variable
 ,"-g" -- Generate debug info
 ,"-gnatwa" -- Enable all warnings
 ,"-gnatw.X" -- Disable warnings for
                --No_Exception_Propagation
 ,"-gnatVa" -- All validity checks
 ,"-gnaty3" -- Specify indentation level 3
 ,"-gnatya" -- Check attribute casing
 ,"-gnatyA" -- Use of array index numbers
               -- in array attributes
 ,"-gnatyB" -- Check Boolean operators
 ,"-gnatyb" -- Blanks not allowed at
              -- statement end
 ,"-gnatyc" -- Check comments
 ,"-gnaty-d" -- Disable check no DOS line
               -- terminators present
 ,"-gnatye" -- Check end/exit labels
 ,"-gnatyf" -- No form feeds or vertical
              -- tabs
 ,"-gnatyh" -- No horizontal tabs
 ,"-gnatyi" -- Check if-then layout
 ,"-gnatyI" -- check mode IN keywords
 ,"-gnatyk" -- Check keyword casing
 ,"-gnatyl" -- Check layout
 ,"-gnatym" -- Check maximum line length
 ,"-gnatyn" -- Check casing of entities in
              -- Standard
 ,"-gnatyO" -- Check that overriding
```

```
                -- subprograms are Ç
                -- explicitly marked as such
        ,"-gnatyp" -- Check pragma casing
        ,"-gnatyr" -- Check identifier
                -- references casing
        ,"-gnatyS" -- Check no statements
                -- after THEN/ELSE
        ,"-gnatyt" -- Check token spacing
        ,"-gnatyu" -- Check unnecessary blank
                -- lines
        ,"-gnatyx" -- Check extra parentheses
        ,"-gnatW8" -- UTF-8 encoding for wide
                -- characters
    );
```

and

```
toolchain.use.gnat=gnat_native=14.2.1
toolchain.use.gprbuild=gprbuild=22.0.1
toolchain.external.gnat=false
```

Have a nice day!

From: jere
Date: Sun, 1 Jun 2025 15:28:54 +0000

Out of curiosity, if you remove the Put_Image function and the aspect for it on your main type does that change the timing at all? Asking because Put_Image is a relatively new feature, so maybe the implementation for it isn't quite optimized.

Also are you using HDD or SSD type drives? The binding stage is going to generate files and if your antivirus scans them (a lot do by default unless disabled), that might be getting slow on an HDD type drive. Not saying it's an acceptable time loss, but trying to isolate the cause.

From: Shuihuzhuan
Date: Sun, 1 Jun 2025 16:11:12 +0000

Thanks for the advice.

Without the Put_Image aspect, I get almost the same compilation time (37s vs 39s with 10 instantiations).

I'm using an SSD drive and no antivirus (I'm on linux).

From: JC001
Date: Tue, 3 Jun 2025 09:31:55 +0000

> Without my own generic package, the time penalty is not as negligible as I thought. I had just tried with 2-3 copy-pastes. With the whole use cases in my code, I get around the same compile time penalty.

So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication, since GNAT does instantiation by macro expansion. What might save you time is using a compiler that implements shared-code generics (DEC Ada or Janus Ada).

> it's nicer to have 10s compilation time instead of around 40s

The first Ada compiler I used was the Rolm/Data General compiler, which took about 10 minutes to compile a "Hello, World!" program, so I don't know what you're complaining about.

From: DirkCraeynest
Date: Tue, 3 Jun 2025 09:51:34 +0000

> The first Ada compiler I used was the Rolm/Data General compiler, which took about 10 minutes to compile a "Hello, World!" program, so I don't know what you're complaining about.

Mind you, for the younger people here, what Jeffrey is talking about is a compiler dating from the early 1980's (1983 or 1984 IIRC). In fact, that compiler got Validation Certificate nr. 2, after NYU's Ada/Ed that got the first certificate (but the latter was an interpreter written in Setl, a language based on sets and interpreted as well, i.e. two levels of interpretation). So ROLM/Data General was the first validated Ada 1983 /compiler/. Note that it was still in serious use at least until 2006. Been there, done that…

Historic info added to avoid Jeffrey's posting would be used to falsely claim "Ada is slow" [sic]. 40+ years ago, compiling and running a Java or Rust program would have taken infinitely longer…

From: Shuihuzhuan
Date: Tue, 3 Jun 2025 11:38:19 +0000

> So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication, since GNAT does instantiation by macro expansion.

It somehow has to do. It's just not my own generic package but the indefinite_vector generic that takes time to compile!

From: JC001
Date: Wed, 4 Jun 2025 09:28:41 +0000

> It's just not my own generic package but the indefinite_vector generic that takes time to compile!

Well, of course. An instance of Indefinite_Vectors is a pretty large package. But if you hand-instantiate it, it will still take about as long to compile.

From: dmitry-kazakov
Date: Wed, 4 Jun 2025 09:46:41 +0000

From my experience the package length had little to do with the problem. I had a lot of small generics instantiated for various scalar types (signed, unsigned x size). It blew the GNAT Pro out.

I would propose to consider instead of generics in generics to look if

- child generics
- formal generic instance as an argument

do better.

(Even better, avoid generics if you can, and use tagged types instead!)

From: dmitry-kazakov
Date: Tue, 3 Jun 2025 13:02:05 +0000

> So this has nothing to do with generics; it's due to the amount of code that you're compiling, which is roughly the same whether you use generics or code duplication

It has all to do with generics. GNAT implementation is extremely slow and consumes incomprehensible amounts of memory. It has become much better in recent decades. The production code that actively used generics compiled several /days/ then and I could not compile it on a 32-bit system. It ran out of memory!

I think AdaCore added a lot of optimisation since then, which does not kick in this concrete case because of new Ada features.

From: cantanima
Date: Sun, 1 Jun 2025 23:02:32 +0000

You mention the compile time, but how does the /execution/ time change, if at all?

If the execution time is faster, then the longer compile time might be an acceptable tradeoff.

From: Shuihuzhuan
Date: Mon, 2 Jun 2025 05:12:44 +0000

It doesn't change anything at runtime. I was just surprised to have a 3s compilation time added each time I added an instantiation. And when developing, it's nicer to have 10s compilation time instead of around 40s :->

From: jcmoyer
Date: Tue, 3 Jun 2025 02:56:35 +0000

I have found that it helps immensely to pre-instantiate expensive-to-compile generics outside of the file where you use them. This prevents them from being re-instantiated every time you compile modified code.

For example, a string_vectors.ads might only contain:

```
with Ada.Containers.Indefinite_Vectors;

package String_Vectors is new
Ada.Containers.Indefinite_Vectors
  (Index_Type => Positive,
   Element_Type => String);
```

Then String_Vectors will be instantiated once, cached, and reused across compilations (at least until you modify the generic body).

Specifying Major Order for Packed Matrix

From: mosteo
Subject: Specifying Major Order for Packed Matrix
Date: Sun, 8 Jun 2025 19:01:32 +0000
Newsgroups: forum.ada-lang.io

See this declaration for 8 bits arranged in a matrix of 4x2, indexed by (row, column):

```
type Cell_Matrix is array (1..4, 1..2) of
Boolean with
   Component_Size => 1,
   Size => 8,
   Convention => Fortran;
```

I was hoping that this would give column-major order but Unchecked_Conversion reveals row-major order.

After a few searches and ARM digging I could not find an answer to the following:

- Can a matrix major order be specified somehow? [Couldn't find a way]
- If not, are Ada matrices guaranteed to be row-major? [I think not]

I have several workarounds at hand, no need to suggest alternatives. I'm interested on the normative behind it only. Thanks!

From: dmitry-kazakov
Date: Sun, 8 Jun 2025 20:11:09 +0000

Hmm, considering this:

```
with Ada.Text_IO; use Ada.Text_IO;
with Interfaces.Fortran; use
Interfaces.Fortran;

procedure Test is
   type Cell_Matrix is array (1..4, 1..2) of
   Real with Convention => Fortran;

   type Matrix is array (1..2, 1..4) of Real;

   A : Cell_Matrix := ((1.1, 1.2), (2.1, 2.2),
   (3.1, 3.2), (4.1, 4.2));
   B : Matrix;
   pragma Import (Ada, B);
   for B'Address use A'Address;
begin
   for I in A'Range (1) loop
      for J in A'Range (2) loop
         Put (Real'Image (A (I, J)));
         Put (' ');
      end loop;

      New_Line;
   end loop;
   New_Line;
   for I in B'Range (1) loop
      for J in B'Range (2) loop
         Put (Real'Image (B (I, J)));
         Put (' ');
      end loop;
      New_Line;
   end loop;
end Test;
```

The output is:

```
 1.10000E+00  1.20000E+00
 2.10000E+00  2.20000E+00
 3.10000E+00  3.20000E+00
 4.10000E+00  4.20000E+00

 1.10000E+00  2.10000E+00  3.10000E+00
 4.10000E+00
 1.20000E+00  2.20000E+00  3.20000E+00
 4.20000E+00
```

It looks correct to me.

From: JC001
Date: Mon, 9 Jun 2025 11:00:16 +0000

It is only Implementation Advice that arrays with Convention Fortran use column-major order, so a compiler does not have to do so. However, I know that GNAT does so, at least for unpacked arrays of numbers. You're packing your array into a single byte, and that may conflict with or take precedence over the convention. I tried

```
with Ada.Text_IO;
with Ada.Unchecked_Conversion;

procedure Fortran_Test is
   type Cell_Base is array (1 .. 4, 1 .. 2) of
   Boolean;
   type Cell_Fortran is new Cell_Base with
   Convention => Fortran;
   type Cell_Packed  is new Cell_Base with
   Component_Size => 1, Size => 8;
   type Cell_PF is new Cell_Packed with
   Convention => Fortran;

   B : constant Cell_Base := ( (False, False),
   (False, True), (True, False), (True, True));
   F : constant Cell_Fortran := Cell_Fortran
   (B);
   PB : constant Cell_Packed  :=
   Cell_Packed (B);
   PF : constant Cell_PF      := Cell_PF (F);

   type Byte is mod 256;

   function To_Byte is new
   Ada.Unchecked_Conversion (Source =>
  Cell_Packed, Target => Byte);
   function To_Byte is new
   Ada.Unchecked_Conversion (Source =>
   Cell_PF, Target => Byte);
begin -- Fortran_Test
   Ada.Text_IO.Put_Line (Item => "PB =" &
   To_Byte (PB)'Image & " PF =" & To_Byte
   (PF)'Image);
end Fortran_Test;
```

with GNAT and got 216 for both. With ObjectAda I get PB = 216 PF = 2; the value for PF is clearly wrong.

From: mosteo
Date: Mon, 9 Jun 2025 12:14:43 +0000

So it's the packing that in the case of GNAT is nullifying the Fortran convention.

Anyway, as long as those are recommendations and there's no way to be sure that the compiler either gives you what you want or a compilation error, I'm not happy with relying on any behavior. And GNAT at least is not warning about not honoring the convention.

Thank you both for your tests.

From: jere
Date: Mon, 9 Jun 2025 14:28:37 +0000

It might be worth putting in a report with AdaCore and see if it is intended. Even though it is implementation advice, I always took that to be more of the advice was "to either provide the convention or not and if you do here are the rules". It would be odd to have the convention only if it doesn't actually map to the language types correctly in some situations (without warnings).

From: Irvise
Date: Mon, 9 Jun 2025 17:46:52 +0000

This goes beyond Ada.

Fortran does have booleans and arrays of booleans. HOWEVER, it *does not have bit arrays (packed arrays of booleans)*. Bit arrays in Fortran, as expected by the standard and by its users, are Arrays of integers on which bit operators can be performed (or just a single INT*8/16/…).

Therefore, what you are trying to do here is illegal in Fortran and its behaviour is either undefined or implementation defined. I think this is good. Though true, GNAT could issue a warning!

Also, I would like to see more Fortran/Ada interop

From: OneWingedShark
Date: Mon, 9 Jun 2025 18:33:16 +0000

This is not a very satisfactory answer: the notion of an array of integers —even constrained to the range 0 to 1, inclusive— indexed in row- or column-major ordering is separate and distinct from the mere implementation-detail of how much space said integers take up.

From: Irvise
Date: Mon, 9 Jun 2025 19:00:04 +0000

Yes, I agree, but the problem remains. I also did not want to focus too much on the space required for the integers… Though I understand that the underlying memory should be the same, so Ada/GNAT could try a bit harder or just be honest and say that this is a mistake

From: mosteo
Date: Tue, 10 Jun 2025 11:37:59 +0000

> HOWEVER, it *does not have bit arrays (packed arrays of booleans)*

This is a valid point, but I'm really not interested in interoperating with Fortran, it was just the only thing I found that could affect the component layout. I would be perfectly happy with something like

```
type Matrix is array (1 .. 4, 1 .. 2) of Boolean
with Storage_Layout => Column_Major;
```

I find it interesting that you can be precise (and get guarantees) with record representation clauses, but there seems to be no similar capabilities for arrays.

From: dmitry-kazakov
Date: Tue, 10 Jun 2025 12:54:16 +0000

The next step would be supporting packed symmetrical matrixes and three-diagonal matrices, video frame buffers etc…

From: JC001
Date: Wed, 11 Jun 2025 09:52:54 +0000

Note that even for something like

```
type Bit_Map is array (1 .. 8) of Boolean
with Component_Size => 1, Size => 8;
```

the language doesn't specify which index corresponds to which bit position. If you need to use indices to access specific bit positions, it's probably best to use a private type with appropriate operations.

Ada Frontiers

Plan for Next Version of International Ada Standard

From: sttaft
Subject: Plan for Next Version of International Ada Standard
Date: Fri, 18 Apr 2025 15:00:53 +0000
Newsgroups: forum.ada-lang.io

As the ISO WG9 Ada Rapporteur Group (ARG) finishes up its work on Ada 2022 and its first "corrigendum", we are beginning the next major revision cycle. At this point we are looking for brainstorming and prioritizing of the next set of features for Ada. If you want to provide some inputs, please visit

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/134

From: LionelDraghi
Date: Mon, 21 Apr 2025 23:50:35 +0000

As it happens, Brian Goetz, the lead architect of Java just gave a conference titled " Where is the Java language going?"

Very interesting to see the process, the major identified topics, the sources of inspiration (which do not include Ada, and yet!), to see how initial choices weigh in 30 years later, to hear the discourse on 'we went too far' with objects, the consequences of hardware evolution that now makes the widespread use of pointers costly, etc.

By the way, on this last point, it seems to me that with its more favorable stack vs. heap balance and its ability to use fewer pointers than other languages, Ada should position itself in the data-oriented wave.

It would be quite amusing if, decades after being considered heavy and slow, Ada made a name for itself in the arena of high-performance languages!

From: pmnw
Date: Tue, 22 Apr 2025 19:59:24 +0000

Thanks, I will watch that with great interest.

You make a very good point with regards to the increasing focus on data flows and computing rather than just applications.

It's interesting to consider Ada as a language that enables that sort of programming.

Arguments about definitions of what is "data-orientation" aside, I have my doubts given the shape of Ada's standard library which offers very little data-oriented functionality - sorts, finds, min, max, etc. are far from being sufficient for work with "big" data.

To make the thing worthwhile, it would have to be competitive in offering high-level abstractions lifted from functional programming languages because serious data processing cannot be done without them (~ the language expressing the data flow through functional transforms and saving users from writing raw loops).

For example, C++ has lately went full on ranges and views with operations such as applies, folds, filters, joins, concats, splits, zips, chunks, slides, strides, and so on, along with piping that together make it feel almost like programming in R's Tidyverse.

It has generators and coroutines and a state-of-the-art random library.

It all works well with Stepanov's original algorithm designs like rotations, etc.

It could absolutely be done in Ada and I expect the result would be much more elegant and beautiful, I have no doubt, but I cannot imagine Ada getting a fraction of this functionality.

From: OneWingedShark
Date: Wed, 23 Apr 2025 03:15:06 +0000

> Arguments about definitions of what is "data-orientation" aside, I have my doubts given the shape of Ada's standard library which offers very little data-oriented functionality - sorts, finds, min, max, etc. are far from being sufficient for work with "big" data.

There is some argument that "big data" is more often than not "stupid data" — like using textual '1' and '0' to represent bits, for every bit in a byte, for every byte in a word. (Not entirely wrong: IIUC, the recent killer jump in AI came from dropping float-sizes; I expect it would have been even cheaper if they went to fixed-point.)

"The Industry" idiocy aside, you are right that there ought to be some development of a suite of algorithms and, I presume, data-structures… Welcome to 1985. — It was always the idea that an /Ada Programming Support Environment/ [APSE] would have such functionality; see:

- Adabase: a data base for Ada programs [1] (1983)
- Workspaces and experimental databases: Automated support for software maintenance and evolution [2] (1987)
- The importance of Ada programming support environments [3] (1982)
- A Revised STONEMAN for Distributed Ada Support [4] (1983)

But there's a huge point of wisdom here that you may be missing: these capabilities were wisely kept from the /Ada Programming Language/ Standard. — This isn't to say that we shouldn't have good, well-built libraries, but that there shouldn't be a standard library that includes everything-and-the-kitchen-sink. (Which is perhaps the /other/ "we went too far" of Java.)

[1] https://dl.acm.org/doi/pdf/10.1145/3304133.3304141

[2] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f5980272e4b21a5e7c44e0c1b1af3be9beae5e38

[3] https://dl.acm.org/doi/pdf/10.1145/1500774.1500815

[4] https://apps.dtic.mil/sti/tr/pdf/ADA137940.pdf

From: pmnw
Date: Wed, 23 Apr 2025 09:21:53 +0000

> But there's a huge point of wisdom here that you may be missing: these capabilities were wisely kept from the /Ada Programming Language/ Standard. — This isn't to say that we shouldn't have good, well-built libraries, but that there shouldn't be a standard library that includes everything-and-the-kitchen-sink.

I agree, I made this exact point… implicitly, and really there is just too much that can be put behind "data-driven" or "data-oriented" for a meaningful discussion without numerous preconditions.

If Ada users want more performance à la @LionelDraghi just above, then I'm all for it, but it won't enable what I understand as "data-oriented" programming.

There is also a chasm between offering performance and people making use of it.

More wrt the /Plan for next version of International Ada Standard/

Looking at the JavaOne video - the language is getting functional imports like algebraic data types and pattern matching and syntax sugars. Working with data requires an ergonomic DSL and they seem to think these features will serve as such. It may work for them, but Ada seems to me in need of shaving and polishing more than inflation.

So, polishing what's there - yes, opening new fronts - no.

I'm saying this as an FP appreciator. Until recently I thought it would be nice for Ada to also import some of the functional features, but after some more thought, I'm thinking that wouldn't work. It would further complicate an already complicated language.

Ideas I like:

- unicode
- cleanup & annex J
- libraries
- searchable documentation

Unicode and language cleanup should be solved on the language-stdlib level.

In a perfect world, simplifying the language by removing obsolescent or duplicated features would be the utmost priority. “Stop adding stuff, start removing it.”

Ideas I dislike (Github):

- allowing instantiation at the point of use, and with more parameters inferred - against Ada design.
- safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother.
- standard packages for XML and JSON.

The last point really makes you think… I’m unaware of any algorithms libraries in Ada, but there’s a proposal to include XML in the standard library. The standard library offers no tools for high-level data structure transforms beyond the most pedestrian operations like find, and it’s not being discussed - but it could soon have XML.

That’s interesting and seems to confirm the emergent consensus (what @Lucretia said in the GitHub discussion) that Ada is a language for embedded development.

From: Lucretia
Date: Wed, 23 Apr 2025 10:17:56 +0000

> safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother

Adding a modern profile where other pointer facilities are the default, like nullable (easier to add?) or optional (requires fp afaik) instead of access types.

From: jere
Date: Wed, 23 Apr 2025 17:01:51 +0000

> safe use of pointers, potentially including support for pointer ownership - I’d rather be reading about banning pointers and null, and in any case, you won’t out-Rust Rust, so why bother

I think if you eliminated access types entirely, then with the current Ada it would be shooting yourself in the foot. There’s very little language support in managing collections of limited objects. Right now you have to rely on access types or global variable declarations if you want to do complex stuff with limited types. Also there’s definitely a middle ground between Rust level semantics and where we are now. I think anything that removes the need for access types or at least makes them safer to use in common situations is a good thing to consider.

I do think that the standard library and the language should increase the ability to avoid pointers in more situations (it already covers a lot of area in this aspect, but still there are outliers, and some of the existing solutions are not great / readable / maintainable).

> In a perfect world, simplifying the language by removing obsolescent or duplicated features would be the utmost priority. “Stop adding stuff, start removing it.”

I agree here about removing things (I don’t think we stop adding stuff if we need it), but I will say based on my experience reading the ARG discussions, it seems very unlikely. A lot of folks on the committee are worried about breaking older versions of the language, which for me makes little sense, since people using older versions of the language can use the compilers they always used without the new features, or restrict themselves to an older version of the language via compiler switches/pragmas. I feel like compiler vendors can make the business decision based on their users if they want to maintain / update those older versions or just move forward (or do both).

From: sttaft
Date: Wed, 23 Apr 2025 17:48:26 +0000

> I agree here about removing things [...]

We do move things into Annex J, the Obsolescent Features annex, when we believe we have replaced them with something clearly better. This can make the description of the language simpler (presuming you ignore Annex J!), without necessarily breaking old code.

At some point if a feature has been in the Obsolescent Features Annex for several language versions, I suspect some compiler implementors might stop supporting it, though if they have customers using old versions, they might more likely just presume that customers who care can use the Restriction No_Obsolescent_Features (see RM 13.12.1(4/3)).

From: pmnw
Date: Wed, 23 Apr 2025 17:57:04 +0000

> I think if you eliminated access types entirely, then with the current Ada it would be shooting yourself in the foot.

Of course, I am just being grumpy about the sudden need for smart pointers or borrowing in a 40-yo language. Perhaps this is politics, idk.

I’m mostly worried about the language playing catch-up with Rust at the cost of more and more additions. Even then, if more is better, I’d prefer additions to be higher-level than memory management facilities.

From: Lucretia
Date: Tue, 22 Apr 2025 20:29:03 +0000

> on ‘we went too far’ with objects

They definitely did, Java really is spaghetti code where you never know what object you’re using, it’s a mess really and I really think caused the modern hatred of oop.

> Ada made a name for itself in the arena of high-performance languages!

Since when was Ada /not/ high-performance??

From: LionelDraghi
Date: Tue, 22 Apr 2025 20:36:14 +0000

> Since when was Ada /not/ high-performance??

I was speaking of perception only!

From: LionelDraghi
Date: Tue, 22 Apr 2025 22:01:20 +0000

> Arguments about definitions of what is “data-orientation” aside

I was thinking of the performance aspect of Data Oriented Design.

For example, the AdaCore proposal of a Max_Size aspect for classwide type [1]

The Motivation chapter says “Being able to use classwide types where definite types are required in Ada, which in turns allows to either use tagged types where they couldn’t be used before”, and then discuss the benefit of removing dynamic allocation, and getting tagged types more used in embedded context.

From my point of view, the main benefit would be to enable arrays of classwide instances of such types, without pointers, that is with a dramatic effect on locality and a potential huge gain in performance.

With only the “with Maxsize => x” aspect added to the declaration of the root tagged type (and maybe some other aspect to lock the class hierarchy).

I dream I could just declare

```
Team : array (1 .. 10) of Player'Class;
```

No access type, no indefinite containers: not only the performances are much better, but the code is simpler than ever.

[1] https://hackmd.io/q0NXV7J8RdiambtId8CMsg

From: jere
Date: Tue, 22 Apr 2025 22:53:22 +0000

Space performance would potentially be worse, but I can see a better speed increase. On the GNAT specific front they are adding a settable aspect Class'Size => in GCC 15 that might facilitate the above. The normal problem is that the size of Player’Class is unknown at compile time (in the general case). That new aspect allows one to set an upper bound, which I would imagine makes array type declarations possible.

From: sidisyom
Date: Sun, 27 Apr 2025 20:45:42 +0000

> I dream I could just declare

It looks like this is included [1] (see feature #2) in the latest version of GNAT Pro (with -gnatX0).

+1 for the class hierarchy lock-down. In the jvm space Kotlin has this nice feature called /Sealed Classes/ which makes all subclasses statically known during compilation time and I assume allows a compiler to apply some nice performance optimisations since it is no longer restricted by the dynamic class loading requirement of the jvm.

At the code level, it offers some nice readability benefits when casing (or in Kotlin parlance whening) on the type of a function argument.

In the jvm of course, /all/ allocations are heap-based so it is a different world there (though, that is not strictly accurate with recent (JIT) compilers).

[1] https://blog.adacore.com/lets-play-7-differences-in-ada

From: Micronian2
Date: Wed, 23 Apr 2025 18:30:37 +0000

I too think more effort should be placed on improving the existing parts of the language and less on major language additions that would take longer to get implemented (if ever, which depends on what AdaCore does). More standard libraries would be good too. I also don't want more proliferation of anonymous access types (yuck!). I absolutely do *not* want GNAT's new "class" type extension (i.e. the one that is similar to C++ classes) to become standard either.

From: sttaft
Date: Wed, 23 Apr 2025 19:17:12 +0000

> I absolutely do *not* want GNAT's new "class" type extension (i.e. the one that is similar to C++ classes) to become standard either.

I think many members of the ARG agree with you.

From: OneWingedShark
Date: Wed, 23 Apr 2025 20:42:33 +0000

> I think many members of the ARG agree with you.

I certainly don't want it: it adds no significant value for additional complexity.

The bizarre thing, though is the /*Why!?*/ — If you wanted to do something much more usable with the effort, implement an updated Lamb-Nestor IDL (which DIANA was an instance of), bringing its capabilities (and possibly syntax) more in-line with Ada/SPARK, with an eye toward producing Ada/SPARK & ASN.1, then you have something /far/ more usable, and which could be used both to generate your classes, with SPARK-annotations, and to enable robust heterogenous-system comms/interchange.

From: waleedmebane
Date: Sun, 27 Apr 2025 21:18:55 +0000

About removing things and whether Ada (or Java) is too big: It seems to me that one of the appeals of the language is that decades old software still works, and one can presumably expect code one writes today to work for decades with little change. The same seems true for Java, but it likely wouldn't be if programmers had to rely on third-party libraries to provide common features since such third-parties are usually less concerned about supporting old features than compiler writers. It's not just a matter of using an old version of the compiler necessarily since sometimes the machines and OSes that they ran on cease to be produced. (For Java, that might be okay as long as the byte-code syntax and semantics don't change.)

What if POSIX hadn't been so successful? Then, much less C code would be working today on common machines and OSes unless there were some similarly long-lived standard and library implementations for those machines (and maybe also an easy portability path).

So, it seems like the large scope of the language standard is a big part of what makes such longevity for programs possible without much maintenance burden.

SPARK/Ada

Importing C with SPARK

From: Lucretia
Subject: Importing C with Spark
Date: Thu, 24 Apr 2025 09:45:00 +0000
Newsgroups: forum.ada-lang.io

Can it be done? The docs are 4 lines and says:

> This section describes features for mixed-language programming in SPARK, covering facilities offered by Ada's Annex B.

> Package Interfaces can be used in SPARK, including its intrinsic "Shift" and "Rotate" functions.

> Other packages are not directly supported.

From: jere
Date: Thu, 24 Apr 2025 17:22:42 +0000

I don't know SPARK, but the reference manual seems to indicate both Convention and Import are supported, so my guess is that wording implies that what languages are supported depends on the vendor.

[1] https://docs.adacore.com/spark2014-docs/html/lrm/language-defined-aspects-and-attributes.html

From: Lucretia
Date: Thu, 24 Apr 2025 17:26:22 +0000

I did a quick test and it seems you can.

From: sttaft
Date: Thu, 24 Apr 2025 22:19:45 +0000

This has been changing recently, I believe, as more customers are using SPARK and C together. And as you discovered, it does work, even if it isn't "directly supported".

SPARK-verified Postfix Expression Calculator

From: pyj
Subject: Spark Verified Postfix Expression Calculator
Date: Thu, 12 Jun 2025 02:25:49 +0000
Newsgroups: forum.ada-lang.io

I got inspired by @streaksu's Ironclad work and @OneWingedShark's idea to write a verified seedForth interpreter in SPARK. That seemed a bit large as a starting SPARK project, so I wrote a SPARK verified postfix expression calculator [1].

I had a great time doing this, so I might try to fork that calculator into a simple Forth interpreter. I don't have formal methods training, but I wrote up my experience if anyone is curious [2].

I would really appreciate any guidance on the current code or related post from those with formal methods experience. The only reference I have right now is McCormick/Chapin's "Building High Integrity Applications with SPARK" book, but that has been very helpful so far.

[1] https://github.com/pyjarrett/postfix_calc

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

From: Fabien.C
Date: Thu, 12 Jun 2025 07:47:36 +0000

Nice project! I'd love to see you continue

From: gast
Date: Thu, 12 Jun 2025 13:35:20 +0000

Thank you! Could you please also add the prover output to the README in [1]? This way, other non-Ada users can see what it looks like without having to install and run anything.

[1] https://github.com/pyjarrett/postfix_calc

From: OneWingedShark
Date: Thu, 12 Jun 2025 15:47:52 +0000

Very nice!

Glad I was somewhat inspiring.

(Reading the article now.)

From: mgrojo
Date: Sat, 14 Jun 2025 16:46:55 +0000

Regarding " Floating point simplifications", what I've found useful in CoAP-SPARK [1] for proving floating point properties is the COLIBRI prover. It doesn't come with the Alire distribution of gnatprove, but can be separately obtained from the GNAT Community 2021 edition. Then, just putting the binary on the path is enough to make gnatprove

find it and use it when requested by the arguments. This workflow [2] automating the proof process of my project is doing the download and extraction from GNAT Community.

[1] https://github.com/mgrojo/coap_spark

[2] https://github.com/mgrojo/coap_spark/blob/3dc855f1cf5c730b512c9c782c82973a10f95c81/.github/workflows/prove.yml

From: Irvise
Date: Sun, 15 Jun 2025 06:50:25 +0000

You can download the Colibri and Colibri2 (still highly in beta) from their website: Colibri [1]

Their repositories (where the binaries come from) are the following:

- Colibri: pub / COLIBRI · GitLab [2]
- Colibri2 (colibrics): pub / colibrics · GitLab [3]

I am pointing this out as they are open source projects that have seen quite a bit of development since the GNAT Community pack was discontinued. I believe it may also be much easier to download the binaries (or compressed binary) directly and just put it into the Path. I have tested Colibri and it did not do much better than the others, but as @mgrojo points out, each solver has their strengths, and Colibri's lies in numeric stuff rather than program flow or similar topics.

I hope this helps you all!

[1] https://colibri.frama-c.com/download.html

[2] https://git.frama-c.com/pub/colibri

[3] https://git.frama-c.com/pub/colibrics/

From: mgrojo
Date: Sun, 15 Jun 2025 14:42:33 +0000

Using the latest Colibri or Colibri2 releases from the repositories, as you mention, was my first approach, but they aren't really compatible with gnatprove yet, as AdaCore confirmed to me through the GNAT Academic program. But the tool fails silently and you don't see any difference. The working version is Colibri 2020.9 [1], but building from source is not trivial. I finally discovered that the correct version was present in the latest Community Edition. The license [2] is LGPL since 2020 [3], so there are no possible worries about using or distributing it.

[1] https://git.frama-c.com/pub/colibri//commit/ba7a7755ab9c58043e693fcadb6f3aaf16f63ad5

[2] https://git.frama-c.com/pub/colibri//blob/ba7a7755ab9c58043e693fcadb6f3aaf16f63ad5/LICENCE

[3] https://git.frama-c.com/pub/colibri//commit/03b6e2900abb24ef072541733517bb04a67260a9

From: Irvise
Date: Sun, 15 Jun 2025 19:37:10 +0000

> but they aren't really compatible with gnatprove yet, as AdaCore confimed me through the GNAT Academic program. But the tool fails silently and you don't see any difference.

Oh, that explains a few things I saw during my testing…

That is actually a bit surprising to me as all the tools are using Why3 and there are already four provers supported (Z3, alt-ergo, cvc5 and Coq with some tweaks…). Oh well… That is quite sad. I know however that AdaCore was looking/working in Colibri, which is fully confirmed by the fact that it was distributed with the Community Edition…