ADA USER JOURNAL

Volume 45
Number 1
March 2024

Contents

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

[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,

Once more, the flagship Ada conference is upon us [1], this year taking place in Barcelona, Spain. Furthermore, among its satellite activities is an "Ada Developers Workshop" [2] that aims to fill in for the sorely missed "Ada Developer Room" of FOSDEM past.

For lovers of Ada nitty-gritty details, this period includes a discussion of Container and Cursor semantics [3] with head-butting positions, so the reader can take sides (or hold their unopposed personal truth at home ;-)).

Sincerely,

Alejandro R. Mosteo.

[1] "AEiC 2024 - Ada-Europe Conference - Deadlines Approaching", in Ada-related Events.

[2] "Ada Developer Workshop @ AEiC 2024, a New "FOSDEM DevRoom" for the Community", in Ada-related Events.

[3] "Re: Map Iteration and Modification", in Ada Practice.

Ada-related Events

Ada-Europe Conference - 31 Jan Journal Track Extended Deadline

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe conference - 31 Jan Journal Track Extended Deadline
Date: Mon, 8 Jan 2024 10:43:48 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

UPDATED Call for Contributions

28th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024)

11-14 June 2024, Barcelona, Spain

www.ada-europe.org/conference2024

*** Journal track deadline EXTENDED to 31 January 2024 ***

*** Other submissions by 26 February 2024 ***

Organized by Ada-Europe and Barcelona Supercomputing Center (BSC), in cooperation with ACM SIGAda, ACM SIGBED, ACM SIGPLAN, and Ada Resource Association (ARA)

#AEiC2024 #AdaEurope #AdaProgramming

General Information

The 28th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024) will take place in Barcelona, Spain.

AEiC is a leading international forum for providers, practitioners, and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development, and maintenance of long-lived, high-quality software systems for a challenging variety of application domains. The program will also include keynotes, Q&A and discussion sessions, and social events. Participants include practitioners and researchers from industry, academia, and government organizations active in the development of reliable software technologies.

The topics of interest for the conference include but are not limited to (more specific topics are described on the conference web page):

* Formal and Model-Based Engineering of Critical Systems;
* High-Integrity Systems and Reliability;
* AI for High-Integrity Systems Engineering;
* Real-Time Systems;
* Ada Language;
* Applications in Relevant Domains.

The conference comprises different tracks and co-located events:

* Journal track papers present research advances supported by solid theoretical foundation and thorough evaluation.
* Industrial track contributions highlight industrial open challenges and/or the practitioners' side of a relevant case study or industrial project.
* Work-in-progress track papers illustrate novel research ideas that are still at an initial stage, between conception and first prototype.
* Tutorials guide attenders through a hands-on familiarization with innovative developments or with useful features related to reliable software.
* Workshops provide discussion forums on themes related to the conference topics.
* Vendor presentations and exhibitions allow for companies to showcase their latest products and services.

Important Dates

31 January 2024 EXTENDED submission deadline for journal track papers

26 February 2024 Deadline for submission of industrial track papers, work-in-progress papers, tutorial and workshop proposals

22 March 2024 First round notification for journal track papers, and notification of acceptance for all other types of submissions

11-14 June 2024 Conference

Call for Journal Track Submissions

Following a journal-first model, this edition of the conference includes a journal track, which seeks original and

high-quality papers that describe mature research work on the conference topics. Accepted journal track papers will be published in a Special Issue of Elsevier JSA - the Journal of Systems Architecture (Q1 ranked, CiteScore 8.5, impact factor 4.5). Accordingly, the conference is listed as "Journal Published" in the latest update of the CORE Conference Ranking released in August 2023. Contributions must be submitted by 31 January 2024. Submissions should be made online at https://www.editorialmanager.com/jsa/, selecting the "Ada-Europe AEiC 2024" option (submission page open from 15 November 2023) as article type of the paper. General information for submitting to the JSA can be found at the Journal of Systems Architecture website.

JSA has adopted the Virtual Special Issue model to speed up the publication process, where Special Issue papers are published in regular issues, but marked as SI papers. Acceptance decisions are made on a rolling basis. Therefore, authors are encouraged to submit papers early, and need not wait until the submission deadline. Authors who have successfully passed the first round of review will be invited to present their work at the conference. The abstract of the accepted contributions will be included in the conference booklet.

The Ada-Europe organization will waive the Open Access fees for the first four accepted papers (whose authors do not already enjoy Open Access agreements). Subsequent papers will follow JSA regular publishing track. Prospective authors may direct all enquiries regarding this track to the corresponding chairs, Bjorn Andersson (baandersson@sei.cmu.edu) and Luis Miguel Pinho (lmp@isep.ipp.pt).

Call for Industrial Track Submissions

The conference seeks industrial practitioner presentations that deliver insight on the challenges of developing reliable software. Especially welcome kinds of submissions are listed on the conference website. Given their applied nature, such contributions will be subject to a dedicated practitioner-peer review process. Interested authors shall submit a 1-to-2 pages abstract, by 26 February 2024, via EasyChair at https://easychair.org/my/conference?conf=aeic2024, selecting the "Industrial Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstract of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be invited to expand their contributions into full-fledged articles for publication in the Ada User Journal, which will form the proceedings of the industrial track of the Conference. Prospective authors may direct all enquiries regarding this track to its chairs Luciana Provenzano (luciana.provenzano@mdu.se) and Michael Pressler (Michael.Pressler@de.bosch.com).

Call for Work-in-Progress Track Submissions

The work-in-progress track seeks two kinds of submissions: (a) ongoing research and (b) early-stage ideas. Ongoing research submissions are 4-page papers describing research results that are not mature enough to be submitted to the journal track. Early-stage ideas are 1-page papers that pitch new research directions that fall within the scope of the conference. Both kinds of submissions must be original and shall undergo anonymous peer review. Submissions by recent MSc graduates and PhD students are especially sought. Authors shall submit their work by 26 February 2024, via EasyChair at https://easychair.org/my/conference?conf=aeic2024, selecting the "Work-in-Progress Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.ada-europe.org/auj/guide.

The abstract of the accepted contributions will be included in the conference booklet. The corresponding authors will get a presentation slot in the prime-time technical program of the conference and will also be offered the opportunity to expand their contributions into 4-page articles for publication in the Ada User Journal, which will form the proceedings of the WiP track of the Conference. Prospective authors may direct all enquiries regarding this track to the corresponding chairs Alejandro R. Mosteo (amosteo@unizar.es) and Ruben Martins (rubenm@andrew.cmu.edu).

Awards

The organization will offer an honorary award for the best technical presentation, to be announced in the closing session of the conference.

Call for Tutorials

The conference seeks tutorials in the form of educational seminars on themes falling within the conference scope, with an academic or practitioner slant, including hands-on or practical elements. Tutorial proposals shall include a title, an abstract, a description of the topic, an outline of the presentation, the proposed duration (half-day or full-day), the intended level of the contents (introductory, intermediate, or advanced), and a statement motivating attendance. Tutorial proposals shall be submitted at any time but no later than the 26 February 2024 to the respective chair Maria A. Serrano (maria.serrano@nearbycomputing.com), with subject line: "[AEiC 2024: tutorial proposal]". Once submitted, each tutorial proposal will be evaluated by the conference organizers as soon as possible, with decisions from January 1st. The authors of accepted full-day tutorials will receive a complimentary conference registration, halved for half-day tutorials. The Ada User Journal will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

The conference welcomes satellite workshops centred on themes that fall within the conference scope. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the AEiC conference. Workshop organizers shall also commit to producing the proceedings of the event, for publication in the Ada User Journal. Workshop proposals shall be submitted at any time but no later than the 26 February 2024 to the respective chair Sergio Saez (ssaez@disca.upv.es), with subject line: "[AEiC 2024: workshop proposal]". Once submitted, each workshop proposal will be evaluated by the conference organizers as soon as possible, with decisions from January 1st.

Academic Listing

The Journal of Systems Architecture, publication venue of the journal track proceedings of the conference, is Q1 ranked, with CiteScore 8.5 and Impact Factor 4.5. The Ada User Journal, venue of all other technical proceedings of the conference, is indexed by Scopus and by EBSCOhost in the Academic Search Ultimate database.

Call for Exhibitors and Sponsors

The conference will include a vendor and technology exhibition with the option of a 20 minutes presentation as part of the conference program. Interested providers should direct inquiries to the Exhibition & Sponsorship Chair Ahlan Marriot (ahlan@ada-switzerland.ch).

Venue

The conference will take place in Barcelona, Spain. Barcelona is a major cultural, economic, and financial centre, known for its architecture, culture, and Mediterranean atmosphere, a hub for technology and innovation. There's plenty to see and visit in Barcelona, so plan in advance!

Organizing Committee

- Conference Chair

Sara Royuela, Barcelona Supercomputing Center, Spain
sara.royuela@bsc.es

- Journal Track Chairs

Bjorn Andersson, Carnegie Mellon University, USA
baandersson@sei.cmu.edu

Luis Miguel Pinho, ISEP & INESC TEC, Portugal
lmp@isep.ipp.pt

- Industrial Track Chairs

Luciana Provenzano, Mälardalen University, Sweden
luciana.provenzano@mdu.se

Michael Pressler, Robert Bosch GmbH, Germany
Michael.Pressler@de.bosch.com

- Work-In-Progress Track Chairs

Alejandro R. Mosteo, CUD Zaragoza, Spain
amosteo@unizar.es

Ruben Martins, Carnegie Mellon University, USA
rubenm@andrew.cmu.edu

- Tutorial Chair

Maria A. Serrano, NearbyComputing, Spain
maria.serrano@nearbycomputing.com

- Workshop Chair

Sergio Saez, Universitat Politècnica de València, Spain
ssaez@disca.upv.es

- Exhibition & Sponsorship Chair

Ahlan Marriott, White Elephant GmbH, Switzerland
ahlan@Ada-Switzerland.ch

- Publicity Chair

Dirk Craeynest, Ada-Belgium & KU Leuven, Belgium
Dirk.Craeynest@cs.kuleuven.be

- Webmaster

Hai Nam Tran, University of Brest, France
hai-nam.tran@univ-brest.fr

- Local Chair

Nuria Sirvent, Barcelona Supercomputing Center, Spain
nuria.sirvent@bsc.es

Journal Track Committee

Al Mok, University of Texas at Austin, USA

Alejandro Mosteo, CUD Zaragoza, Spain

Alwyn Godloe, NASA, USA

António Casimiro, University of Lisbon, Portugal

Barbara Gallina, Mälardalen University, Sweden

Bernd Burgstaller, Yonsei University, South Korea

C. Michael Holloway, NASA, USA

Cristina Seceleanu, Mälardalen University, Sweden

Doug Schmidt, Vanderbilt University, USA

Frank Singhoff, University of Brest, FR

George Lima, Universidade Federal da Bahia, Brazil

Isaac Amundson, Rockwell Collins, USA

Jérôme Hugues, CMU/SEI, USA

José Cruz, Lockeed Martin, USA

Kristoffer Nyborg Gregertsen, SINTEF Digital, Norway

Laurent Pautet, Telecom ParisTech, France

Leonidas Kosmidis, Barcelona Supercomputing Center, Spain

Mario Aldea Rivas, University of Cantabria, Spain

Matthias Becker, KTH - Royal Institute of Technology, Sweden

Patricia López Martínez, University of Cantabria, Spain

Sara Royuela, Barcelona Supercomputing Center, Spain

Sergio Sáez, Universitat Politècnica de València, Spain

Tucker Taft, AdaCore, USA

Tullio Vardanega, University of Padua, Italy

Xiaotian Dai, University of York, England

Industrial Track Committee

Aida Causevic, Alstom, Sweden

Alexander Viehl, Research Center for Information Technology, Germany

Ana Rodríguez, Silver Atena, Spain

Aurora Agar, NATO, Netherlands

Behnaz Pourmohseni, Robert Bosch GmbH, Germany

Claire Dross, AdaCore, France

Elena Lisova, Volvo CE, Sweden

Enricco Mezzeti, Barcelona Supercomputing Center, Spain

Federico Aromolo, Scuola Superiore Sant'Anna, Italy

Helder Silva, Edisoft, Portugal

Hugo Torres Vieira, Evidence Srl, Italy

Irune Agirre, Ikerlan, Spain

Jordi Cardona, Rapita Systems, Spain

José Ruiz, AdaCore, France

Joyce Tokar, Raytheon, USA

Luciana Alvite, Alstom, Germany

Marco Panunzio, Thales Alenia Space, France

Patricia Balbastre Betoret, Valencia Polytechnic University, Spain

Philippe Waroquiers, Eurocontrol NMD, Belgium

Raúl de la Cruz, Collins Aerospace, Ireland

Santiago Urueña, GMV, Spain

Stef Van Vlierberghe, Eurocontrol NMD, Belgium

Work-in-Progress Track Committee

Alan Oliveira, University of Lisbon, Portugal

J. Javier Gutiérrez, University of Cantabria, Spain

Jérémie Guiochet, LAAS-CNRS, France

Kalinka Branco, University of São Paulo, Brazil

Katherine Kosaian, University of Iowa, USA

Kevin Cheang, AWS, USA

Kristin Yvonne Rozier, Iowa State University, USA

Leandro Buss Becker, University of Manchester, UK

Li-Pin Chang, National Yang Ming Chiao Tung University, Taiwan

Mathias Preiner, Stanford University, USA

Raffaele Romagnoli, Carnegie Mellon University, USA

Robert Kaiser, RheinMain University of Applied Sciences, Germany

Sara Abbaspour, Mälardalen University, Sweden

Sergi Alcaide, Barcelona Supercomputing Center, Spain

Simona Bernardi, Unizar, Spain

Stefan Mitsch, School of Computing at DePaul University, USA

Teresa Lázaro, Aragon's Institute of Technology, Spain

Tiago Carvalho, ISEP, Portugal

Yannick Moy, AdaCore, France

Previous Editions

Ada-Europe organizes annual international conferences since the early 80's. This is the 28th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08), Brest, France ('09), Valencia, Spain ('10), Edinburgh, UK ('11), Stockholm, Sweden ('12), Berlin, Germany ('13), Paris, France ('14), Madrid, Spain ('15), Pisa, Italy ('16), Vienna, Austria ('17), Lisbon, Portugal ('18), Warsaw, Poland ('19), online from Santander, Spain ('21),

Ghent, Belgium ('22), and Lisbon, Portugal ('23).

Information on previous editions of the conference can be found at www.ada-europe.org/confs/ae.

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2024 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 28th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2024)

* June 11-14, 2024, Barcelona, Spain, www.ada-europe.org/conference2024

(V4.1)

AEiC 2024 - Ada-Europe Conference - Deadlines Approaching

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: AEiC 2024 - Ada-Europe conference - Deadlines Approaching
Date: Fri, 16 Feb 2024 19:07:10 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

UPDATED Call for Contributions - Additional Tracks

28th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024)

11-14 June 2024, Barcelona, Spain

*** DEADLINES approaching: 26 February and 4 March 2024 ***

www.ada-europe.org/conference2024

*** Submission DEADLINE 26 February 2024 ***

Workshops: submit to Workshop Chair, Sergio Saez ssaez@disca.upv.es subject "[AEiC 2024: workshop proposal]"

Tutorials: submit to Tutorial and Education Chair,
Maria A. Serrano
maria.serrano@nearbycomputing.com
subject "[AEiC 2024: tutorial proposal]"

*** EXTENDED submission DEADLINE 4 March 2024 ***

Industrial- and Work-in-Progress-track: submit via https://easychair.org/my/conference?conf=aeic2024
select "Industrial Track" or "Work in Progress Track"

For more information please see the full Call for Papers at www.ada-europe.org/conference2024/cfp.html

---

Organized by Ada-Europe and Barcelona Supercomputing Center (BSC), in cooperation with ACM SIGAda, ACM SIGBED, ACM SIGPLAN, and Ada Resource Association (ARA)

#AEiC2024 #AdaEurope #AdaProgramming

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2024 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 28th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2024)

* June 11-14, 2024, Barcelona, Spain, www.ada-europe.org/conference2024

(V6.1)

Ada Developer Workshop @ AEiC 2024, a New "FOSDEM DevRoom" for the Community

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: Ada Developer Workshop @ AEiC 2024, a new "FOSDEM DevRoom" for the community
Date: Sat, 24 Feb 2024 22:30:03 +0100
Newsgroups: comp.lang.ada

Dear Ada community,

I come with great news! For the past two years, there was no Ada DevRoom over @ FOSDEM, a place where the Ada community used to meet and share their work and projects. Some of us wanted to keep having such experience as we believed it to be a greatly beneficial aspect to the wider Ada community.

For this reason, Fabien Chouteau, Dirk Craeynest and Fernando Oleo Blanco, made a proposal to the Ada-Europe International Conference on Reliable Software Technologies (AEiC 2024 aka Ada-Europe 2024) in order to have a "devroom" for the wider Ada community, just like in FOSDEM.

We were accepted and you can already find all the information over at the Ada Developer Workshop webpage [1]!

I would encourage everybody to take a look at it! Nonetheless, here is a quick summary highlighting some of the points:

- It will take place on Friday, 14th of June in Barcelona. Friday was chosen in order to minimise the amount of free days/holidays that we would need to take off from our jobs and allow us to then use the weekend to visit and enjoy Barcelona.
- The cost will be lower than for the main conference. Our goal is to make it completely free, just like FOSDEM, but this is still a Work-In-Progress (WIP).
- The nature of the event is similar to any past DevRoom that took place @ FOSDEM. The main difference is that now, being an open-source project will not be a requirement.
- March 31st, 2024 is the (current) deadline for submissions. If you would like to present your work or discuss topics, please, please please, keep this date in mind!

We are eager to hear from all of you. And if you have any questions, please, let us know!

[1] https://www.ada-europe.org/conference2024/adadev.html

From: Streaksu <streaksu@mailbox.org>
Date: Tue, 27 Feb 2024 06:51:03 +0100

That sounds amazing! Thank you so much for your work and to the people at AEiC for making it happen.

> The cost will be lower than for the main conference.

That would be a huge deal. I have not checked this edition's registrations, but if 2023's are anything to go by, as a hobbyist Ada developer, I don't think I can justify it for myself. But a cheaper event would be a great alternative. Please do keep us updated!

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Date: Fri, 22 Mar 2024 19:18:04 +0100

Hi Ada community!

This is a kind reminder that you can still submit any talks to the Ada Developer Workshop that will take place during the AEiC 2024, on the 14th of June in Barcelona!

Entry prices should be published shortly in the AEiC website. Nonetheless, we are still looking for some sponsorships :)

For more information see http://www.ada-europe.org/conference2024/adadev.html or email any of the organisers (Fabien, Dirk and Fernando).

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Date: Mon, 25 Mar 2024 23:18:42 +0100

Great news everybody! This was posted by Dirk on the Ada-Lang forum.

Hot news! Thanks to AdaCore sponsoring the Ada Developer Workshop in Barcelona, the early registration fee for in-person participation will be only 10 EUR, including lunch and coffee breaks.

That's as low-cost as attending an Ada Developer Room at FOSDEM in

Brussels, as you easily spend 10 EUR on food and drinks there… ;)

Registration info, for the conference, tutorials, workshops, social events, will shortly be added to the conference website at Ada-Europe 2024 [1].

Hope to see many of you there!

And remember, submissions are still welcome!

[1] http://www.ada-europe.org/conference2024/

Ada Monthly Meetup 2024

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Subject: Ada Monthly Meetup 2024
Date: Sun, 3 Mar 2024 20:31:05 +0100
Newsgroups: comp.lang.ada

Dear all, this is just a quick reminder that the next Ada Monthly Meetup will take place on Saturday 9th of March!

No topics were proposed for this meetup. Nonetheless, I will take the opportunity to talk a bit about FOSDEM (and WolfSSL), the newly proposed Ada Developer Workshop during AEiC, remind people about the newly released Alire v2.0-RC1 and a few other topics if we have time.

From: Fernando Oleo / Irvise <irvise_ml@irvise.xyz>
Date: Sun, 17 Mar 2024 10:10:26 +0100

Hello everybody!

I would like to announce the April (2024) Ada Monthly Meetup which will be taking place on the 6th of April at **13:00 UTC time (15:00 CEST)**. As always the meetup will take place over at Jitsi. The Meetup will also be livestreamed to Youtube.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no topics :wink:

Though I will try to focus more on Ada and I would like to bring people's attention to [Tsoding's Ada livestreams] (https://forum.ada-lang.io/t/making-a-game-in-ada-with-raylib/704).

Here are the connection details from previous posts: The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is [Jitsi Meet] (https://meet.jit.si/AdaMonthlyMeetup) The room name is “AdaMonthlyMeetup” and in case it asks for a password, it will be set to “AdaRules”.

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

Best regards and see you soon! Fer

P.S: it is that time of year when clocks have their time changed. So please, take a look at whether this affects you. (Central) Europe will now go from CET to CEST, so +2h. USA and related countries already had their time changed last week.

Ada-related Resources

[Delta counts are from February 19th to May 28th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 28 May 2024 13:23 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: _705 (+144) members [1]
- LinkedIn: 3_509 (+30) members [2]
- Stack Overflow: 2_405 (+12) questions [3]
- Gitter: 253 (+10) people [4]
- Ada-lang.io: 219 (+37) users [5]
- Telegram: 201 (+28) users [6]
- Libera.Chat: 75 (-1) concurrent users [7]

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

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

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

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

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

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

[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: 28 May 2024 13:33 CET
To: Ada User Journal readership

GitHub: >1_000* (=) developers [1]
Rosetta Code: 950 (+10) examples [2]
42 (+4) developers [3]
Alire: 405 (+12) crates [4]
1_048 (new) releases [5]
Sourceforge: 251 (+3) projects [6]
Open Hub: 214 (=) projects [7]
Codelabs: 57 (=) repositories [8]
Bitbucket: 38 (+1) repositories [9]

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

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

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

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

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

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

[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: 28 Feb 2024 13:43 CET
To: Ada User Journal readership

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

- TIOBE Index: 22 (+3) 0.83% (+0.06%) [1]
- PYPL Index: 19 (-4) 0.82% 1.08% (-0.26%) [2]
- Languish Trends: 180 (new) 0.01% [3]
- Stack Overflow Survey: 42 (=) 0.77% (=) [4]
- IEEE Spectrum (general): 36 (=) Score: 0.0107 (=) [5]
- IEEE Spectrum (jobs): 29 (=) Score: 0.0173 (=) [5]
- IEEE Spectrum (trending): 30 (=) Score: 0.0122 (=) [5]

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

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

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

[4] https://survey.stackoverflow.co/2023/

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

Certificate Error Accessing Adapower.com

From: Juanmiuk <juanmiuk@gmail.com>
Subject: Certificate Security Error when. access adapower.com
Date: Wed, 24 Jan 2024 05:30:39 -0800
Newsgroups: comp.lang.ada

When I tried to access adapower.com from the last version of Chrome and

NordVPN VPN the browser shows me this error:

Your connection isn't private. The web page you are trying to enter is not certified by a known certifying authority. Attackers might be trying to steal your information (for example, passwords, messages, or credit cards).

This error did not happen with Safari or Microsoft Edge (last version)

What's going on?

From: Stéphane Rivière <stef@genesix.org>
Date: Wed, 24 Jan 2024 16:11:42 +0100

Simply no TLS certificates (see the padlock status before the URL)

This site is in ruins, out of date and should no longer exist.

What's more, a Google search turns up some dubious links.

Ada-related Tools

NeoVim Plugin to Publish Alire Packages

From: Tama Mcglinn <t.mcglinn@gmail.com>
Subject: NeoVim plugin to publish Alire packages
Date: Sat, 17 Feb 2024 00:01:47 -0800
Newsgroups: comp.lang.ada

In case there's any NeoVim users who also publish Alire packages, I wrote a plugin for that;

https://github.com/TamaMcGlinn/nvim-alire-tools

allows you to bind or call `:AlirePublish` which handles everything for your Alire toml file, and intelligently sees where you are in the version publishing process.

Aunit.Checks

From: Simon Wright <simon@pushface.org>
Subject: AUnit.Checks
Date: Sun, 24 Mar 2024 09:19:38 +0000
Newsgroups: comp.lang.ada

Has anyone come across this package? AFAICT it doesn't appear in the AUnit repo on Github.

Even the spec would be invaluable!

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Mar 2024 11:17:06 +0000

Cancel that! It's in Stephe Leake's AUnit extensions, encountered in ada-mode.

Ada Practice

Re: Map Iteration and Modification

[Continues from AUJ 44-4, December 2023. The discussion initially addressed how to modify a container during iteration, to later move onto iteration semantics. —arm]

From: G.B. <bauhaus@notmyhomepage.invalid>
Subject: Re: Map iteration and modification
Date: Mon, 1 Jan 2024 20:27:51 +0100
Newsgroups: comp.lang.ada

>> Suppose that there is a way of orderly proceeding from one item to the next. It is probably known to the implementation of map. Do single steps guarantee transitivity, though, so that an algorithm can assume the order to be invariable?

> An insane implementation can expose random orders each time.

An implementation order should then not be exposed, right? What portable benefits would there be when another interface is added to that of map, i.e., to Ada containers for general use? Would it not be possible to get these benefits using a different approach? I think the use case is clearly stated:

First, find Cursors in map =: C*.

Right after that, Delete from map all nodes referred to by C*.

> Unless removing element invalidates all cursors. Look, insanity has no bounds. Cursors AKA pointers are as volatile as positions in certain implementations. Consider a garbage collector running after removing a pair and shuffling remaining pairs in memory.

> [...]

> you assume that cursors are ordered and the order is preserved from call to call. [...]

Yes, given the descriptions of Ordered_Maps, so long as there is no tampering, a Cursor will respect an order. Likely the one that the programmer has in mind.

[...]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 1 Jan 2024 21:55:12 +0100

> An implementation order should then not be exposed, right?

IMO, an order should be exposed. Not necessarily the "implementation order" whatever that might mean.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 2 Jan 2024 21:15:01 -0600

>> There is no "natural" order to the key/element pairs; they are effectively unordered.

> Iteration = order. It is the same thing. If you provide iteration of pairs in the mapping by doing so you provide an order of.

Certainly not. An iteration presents all of the elements in a container, but there is no requirement that there is an order. Indeed, logically, all of the elements are presented at the same time (and parallel iteration provides an approximation of that).

If you try to enforce an order on things that don't require it, you end up preventing useful parallelism (practically, at least, no one has succeeded at providing useful parallelism to sequential code and people have been trying for about 50 years -- they were trying when I was a university student in the late 1970s).

>> [...] Certainly, no concept of "forward" or "reverse" applies to such an ordering (nor any stability requirement).

> It does. You have a strict total order of pairs which guarantees existence of previous and next pairs according to.

Again, this is unrelated. Iteration can usefully occur in unordered containers (that is, "foreach"). Ordering is a separate concept, not always needed (certainly not in basic structures like maps, sets, and bags).

[...]

Ada requires that cursors continue to designate the same element through all operations other than deletion of the element or movement to a different container. Specific containers have additional invariants, but this is the most general one. No other requirement is needed in many cases.

> Yes, position is a property of enumeration.

Surely not. This is a basis for my disagreement with you here. The only requirement for enumeration is that all elements are produced. The order is an artifact of doing an inherently parallel operation sequentially. We don't care about or depend on artifacts.

[...]

>> You have some problem with an iterator interface as opposed to an array interface??

> Yes, I am against pointers (referential semantics) in general.

This is nonsense - virtually everything is referential semantics (other than components). Array indexes are just a

poor man's pointer (indeed, I learned how to program in Fortran 66 initially, and the way one built useful data structures was to use array indexes as stand-ins for pointers). In A(1), 1 is a reference to the first component of A.

So long as you are using arrays, you are using referential semantics. The only way to avoid it is to embed an object directly in an enclosing object (as in a record), and that doesn't work for many problems.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 2 Jan 2024 21:22:00 -0600

> Cursor is merely a fat pointer.

A cursor is an abstract reference. It *might* be implemented with a pointer or with an array index. Indeed, the bounded containers pretty much have to be implemented with an underlying array.

It would be nice if there was some terminology for abstract references that hadn't been stolen by some programming language. Terms like "pointer" and "access" and "reference" all imply an implementation strategy. That's not relevant most of the time, and many programming language design mistakes follow from that. (Anonymous access types come to mind).

From: Moi <findlaybill@blueyonder.co.uk>
Date: Wed, 3 Jan 2024 04:05:59 +0000

> It would be nice if there was some terminology for abstract references that hadn't been stolen by some programming language. [...]

What about "currency", as used in DB systems?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 3 Jan 2024 11:04:58 +0100

> Certainly not. An iteration presents all of the elements in a container, but there is no requirement that there is an order.

The meaning of the word "iterate" is doing something (e.g. visiting an element) again. That *is* an order.

> Indeed, logically, all of the elements are presented at the same time (and parallel iteration provides an approximation of that).

Parallel iteration changes nothing because involved tasks are enumerated and thus ordered as well.

> If you try to enforce an order on things that don't require it, you end up preventing useful parallelism [...]

Ordering things does not prevent parallelism. But storing cursors for later is a mother of all Sequentialisms! (:-))

Whether container elements can be effectively deleted in parallel is an interesting but rather impractical one. Nobody, literally nobody, cares because any implementation would be many times slower than the worst sequential one! (:-))

> [...] Iteration can usefully occur in unordered containers (that is, "foreach").

"An enumeration is a complete, ordered listing of all the items in a collection."

-- Wikipedia

If "foreach" exposes an arbitrary ordering rather than some meaningful (natural) one, that speaks for "insanity" but changes nothing.

> Ordering is a separate concept, not always needed

Right. But no ordering means no iteration, no foreach etc. If I can iterate, that I can create an ordered set of (counter, element) pairs. Done.

> Surely not. This is a basis for my disagreement with you here.

Then you are disagreeing with core mathematics... (:-))

> The only requirement for enumeration is that all elements are produced.

Produced in an order. Elements only produced" is merely an opaque set. Enumeration of that set is ordering its elements.

> The order is an artifact of doing an inherently parallel operation sequentially.

Yes, ordering is an ability to enumerate elements of a set. It is not an artifact it is the sole semantics of.

[...]

> So long as you are using arrays, you are using referential semantics. [...]

The key difference is that index does not refer to any element. It is container + index that do.

From the programming POV it is about avoiding hidden states when you try to sweep the container part under the rug.

[...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 3 Jan 2024 22:07:30 -0600

> Parallel iteration changes nothing because involved tasks are enumerated and thus ordered as well.

Nonsense. There is no interface in Ada to access logical threads (the ones created by the parallel keyword).

> Ordering things does not prevent parallelism.

Yes it does, because it adds unnecessary constraints. It's those constraints that make parallelizing normal sequential code hard. A parallelizer has to guess which ones are fundamental to the code meaning and which ones are not.

[...]

You are adding an unnecessary property to the concept of iteration. Iteration does not necessarily imply enumeration (it can, of course). Iteration /= enumeration.

[...]

Iteration is not necessarily enumeration. It is applying an operation to all elements, and doing that does not require an order. Some specific operations might require an order, and clearly for those one needs to use a data structure that inherently has an order.

> The key difference is that index does not refer to any element. It is container + index that do.

That's not a "key difference". That's exactly how one should use cursors, especially in Ada 2022. The Ada containers do have cursor-only operations, but those should be avoided since it is impossible to provide useful contracts for those operations (the container is unknown, so the world can be modified, which is bad for parallelism and understanding). Best to consider those operations obsolete. (Note that I was *always* against the cursor-only operations in the containers.)

So, using a cursor implies calling an operation that includes the container of its parameter.

> From the programming POV it is about avoiding hidden states when you try to sweep the container part under the rug.

That's easily avoided -- don't use the obsolete operations. (And a style tool like Jean-Pierre's can enforce that for you.)

> [...] Usability always trumps performance.

That's the philosophy of languages like Python, not Ada. If you truly believe this, then you shouldn't be using Ada at all, since it makes lots of compromises to usability in order to get performance.

> And again, looking at the standard containers and all these *tagged* *intermediate* objects one needs in order to do elementary things, I kind of have doubts... (:-))

The standard containers were designed to make *safe* containers with decent performance. As I noted, they're not a built-in part of the programming language, and as such have no impact on the performance of the language proper. One could easily replace them with an unsafe design to get maximum performance -- but that would have to return pointers to elements, and you've said you don't like referential semantics. So you would never use those.

You also can avoid all of the "tagged objects" (really controlled objects) by using function Element to get a copy of the element rather than some sort of reference to it. That's preferred if it doesn't cost too much for your application.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 4 Jan 2024 12:28:04 +0100

> Iteration is not necessarily enumeration. It is applying an operation to all elements, and doing that does not require an order.

That is not iteration, it is unordered listing, a totally useless thing because the result is the same unordered set.

You could not implement it without prior ordering of the elements you fed to the threads. If the threads picked up elements concurrently there would be no way to do that without ordering elements into a taken / not yet taken order. You cannot even get an element from a truly unordered set, no way! If the programmer tried to make any use of the listing he would again have to impose ordering when collecting results per some shared object.

The unordered listing is a null operation without ordering.

> [...] So, using a cursor implies calling an operation that includes the container of its parameter.

OK. It is some immensely over-designed index operation, then! (:-)) So, my initial question is back, why all that overhead? When you cannot do elementary things like preserving your indices from a well-defined set of upon deleting elements with indices outside that set?

[...]

> Specifically, the containers are separate from Ada.

Not really. Like STL with C++ it massively influenced the language design motivating adding certain language features and shifting general language paradigm in certain direction.

>> Usability always trumps performance.

> That's the philosophy of languages like Python, not Ada.

Ah, this is why Python is totally unusable? (:-))

Ada is usable and performant because of the right abstractions it deploys. If you notice performance problems then, maybe, just my guess, you are using the wrong abstraction?

> The standard containers were designed to make *safe* containers with decent performance.

Well, we always wish for the best... (:-))

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 4 Jan 2024 20:00:37 -0600

> [...]

> Ah, this is why Python is totally unusable? (:-))

I would tend to argue that it is indeed the case that you get dubious results when you put usability first. Ada puts readability/understandability, maintainability, and consistency first (along with performance). Those attributes tend to provide usability, but not at the cost of making things less consistent or understandable.

I wrote an article on this topic a year and a half ago that I wanted to publish on Ada-Auth.org. But I got enough pushback about not being "neutral" that I never did so. (I don't think discussing why we don't do things some other languages do is negative, but whatever.) I've put this on RR's blog at
http://www.rrsoftware.com/html/blog/consequences.html
so it isn't lost.

From: Simon Wright <simon@pushface.org>
Date: Fri, 05 Jan 2024 09:26:03 +0000

> http://www.rrsoftware.com/html/blog/consequences.html

Thanks for this!

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 5 Jan 2024 12:51:50 +0100

> http://www.rrsoftware.com/html/blog/consequences.html

Thanks for posting this.

I disagree with what you wrote on several points:

1. Your premise was that use = writing. To me using includes all aspects of software developing and maintenance process. Writing is only a small part of it.
2. You argue for language regularity as if it were opposite to usability. Again, it is pretty much obvious that a regular language is easier to use in any possible sense.
3. Removing meaningless repetitions contributes to usability. But X := X + Y is only one instance where Ada required such repetition. There are others. E.g.

```
if X in T'Class then
    declare
XT : T'Class renames T'Class (X);
```

   T'Class is repeated 3 times. A discussion point is whether a new name XT could be avoided etc.

   Introducing @ for a *single* purpose contradicts the principle of regularity. I would rather have a regular syntax for most if not all such instances.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 6 Jan 2024 01:25:46 -0600

> 1. Your premise was that use = writing.

Perhaps I didn't make it clear enough, but my premise was that many people making suggestions for Ada confuse "ease-of-use" with "ease-of-writing". I said "mischaracterized" for a reason (and I see that "mis" was missing from the first use, so I just added that). "Ease-of-writing" is not a thing for Ada, and it isn't considered while the other aspects are weighed. And as I said in my last message, there is a difference in that writing more can help understandability, but it never helps writing.

[...]

> T'Class is repeated 3 times. A discussion point is whether a new name XT could be avoided etc.

Of course, this example violates OOP dogma, and some people would argue that it should be harder than following it. That's the same reason that Ada doesn't have that many implicit conversions. In this particular example, I tend to think the dogma is silly, but I don't off-hand see a way to avoid the conversion being somewhere (few implicit conversions after all).

> Introducing @ for a *single* purpose contradicts the principle of regularity.

@ is regular in the sense that it is allowed anywhere in an expression. If you tried to expand the use to other contexts, you would have to differentiate them, which would almost certainly require some sort of declaration. But doing that risks making the mechanism as wordy as what it replaces (which obviously defeats the purpose).

We looked at a number of ideas like that, but they didn't seem to help comprehension. In something like:

```
LHS:(X(Y)) := LHS + 1;
```

(where LHS is an arbitrary identifier), if the target name is fairly long, it could be hard to find where the name for the target is given, and in any case, it adds to the name space that the programmer has to remember when reading the source expression. That didn't seem to add to readability as much as the simple @ does.

In any case, these things are trade-offs, and certainly nothing is absolute. But @ is certainly much more general than ":=+" would be, given that it works with function calls and array indexing and attributes and user-defined operations rather than just a single operator.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 7 Jan 2024 16:06:10 +0100

> [...] But @ is certainly much more general than ":=+" would be [...]

For the 9X and 0X revisions I suggested adding "when <condition>" to return and raise statements, similar to its use on exit statements. This was rejected because the language already has a way to accomplish this: if statements.

Given that one can do

```
declare
   V : T renames Very_Long_Identifier;
begin
  V := V - 23;
end;
```

it seems that @ should also have been rejected. Probably more so, since @ is completely new syntax rather than reusing existing syntax on some additional statements. What is the justification of accepting @ while still rejecting the other?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 8 Jan 2024 22:46:59 -0600

> For the 9X and 0X revisions I suggested adding "when <condition>" to return and raise statements, similar to its use on exit statements.

I don't recall ever seriously considering this (might just my memory getting old). I suspect that didn't get rejected so much as not making the cut as important enough. We do try to limit the size of what gets added, not just adding everyone's favorite feature.

I'd guess that "raise Foo when Something" would get rejected now as it would be confusing with "raise Foo with Something" which means something very different. (At least the types of "Something" are different in these two.) OTOH, we added "when condition" to loops (which I thought was unnecessary, but I lost that), so arguably it would be consistent to add it to other statements and expressions as well. Perhaps you should raise it again on Github.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Tue, 9 Jan 2024 10:43:38 +0100

>I suspect that didn't get rejected so much as not making the cut as important enough.

I don't consider special syntax to shorten names in assignment statements important at all. We have renames for that, and it is a more general mechanism, applying to more than just assignments.

"Usability" (was Re: Map Iteration and Modification)

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Subject: Re: "Usability" (was Re: Map iteration and modification)
Date: Sat, 6 Jan 2024 02:54:09 -0000
Newsgroups: comp.lang.ada

> http://www.rrsoftware.com/html/blog/consequences.html

Without reading that, I would never have understood "usability" to mean "ease of writing". I learned from early on in my programming career that readability was more important than writability. So "using" a language doesn't end with writing the code: you then have to test and debug it-- basically lick it into shape--then maintain it afterwards.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 6 Jan 2024 01:03:05 -0600

> Without reading that, I would never have understood "usability" to mean "ease of writing". [...]

Usability is of course not just ease-of-writing, but a lot of people tend to co-mingle the two. For readability, too little information can be just as bad as too much. For writability, the less you have to write, the better.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 6 Jan 2024 10:14:07 +0200

> Usability is of course not just ease-of-writing, but a lot of people tend to co-mingle the two. For readability, too little information can be just as bad as too much. For writability, the less you have to write, the better.

I feel that is too narrow a definition of writability (and perhaps you did not intend it as a definition). Before one can start typing code, one has to decide what to write -- which language constructs to use. A systematically constructed, regular language like Ada makes that mental effort easier, even if it results in more keystrokes; a plethora of special-case syntaxes and abbreviation possibilities makes it harder.

Perhaps "writability" should even be taken to cover the whole process of creating /correct/ code, and include all the necessary testing, debugging and corrections until correct code is achieved. Here of course Ada shines again, with so many coding errors caught at compile time.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 6 Jan 2024 21:21:30 -0400

> Usability is of course not just ease-of-writing, but a lot of people tend to co-mingle the two.

Yes, I'm always surprised to see many languages (including Rust) praising themselves for being "concise". Apart from saving some keystrokes, I fail to see the benefit of being concise...

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Tue, 09 Jan 2024 15:19:52 +0000

> [...] Apart from saving some keystrokes, I fail to see the benefit of being concise...

Agreed. However, it is a bit of a totem in the FP cult.

Limited with Too Restrictive?

From: Blady <p.p11@orange.fr>
Subject: Limited with too restrictive?
Date: Sat, 13 Jan 2024 17:11:35 +0100
Newsgroups: comp.lang.ada

I want to break some unit circularity definitions with access types as for instance with record:

```
type R1;
type AR1 is access R1;
type R1 is record
   Data : Natural;
   Next : AR1;
end record;
```

In my case, I have a unit:

```
package test_20240113_modr is
   type R2 is record
      Data : Natural;
   end record;
   type AR2 is access R2;
end test_20240113_modr;
```

"limited withed" in:

```
limited with test_20240113_modr;
package test_20240113_mods is
end;
```

Let's imagine the circularity, thus PS1 and PS2 definitions are legal.

Of course the following isn't legal:

```
type AS1 is array (1..2) of
test_20240113_modr.R2; -- illegal
```

However why not with access type:

```
type AS2 is array (1..2) of
test_20240113_modr.AR2; -- illegal
```

Likewise, why not:

```
type AS3 is record
   Data : Natural;
   Next : test_20240113_modr.AR2; -- illegal
end record;
```

Isn't "limited with" too restrictive, is it?

Well, I could make some code transfers from unit to another or access conversions, that's what I actually do but at heavy cost.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 13 Jan 2024 22:31:12 -0600

> However why not with access type:

> type AS2 is array (1..2) of test_20240113_modr.AR2; -- illegal

For a limited with, one only knows the syntactic declarations (we cannot assume any analysis). Therefore, we cannot know the representation of any type, including access types.

Specifically, compilers may support multiple representations for access types, for a variety of reasons (the underlying machine has different representations, as on the 8086 and U2200 that we did compilers for; because additional data needs to be carried along to implement Ada semantics - GNAT did that for access to unconstrained arrays, and so on). The representation can depend upon aspect specifications, the designated subtype, and more, none of which is known at the point of a limited with.

We couldn't restrict implementations to a single representation for access types, and thus limited with has to treat them the same as other types.

It's necessary to declare local access types for entities that are accessed from a limited view. The reason that anonymous access types were expanded was to make that less clunky -- but I don't think it succeeded.

> Well, I could make some code transferts from unit to another or access conversions, that's what I actually do but at heavy cost.

Yup, but the alternative is worse - requiring all access types to be the most general representation (which can have a heavy performance cost).

String_Access in Unbounded String Handling?

From: Blady <p.p11@orange.fr>
Subject: String_Access in unbounded string handling?
Date: Sun, 14 Jan 2024 12:05:40 +0100
Newsgroups: comp.lang.ada

String_Access is defined in A.4.5 Unbounded-Length String Handling:

7 type String_Access is access all String;

and note:

75 The type String_Access provides a (nonprivate) access type for explicit processing of unbounded-length strings.

I wonder what String_Access is for and what could be "explicit processing"?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 14 Jan 2024 12:17:25 +0100

String_Access is a mistake that should not exist.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 14 Jan 2024 16:12:31 +0100

> String_Access is a mistake that should not exist.

Well, from one point of view, surely.

However I frequently need such a type because I in general refrain from using Unbounded_String. Now, it would be no problem to declare it as needed, except for generics! If you have generic packages like:

```
generic
   type Object_Type (<>) is private;
   type Object_Access_Type
      is access all Object_Type;
```

You want all instances to share the same String_Access. So it is conflicting. One is true, it has no place there. It should have been the package Standard or none.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 16 Jan 2024 19:24:40 -0600

> String_Access is a mistake that should not exist.

I agree with Jeffrey. Whatever reason it was initially put into the package has long since ceased to be relevant. And, as Dmitry notes, when you want such a type, it's usually because you didn't want to use Ada.Strings.Unbounded (or Bounded). So the placement is odd at best.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 16 Jan 2024 19:30:57 -0600

> ... It should have been the package Standard or none.

None for me. ;-)

One really doesn't want to put anything in Standard that isn't widely needed, as those names become hard to use in other circumstances. In particular, declarations in Standard hide anything that is use-visible with the same name, so adding something to Standard can be rather incompatible.

One could mitigate use-visibility problems by allowing more extensive overloading (for instance, of objects), but that causes rare and subtle cases where a program could change meaning without any indication. (Where a different object would be used, for instance.) That makes that too risky a change for Ada.

From: Blady <p.p11@orange.fr>
Date: Wed, 17 Jan 2024 10:54:24 +0100

Thanks for all your answers,

This is probably a very minor subject, however I submitted it:
https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/79

From: Tucker Taft <tucker.taft@gmail.com>
Date: Wed, 17 Jan 2024 05:34:12 -0800

> I wonder what String_Access is for and what could be "explicit processing"?

The idea was to support the explicit use of new String'(...), X.all, and Unchecked_Deallocation rather than the implicit use of the heap inherent in Unbounded strings. It was recognized that you need a single global access type to avoid having to do conversions all over the place. This predated the availability of stand-alone objects of an anonymous access type (aka "SAOOAAATs" ;-), but those are not universally loved either. It certainly cannot be removed now without potentially very painful disruption of existing users. It could be moved to a different package without too much disruption, but I haven't seen any groundswell of interest in doing that either.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 18 Jan 2024 19:36:59 -0600

>[...] It certainly cannot be removed now without potentially very painful disruption of existing users.

I'm dubious that there are any such users. Certainly, in the handful of cases where I needed such a type, I just declared it (strong typing, you know?) and never thought of Ada.Strings.Unbounded as being a place to find such a type already defined. It is such an odd place I doubt anyone outside of perhaps the people who defined the type ever used it.

OTOH, I agree that the compatibility impact is non-zero (anyone who did use it would have to change their code), and the benefit of removing the type at this point is close to zero (junk declarations abound in long-term Ada packages, what's one more; and certainly there is a lot of unused stuff in any particular reusable package and any particular use), so the cost-benefit ratio doesn't seem to make a change here worth it. An Ada successor language would design Ada.Strings.Unbounded rather differently (so as to be able to use string literals directly with the type) and probably would include universal character support as well, so it's hard to find an important reason to change this.

Also, I'm pretty sure we're discussed this within the ARG several times in the past, so this is well-trodden ground.

From: Blady <p.p11@orange.fr>
Date: Tue, 30 Jan 2024 16:53:22 +0100

At least, the type String_Access could be tagged as obsolescent.

Choice Must Be Static?

From: Blady <p.p11@orange.fr>
Subject: error: choice must be static?
Date: Sun, 11 Feb 2024 13:29:59 +0100
Newsgroups: comp.lang.ada

I've got the following GNAT error:

```
$ GCC -c -gnat2022 -gnatl
2024/test_20240211_static_choice.adb
GNAT 13.2.0
1. procedure test_20240211_static_choice is
2.
3. package Maps is
4. type Map_Type is private
5. with Aggregate =>  (Empty =>
   Empty_Map,
6. Add_Named => Add_To_Map);
7. procedure Add_To_Map (M : in out
   Map_Type; Key : in Integer; Value : in
    String);
8. Empty_Map : constant Map_Type;
9. private
10. type Map_Type is array (1..10) of String
     (1..10);
11. procedure Add_To_Map (M : in out
     Map_Type; Key : in
     Integer; Value : in String) is null;
12. Empty_Map : constant Map_Type :=
     [1..10 => "          "];
-- error: choice must be static
>>> error: choice must be static
```

I wonder what more static it should be. Any clue?

[Full source code removed. —arm]

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 11 Feb 2024 21:56:17 +0100

I don't know what this means, but it's definitely related to the Aggregate aspect. This compiles:

```
Empty_Base : constant Map_Base :=
                  (1 .. 10 => (1 .. 10 => ' ') );
```

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 12 Feb 2024 09:12:37 +0100

Square brackets are the root of all evil! (:-))

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 12 Feb 2024 20:12:01 -0600

Looks like a compiler bug to me. The nonsense message gives that away... :-)

From: Simon Wright
<simon@pushface.org>
Date: Tue, 13 Feb 2024 11:45:17 +0000

> Looks like a compiler bug to me. The nonsense message gives that away... :-)

GCC 14.0.1 says

[...]

```
 4. type Map_Type is private
 5. with Aggregate => (Empty  =>
Empty_Map,
>>> error: aspect "Aggregate" can only be
applied to non-array type
[...]
14. Empty_Map : constant Map_Type :=
[1..10 => "          "];
>>> error: choice must be static
```

I think the first is because of ARM 4.3.5(2), "For a type other than an array type, the following type-related operational aspect may be specified"[1] and the second is a "nonsense" consequence.

[1] http://www.ada-auth.org/standards/22rm/html/RM-4-3-5.html#p2

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Tue, 13 Feb 2024 22:28:22 -0600

Ah, yes, I didn't notice that part. One cannot give the Aggregate aspect on an array type, directly or indirectly. That's because container aggregates are designed to work like array aggregates, and we didn't want visibility to determine the interpretation of an aggregate (especially where the same syntax could have a different meaning in different visibility).. Thus, there can be no point where a single type can have both array aggregates and container aggregates.

Note that record aggregates and container aggregates are always syntactically different, and thus it is OK to have both in a single location (that's one of the reasons that we adopted square brackets for container aggregates). That seemed important as the majority of private types are completed by record types, and not allowing record types in this context would be difficult to work around.

From: Blady <p.p11@orange.fr>
Date: Sat, 17 Feb 2024 09:51:39 +0100

Thanks Randy for the explanation, it helps.

In-Memory Stream

From: Drpi <314@drpi.fr>
Subject: In memory Stream
Date: Fri, 16 Feb 2024 10:41:12 +0100
Newsgroups: comp.lang.ada

I want to transfer some data between applications through a memory buffer. The buffer transfer between applications is under control. My problem is with the buffer content. I thought I'll use a Stream writing/reading in/from the memory buffer. How can I achieve this? I've found no example doing this.

Note: I use Ada 2012.

From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 16 Feb 2024 11:40:54 +0100

I don't know if this is what you want, but at least it is an example of using streams…

Package Storage_Streams, from Adalog's components page:
https://adalog.fr/en/components.html#Storage_Stream

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Fri, 16 Feb 2024 13:40:27 +0100

> How can I achieve this? I've found no example doing this.

It of course depends on the target operating system. You need to create a shared region or memory mapped file etc. You also need system-wide events to signal the stream ends empty or full.

Simple Components has an implementation interprocess streams for usual suspects:
http://www.dmitry-kazakov.de/ada/components.htm#12.7

> Note : I use Ada 2012.

No problem, it is kept Ada 95 compatible.

From: Pascal Obry <pascal@obry.net>
Date: Fri, 16 Feb 2024 13:49:54 +0100

AWS comes with a memory stream implementation.

https://github.com/AdaCore/aws/blob/master/include/memory_streams.ads

You may want to have a look here.

From: Simon Wright
<simon@pushface.org>
Date: Fri, 16 Feb 2024 20:19:42 +0000

A spec and body for an implementation I've had since 2008:
https://github.com/simonjwright/coldframe/blob/alire/src/common/coldframe-memory_streams.ads

https://github.com/simonjwright/coldframe/blob/alire/src/common/coldframe-memory_streams.adb

From: Drpi <314@drpi.fr>
Date: Sat, 17 Feb 2024 14:36:46 +0100

Concerning the OS and the buffer transfer mechanism, as I said, this is under control. I use Windows and the WM_COPYDATA message.

My usage is a bit special. The writing process writes a bunch of data in a memory buffer then requests this buffer to be transferred to another process by way of WM_COPYDATA. The receiving process reads the data from the "new" memory buffer. I say "new" since the address is different from the one used in the writing process (of course it cannot be the same).

The library Jean-Pierre pointed me to perfectly matches this usage. Light and easy to use. Thanks.

One enhancement I see is to manage the buffer size to avoid buffer overflow (or did I miss something?).

Thanks again to everybody.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 17 Feb 2024 15:26:45 +0100

> The library Jean-Pierre pointed me to perfectly matches this usage. Light and easy to use. Thanks.

:-)

> One enhancement I see is to manage the buffer size to avoid buffer overflow (or did I miss something?).

I don't see what you mean here... On the memory side, we are reading/writing bytes from memory, there is no notion of overflow. And the number of bytes processed by Read/Write is given by the size of Item, so no overflow either...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 17 Feb 2024 15:28:54 +0100

You ask Windows to copy a chunk of memory from one process space into another, so yes, it is physically different memory. Different or same address tells nothing because under Windows System.Address is virtual and can point anywhere.

As you may guess it is a quite heavy overhead, not only because of copying data between process spaces, but also because of sending and dispatching Windows messages.

Note, that if you implement stream Read/Write as individual Windows messages it will become extremely slow. GNAT optimizes streaming of some built-in objects, e.g. String. But as a general case you should expect that streaming of any non-scalar object would cause multiple calls to Read/Write and thus multiple individual Windows messages.

An efficient way to exchange data under Windows is a file mapping. See CreateFileMapping and MapViewOfFile.

https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createfilemappinga

https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-mapviewoffile

Then use CreateEvent with a name to signal states of the stream buffer system-wide. Named Windows events are shared between processes.

https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-createeventa

[This is how interprocess stream is implemented for Windows in Simple Components]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 17 Feb 2024 15:48:05 +0100

> On the memory side, we are reading/writing bytes from memory, there is no notion of overflow.

In the Simple Components there is a pipe stream.

```
type Pipe_Stream
   (Size : Stream_Element_Count)  is
    new Root_Stream_Type with private;
```

When a task writes the stream full (Size elements), it gets blocked until another task reads something out.

Another implementation

```
type Storage_Stream
   (Block_Size : Stream_Element_Count)
    is new Root_Stream_Type with private;
```

rather allocates a new block of memory. The allocated blocks get reused when their contents are read out.

From: Drpi <314@drpi.fr>
Date: Sat, 17 Feb 2024 15:56:34 +0100

> [...] As you may guess it is a quite heavy overhead [...]

In my use case, there is no performance problem. The purpose is to make an editor single instance. When you launch the editor the first time, everything is done as usual. Next time you launch the editor (for example by double clicking on a file in file explorer) the init code of the editor detects an instance of the editor is already running, transfers the command line arguments to the first instance and exits.

The buffer transfer occurs once when starting a new instance of the editor.

However, I keep your solution in mind. I might need it one day.

From: Simon Wright <simon@pushface.org>
Date: Sat, 17 Feb 2024 18:09:02 +0000

> But as a general case you should expect that streaming of any non-scalar object would cause multiple calls to Read/Write and thus multiple individual Windows messages.

Our motivation for the memory stream was the equivalent of this for UDP messages; GNAT.Sockets behaves (behaved?) exactly like this, so we buffered the result of 'Output & wrote the constructed buffer to the socket; on the other side, we read the UDP message, stuffed its contents into a memory stream, then let the client 'Input.

I can't remember at this distance in time, but I think I would have liked to construct a memory stream on the received UDP packet rather than copying the content; the compiler wouldn't let me. Perhaps worth another try.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 17 Feb 2024 19:52:00 +0100

> One enhancement I see is to manage the buffer size to avoid buffer overflow (or did I miss something?).

The purpose of this stream is to access raw memory, so there is no notion of "buffer size". It is up to you to match your (user) buffer with the memory buffer. Of course, you can add a layer with all the checks you want...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 17 Feb 2024 22:33:17 +0100

> I can't remember at this distance in time, but I think I would have liked to construct a memory stream on the received UDP packet rather than copying the content; the compiler wouldn't let me.

UDP is a kind of thing... Basically, there is no use of UDP except for broadcasting, e.g. in LAN discovery.

In all other cases it is either TCP or multicast. Since UDP does not guarantee either delivery or ordering. It would be a huge overhead to implement reliable buffered streams on top of UDP, with sequence numbers, acknowledgements, re-sending, reordering etc.

As for taking apart a UDP packet, it is straightforward. You simply declare a stream element array of the packet size and map it on the packet using:

```
pragma Import (Ada, A);
for A'Address use UDP_Packet'Address;
```

And somewhere

```
pragma Assert (Stream_Element'Size = 8);
```

just in case...

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 18 Feb 2024 00:00:11 -0000

> The writing process writes a bunch of data in a memory buffer then requests this buffer to be transferred to another process by way of WM_COPYDATA.

I thought Windows had pipes.

From: Lawrence D'Oliveiro <ldo@nz.invalid>
Date: Sun, 18 Feb 2024 00:02:33 -0000

> When writing in the stream, you have to care to not overflow the buffer.

With pipes, the OS takes care of this for you. Once its kernel buffer is full, further writes are automatically blocked until a reader has drained something from the buffer.

It’s called “flow control”.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Feb 2024 11:06:16 +0100

> I thought Windows had pipes.

Yes it has, but very rarely used though much better designed than UNIX pipes. See https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createnamedpipea

In general Windows has much richer and better API regarding interprocess

communication than Linux. After all Windows NT was sort of descendant of VMS, which was light years ahead of UNIX Sys V. In recent times Linux improved, e.g. they added futex stuff etc. BSD is far worse than Linux in respect of API.

From: Simon Wright <simon@pushface.org>
Date: Sun, 18 Feb 2024 10:06:46 +0000

> UDP is a kind of thing... Basically, there is no use of UDP except for broadcasting, e.g. in LAN discovery.

Worked for us, sending radar measurements p-2-p at 200 Hz

> for A'Address use UDP_Packet'Address;

OK if the participants all have the same endianness. We used XDR (and the translation cost is nil if the host is big-endian, as PowerPCs are; all the critical machines were PowerPC).

From: Björn Lundin <bnl@nowhere.com>
Date: Sun, 18 Feb 2024 12:36:54 +0100

> I thought Windows had pipes.

It does, we use it for our IPC in both Linux and Windows. Works very well. We use named pipes - where each process knows its name through via env-var At start they create a named pipe with that name

We use anonymous pipes for client communication

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 18 Feb 2024 14:02:32 +0100

> OK if the participants all have the same endianness. We used XDR [...]

I always override stream attributes and use portable formats. E.g. some chained code for integers. Sign + exponent + normalized mantissa for floats, again chained. That is all. There is no need in XDR, JSON, ASN.1 or other data representation mess. They are just worthless overhead.

Raise Expressions from AARM

From: Blady <p.p11@orange.fr>
Subject: Raise expressions from AARM.
Date: Sat, 24 Feb 2024 10:50:31 +0100
Newsgroups: comp.lang.ada

AARM Ada 2022 section 11.3 presents some uses of raise expressions including this one:
(http://www.ada-auth.org/standards/22aarm/html/AA-11-3.html)

2.a.10/4 ...

```
B : Some_Array := (1, 2, 3, others =>
   raise Not_Valid_Error);
```

What could be the use cases?

My guess: whatever the size of Some_Array (greater than 3), B is elaborated but raises Not_Valid_Error when accessing component beyond position 3:

```
type Some_Array is array
   (Positive range 1..10) of Natural;
...
B : Some_Array := (1, 2, 3, others =>
   raise Not_Valid_Error);
...
begin
X := B (2); -- OK
X := B (6); -- raises Not_Valid_Error
end;
```

Is it correct?

NB: GNAT 13.2 issues a compilation error:
>>> error: "others" choice not allowed here
see: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=113862

Thanks, Pascal.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 24 Feb 2024 11:39:08 +0100

> Is it correct?

No. This will raise the exception upon the elaboration of B.

The only use of this that I can imagine is if the length of Some_Array is 3. Then the others choice is null, so the raise expression is never evaluated. But if someone changes the definition of Some_Array to be longer, then the exception will be raised.

> NB: GNAT 13.2 issues a compilation error:

> >>> error: "others" choice not allowed here

> see: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=113862

The example in the error report has Some_Array unconstrained, in which case an others choice is not allowed. With the constrained definition given above, the aggregate is valid.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Sat, 24 Feb 2024 12:39:43 +0200

> What could be the use cases?

The point of these examples (which are only in the discussion annotation, not in the normative standard) is to discuss what is syntactically legal and why. The examples need not make practical sense.

> My guess: [...] raises Not_Valid_Error when accessing component beyond position 3:

No. A raise-expression is not a value that can be stored in an array or passed around; its evaluation raises an exception /instead/ of yielding a value.

In this example, if the evaluation of the array aggregate that initializes B evaluates the expression supplied for the "others" choice, this evaluation will raise Not_Valid_Error and disrupt the initialization of B.

It is not clear to me if the RM requires the evaluation of the "others" expression if there are no "other" indices.
Experimenting with GNAT (Community 2019) shows that if the Some_Array type has 'Length = 3, the exception is not raised (so the "others" value is not evaluated), while if the 'Length is greater than 3 the exception is raised.

> type Some_Array is array (Positive range 1..10) of Natural;

> B : Some_Array := (1, 2, 3, others => raise Not_Valid_Error);

That should raise Not_Valid_Error during the initialization of B.

From: Blady <p.p11@orange.fr>
Date: Sun, 25 Feb 2024 12:09:08 +0100

If I understand well, no compiler error nor warning at compilation time but Not_Valid_Error raised at run time elaboration.

To be compared with:

```
B1 : Some_Array := (1, 2, 3);
```

No compiler error, one compiler warning "Constraint_Error will be raised at run time" and Constraint_Error range check failed raised at run time elaboration.

From: Blady <p.p11@orange.fr>
Date: Sun, 25 Feb 2024 12:23:48 +0100

> The examples need not make practical sense.

Well, despite I knew that, I wanted to draw some use cases from them.

For instance:

```
A : A_Tagged  := (Some_Tagged'
   (raise TBD_Error) with Comp => 'A');
```

It will raise TBD_Error if Some_Tagged is not a null record, good to know, isn't it?

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 26 Feb 2024 22:01:23 +0200

> It will raise TBD_Error if Some_Tagged is not a null record, good to know, isn't it?

Hm, not raising the exception for a null record seems weird to me, and I cannot deduce it from the RM. Moreover, for a plain qualified expression

Some_Tagged'(raise TBD_Error)

not in an extension aggregate GNAT raises the exception even if the type is a null record. I suspect that not raising the exception for an extension aggregate where the ancestor type is a null record is a bug in GNAT.