# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editorial Policy for Ada User Journal</td>
<td>2</td>
</tr>
<tr>
<td>Editorial</td>
<td>3</td>
</tr>
<tr>
<td>Quarterly News Digest</td>
<td>4</td>
</tr>
<tr>
<td>Conference Calendar</td>
<td>25</td>
</tr>
<tr>
<td>Forthcoming Events</td>
<td>33</td>
</tr>
<tr>
<td>Proceedings of the “HILT’22 - Supporting a Rigorous Approach to Software Development Workshop”</td>
<td>35</td>
</tr>
<tr>
<td>H. Ausden. “Achieving 100% Availability in the ERAM Air Traffic Control System”</td>
<td></td>
</tr>
<tr>
<td>D. Larraz, A. Viswanathan, C. Tinelli, M. Laurent. “Beyond Model Checking of Idealized Lustre in KIND 2”</td>
<td>44</td>
</tr>
<tr>
<td>Proceedings of the “ADEPT: AADL by its practitioners Workshop” of AEiC 2022</td>
<td>63</td>
</tr>
<tr>
<td>J. C. Roger, P. Dissaux. “AADL Modelling with SysML v2”</td>
<td></td>
</tr>
<tr>
<td>G. Bardaro, M. Matteucci. “Modelling Robot Architectures with AADL”</td>
<td>80</td>
</tr>
<tr>
<td>E. Senn, L. W. J. Bourdon. “Modeling ROS Based Applications with AADL”</td>
<td>84</td>
</tr>
<tr>
<td>D. Blouin, P. Crisafulli, C. Maxim, F. Caron. “An Introduction to ALISA and Its Usage for an Industrial Railway System Case Study”</td>
<td>88</td>
</tr>
<tr>
<td>Ada-Europe Associate Members (National Ada Organizations)</td>
<td>92</td>
</tr>
<tr>
<td>Ada-Europe Sponsors</td>
<td>Inside Back Cover</td>
</tr>
</tbody>
</table>
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 4
Ada-related Events 4
Ada and Education 7
Ada-related Resources 7
Ada-related Tools 7
Ada and Other Languages 8
Ada Practice 11
Ada in Jest 24

[5] “Table Game”, in Ada and Education.

Sincerely,
Alejandro R. Mosteo.

Ada-related Events

Babbage & Lovelace - The Opera

From: Simon Wright
<simon@pushface.org >
Subject: Babbage & Lovelace - The Opera
Date: Mon, 16 Jan 2023 18:30:56 +0000
Newsgroups: comp.lang.ada
https://guerillaopera.org/repertoire/thrilling-adventures

As I reacted on Twitter when I saw Sydney Padua @sydneypadua announcing this opera adaption of her graphic novel “The Thrilling Adventures of Lovelace and Babbage”:

-----start-quote-------
The late Robert Dewar, of #AdaProgramming language fame, would have loved this. For some history, look for “The Maiden and the Mandate” in https://ada-europe.org/archive/aui/ aui-41-1-withcovers.pdf and https://adacore.com/adacore25...

-----end-quote-------

Those were hilariously funny fully staged musical performances at several ACM SIGAda and Ada-Europe conferences, which I was lucky enough to attend. It would be great if AdaCore could put online one of the video recordings that were made at the time.

Dirk

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/Ada-Europe/SIGAda/WG9)

AEiC 2023 - Ada-Europe Conference - Final Deadline Approaching

[For the record, as the deadline is past. —arm]

From: Dirk Craeynest
<dirk@orka.cs.kuleuven.be >
Subject: AEiC 2023 - Ada-Europe conference - Final Deadline Approaching
Date: Thu, 16 Feb 2023 09:39:33
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

FINAL Call for Contributions
27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023)
13-16 June 2023, Lisbon, Portugal
www.ada-europe.org/conference2023

*** FINAL submission DEADLINE 27 February 2023 ***

Organized by Ada-Europe in cooperation with ACM SIGAda (approval pending) and the Ada Resource Association (ARA)
General Information

The 27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023) will take place in Lisbon, Portugal. The conference schedule comprises a journal track, an industrial track, a work-in-progress track, a vendor exhibition, parallel tutorials, and satellite workshops.

* Journal-track submissions present research advances supported by solid theoretical foundation and thorough evaluation.
* Industrial-track submissions highlight the practitioners' side of a challenging case study or industrial project.
* Work-in-progress-track submissions illustrate a novel research idea that is still at an initial stage, between conception and first prototype.
* Tutorial submissions guide attenders through a hands-on familiarization with innovative developments or with useful features related to reliable software.

Schedule

[CLOSED] Extended submission deadline for journal-track papers

27 February 2023: Submission deadline for industrial-track and work-in-progress-track papers, tutorial & workshop proposals

20 March 2023: First round notification for journal-track papers, acceptance notification for other submission types

13-16 June 2023: Conference

Scope and Topics

The conference 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 allow ample time for keynotes, Q&A sessions and discussions, and social events.

Participants include practitioners and researchers from industry, academia, and government organizations active in the promotion and development of reliable software technologies.

The topics of interest for the conference include but are not limited to:

- Formal and Model-Based Engineering of Critical Systems;
- Real-Time Systems;
- High-Integrity Systems and Reliability;
- Ada Language;
- Applications in a variety of domains.

More specific topics are described on the conference web page.

Call for Journal-track Submissions

Following a journal-first model, this edition of the conference again 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 the "Reliable Software Technologies (AEiC2023)" Special Issue of JSA -- the Journal of Systems Architecture (Scimago Q1 ranked, impact factor 5.936).

[Submission details removed. Call is closed now.]

Authors who have successfully passed the first round of review will be invited to present their work at the conference. Please note that the AEiC 2023 organization committee will waive the Open Access fees for the first four accepted papers, which do not already enjoy OA from personalized bilateral agreements with the Publisher. Subsequent papers will follow JSA regular publishing track.

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 web site. Given their applied nature, such contributions will be subject to a dedicated practitioner-peer review process. Interested authors shall submit a one-to-two pages abstract, by 27 February 2023, via EasyChair at https://easychair.org/my/conference?conf=aeic2023, selecting the "Industrial Track". The format for submission is strictly in PDF, following the Ada User Journal style. Templates are available at http://www.adaeurope.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 Bjorn Anderson (baandersson at seil.cmu.edu) and José Cecilio (jmcceciio at fc.ul.pt).

Awards

Ada-Europe 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 by e-mail to Tutorial and Education Chair, Luís Miguel Pinho (Imp at isep.ipp.pt), with subject line: “[AEiC 2023: tutorial proposal]”. Tutorial proposals shall be submitted by 27 February 2023. 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 by e-mail to the Workshop Chair, Frank Singhoff (singhoff at univ-brest.fr), with subject line: “[AEiC 2023: workshop proposal]”. Workshop proposals shall be submitted at any time but no later than the 27 February 2023. Once submitted, each workshop proposal will be evaluated by the conference organizers as soon as possible.

Call for Exhibitors

The conference will include a vendor and technology exhibition. Interested providers should direct inquiries to the Exhibition & Sponsorship Chair, Ahlan Marriott (ahlan at Ada-Switzerland.ch).

Venue

The conference will take place at the Hotel Fénix Lisboa, near downtown Lisbon, Portugal. June is full of events in Lisbon, including the festivities in honour of St. António (June 13 is the town holiday), with music, grilled sardines, and popular parties in Alfama and Bairro Alto neighbourhoods. There's plenty to see and visit in Lisbon, so plan in advance!

Organizing Committee

- Conference Chair
  António Casimiro,
  University of Lisbon, Portugal
casim at ciencias.ulisboa.pt

- Journal-track Chair
  Elena Troubitsyna,
  KTH Royal Inst. of Technology, Sweden
eleknat at kth.se

- Industrial-track Chairs
  Alexandre Skrzyniarz,
  Thales, France
alexandre.skrzyniarz at fr.thalesgroup.com

- Work-in-Progress-track Chairs
  Bjorn Andersson,
  Carnegie Mellon University, USA
baandersson at sei.cmu.edu

José Cecilio,
University of Lisbon, Portugal
jmceciilio at fc.ul.pt

- Tutorial and Education Chair
  Luis Miguel Pinho,
  ISEP, Portugal
lim at isep.ipp.pt

- Workshop Chair
  Frank Singhoff,
  University of Brest, France
singhoff at univ-brest.fr

- Exhibition & Sponsorship Chair
  Ahlan Marriott,
  White Elephant GmbH, Switzerland
ahlan at Ada-Switzerland.ch

- Publicity Chair
  Dirk Craeynest,
  Ada-Belgium & KU Leuven, Belgium
Dirk.Craeynest at cs.kuleuven.be

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

Previous Editions

Ada-Europe organizes annual international conferences since the early 80’s. This is the 27th 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), and Ghent, Belgium ('22).

Information on previous editions of the conference can be found at http://www.adaeurope.org/confs/ae.

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2023 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 27th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2023)
(V3.1)

Ada-Europe Conference - 6 March Extended Final Deadline

From: Dirk Craeynest
dirk@doka.cs.kuleuven.be

Subject: Ada-Europe Conference - 6 March Extended Final Deadline

Date: Sat, 25 Feb 2023 14:33:00 -0000

Newsgroups: comp.lang.adacomp.lang.ada, fr.comp.lang.adacomp.lang.misc

The recently posted reminder for the Ada-Europe 2023 Conference triggered several requests for extra time. To give all authors the same opportunity to further refine their submission, the organizers decided that the deadline for industrial- and work-in-progress-track abstracts, and for tutorial and workshop proposals will be extended by 1 week until Monday, 6 March 2023. 1+ week remains!

--------------------------------------------------

FINAL UPDATED Call for Contributions

27th Ada-Europe International Conference on Reliable Software Technologies (AEiC 2023)
13-16 June 2023, Lisbon, Portugal

*** EXTENDED FINAL submission DEADLINE 6 March 2023 ***

Industrial- and Work-in-Progress-track:
submit via https://easychair.org/my/conference?conf=aetic2023
select "Industrial Track" or "Work in Progress Track"

Tutorials: submit to Tutorial and Education Chair,
Luís Miguel Pinho <lim @ isep.ipp.pt> subject “[AEiC 2023: tutorial proposal]”

Workshops: submit to Workshop Chair, Frank Singhoff
<singhoff @ univ-brest.fr> subject “[AEiC 2023: workshop proposal]”

For more information please see the full Call for Papers at www.adaeurope.org/conference2023
#AEiC2023 #AdaEurope #AdaProgramming

Our apologies if you receive multiple copies of this announcement.

Please circulate widely.

Dirk Craeynest, AEiC 2023 Publicity Chair
Dirk.Craeynest@cs.kuleuven.be

* 27th Ada-Europe Int. Conf. Reliable Software Technologies (AEiC 2023)
(V4.1)
Ada and Education

Table Game

From: Mockturtle
<framefritti@gmail.com>
Subject: Table game
Date: Tue, 17 Jan 2023 05:56:31 -0800
Newsgroups: comp.lang.ada

Really?!?
https://www.amazon.com/Artana-AAX14001-Lovelace-Babbage/dp/B07WHMG5Y8

The link is for a tabletop game with the
blurb “Play as a pioneer of early computing, like Ada Lovelace or Charles Babbage, to build a program that solves problems for famous patrons like Charles Darwin, Mary Shelley, and more!”. It is priced at 19.98$ and has 4.5/5 stars rating with 50 reviews at the time of this writing. —arm]

Ada-related Resources

[Delta counts are from February 12th to April 5th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 5 Apr 2023 17:36 CET
To: Ada User Journal readership

Ada groups on various social media:
- Reddit: 8,349 (+58) members
- LinkedIn: 3,436 (+18) members
- Stack Overflow: 2,323 (+14) questions
- Telegram: 160 (+1) users
- Gitter: 219 (+68*) people
- Ada-lang.io: 107 (+6) users
- Libera.Chat: 74 (-8) concurrent users
- Twitter: 22 (-10) tweeters
  44 (-5) unique tweets

* Gitter has migrated its messaging to the Matrix open standard. The [5] reference has been updated accordingly.

[1] https://www.reddit.com/r/ada/
[4] https://t.me/ada_lang

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 5 Apr 2023 17:45 CET
To: Ada User Journal readership

Rosetta Code: 924 (+4) examples
40 (+1) developers
GitHub: 763* (=) developers
Alire: 337 (+13) crates
Sourceforge: 240 (=) projects
Open Hub: 214 (=) projects
Codelabs: 54 (=) repositories
Bitbucket: 31 (=) repositories

* This number is unreliable due to GitHub search limitations.

[6] https://www.openhub.net/tags?name=ada
[7] https://git.codelabs.ch/?a=project_index
[8] 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: 5 Apr 2023 17:36 CET
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. —arm]
- TIOBE Index: 28 (-5) 0.42% (-0.18%)
- PYPL Index: 19 (-2) 0.83% (-0.11%)
- IEEE Spectrum (general): 35 (=) Score: 1.16
- IEEE Spectrum (jobs): 33 (=) Score: 0.79

- IEEE Spectrum (trending): 32 (=) Score: 3.95

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

Ada-related Tools

Embedded AVR Ada Setup - Linux Edition

From: Stéphane Rivière <stef@genesix.org>
Subject: ANN: Embedded AVR Ada Setup - Linux edition
Date: Thu, 12 Jan 2023 11:21:30 +0100
Newsgroups: comp.lang.ada

Hi all,
Thanks to the work of Rolf Ebert (AVR-Ada and AVR-Ada to Alire conversion), Fabien Chouteau and AdaCore (GNAT-Ada, GNAT-ARV to Alire conversion, Alire promotion) and their friendly help, here is a tutorial to get the most pleasant environment to develop in Ada on 8-bit AVR targets under Linux.

Based on Alire and GNAT Studio 23 it allows real-time debugging in GNAT Studio as if you were in a native X86_64 environment.

This was an opportunity to get acquainted with Alire while keeping our usual GNAT Studio based environment, which integrates perfectly with Alire. Thanks to the author Alejandro R. Mosteo, who also wrote a very interesting presentation of Alire in AUJ Vol 39, Number 3, Sept 2018, P 189.

This work is part of a more general desire to empower the Ada community with respect to the defunct GNAT CE. We therefore adhere to this new policy of AdaCore. Between this new direction, the arrival of Alire, the availability of many Crates, the first successes of the community in building GNAT Studio independently, the arrival of Rust which is good for the visibility of our favorite language, Ada is certainly entering a new era :)
Short Video on Getting Started with GtkAda in 2023
From: Stephen Merrony
<smerrony@gmail.com>
Subject: A Short Video on Getting Started with Gkidda in 2023
Date: Sat, 14 Jan 2023 01:01:13 -0800
Newsroups: comp.lang.ada

I made a quick video showing how easy it is to get started writing a Gtk application in Ada these days...
https://youtu.be/IforV5hsUvg
[Video running time is 11:03 minutes. —arm]

Gnu Emacs Ada Mode 8.0.4 Released
From: Stephen Leake
stephen.leake84@gmail.com
Subject: Gnu Emacs Ada mode 8.0.4 released.
Date: Wed, 25 Jan 2023 05:27:57 -0800
Newsroups: comp.lang.ada

Gnu Emacs Ada mode 8.0.4 is now available in GNU ELPA.
All Ada mode executables can now be built with Alire (https://alirie.adadev/); this greatly simplifies that process.
gpr_query and gpr_mode are split out into separate GNU ELPA packages. You must install them separately (Emacs install-package doesn’t support “recommended packages” like Debian does).
Ada mode can now be used with Eglot; this is controlled by new variables:

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

gpr_query and gpr_mode are default to wisi, eglot, none

In addition, name completion is provided by eglot if any of the other backends are using eglot; eglot completion is always better than wisi.
The current AdaCore language server (version 23) supports face but not indent. The current version of eglot (1.10) does not support face. The Language Server Protocol does not support statement motion. So for now, eglot + ada_language_server only provides xref and completion.
The AdaCore language server ada_language_server is installed with GNATStudio (which ada-mode will find by default), or can be built with Alire. If you build it with Alire, either put it in PATH, or set gnat-lsp-server-exec.
I have not tested ada-mode with lsp_mode. You can set ada-*backend to other to experiment with that, or tree-sitter, or some other backend. tree-sitter will be fully supported in the next ada-mode release.
The required Ada code requires a manual compile step, after the normal list-packages installation:
cd ~/.emacs.d/elpa/ada-mode-7.3beta* /build.sh
/install.sh
If you have Alire installed, these scripts use it.

Aда and Other Languages
Carbon New Language
From: Gautier Write-Only Address
<gautier_niozez@hotmail.com>
Subject: Carbon
Date: Fri, 22 Jul 2022 14:13:08 -0700
Newsroups: comp.lang.ada

[This thread is a bit dated as it was deemed less of a priority due to space constraints in past issues. —arm] Next attempt to replace C/C++ without really replacing it: Carbon!
You will notice, as usual, a few aspects borrowed from Ada - and one point inspired by Ada 83 (which was relaxed in a later Ada version):)
https://devclass.com/2022/07/20/google-brands-carbon-language-as-experimental-successor-to-c/
https://9to5google.com/2022/07/19/carbon-programming-language-google-cpp/
https://thenewstack.io/google-launches-carbon-an-experimental-replacement-for-c/
From: John Mccabe
<joh@nospam.mccabe.org.uk>
Date: Sat, 23 Jul 2022 09:09:57 -0000

I read that stuff yesterday and, yet again, shook my head in disbelief:-(
The bit where I laughed was where it was claimed that C++ is building technical debt because it’s not changing quickly enough; C++ is currently a mess because it’s changing too quickly! Half-baked, and half-implemented ideas are going into ‘standards’ in the full knowledge that they’ll change again in the next one. Even g++ doesn’t provide 100% support for C++17 (https://gcc.gnu.org/projects/cpp-xxx-status.html#cxx17)!
Carbon is likely to be even worse; every ‘new’ language that promises the earth, without being designed in a rigorous way, ends up with the same problems. Java - I started playing with that in the 90s and got frustrated that every update brought more and more deprecation warnings in. Python - 2.x -> 3.0 was a massive jump (and took years to gain traction) because the ‘designers’ just hadn’t done a very good job to start with! Rust? Mmm
As for the ‘reuse C++ syntax’: why the obsession with that? C++ syntax is really bad! (Semantics, in some cases, are another level - how many languages need a book like “C++ Gotchas”!?)
Aaaaarrrrgghhh!
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Sat, 23 Jul 2022 15:14:15 +0200
> Next attempt to replace C/C++ without really replacing it: Carbon!
We have just learned how dangerous carbon is for our climate. Yet these few privileged keep on pumping it up! (:-))
From: Stéphane Riviére
<stef@genesix.org>
Date: Sat, 23 Jul 2022 15:49:05 +0200
> We have just learned how dangerous carbon is for our climate. Yet these few privileged keep on pumping it up! (:-))
Carbon language bad, green language good
From: Luke A. Guest
<luaguest@archeia.com>
Date: Sun, 24 Jul 2022 10:38:58 +0100
> Next attempt to replace C/C++ without really replacing it: Carbon!
Saw this last week and immediately thought they’d failed on one of their “design goals,” i.e. to be “readable”.
> You will notice, as usual, a few aspects borrowed from Ada - and one point inspired by Ada 83 (which was relaxed in a later Ada version):-)}
Ada and Other Languages

What did they take from Ada?
From: John McCabe
<johng@nospam.mccabe.org.uk>
Date: Tue, 26 Jul 2022 10:31:42 -0700
> What did they take from Ada?
Certainly not the approach to making life easier and less error-prone for developers.

I've got involved in a couple of discussions on their forum, and I'm inclined to think they just want C++ but taken out of the control of ISO/IEC WGs steering committees.

They're pretty much not considering changing any of the aspects of C++ that make it such a heap of junk (IMO, of course), including, but not limited to:
- Arrays
- Enums
- (both of the above when used together: -)
- Symbols - overuse, duplication, inconsistency
- Implicit stuff
- Pretend strong typing
- Forcing developers to deal manually with numeric values that don't fit into an n-byte range, where n is a whole number

It really is shockingly soul-destroying watching all that. What's worse is that, from what I've seen over the years, the new languages that have been developed in a more 'relaxed' way than Ada (well, evolved, really, like Java, Python etc) and have become relatively successful have taken a good 10 years or so to get to that point, yet the discussions on the Carbon forum are all about how to appeal to_current_developers who're used to C++; not_future_developers who, ideally, would_never_be used to C++!

From: Nasser M. Abbasi
<nma@1200.org>
Date: Thu, 28 Jul 2022 18:48:49 -0500

Since Ada has solved these problems a long time ago, then why are people still reinventing the wheel? Why are they not just using Ada? Ada is free software.

Maybe there is something in Ada that prevents it from being widely adopted and used? [...]

From: John McCabe
<johng@nospam.mccabe.org.uk>
Date: Fri, 29 Jul 2022 11:03:36 -0000
> why are people still reinventing the wheel?
Possibly for the same reason that I was so anti-Ada in my early years; it takes getting used to and people are lazy.

Looking at some of the languages that have come out in recent years, it's obvious that people can't be bothered to type much; "fn/"def" (or, even, nothing!) instead of "function/"procedure", "{"/"}" instead of "begin/"end", ",&" instead of "and", ":[/]" instead of "or" (!!!) etc.

From what I can see, some of the "moderators" on that Carbon group don't have much real professional software development experience, so I suspect they really have no clue about what they could achieve with Ada, and have little understanding of some of the constraints that embedded, especially bare-metal, systems impose on what you can and can't include in a program. I'm thinking here of things like heap-unfriendly container classes, such as (in Swift) arrays that are automatically expandable when you append a new item, rather than being fixed size etc.

There also seems to be a bit of an obsession with the time between "empty editor window" and "executable available", rather than "empty editor window" and "executable that actually does what you want!

Also, as Devin says, compiler availability is an issue, from the point of view of actually_using_Ada.

However, from the point of view of creating a new language, the fact that so many people clearly think it has_to be the C/C++ way is quite disturbing, especially since, as I think I mentioned, it's going to be a number of years until any new language really makes its mark, so new languages should be taking future developers into account, not just pandering to the laziness of existing ones!

At this point I think I should make it clear that, although I think Ada has some great features (and I regularly espouse them amongst my colleagues), I don't use Ada in the software I'm developing. I'd like to, but it would take me a lot of time to get back to a level in Ada where I'd be comfortable creating a relatively substantial codebase from scratch. The alternative would be to go and join a team that's already using Ada, but every Ada job I've seen come up locally is to support code that was written in Ada 95; I'd rather be looking at Ada 2005 -> if I was to make that jump.

From: Gautier Write
<gautier_niouzes@hotmail.com>
Date: Fri, 29 Jul 2022 11:59:21 -0700

[...] IMHO the only way to make Ada more popular is to create popular applications with it.

From: Dennis Knorr
<dennis.knorr@gmx.net>
Date: Sat, 6 Aug 2022 16:18:12 +0200

> Maybe there is something in Ada that prevents it from being widely adopted and used?

An opinion from a bystander who wants to like Ada, this is only after I looked the resources and the community up a bit two years ago again. you do not have to agree, it's just my experience and sometimes gut feeling.

* Bad to no marketing
* Sometimes elitism by members of the community/Ada fans
* No modern feeling toolchain (Even Lazarus+Pascal or Gambas has a more modern feeling toolchain, and that says a lot)
* Not much free software built with it
* Not much free software for the toolchain available
* Not much libraries which are ready and easy to use as a beginner
* No modern/up2date books and articles (especially in other languages than in English) seem to be available.
* The free Ada Compiler seems slow and a while back it generated relatively big binaries and the result was not very fast.

Just a few concrete examples to back that up:

* Is there a web playground or repl shell trying or learning/trying Ada or some of its prominent modules?
* There's no modern book in German about modern Ada and its libraries
* There's no syntax highlighting package in vim for Ada
* No exercises like for example Ruby Koans
* It *Looks* like there are no libraries which make it easy use Ada for programming (think json/document formats, http/mail/mime protocols, algorithms or cryptography libraries)

I know there are libraries out there, but they are hard to find, not promoted/marketed and I saw developers (also in other languages, I admit that) talking like, if you do not understand it, you should go back to toy languages like python.

I also know that not all bullet points above are really true to the fullest, but most of them from the outside look like it and also have at least some grain of truth in there.

If someone would write a book in German, how to write Ada and use Scrytoolibrary, Snetworklibrary and how to integrate it in one's favorite development software, this surely would be very interesting to many.

The ONLY thing where I see Ada Marketing in the free software world is FOSDEM. But it is in its own Room. Ada people would need to go out and say: hey, look we also can do good stuff, look, an
https server with letsencrypt support with library in 30 lines.

To be honest, I am curious how the community here will react to it. I mean, I got the Book "Programming in Ada 2005" as a present and I liked it, but after reading the introduction (first 2-3 chapters I think) back then (was like over 15 years ago) I saw no libraries which I can use, and I was not that big a programmer to write them myself.

From: A.J. sianozja@gmail.com>
Date: Sat, 6 Aug 2022 10:48:00 -0700

I agree with you on some of these points. Ada never seemed to be big on marketing, at least outside of specific niches, and from a learning & resources aspect, it took me reading Barnes' Programming in Ada 2012 cover-to-cover to properly grok the language. With that being said, things have been changing a lot in the last two years.

https://learn.adacore.com is a decent resource in that it gives you a little Ada interpreter with code snippets you can test out yourself right in the browser. It's not exactly a "web playground or repl shell" but it's pretty good and seems to support the standard library.

From a library and tooling standpoint, I would check out Aline. It takes a matter of minutes to get from not having any Ada compilers installed at all to compiling your own hello example and there's a lot of libraries already supported (https://adire.adacore.org/crates.html). To bring, for example, Gnacccoll_sqlite, into your project, you would simply just type "alire gnacccoll_sqlite" while in that directory.

Then of course there's the awesome-ada repository that has some nice resources, albeit they seem to mostly be in English: https://github.com/ahlenstein/awesome-ada

From: G.B. <bahuhas@notmyhomepage.invald>  
Date: Sun, 7 Aug 2022 11:08:43 +0200  

> * There's no modern book in German about modern Ada and its libraries

What's the competition, considering C#, Swift, Java or C? I.e., an original work about Ada tools are also focusing on topics of embedded computers, a fairly large and attractive market. JSON or MIME, perhaps even interpreters are present, but I think not central to control stuff near sensors and actuators. How does one compute deterministic responses before a deadline using Node.js?

> From: Dennis Knorr  
< dennis.knorr@gmx.net>  
Date: Mon, 8 Aug 2022 23:38:59 +0200  

> What's the competition, considering C#, Swift, Java or C?

From the absolute amount in English, these languages or Python or Rust have more books. Hell, even Raku has more books.

Python, Kotlin(!), C# have more german books and also more current ones. I bet in five years from now there will be more German books about Carbon than about Ada, even if you include the old ones. [...]

> * No exercises like for example Ruby Koans
> * It *Looks* like there are no libraries which make it easy use Ada for programming (think json/document formats, http/mail/mime protocols, AWS, GNATColl, $ alr with json.
> algorithms or cryptography libraries

Just use one that you can trust. If you need it to be more Ada-ish, ChaCha20 cipher and Poly1305 digest have just been mentioned a few postings ago. If algorithms can address securing the entire computation...

There used to be the PAL, which is the Public Ada Library, easy to find. A bit dated, and reflecting the hype back then, I guess.

I gather that, currently, and in the past, Ada tools are also focusing on topics of embedded computers, a fairly large and attractive market. JSON or MIME, perhaps even interpreters are present, but I think not central to control stuff near sensors and actuators. How does one compute deterministic responses before a deadline using Node.js?

> From: Dennis Knorr  
< dennis.knorr@gmx.net>  
Date: Mon, 8 Aug 2022 23:38:59 +0200  

> What's the competition, considering C#, Swift, Java or C?

From the absolute amount in English, these languages or Python or Rust have more books. Hell, even Raku has more books.

Python, Kotlin(!), C# have more german books and also more current ones. I bet in five years from now there will be more German books about Carbon than about Ada, even if you include the old ones. [...]

> * There's no syntax highlighting package in vim for ada
> :syntax enable

Okay, that I did not know.

> (Does vim feature in a modern feeling tool chain, though?)

Well, okay, Intellij is called more modern of course or VSCode, but you still can craft modern tooling onto vim and it works well.

> From: Randy Brukardt  
< randy@rrsoftware.com>  
Date: Mon, 8 Aug 2022 23:12:44 -0500

> P.S. Nobody writes Ada books these days because they do not sell.

Do *any* programming books really sell? If so, why? :-)

Ada and Other Languages

There are plenty of free, on-line resources for pretty much any programming language. Why pay for something you can get free?

When someone starts talking about books, I think they’re a troll. I can understand complaints about having trouble finding stuff (although Google should find AdaIC.org pretty easily, it’s usually pretty high in Ada results, and most of the good stuff is linked from there), and lack of hype, and so on. But there’s lots of good stuff if one looks (or asks here - if someone knows about here they’re ready to use Ada).

AdaIC has an Ada-specific search engine which hopefully makes it easier to find Ada stuff than a general engine like Google.
Ada Practice

Working with Library Versions

[This thread splintered veered off from the announcement of ada-lang-io towards library management. Other topics have been pruned out. —arm]

From: Paul Jarrett
<jarrett.paul.young@gmail.com>
Date: Tue, 11 Oct 2022 21:21:31 -0700

> Ada-home.com is sort of like that, but it is run by some company and hasn’t been updated in forever.

https://ada-lang.io/ is designed to be updateable for a long time and open to community contributions by being completely open source. There are already multiple people who have permissions to merge changes to help ensure longevity.

ada-lang.io is indexed using Algolia, so the entire site (including the Ada 2022 draft RM) is searchable.

Someone else wrote a tool for searching through all code in Alire crates at https://search.synack.me/

> I am not sure if package manager is a good idea if it does not refer to the target system’s packaging tools, e.g. DEB, RPM, MSI etc. The main problem with that stuff is usually architectural. Most of it is plain aggregation of source code, which is utterly wrong. The very idea to rebuild everything each time on each client is atrocious both with regard to wasting computing resources as well as testing, safety, consistency, interoperability inside the target.

Alire can do additional build steps and other things.

As an application developer, having the code available helps in auditing third-party software for security reasons, build it in a debug configuration for troubleshooting, and also provides the means to locally fix bugs or adapt the library if needed. Isolating libraries and including them with a package manager on a per project basis eases setup also by not making developers have to look up or use multiple installers.

I’ve seen inconsistencies in builds when developers who rely on the system libraries (installed by things like apt) join the project at different times — the earliest developers might be on libfoo-1.2 whereas newer developers are on libfoo-1.4. You don’t run into this problem if the repo points to the applicable dependencies and everyone builds everything locally. It also avoids other problems such as if your system’s package manager doesn’t have a particular library version, and the project builds that library from source. It’s not perfect and there’s other problems that you run into, but it often does help understanding what is being built in the project more clearly. Alire even takes this an entire step further by being able to install and manage the toolchain as well (gprbuild and GNAT).

Package managers also simplify having multiple projects using the same library, but different and possibly incompatible versions on the same system. You get a snapshot in time and a more consistent path to get a build working for new developers, or on a new system. There are limitations due to what systems open source library writers have available to test on, so you shouldn’t just blanket trust code you pull in though, and you should be careful how you use it.

Overall, Alire makes the experience building and developing in Ada for me on Windows, Mac and Linux, considerably simpler and more efficient, by providing the same interface for use across all of these systems.

With the beautiful site styling done by onox, someone pointed to ada-lang.io should be able to download Alire, install a toolchain, make a project and build in less than 15 minutes or so (depending on download and install time). The work done by Fabien and Alejandro, and everyone else who has contributed to Alire to make this happen within the last couple years is absolutely incredible. Combined with Maxin’s fantastic work on the Ada language plugin for Visual Studio Code, it’s a great experience for first-time users of the language.

> Maybe a web forum would be a good idea, because many people nowadays see Usenet newsgroups as an outdated thing. So the fact that the community mostly relies on comp.lang.ada may turn them off.

There’s a dedicated forum now at https://forum.ada-lang.io/
subject to the same version checks as other Alire crates.
> The very idea to rebuild everything
each time on each client is atrocious
both with regard of wasting computing
resources as well as testing, safety, consistency, interoperability inside the
target.
Actually, it’s better for consistency; that’s
why Alire does it.
I don’t understand what you mean by "testing" here; how does compiling from
source affect testing?
Same for "interoperability".
> I’ve seen inconsistencies in builds when
developers who rely on the system
libraries [...] More precisely, an Alire crate can specify
precisely which version of each
dependency it requires/is compatible with.
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Thu, 13 Oct 2022 08:58:16 +0200
> Actually, it’s better for consistency; that’s
why Alire does it.
Consistency is easier to enforce on pre-
built deployments, obviously. Moreover
libraries usually provide integrated checks
and/or have some target platform policy,
e.g. naming and placement conventions.
> I don’t understand what you mean by "testing" here; how does compiling from
source affect testing?
Because one can run tests on pre-built
packages impossible to run on the
sources. For example, network/hardware
protocols in order to test a protocol
implementation one needs complex mock
setups the client simply does not have.
Such tests may run for many hours etc.
> Same for "interoperability".
See above. You cannot run integration
tests on the client, it is just silly.
> [...] You don’t run into this problem if
the repo points to the applicable
dependencies and everyone builds
everything locally.
No difference whether deployment is in
source or pre-built. Dependencies must be
enforced regardless. However it is far
easier to do with pre-built packages.
> More precisely, an Alire crate can
specify precisely which version of each
dependency it requires/is compatible with.
It seems so. Multiple versions at once are
not supported. E.g. when you are working
on two projects both dependent on
different versions of another project:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>A.1</td>
</tr>
<tr>
<td>C</td>
<td>A.2</td>
</tr>
</tbody>
</table>
Or even the same project, e.g. when doing
some migration from one version to
another.
From: Fabien Chouteau
<fabienn.chouteau@gmail.com>
Date: Fri, 14 Oct 2022 01:41:32 -0700
> Multiple versions at once are not
supported. [...] Yes of course, different crates can depend
on different versions of the same crate.
> Or even the same project, e.g. when
doing some migration from one version
to another.
Not sure how you would do that? Link
two different versions of the same library
in an executable? That's not going to
work.
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Fri, 14 Oct 2022 12:05:00 +0200
> Yes of course, different crates can depend
on different versions of the same crate.
It is about whether both A's can be
installed and coexist on the same
machine.
> Not sure how you would do that? Link
two different versions of the same
library in an executable? That's not
going to work.
Same as above. You have B.1 -> A.1 and
B.* -> A.2. You want to install both A.1
and A.2 and work on B.* while checking
on B.1.
In the long gone time of common sense, a
project code management system would
use a virtual file system and map different
parts of the project's graph onto a
structure of folders arranged by versions.
Today one would use something ugly like a
virtual machine or incredibly ugly like a
docker.
From: Stephen Leake
<stephen_leake@stephe-leake.org>
Date: Fri, 14 Oct 2022 04:19:05 -0700
> It is about whether both A's can be
installed and coexist on the same
machine.
In Alire, "installed" means "checked out
the source code into a local directory".
If A depends on a system library that is a
shared object file, and those are different
versions, then it depends on the OS; Debian
can handle this nicely, Windows
only via separate directories and search
paths.
> Same as above. You have B.1 -> A.1
and B.* -> A.2. You want to install
both A.1 and A.2 and work on B.*
while checking on B.1.
And the solution is the same as well.
> [...] a project code management system
would use a virtual file system and map
different parts of the project's graph
onto a structure of folders arranged by
versions.
What prevents that now?
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Fri, 14 Oct 2022 15:05:06 +0200
> What prevents that now?
Nothing except that it is to be done
manually. Why not download a source
archive and bother with anything? It is
Turing-complete, after all... (::-
)
The advantage of a file system is that
developing image will be automated and
consistent. And you would not need
to move any files physically. Alire is
extremely slow because it must pull all
files [and then compile them on top of
that].
Furthermore, a virtual file system shares
duplicates of the same version of the
same file. When you work with naked Git
you must have as many copies as you have
projects. Same applies to virtual machines
and dockers. It is a huge overhead for
nothing.
Moreover, a virtual file system is instant
so long you do not access a file for read or
write. Which is the case for gprbuild,
makes and other tools which use
timestamps and then never look into files.
With a virtual file system you can
automatically check in all files on closing
if it was open for write and never worry
about command-line mess or plug-ins.
Any tool will work out of the box.
From: G.B.
<bauhaus@notmyhomepage.invalid>
Date: Sun, 16 Oct 2022 10:54:57 +0200
> Furthermore, a virtual file system shares
duplicates of the same version of the
same file. When you work with naked
Git you must have as many copies as you
have projects. Same applies to
virtual machines and dockers. It is a
huge overhead for nothing.
Inasmuch as versions are subject to
business, software configuration
management is just work that requires
resources to get it done. Problem solved.
(Well, not for the small shop on a budget,
granted.)
To what extent can static linking make
B.1 and B.2 exist on the same system?
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Sun, 16 Oct 2022 11:20:33 +0200
> Inasmuch as versions are subject to
business, software configuration
management is just work that requires
resources to get it done.
Yes, human resources especially. It is a
self-feeding system that exists in each
organization. It creates problems in order
to justify its continuous growth. Modern
time tools excel at wasting and perfect
outright meaninglessness.

> Problem solved. (Well, not for the small
shop on a budget, granted.)

I cannot say that ClearCase, which did
things more or less right 20 years ago,
was for small business either. (--) AFAIK
it is still available and GNAT Studio
supports it. However, IBM (Rational,
actually) fulfills its existential end goal of
wasting personal and hardware resources
by other, no less efficient, techniques... (:-
))

> To what extent can static linking make
B.1 and B.2 exist on the same system?
To a full extent! (:-))

Sorry, I do not understand your question...

Arrays with Discriminated Task Components

From: Adamagica
<christ-usch.grein@t-online.de>
Subject: Arrays with discriminated task components
Date: Sat, 24 Dec 2022 03:44:27 -0800
Newsgroups: comp.lang.ada

I've got a task type with a discriminant:

```ada
type Index is range 1 .. N;
task type T (D: Index);
```

Now I want an array of these tasks, where
each task knows its identity (the index)
via the discriminant, an
iterated_component_association:

```ada
Arr: array (Index) of T :=
  (for I in Index => ???);
```

How can I do this?

This works with access, but I find this
extremely ugly:

```ada
Arr: array (Index) of access T :=
  (for I in Index => new T (I));
```

Alternatively, I could use the traditional
method with a Start entry with the index
as parameter:

```ada
task type T is
  entry Start (D: Index);
end T;
```

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Sat, 25 Dec 2022 18:16:56 +0200

This seems to work with GNAT, but I'm
not entirely sure if it is legal (could there
be a conflict between the default value of
the task discriminant, which is the same
for all tasks in the array, and the actual
discriminants which are different for each
task in the array?):

```ada
N : constant := 10;
type Index is range 1 .. N;
task type T (D: Index := Index'First);
  -- A default value for D is needed to make
  -- the type constrained, as
  -- required by the Arr declaration below.
function New_T (I : in Index) return T is
  begin
    return T (D => I);
  end T;
end return New_T;
```

Whether this is any less ugly than the
heap allocation method is doubtful.
From: Adamagica <christ-usch.grein@t-
online.de>
Date: Mon, 26 Dec 2022 08:39:23 -0800

Sockets, Streams, and
Element_Arrays

From: Mark Gardner
<mgardner2017@gmail.com>
Subject: Sockets, Streams, and
Element_Arrays: Much confusion
Date: Sat, 31 Dec 2022 14:11:11 +0100

GNAT.Sockets gives me a
Stream_Element_Array, which I can't find
any documentation on how to make use of
other than "You should also be able to get
a Stream, which you should use instead"
(About ten years ago, on this very
newsgroup, somebody said not to use
streams with UDP, or at least not
GNAT.Sockets.Stream).

Adasockets gives me a String, which I can
work with, except it throws away the data
recvfrom gives it, apparently making it
impossible to reply to the querying address.

At this point, I'm half-tempted to make
my own binding, but as I've never done
that sort of thing before, I thought I'd ask
the wisdom of the Usenet if there is a way
to convert a Stream_Element_Array into
the exotic types of Unsigned_16 and
String.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Sat, 31 Dec 2022 14:11:11 +0100

GNAT.Sockets gives me a
Stream_Element_Array [...]

Stream_Element_Array is declared in
Ada.Streams as

```ada
type Stream_Element_Array is
  array (Stream_Element_Offset range <>)
  of aliased Stream_Element;
```

For communication purposes it is an array
of octets. Your datagram is
represented as a Stream_Element_Array or a slice of.

As for streams, yes, it does not make
sense to use them for networking, unless
you override all stream primitives. The
reasons for that are
- non-portability of predefined primitives
- low efficiency for complex data types
- encoding inefficiency as well

You will need to handle some application
protocol artifacts, checksums, counters,
strange encodings, sequence numbers etc.
It is easier to do this directly on the
Stream_Element_Array elements.
And, well, do not use UDP, except for broadcasting. There is no reason to use it. For multicast consider delivery-safe protocols like PGM. For single cast use TCP/IP. (If you need low latency see the socket NO_DELAY option)

From: Mark Gardner  
<magardner2017@gmail.com>  
Date: Sat, 31 Dec 2022 15:50:29 +0200

> For communication purposes it is an array of octets.

According to RM 13.13.1, "Stream_Element is mod implementation-defined" which to me says there is no guarantee that they will be octets, unless this is specified elsewhere?

> You will need to handle some application protocol artifacts, checksums, counters, strange encodings, sequence numbers etc. It is easier to do this directly on the Stream_Element_Array elements.

So, how would I do this directly on the elements? I mean, if it is an octet-array to a string, I expect an element-to-element copy, or type conversion to work, but what about integers? Do I need to do something like My_Int := Unsigned_8(octet(1)) + 2**8 * Unsigned_8(octet(2)); or whatever endianness demands? Or is this the time to learn how to Unchecked_Conversion?

> And, well, do not use UDP, except for broadcasting.

Well, my use case just so happens to be broadcasting, and re-broadcasting data across a binary-tree-like p2p network.

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Sat, 31 Dec 2022 15:16:05 +0100

> According to RM 13.13.1, "Stream_Element is mod implementation-defined"

GNAT.Sockets is GNAT-specific. All GNAT compilers have Stream_Element 8 bits. I can imagine some DSP implementation with Stream_Element of 32 bits. But realistically add pragma Assert (Stream_Element'Size >= 8);

and be done with that.

[..]

From: Jeffrey R.Carter  
<spam.jrcarter.not@spam.acm.org.not>  
Date: Sat, 31 Dec 2022 16:18:50 +0100

> According to RM 13.13.1, "Stream_Element is mod implementation-defined"

The ARM has always tried to ensure that the language could be implemented on any kind of processor. Thus you have implementation-defined separate definitions of Storage_Element and Stream_Element, which need not be the same, and no guarantee that Interfaces contains declarations of Integer_8 or Unsigned_8.

But these days almost everything is byte oriented, so unless you need what you're writing to work on some unusual H/W, you can presume that both of these are bytes, and that Interfaces contains those declarations.

From: Simon Wright  
<simon@pushface.org>  
Date: Sat, 31 Dec 2022 17:39:07 +0000

> About ten years ago, on this very newsgroup, somebody said not to use streams with UDP, or at least not GNAT.Sockets.Stream.

The reasoning behind the recommendation not to use streams with UDP was as follows (there's a faint possibility that it no longer applies!)

If the data type you want to send is e.g. type Message is record  
Id : Integer;  
Val : Boolean;  
end record;

and you create a datagram socket and from that a stream, then use Message'Write to the stream, GNAT will transmit each component of Message separately in canonical order (the order they're written in the type declaration). This results in two datagrams being sent, one of 4 bytes and one of 1 byte.

If you take the same approach at the destination, Message'Read reads one datagram of 4 bytes, and one of 1 byte, and it all looks perfect from the outside. If the destination is expecting a 5 byte record, of course, things won't work so well.

The approach we adopted was to create a 'memory stream', which is a chunk of memory that you can treat as a stream (see for example ColdFrame.Memory_Streams at [1]). With Ada2022, you should be able to use Ada.Streams.Storage.Bounded [2].

Message'Write the record into the memory stream; transmit the written contents as one datagram. I'm guessing with Memory_Stream'Write(Socket_Stream, Buffer);

> To read, create a memory stream large enough for the message you expect

Does this second buffer need to be added? If the datagram arrives (UDP), shouldn't GNAT.Sockets.Stream() be able to handle it?

> You can use gnbind's switch -xdr to "Use the target-independent XDR protocol for stream oriented attributes [...]"

Oh fun, I didn't think of that aspect. Thanks! Would I have to pass it as a command line flag, or would there be some kind of pragma I could use?

Thanks for the help so far, and happy new year!

boblap/11.html#11.12

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Sat, 31 Dec 2022 21:16:18 +0100

> I'm guessing with Memory_Stream'Write(Socket_Stream, Buffer);

No, you create a memory stream object. Then you write your packet into it:  
My_Message'Write (My_Memory_Stream'Access);

Once written you use the accumulated stream contents to write it into the socket. An implementation of a memory-resident stream is very simple. E.g. see:  
http://www.dmitry-kazakov.de/ada/strings_edit.htm#Strings_Edit.Streams

My advice would be not to do this. It is wasting resources and complicated being indirect when 'Write and 'Read are compiler-generated. If you implement
Write and Read yourself, then why not call these implementations directly. It just does not make sense to me. I always wonder why people always overdesign communication stuff.

Build messages directly in a Stream_Element_Array. Use system-independent ways to encode packet data. E.g. chained codes for integers. Mantissa + exponent for real numbers. If you have Booleans and enumerations it is a good idea to pack them into one or two octets to shorten the packets. All this is very straightforward and easy to implement.

You can also consider using some standard data representation format, e.g. ASN.1. An Ada ASN.1 implementation is here:

http://www.dmitry-kazakov.de/ada/components.htm#ASN.1

You describe your message in ASN.1 as an Ada tagged type derived from building blocks. Then you can encode and decode it directly from Stream_Element_Array. I would not recommend that either. ASN.1 is quite overblown.

Happy New Year!

From: philip...@gmail.com
Date: Sat, 31 Dec 2022 14:32:17

I have to disagree here. UDP is perfectly fine for RPC-like (Remote Procedure Call) transactions on a local area network. And it is orders of magnitude easier to implement on microcontrollers than TCP.

An Ada program using UDP to communicate with data collecting microcontrollers makes perfect sense in some contexts. I use it for my Remote I/O Protocol.

The only trick is that the server (or responder, as I like to call it) and client (or initiator) can’t quite use the same code.

Here is my generic package for UDP with fixed length messages:

https://github.com/pmunts/libsimpleio/blob/master/ada/objects/messaging-fixed-gnat_udp.ads

Getting between Stream_Element_Array and a byte array is a pain and I wound up just looping over arrays, copying one byte at a time. If somebody has a better idea, let me know.

From: Jeffrey R. Carter
Date: Sat, 31 Dec 2022 23:49:33 +0100

I’d have thought that building messages directly in a stream element array would be the least desirable way to do it.


Real_Arrays on Heap with Clean Syntax

From: Jim Paloander
Date: Sun, 22 Jan 2023 13:34:18 -0800

Dear Ada lovers,

with stack allocation of Real_Vector (1..N) when N >= 100,000 I get STACK_OVERFLOW ERROR while trying to check how fast operator overloading is working for an expression

X := A + B + C + C + A + B, where A,B,C,X are all Real_Vector (1..N).

So my only option was to allocate on the heap using new. But then I lost the clean syntax

X := A + B + C + C + A + B + and I had to write instead:

X := A.all + B.all + C.all + C.all + A.all + B.all.

This is really ugly and annoying because when you are using Real_Arrays for implementing some linear algebra method which relies heavily on matrix vector products and vector updates, you do need to allocate on the heap (sizes are determined in runtime) and you do need a clean syntax. So, is there any way to simplify my life without using the .all or even without declaring A,B,C,X as access Real_Vector?

Thanks for your time!

From: Joakim Strandberg
Date: Sun, 22 Jan 2023 13:56:27 -0800

Easiest solution is probably to declare a new task and specify the stack size using the Storage_Size attribute. Allocate as much stack space as you need to be able to do the calculations and do all the allocations on the declared task, not on the environment task. You will avoid the unnecessary heap allocations and have nice clean syntax.

From: Dmitry A. Kazakov
Date: Sun, 22 Jan 2023 23:13:44 +0100

You can define "*" on the access type, which should probably be an arena pointer for performance reasons:

Arena : Mark_And_Release_Pool;
type Real_Vector_Ptr is access;
Real_Vector:
for Real_Vector_Ptr'Storage_Pool use Arena;
function "*" (Left, Right : Real_Vector_Ptr)
return Real_Vector_Ptr
begin
if Left'Length /= Right'Length then
raise Constraint_Error;
end if;
return Result := Real_Vector_Ptr :=
new Real_Vector (Left'Range) do
for i in Result'Range loop
Result (i) :=
Left (i) + Right (i - Left'First + Right'First);
end loop;
end return;
end "*";
You can overload that with
function "*" (Left : Real_Vector; Right : Real_Vector)
return Real_Vector_Ptr
begin
if Left'Length /= Right'Length then
raise Constraint_Error;
end if;
return Result := Real_Vector_Ptr :=
new Real_Vector (Left'Range) do
for i in Result'Range loop
Result (i) :=
Left (i) + Right (i - Left'First + Right'First);
end loop;
end return;
end "*";
and with
function "*" (Left : Real_Vector; Right : Real_Vector_Ptr)
return Real_Vector_Ptr;
Then you will be able to write:
X := A + B + C + C + A + B;
Destroy (X);
-- Pop all arena garbage
But of course, the optimal way to work
large linear algebra problems is by using
in-place operations, e.g.
procedure Add (Left : in out Real_Vector; Right : Real_Vector);
end;
Regards,
Dmitry A. Kazakov
http://www.dmitry-kazakov.de
From: Jim Palaoader
<dhmos.altiotis@gmail.com>
Date: Sun, 22 Jun 2023 14:49:09 -0800
> It is my impression that in the Ada
community the preferred way of
working is in general stack only. [...]
I am not sure what you mean. It is quite possible to design a wrapper datatype allocating vectors/matrices in the pool. E.g. Ada’s Unbounded_String is such a thing. Real_Arrays were not designed this way because see above.

> But in C++ you can reproduce the same functionality as Fortran using Expression Templates and Template Metaprogramming.

Nothing prevents you from wrapping Real_Array in a generic way:

```plaintext
generic
with package Real_Array is new
Numerics.Generic_Real_Arrays (<>);
package Generic_Pool_Real_Arrays is
end Generic_Pool_Real_Arrays;
> Perhaps Ada should allow something like that. Because for maintainability reasons the best would be to write the mathematical expressions as close as possible to the mathematical formulas.
```

There is no problem with that as you can define operations on pointers.

```plaintext
From: G.B.
<baubaus@notmyhomepage.invalid>
Date: Mon, 23 Jan 2023 09:39:39 +0100

Are you aware of any libraries similar to Real_Arrays, but who allocated memory internally using heap?
```

The most natural way to work with an array of FPT numbers is for the programmer to declare an array indexed by some index type. Done. If GNAT gets in the way there, it might be worth a note sent to its maintainers. Whenever a programmer is tasked with considering memory allocation, then depending on one’s propensity towards working on memory allocation it is inconvenient and distracting. Math programs don’t make you do this, I think.

Also, std::vector and its relatives shield the programmer from the absurdly clever, yet unreadable memory allocation that needs to be stuffed behind the scenes. More importantly, though, C++ introduced std::move semantics after a few decades of its existence, to address copying when using chains of +. It might be interesting to see Ada’s in-situ construction of return values in comparison.

> Similarly to the Containers.Vector. But Vector has such an awful syntax. There should be something like an indexer [i] similarly to the C++ std::vector to make things simpler

```plaintext
... v.at(k) = 4;
less awful than
v(k) := 4;
```

Another thing: Mathematical notation has ellipsis, thus

```plaintext
A + B + … + Y + Z;
```

Most general purpose languages don’t have ellipsis for this kind of expression. However, even mathematical formulas use what programmers can usually achieve, too. The usual

```plaintext
\sum_k A_k.
```

No “+” at all, and an array of vectors, not single ones. Going further, some like to write

```plaintext
reduce(“+”, A);
```

In Ada, you could have a generic function for this, or use a function pointer.

The .all thing vanishes automatically whenever you want to refer to a particular component of the pointed-at object, as opposed to all of them. So, A.all(K) is the same as A(K). Likewise, .all can be dropped if want to invoke the pointed-at subprogram if it has parameters.

**Broadcast / Iterate to All**

**Connection Objects via Simple Components?**

> It was in 2016:

```plaintext
2016:
> https://archive.fosdem.org/2016/schedule/ event/ada_memory/
```

Thanks Egil, you were faster than me...

I also have a full tutorial at several Ada-Europe conferences. No video, but I can send the slides to those interested.

```plaintext
From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 23 Jan 2023 09:28:46 +0100

I was not sure whether or not it can be avoided with Implicit_Dereference,
> type Accessor (Data: not null access
> Element) is limited private with Implicit_Dereference => Data;
```

If you create a new wrapper type, anyway, then it is easier to define operations directly on that new type.

```plaintext
> Otherwise what you described for operator+ one has to do for every operator overloaded inside Real_Arrays package.
```

You should not use the standard library anyway. It is not intended for large problems, which require specific approaches and methods, like sparse matrices, concurrent processing and so on.

> The optimal way to work large linear algebra problem is what you describe because unfortunately Ada does not allow what Fortran does since 30 years ago or more.

In an effort to better learn network programming in Ada, I’ve been working through the Protohackers Challenges (https://protohackers.com/), and the current challenge (number 3) is to create a chat server.

I am using a TCP Connections Server with Simple Components, specifically a Connection State Machine, but I’ve run into a problem. I’m trying to send a message received via “procedure Process_Packet (Client : in out Server_Connection)” to all connected Clients.

My (potentially incorrect) thought on how to accomplish this is to iterate through all of the clients currently connected, and use Send to send the message received to those clients. I’ve been struggling with how to actually do this though, since I couldn’t use “function Get_Clients_Count (Listener : Connections_Server) return Natural” from within Process_Packets.

Another thought I had could be to just place every message received in a central queue, and then once all of the packets have been received, to then process that queue and send the results to every connected client.
I tried overriding "procedure On_Worker_Start (Listener : in out Connections_Server)". thinking that I could use it to read such a queue, but it never seemed to be called from within my program and I'm still unsure how to iterate through the Connection objects anyway.

Am I approaching this the right way, or am I missing something very obvious? I've read the test files that came with Simple Components, including the data server but couldn't see a way to get each client to interact with each other. If I didn't explain this well enough, please let me know, I'll be happy to clarify.

From: Jeffrey R. Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Wed, 8 Feb 2023 10:55:11 +0100

For an example of this, see the Chattanooga demo that comes with Gnoga

From: Emmanuel Briot
<briot.emmanuel@gmail.com>
Date: Mon, 13 Feb 2023 00:44:01 -0800

I am not sure how familiar you are with Network programming in general (not just as it would be done in Ada). Using a blocking Send could actually kill your performance. You mentioned you would be sending a message to one client after another. Imagine one of the clients has small socket buffers, and is busy doing something else at the moment so not reading your message immediately. If you are sending a large message, your server would only be able to send part of the message, then it would block until the client has read enough that there is space again in the socket buffers to send the rest of the message. That could take ... days. In the meantime, your server is not doing anything else, and no other client gets sent anything...

Instead, you need to use non-blocking sockets. When Send returns, it has sent whatever it could for the moment. You then need to monitor the socket (and all other similar ones) using something like select (which is limited to sockets < 1024, so pretty useless for an actual server in practice) poll (better version of select) or epoll (the best in my opinion). I have written a similar server that has 25000 concurrent clients, and serves them all with 10 worker tasks. That would never fly with blocking sockets.

A similar approach when receiving messages from clients, by the way. The message might have sent only part of its message, so you need to give up temporarily, and come back to it when poll tells you there is something new to read.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 09:30:22 +0100

> Using a blocking Send could actually kill your performance. [...] A similar approach when receiving messages from clients, by the way.

Yes. All networking in Simple components is built on non-blocking sockets (socket select).

P.S. This poses difficulties for users, who see all communication turned upside down being driven by arbitrary socket events rather than by the protocol logic. This was a reason I argued for introducing co-routines with task interfaces in Ada.

From: Emmanuel Briot
<briot.emmanuel@gmail.com>
Date: Mon, 13 Feb 2023 11:55:07 +0100

> > sockets (socket select).
> Have you taken a look at epoll(), on Linux it is so much more natural to use, and so much more efficient in practice. The example I mentioned above (a server with 25 000 concurrent connections) cannot work with select (which only accepts file descriptors up to 1024), and is slow with poll (since the result of the latter is the number of events, and we need to iterate over all registered sockets every time).

> > This was a reason I argued for introducing co-routines with task interface in Ada.

In my own code, I basically provide an epoll-based generic framework. One of the formal parameters is a 'Job_Type' with one primitive operation 'Execute'. The latter is initially called when a new connection is established, and is expected to do as much non-blocking work as it can (Execute is run in one of the worker tasks). When it cannot make progress, it returns a tuple (file descriptor, type_of_event_to_wait_for) to indicate when it needs to be called again in the future, for instance some data became available to read on the specified file descriptor. The intent is that the 'Job_Type' is implemented as a state machine internally.

Of course, a state machine is one of the two ways I know (along with a task) to implement the equivalent of a co-routine in Ada. So I 100% agree with you that co-routines would be very useful in simplifying user code, in particular in the scenario we are discussing here!

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 11:55:07 +0100

> > Have you taken a look at epoll(), on Linux?
> The implementation is on top of GNAT.Sockets, so no.

> > It is so much more natural to use, and so much more efficient in practice.

Well, if there is Linux kernel level support why it is not used in socket select as it is in epoll? I would expect them do that at some point or drop epoll... (-;)

> > [...] The intent is that the 'Job_Type' is implemented as a state machine internally.

Yes, state machine is what I want to avoid. With complex layered protocols it imposes incredible difficulties requiring auxiliary stacks and buffers

The person who developed GNAT.Sockets has left AdaCore a while ago, so "they" (which I assume is what your message was referring to) are actually unlikely to update that. They also have strong concerns about platform-agnostic support, and epoll is linux-specific at this point (likely also BSD). There exist multiple libraries out there that provide an API common to multiple platforms, and that use epoll on linux. Maybe that's what would make sense, but nowadays with Alire, I would expect someone to build a crate there rather than AdaCore modify GNAT.Sockets.

> > Yes, state machine is what I want to avoid. With complex layered protocols it imposes incredible difficulties requiring auxiliary stacks and buffers

Well, I'd like to have special Ada "tasks" acting as co-routines on top of proper tasks yielding when the socket buffer is empty or full.

From: Emmanuel Briot
<briot.emmanuel@gmail.com>
Date: Mon, 13 Feb 2023 03:07:04 -0800

> > Well, if there is Linux kernel level support why it is not used in socket select as it is in epoll?

Because in practice the Linux developers don't get to modify such APIs, which are mandated by Posix, or Unix, or some RFC. So the API for select and poll will *never* change.

epoll is definitely the modern approach on Linux, until of course someone finds something even better. epoll is fully thread safe too, which is very nice when used from Ada. Using select() is totally outdated at this point, and means you can never handle more than 1000 simultaneous clients, and that only if you do not have other file descriptors open (database, files,...)

The person who developed GNAT.Sockets has left AdaCore a while ago, so "they" (which I assume is what your message was referring to) are actually unlikely to update that. They also have strong concerns about platform-agnostic support, and epoll is linux-specific at this point (likely also BSD). There exist multiple libraries out there that provide an API common to multiple platforms, and that use epoll on linux. Maybe that's what would make sense, but nowadays with Alire, I would expect someone to build a crate there rather than AdaCore modify GNAT.Sockets.

> > Yes, state machine is what I want to avoid. With complex layered protocols it imposes incredible difficulties requiring auxiliary stacks and buffers

Well, I'd like to have special Ada "tasks" acting as co-routines on top of proper tasks yielding when the socket buffer is empty or full.
with errors almost intractable either by testing or by formal proofs.

Tell me about auxiliary stacks: - In practice, in my experience, you can have a single incoming buffer which is used by one state, and then another when the first state is no longer active, ... so we do not need to have too many buffers, but that definitely is not trivial. Currently, I have a stack of iterators reading from a socket, buffering on top of that, then decompressing LZ4 data, then decoding our binary encoding to Ada values.

> I'd like to have special Ada "tasks" acting as co-routines on top of proper tasks yielding when the socket buffer is empty or full.

This is an approach we had discussed at AdaCore before I left. There are multiple drawbacks here: the limited stack size for tasks by default (2MB), the fact that entries cannot return indefinite types, the fact that currently those tasks are assigned to OS threads (so too many of them does impact resource usage)....

A colleague had found an external library that would provide several stacks and thus let people implement actual co-routines. We did not do much more work on that, but it was a nice proof of concept, and efficient. I think things are mostly blocked now, as the ARG has been discussing these topics for quite a few years now.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 12:57:19 +0100

> [...] This is an approach we had discussed at AdaCore before I left. [...] My idea is to have these as pseudo-tasks scheduled by the Ada run-time and not mapped onto any OS threads. A proper thread would pick up such a task and run it until it yields. The crucial point is to use the stack of the pseudo-task in place of the thread's stack or backing it up and cleaning the portion of the stack at the point of yielding, whatever.

> [...] the ARG has been discussing these topics for quite a few years now. I have an impression that ARG's view on co-routines totally ignores the use case of communication stacks and other cases state machines show their ugly faces...

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Mon, 13 Feb 2023 15:22:19 +0200

[snip discussion of network programming details, retain discussion about co-routines]

So your co-routines would (1) have their own stack and (2) be independently schedulable, which implies (3) having their own execution context (register values, instruction pointer, etc.) How is that different from the Ada concept of a "task"? How could the ARG separate between a "task" and a "co-routine" in the Ada RM?

There exist Ada compilers and run-times where the tasking concept is implemented entirely in the run-time system, without involving the underlying OS (if there even is one). That approach was mostly abandoned in favour of mapping tasks to OS threads, which makes it easier to use potentially blocking OS services from tasks without blocking the entire Ada program.

So is your problem only that using OS threads is less "efficient" than switching and scheduling threads of control in the run-time system? If so, that seems to be a quality-of-implementation issue that could be solved in a compiler-specific way, and not an issue with the Ada language itself.

The point (from Emmanuel) that task entries cannot return indefinite types is certainly a language limitation, but seems to have little to do with the possible differences between tasks and co-routines, and could be addressed on its own if Ada users so desire.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 13 Feb 2023 16:10:15 +0100

> So your co-routines would (1) have their own stack and (2) be independently schedulable, which implies (3) having their own execution context (register values, instruction pointer, etc.)

Sure. You should be able to implement communication logic in a natural way:

1. Read n bytes [block until finished]
2. Do things
3. Write m bytes [block until finished]
4. Repeat
   > How is that different from the Ada concept of a "task"?
   > How could the ARG separate between a "task" and a "co-routine" in the Ada RM?

Syntax sugar does not bother me. I trust ARG to introduce a couple of reserved words in the most annoying way... (:-)

> > Is your problem only that using OS threads is less "efficient" than switching and scheduling threads of control in the run-time system?

This too. However the main purpose is control inversion caused by callback architectures. A huge number of libraries are built on that pattern. This is OK for the library provider because it is the most natural and efficient way. For the user implementing his own logic, be it communication protocol, GUI etc. it is a huge architectural problem as it distorts the problem space logic. So the goal is to convert a callback/event driven architecture into plain control flow.

> > If so, that seems to be a quality-of-implementation issue that could be solved in a compiler-specific way, and not an issue with the Ada language itself.

In Ada 83 there was no way to pass a procedure as a parameter. We used a task instead... (:-)

But sure, a possibility to delegate a callback to an entry call without intermediates is certainly welcome.

 [...] From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 13 Feb 2023 16:43:31 +0100

> > that task entries cannot return indefinite types is certainly a language limitation.

That's what Holders are intended for... (changing indefinite types into a definite one)

From: Jeremy Grosser
<jJeremy@synack.me>
Date: Mon, 13 Feb 2023 08:40:05 -0800

> epoll is definitely the modern approach on Linux, until of course someone finds something even better.

For high performance networking, io_uring [1] is the new kid on the block, but the API involves a scary amount of pointer manipulation, so I'm not convinced that it's safe to use yet.

While epoll is thread safe, there are some subtleties. If you register a listening socket with epoll, then call epoll_wait from multiple threads, more than one thread may be woken up when the socket has a waiting incoming connection to be accepted. Only one thread will get a successful return from accept(), the others will return EAGAIN. This wastes cycles if your server handles lots of incoming connections. The recently added (kernel >=4.5) EPOLLEXCLUSIVE flag enables a mutex that ensures the event is only delivered to a single thread.

> > They also have strong concerns about platform-agnostic support, and epoll is linux-specific at this point (likely also BSD). [...] On BSD, the kqueue [2] API provides similar functionality to epoll. I believe kqueue is a better design, but you use what your platform supports.

libev [3] is the library I see used most commonly for cross-platform event driven I/O. It will use the best available polling functions on whatever platform it's compiled for. Unfortunately, it's
composed mostly of C preprocessor macros.

I’ve already written an epoll binding [5] that’s in the Alire index. GNAT.Sockets provides the types and bindings for the portable syscalls.

For the Protohackers puzzles, I’ve written a small evented I/O server using those bindings [6]. Note that this server does not use events for the send() calls yet, which may block, though in practice it isn’t an issue with the size of the payloads used in this application. I do plan to refactor this to buffer data to be sent when the Writable (EPOLLOUT) event is ready.

So far, I’ve found tasks and coroutines to be unnecessary for these servers, though coroutines would make it possible to implement Ada.Streams compatible Read and Write procedures, providing a cleaner interface that doesn’t expose callbacks to the user.

[1] https://lwn.net/Articles/776703/
From: philip...@gmail.com
Date: Mon, 13 Feb 2023 17:55:52 -0800

A coroutine or is it meant for something else?

As I understand it, the parallel execution constructs (parallel blocks and parallel loops) in Ada 2022 are meant to parallelize computations using multiple cores -- that is, real parallelism, not just concurrency.

The Ada2022 RM describes each parallel computation in such a parallel construct as its own thread of control, but all operating within the same task, and all meant to be independent of each other. For example, a computation on a vector that divides the vector into non-overlapping chunks and allocates one core to each chunk.

Within a parallel construct (in any of the parallel threads) it is a bounded error to invoke an operation that is potentially blocking. So the independent computations are not expected to suspend themselves, thus they are not co-routines.

The parallelism in parallel blocks and parallel loops is a "fork-join" parallelism. In other words, when the block or loop is entered all the parallel threads are created, and all those threads are destroyed when the block or loop is exited.

So they are independent threads running "in" the same task, as Dmitry wants, but they are not scheduled by that task in any sense. The task "splits" into these separate threads, and only these, until the end of the parallel construct.

Moreover, there are rules and checks on data-flow between the independent computations, meant to exclude data races. So it is not intended that the parallel computations (within the same parallel construct) should form pipes or have other inter-computation data flows.

Ada Array Contiguity

From: Rod Kay <rodakay5@gmail.com>
Subject: Ada array contiguity.
Date: Mon, 20 Feb 2023 00:34:55 +1100
Newsgroups: comp.lang.ada

I’ve been told that Ada array elements are not guaranteed to be contiguous unless the ‘Convention C’ aspect is applied.

Is this correct?

From: J-P. Rosen <rosen@adalog.fr>
Date: Sun, 19 Feb 2023 15:28:23 +0100

The strength of Ada is that it protects you from all implementation details, thus allowing compilers to choose the most efficient implementation. Therefore, the answer is yes.

(BTW: try to find a definition of "contiguous". At byte level? At word
level? What if the element does not fill a byte?)

From: Niklas Holsti  
<niklas.holsti@tidorum.invalid>  
Date: Sun, 19 Feb 2023 16:59:42 +0200  
> Therefore, the answer is yes.

I tried to find a rule on "contiguity" in the Ada 2022 RM, but failed. Can you point to one? Perhaps this rule is a consequence of C standard rules for arrays (pointer arithmetic), and the general idea that Ada should allow Convention C for a type only if that type is really compatible with the C compiler (in question).

For a constrained array type I would choose to specify the size of the component type, and the size of the array type to be the length of the array times the component size. That should (also) ensure that the elements are stored contiguously (if the Ada compiler accepts this size specification).

It seems (RM B.3(62.4/3)) that Ada compilers are not required to support Convention C for unconstrained array types. RM B.3 (Interfacing with C/C++) declares such types with the Pack aspect, but that may or may not (AIUI) give a contiguous representation.

> (BTW: try to find a definition of "contiguous". At byte level? At word level? What if the element does not fill a byte?)

Indeed. But it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Sun, 19 Feb 2023 16:08:09 +0100  
> it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

Rather: the bit offset of an element is a linear function of its position.

From: J-P. Rosen  
/rosen@adalog.fr>  
Date: Sun, 19 Feb 2023 18:10:44 +0100  
> it seems to me that Arr'Size = Arr'Length * Comp'Size is the meaning usually intended for programming purposes.

Certainly not if Comp'Size is not an integer number of bytes.

From: Niklas Holsti  
<niklas.holsti@tidorum.invalid>  
Date: Sun, 19 Feb 2023 19:54:13 +0200  
> Certainly not if Comp'Size is not an integer number of bytes.

I'm not so certain. By choosing various roundings-up of the component size, you can choose between "bit-contiguous", "byte-contiguous", etc.

For example, bit-contiguous with 2-bit components:

```ada
type Comp is (A, B, C, D) with Size => 2;

type Arr is array (1 .. 10) of Comp
with Pack, Size => 10 * Comp'Size;
```

Nybble-contiguous with Comp'Size => 4, byte- (octet-) contiguous with Comp'Size => 8, etc.

(However, I haven't checked that eg. GNAT does the "right thing" with such Size clauses, just that it accepts them. It does require the Pack aspect for the array type when Comp'Size is not a multiple of 8.)

> Rather: the bit offset of an element is a linear function of its position.

That is ordering by index, but not contiguity: there may still be gaps between elements. However, I assume you meant that the slope of the linear function equals the component size, and then it includes contiguity.

The relationship of index order to memory-location order is certainly an aspect that should be considered when interfacing to C or HW.

Pet peeve: on more than one occasion I have been disappointed that Ada representation clauses do not let me specify the index-order of packed array elements in a word, relative to the bit-numbering, and I have had to fall back to using several scalar-type record components, c1 .. c7 say, instead of one array-type component, c(1..7).

From: Dmitry A. Kazakov  
<mailbox@dmitry-kazakov.de>  
Date: Sun, 19 Feb 2023 20:05:28 +0100  
> That is ordering by index, but not contiguity: there may still be gaps between elements. [...] No gaps = packed = the most dense representation.

Contiguity is rather that the gaps are regular and can be considered a part of each element. E.g. a video buffer with strides is not contiguous.

> The relationship of index order to memory-location order is certainly an aspect that should be considered when interfacing to C or HW.

An definition of contiguous array equivalent to linearity is that the array body representation is isomorphic to slicing.

> Pet peeve [...] This is as blasphemous as asking for n-D slices... (-;)

From: Jeffrey R. Carter  
<spam.jrcarter.not@spam.acm.org.not>  
Date: Sun, 19 Feb 2023 23:02:36 +0100

> I've been told that Ada array elements are not guaranteed to be contiguous unless the 'Convention C' aspect is applied.

The ARM says little about how the compiler represents objects in the absence of representation clauses. However, ARM 13.7(12) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-13-7-1.html#I5633) says, "Storage_Array represents a contiguous sequence of storage elements."

ARM 13.9(17/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-13-9.html#I5679) says that a compiler that supports Unchecked_Conversion should use a contiguous representation for certain constrained array subtypes.

Using convention Fortran should also ensure a contiguous representation, add can apply (unlike convention C) to multidimensional arrays.

From: J-P. Rosen  
/rosen@adalog.fr>  
Date: Mon, 20 Feb 2023 08:12:41 +0100  
> type Comp is (A, B, C, D) with Size => 2;  
> type Arr is array (1 .. 10) of Comp  
> with Pack, Size => 10 * Comp'Size;  
> Nybble-contiguous with Comp'Size => 4, byte- (octet-) contiguous with Comp'Size => 8, etc.

Of course, if you add representation clauses, the compiler will obey them. But the OP's question was whether it was /guaranteed/ to have contiguous representation, and the answer is no - for good reasons.

From: Rod Kay  
<rodakay5@gmail.com>  
Date: Thu, 2 Mar 2023 06:22:25 +1100  
Thank you all for the replies.

To summarise then, contiguity is not guaranteed unless the array is of convention C, convention Fortran or representation clauses are applied.

**Ada.Containers.Vectors Capacity**

From: Rod Kay  
<rodakay5@gmail.com>  
Subject: Is this a compiler bug?  
Date: Sun, 19 Mar 2023 17:17:20 +1100  
Newsgroups: comp.lang.ada

Came across this during a port of the Box2D physics engine.

It's a generic Stack package using 'ada.Containers.Vectors' to implement the stack.

One generic parameter is the 'Initial_Capacity of the stack, used in the 'to_Stack' construction function, via the Vectors 'reserve_Capacity' procedure.
In the 'to_Stack' function, the Capacity is reserved correctly but in the test program when the stack is created and assigned to a variable, the capacity is 0.

Here is the (very small) source code ...
https://gist.github.com/charlie5/7bd4863227a510f834c2bd781dd50ba

The output I get with GCC 12.2.0 is ...
[rod@orth bug]$ /stack_bug
to_Stack ~ Initial Capacity: 256
to_Stack ~ Before reserve: 0
to_Stack ~ After reserve: 256
stack_Bug ~ Actual Capacity: 0

Regards.
From: Jeffrey R Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 19 Mar 2023 11:33:50 +0100

I think this is acceptable behavior. See ARM A.18.2 (147.19/3, 147.20/3, & 147.b/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-A-18.2.html). The first two sections define the behavior of procedure Assign, while the last states "Assign(A, B) and A := B behave identically."

Assign (A, B) only changes the capacity of A if A.Capacity < B.Length.

So if the compiler does not use build-in-place for the initialization of the variable, then the assignment of the function result should not change the capacity of the variable from its (apparent) default of zero (there is, of course, no requirement for the capacity of a default-initialized vector).

The discussion of capacities for vectors is only meaningful for a subset of possible implementations, so messing with capacities may have no meaningful effect at all.


From: Rod Kay <rodakay5@gmail.com>
Date: Mon, 20 Mar 2023 13:24:40 +1100

Thank you, Jeffrey, for the detailed reply. I'm now using a limited record with an extended return for 'build-in-place' initialisation and am getting the behavior I desired.

Why Don't All Initialising Assignments Use 'build-in-place'?

From: Rod Kay <rodakay5@gmail.com>

Subject: Why don't all initialising assignments use 'build-in-place'?
Date: Tue, 21 Mar 2023 23:06:03 +1100

I'm sure there must be a good reason. All I can think of is that it may somehow break backwards compatibility wrt controlled types (a vague stab in the dark).

Any thoughts?
From: Randy Brukarst
<randy@rssoftware.com>
Date: Sat, 25 Mar 2023 03:59:14 -0500

(1) Didn't want to make work for implementers.
(2) You shouldn't be able to tell (since it is required for all cases involved finalization). Finalization is the only way to inject user-defined code into the initialization process.
(3) True build-in-place can be expensive and complex (especially for array types).
(4) Build-in-place requires functions compiled to support it (must pass in the place to initialize into). That might not be the case (especially if a foreign convention is involved). Also see (3) - an implementation might have a cheaper way to return some types that doesn't support build-in-place.

There's probably more, those are off the top of my head. If it is cheap, it would be silly for an implementation to do anything else. (Don't ask what Janus/Ada does. ;-) Otherwise, most people want the fastest possible code.

From: Rod Kay <rodakay5@gmail.com>
Date: Sun, 26 Mar 2023 16:10:33 +1100

Thanks, Randy. I somehow imagined that build-in-place would be faster :/.

So using 'extended return' *everywhere* would decrease performance, I guess.

From: Jeffrey R Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 26 Mar 2023 12:41:00 +0200

> So using 'extended return'
*everywhere* would decrease performance, I guess.

You seem to think that using an extended return requires building in place. This is not required by the ARM.

"Built in place" is defined in ARM 7.6 (17.1.3-17.p/3) (http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-7.6.html#14005). An initial value is required to be built in place when

1. The object (or any part of the object) being initialized is immutably limited
2. The object (or any part of the object) being initialized is controlled and the initialization expression is an aggregate

In all other cases, it is up to the compiler to decide whether or not to build in place.

This holds regardless of the the kind of return statement used if the initialization expression is a function call.

Thus the initialization of an immutably limited object is done in place even if the initialization expression is
* an aggregate
* a function call with a simple return statement

while the initialization of an integer object may be by copy even if the initialization expression is a function call with an extended return statement.

From: Rod Kay <rodakay5@gmail.com>
Date: Mon, 27 Mar 2023 15:44:33 +1100

> You seem to think that using an extended return requires building in place. This is not required by the ARM.

Yes, I did rather think that. Appreciate the correction.

Assignment Access Type with Discriminants

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Subject: Assignment access type with discriminants
Date: Wed, 22 Mar 2023 10:19:28 +0100

I stumbled on a curious fact.

The value of an object with a discriminant can be changed to a value with a different discriminant if the type's discriminants are defaulted.

Right?

Wrong! Not through an access type!

procedure Test is
type F is (F1, F2, F3);
type Foo (K : F := F1) is record
case K is
when F1 =>
X1 : Integer;
when F2 =>
X2 : Float;
when F3 =>
X3 : String (1..2);
end case;
end record;
type Foo_Ptr is access all Foo;
X : aliased Foo;
P : Foo_Ptr := X'Access;
begin
X := (F2, 1.0); -- OK
P.all := (F1, 3.); -- Error!
end Test;

Is this a compiler bug or intentional language design? Any language lawyers?

From: Björn Landin <bml@nowhere.com>
Date: Wed, 22 Mar 2023 10:31:58 +0100

> I stumbled on a curious fact. [...] Is this a compiler bug or intentional language design? Any language lawyers?
unconstrained if the designated subtype is an ... discriminated subtype; otherwise, it is constrained.

4.8(6/3) If the designated type is composite, then ... the created object is constrained by its initial value (even if the designated subtype is unconstrained with defaults).

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Thu, 23 Mar 2023 20:55:48 +0200
I get

Execution of Test terminated by unhandled exception
raised CONSTRAINT_ERROR:

test.adb:18 discriminant check failed
Call stack traceback locations:
0x402c33 0x402b27 0x7f333b5cfd8e
0x7f333b5efc3e 0x402b63
0xfffffffffffffffe

bni@hp-t510:~usr2$ gnatls -v
GNATLS Pro 22.2 (20220605-103)
Linux 64bit - ubuntu 22.04
So it is (also) present on that platform at least

From: G.B.
<bauhaus@notmyhomepage.invalid>
Date: Wed, 22 Mar 2023 15:10:44 +0100
Some experiments point at the general access type.

type Foo_Ptr is access Foo; -- sans `all
X : Foo;
P : Foo_Ptr := new Foo;
type Foo1 is new Foo_Ptr(K => F1);
beg
X := (F2, 1.0); -- OK
P.all := (F1, 3); -- _no_Error!
Foo1(P).all := (F1, 3);
end Test;

(Doesn't rejection for general access types seem reasonable if assignment would otherwise require adjusting the storage layout of a variable, including all access paths to components? Just guessing.)

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Thu, 23 Mar 2023 12:51:03 +0100
I do hope, this answers the question:

> 3.10(14/3) ... The first subtype of a type defined by ... an access_to_object_definition is unconstrained if the designated subtype is an ... discriminated subtype; otherwise, it is constrained.

What do you infer from this, relating to Dmitry's original example code and the error? The "first subtype ... defined" here is the access subtype, and I don't see how that affects an assignment /via/ this access subtype to the accessed object.

(It is not clear to me how an access subtype that is constrained differs from one that is unconstrained. Can someone clarify?)

> 4.8(6/3) If the designated type is composite, then ... the created object is constrained by its initial value (even if the designated subtype is unconstrained with defaults).

That rule applies to objects created by allocators, but the original example code has no allocators (some later variants do). The object in question is created by a declaration (which includes the "aliased" keyword), not by an allocator.

Also, AARM 3.10 contains the following notes on "Wording Changes from Ada 1995":

26.d/2 [A95-00363-01] Most unconstrained aliased objects with defaulted discriminants are no longer constrained by their initial values. [...] 26.k/2 [A95-00363-01] The rules about aliased objects being constrained by their initial values now apply only to allocated objects, and thus have been moved to 4.8, "Allocators".

This seems to mean that aliased objects created by declarations are /not/ constrained by the initial value, so it should be possible to change the discriminant. This seems to be a change from Ada 95 to Ada 2005. I don't see why that change could not be done via an access to the object.

I added some output to Dmitry's original code, with this result:

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>

Test;
An access value is always constrained by its initial value; this is necessary because of constrained access subtypes.

But constrained access subtypes are not allowed for general access types like Foo_Ptr in the example.

Here is a slightly modified version of your example:

> Without this rule, PF2.all would now designate a value whose discriminant is F1!

This error is understandable and valid, because now P.all is PF2.all which is an allocated object and therefore constrained by its initial value with K = F2.

But why should the same apply when P designates X, which is unconstrained? Is it just an optimization (in the RM) so that a general access value does not have to carry around a flag showing whether its designated object is constrained or unconstrained?

Perhaps it would be better to make the assignment P := PF2.all'Access illegal, because it in effect converts a constrained access value (PF2) to an unconstrained access subtype (P), and so in some sense violates the prohibition of constrained subtypes of general access types.

Perhaps it would be better to make the assignment P := PF2.all'Access illegal [...] Yes this is a substitutability violation. Such cases never go without a punishment. In this case it is an implementation overhead.

Consider:

```ada
procedure Set (Destination : in out Foo; Source : Foo) is
   Destination := Source;
end Set;
```

The compiler cannot implement Set in a natural way, because Destination might be arbitrarily constrained by the caller. E.g. when the actual for Destination is P.all. So, the constraint must be passed together with the actual. Quite a burden.

```
From: J-P. Rosen <rosen@adalog.fr>
Date: Fri, 24 Mar 2023 10:41:20 +0100
> But why should the same apply when P designates X, which is unconstrained?

I didn't dig in the RM in all details, but I think this comes from the fact that being constrained (always) is a property of the pointer (more precisely, its subtype), not of the pointed-at object.
```

```
From: Randy Brukardt
<randy@rrsoftware.com>
Date: Sat, 25 Mar 2023 03:51:07 -0500

> But why should the same apply when P designates X, which is unconstrained? [...]

The rule is question is 4.1(9/3):

```
If the type of the name in a dereference is some access-to-object type T, then the dereference denotes a view of an object, the nominal subtype of the view being the designated subtype of T. If the designated subtype has unconstrained discriminants, the (actual) subtype of the view is constrained by the values of the discriminants of the designated object, except when there is a partial view of the type of the designated subtype that does not have discriminants, in which case the dereference is not constrained by its discriminant values.
We have to do that so as otherwise the access value would have to carry a designation as to whether the object was allocated or not.
This rule was inherited from Ada 83.
IMHO, this rule is stupid. It's even more stupid with the hole for types that have partial views without discriminants. The *proper* solution is to get rid of the rarely used and mostly useless access constraints, and then have no extra restrictions on access values. But that's considered too incompatible.
```

```
Ada in Jest
Ada Lovelace Cosplay

From: Mockturtle <framefritti@gmail.com>
Subject: Ada Lovelace cosplay
Date: Mon, 16 Jan 2023 09:48:58 -0800
Newsgroups: comp.lang.ada

Well, yes, someone cosplayed Ada...
https://blog.adafruit.com/2013/10/24/from-scratch-ada-lovelace-costume/
```