ADA USER JOURNAL

Volume 44
Number 3
September 2023

Contents

Quarterly News Digest

Alejandro R. Mosteo

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

Contents

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

Preface by the News Editor

Dear Reader,

Thanks to the efforts of a few Ada enthusiasts, it is now possible to peruse on-line the complete High Order Language Phase 1 reports for the original four candidate languages, of which Ada would eventually emerge. During this phase, language proposals addressed the IRONMAN requirements without providing a prototype implementation. Find the link on [1]; from my initial cursory read, there is plenty of good stuff in there!

If you are an Ada and macOS user and have still not been affected by the problem reported on [2], you probably want to be aware of it, although, hopefully, Apple will have fixed it by the time you read this digest.

Finally, on the practical side, I want to highlight the emerging consensus that when using `Text_IO` to read files, you should rely on the End_Error exception rather than on the End_Of_File subprogram. This is a rather unconventional conclusion, in which the exception becomes the ordinary; find the rationale in [3].

Sincerely,
Alejandro R. Mosteo.

[1] "Common HOL Phase 1 Reports", in Ada-related Resources.

[2] "MacOS: Best Not Upgrade to Xcode/CLT 15.0", in Ada and Operating Systems.

[3] "Get Character and Trailing New Lines", in Ada Practice.

Ada-related Events

October Ada Monthly Meetup 2023

From: Fernando Oleo Blanco <irvise_ml@irvise.xyz>
Subject: Re: Ada Monthly Meetup 2023
Date: Wed, 13 Sep 2023 21:30:10 +0200
Newsgroups: comp.lang.ada

I would like to announce the October Ada Monthly Meetup which will be taking place on the 7th of October at 13:00 UTC time (15:00 CET). As always, the meetup will take place over at Jitsi. Hopefully this time I will not have the same amount of technical issues…

If someone would like to propose a talk or a topic, feel free to do so! I will be talking about FOSDEM, but only for a couple of minutes, since the dates have already been announced (the same as every year, first weekend of February).

Here are the connection details from previous posts: The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is [1]. The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules". I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

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

From: Rod Kay <rodakay5@gmail.com>
Date: Sat, 23 Sep 2023 20:06:51 +1000

I might give a small talk on swig4ada, if that would be of interest.

If it's possible to request talks, I'd love to see an overview of 'Pragmarc' and 'Simple Components' by Jeffrey and Dmitry.

I used the neural net component of Pragmarc many years ago but have not kept up with Pragmarc's development. Also, I've been meaning to look into Simple Components for quite a while.

CfC 28th Ada-Europe Int. Conf. Reliable Software Technologies

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: CfC 28th Ada-Europe Int. Conf. Reliable Software Technologies
Date: Tue, 26 Sep 2023 12:06:18 -0000
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

[CfP is included in the Forthcoming Events Section —arm]

Ada-related Resources

[Delta counts are from July 28th to October 10th. —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 10 Oct 2023 18:22 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 8_422 (+51) members [1]
- LinkedIn: 3_454 (+6) members [2]
- Stack Overflow: 2_365 (+20) questions [3]
- Gitter: 229 (-1) people [4]
- Telegram: 158 (-1) users [5]
- Ada-lang.io: 146 (+13) users [6]
- Libera.Chat: 72 (-1) concurrent users [7]

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

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

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

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

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

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

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

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 10 Oct 2023 18:25 CET
To: Ada User Journal readership

GitHub: 1000* (+13) developers [1]

Rosetta Code: 940 (-1) examples [2]

38 (-3) developers [3]

Alire: 370 (+7) crates [4]

Sourceforge: 247 (+4) projects [5]

Open Hub: 214 (=) projects [6]

Codelabs: 57 (=) repositories [7]

Bitbucket: 37 (+6) repositories [8]

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

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

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

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

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

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

[6] https://www.openhub.net/tags?names=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: 28 Jul 2023 14:53 CET
To: Ada User Journal readership

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

- TIOBE Index: 23 (=) 0.77% (=) [1]
- PYPL Index: 16 (=) 1.04% (-0.02%) [2]
- Stack Overflow Survey: 42 (=) 0.77% (=) [3]
- IEEE Spectrum (general): 36 (-1) Score: 0.0107 (new) [4]
- IEEE Spectrum (jobs): 29 (+4) Score: 0.0173 (new) [4]
- IEEE Spectrum (trending): 30 (+2) Score: 0.0122 (new) [4]

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

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

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

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

IRC Is Still Alive

From: Luke A. Guest <laguest@archeia.com>
Subject: Reminder that the IRC is still alive
Date: Wed, 16 Aug 2023 19:05:10 +0100
Newsgroups: comp.lang.ada

About twice a year we try to advertise the #ada channel on the Libera IRC network. The channel continues to be active and friendly. These days it averages about 63 users at a time, large enough to support lively and informative discussions but small enough so it's not a madhouse. The user numbers did suffer on the move when Freenode imploded.

Topics range all over the map, from building the latest GNAT to writing an OS in Ada to daily Ada programming issues to how to use PolyORB to use the Distributed Systems Annex. The stated topic is discussing Ada in the context of free and open-source software, but commercial users are equally welcome.

So fire up your favorite IRC client and come join us! The network is homed at irc.Libera.chat, but has servers all over the world. Visit www.Libera.chat on the web for details. Hope to see you soon!

Common HOL Phase 1 Reports

From: Luke A. Guest <laguest@archeia.com>
Subject: Common HOL Phase 1 Reports
Date: Wed, 30 Aug 2023 14:48:48 +0100
Newsgroups: comp.lang.ada

[This post refers to the Common High-Order Language program, started in 1975, of which Ada would be the outcome. DTIC stands for Defense Technical Information Center. —arm]

Edward Fish has managed to get the DTIC to scan in the other language's reports.

This has been a combined effort between a few of us on IRC to try to get the other two languages, blue and yellow released so we can see what could've happened.

This report contains all 4 language reports.

https://apps.dtic.mil/sti/trecms/pdf/ADB950587.pdf

From: Luke A. Guest <laguest@archeia.com>
Date: Wed, 30 Aug 2023 21:20:28 +0100

Now we have all 5 reference docs out in the open, yes 5. I want to ask the people who were there at the time, were there any analyses of the Tartan language? Design: https://apps.dtic.mil/sti/citations/ADA062815

From: Stéphane Rivière <stef@genesix.org>
Date: Thu, 31 Aug 2023 06:32:50 +0200

Fascinating (C) Spock.

Big up for this work!

US Government Looking into Memory Safe Programming

From: Ajdude <aj@ianozi.com>
Subject: US Government looking into memory safe programming
Date: Sun, 24 Sep 2023 22:28:56 -0000
Newsgroups: comp.lang.ada

The US Government is requesting information on adoption of memory safe programming languages and open-source software security. They're currently taking comments until October 9th. I think this is a good opportunity to help bring Ada back into the spotlight.

https://www.federalregister.gov/documents/2023/08/10/2023-17239/request-for-information-on-open-source-software-security-areas-of-long-term-focus-and-prioritization

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 25 Sep 2023 08:52:33 +0100

History is repeating itself. How long before they relax the requirements and idiots say "we can use C again, yay!"?

From: Stéphane Rivière <stef@genesix.org>
Date: Mon, 25 Sep 2023 11:59:57 +0200

> History is repeating itself.

+1

> How long before they relax the requirements and idiots say "we can use C again, yay!"?

By the time they discover Rust?

From: J-P. Rosen <rosen@adalog.fr>
Date: Mon, 25 Sep 2023 12:38:54 +0200

> By the time they discover Rust?

Or when they realize that there is only one Rust compiler, and therefore that a single compiler virus could ruin the whole defense system.

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Mon, 25 Sep 2023 17:55:08 +0200

> Or when they realize that there is only one Rust compiler, and therefore that a single compiler virus could ruin the whole defense system.

Maybe, given the emphasis on tools, verification and best practices, they might consider sub-languages, or profiles, of several existing languages.

It's not like memory-safety cannot be made available in languages other than Rust, I should think? Though, it seems to me that Rust has so much better market-aware development strategies than any other language since C, outside Microsoft's or Apple's areas of sales.

Also, I understand that Linux kernel development is steered towards Rust and LLVM. So, they have decided not to go back to the 80s, just pick some good bits and move on, possibly producing grust or crust while at it.

In order to pick well from Ada and the concepts embodied in it, imagine what parts of Ada should be thrown out, ignoring commercial enterprises living off legacy business? What changes to Ada are a good fit while aiming at memory safety, verification support, or light weight and safe parallel execution?

As you can see in [1], there is a suggestion to make money available to refactoring efforts.

[1] https://www.federalregister.gov/d/2023-17239/p-37

From: Luke A. Guest <laguest@archeia.com>
Date: Mon, 25 Sep 2023 17:21:57 +0100

> What changes to Ada are a good fit while aiming at memory safety, verification support, or light weight and safe parallel execution?

I started thinking about that here https://github.com/Lucretia/orenda.

From: Stéphane Rivière <stef@genesix.org>
Date: Tue, 26 Sep 2023 08:55:11 +0200

> Or when they realize that there is only one Rust compiler, and therefore that a single compiler virus could ruin the whole defense system.

Good point!

Still some doubts about their ability to reason that far ;)

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Tue, 26 Sep 2023 11:23:24 -0000

>Still some doubts about their ability to reason that far ;)

Whilst I have in the past refused to use lattice semiconductor hardware due to a CDN preventing secure compiler verification, whilst apparently none or few noticed.

I assume you mean trojaned compiler code inserted upstream to disable protections or ignore unsafe code?

Or do you mean utf-8 library code substitution aimed at a particular compiler?

Ada Advocacy Opportunity

From: Shark8 <onewingedshark@gmail.com>
Subject: Ada Advocacy Opportunity
Date: Wed, 27 Sep 2023 22:07:14 -0700
Newsgroups: comp.lang.ada

The federal government has issued an RFI (Request For Information) on the topic of increasing software security. 10-page max (excluding cover-page and appendix, if any).

Request for Information on Open-Source Software Security: Areas of Long-Term Focus and Prioritization

https://www.federalregister.gov/documents/2023/08/10/2023-17239/request-for-information-on-open-source-software-security-areas-of-long-term-focus-and-prioritization

Ada-related Tools

SPARK Reusable Components

From: Pragmada Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Ann] SparkRC
Date: Tue, 1 Aug 2023 10:58:34 +0200
Newsgroups: comp.lang.ada

Those using SPARK may find this useful:

https://github.com/jrcarter/SparkRC

[From the website: SPARK Reusable Components – A few useful components to complement the Ada.Containers.Formal_* components, primarily to learn about proofs and functional correctness: queues, stacks, an O(logN)-searchable ordered structure, (...) maps. These can be fully proven to correctly implement their contracts and to be free of run-time errors. –arm]

Ada Binding to WolfSSL

From: Joakim Strandberg <joakimds@kth.se>
Subject: Announcing Ada binding to the wolfSSL library
Date: Thu, 3 Aug 2023 14:02:56 -0700
Newsgroups: comp.lang.ada

On the WolfSSL blog I saw the following announcement today:

Today we are happy to announce the availability of an Ada/SPARK binding that enables Ada applications to use post-quantum TLS 1.3 encryption through the WolfSSL embedded SSL/TLS library.

It opens the door to obtaining FIPS 140-3 and DO-178C certifications for Ada and Spark applications that use TLS for their encrypted communications and also makes them quantum-safe.

Check out the Ada/SPARK binding on GitHub here: https://github.com/wolfSSL/wolfssl/tree/master/wrapper/Ada

The Ada port is suitable for anything from IoT, embedded systems to Desktop and Cloud systems.

Contact us at facts@wolfssl.com, or call us at +1 425 245 8247 with any questions, comments, or suggestions.

URL to blog post: https://www.wolfssl.com/announcing-ada-binding-to-the-wolfssl-library/

LibAWS for Debian 12

From: philip...@gmail.com <philip.munts@gmail.com>
Subject: libaws for Debian 12 (Bookworm)
Date: Tue, 8 Aug 2023 09:55:53 -0700
Newsgroups: comp.lang.ada

Debian 12 no longer includes system packages for the AdaCore Ada Web Server Library. (Debian 11 had libaws20-dev et al).

I have managed to build AWS v23.0.0 for Debian 12 and published it in the nascent Munts Technologies Debian 12 Package Repository, available at:

http://repo.munts.com/debian12

The goop to build libaws*.deb is stored at: https://github.com/pmunts/libsimpleio/tree/master/libaws

SweetAda on NEORV32

From: Gabriele Galeotti <gabriele.galeotti.xyz@gmail.com>
Subject: SweetAda on NEORV32
Date: Thu, 10 Aug 2023 06:03:18 -0700
Newsgroups: comp.lang.ada

I've created a NEORV32 target platform in SweetAda (https://github.com/gabriele-galeotti).

NEORV32 (https://github.com/stnolting/neorv32) is a popular RISC-V SoC implementation in VHDL, suited for FPGAs.

The setup so far is blatantly primitive and runs under simulation by means of GHDL, outputting a welcome message inside the simulated UART console and continuously output the value of the mtime timer.

So far I have no FPGA hardware (besides the time) ready to create a real implementation, so if someone is using NEORV32 on real hardware, and is willing to test, it will be very interesting to know about a OK/KO flag feedback. The current setup needs only UART clocking parameters in the CTRL register, which I suppose it depends on the actual clock configuration. In the meantime I will continue to develop things inside the simulated GHDL environment.

WinRT v3

From: Alexg <agamper@bigpond.net.au>
Subject: [Ann] WinRt - version 3
Date: Sat, 26 Aug 2023 22:18:29 -0700
Newsgroups: comp.lang.ada

Dear Ada community

I have created a new git repo for the Ada binding to WinRT (now version 3).

This version is a cleaner implementation than the previous version and includes the following changes:

1) Wide strings are mapped to HStrings.

2) Async operations and actions are handled automatically.

3) code files now contain code at the Namespace level.

Git repo is located here
https://github.com/Alex-Gamper/Ada-WinRt3

GNAT Studio 24.0 for MacOS Ventura.

From: Blady <p.p11@orange.fr>
Subject: [ANN] GNAT Studio 24.0 for macOS Ventura.
Date: Wed, 6 Sep 2023 13:17:51 +0200
Newsgroups: comp.lang.ada

Here is a very preliminary version of GNAT Studio 24.0wa as a standalone app for macOS 13:

https://sourceforge.net/projects/gnuada/files/GNAT_GPL%20Mac%20OS%20X/2023-ventura

See readme for details.

Limitation: Ada Language Server has some latencies and doesn't respond when parsing source code with more 2000 lines. It may be due to some compilation options I missed.

There could be some other limitations that you might meet. Feel free to report them on MacAda list (http://hermes.gwu.edu/archives/gnat-osx.html).

Any help will be really appreciated to fix these limitations.

Simple Components v4.68

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple Components v4.68
Date: Sat, 30 Sep 2023 18:58:05 +0200
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, persistent storage, multiple connections server/client designing tools and protocols implementations. The library is kept conform to the Ada 95, Ada 2005, Ada 2012 language standards.

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

Changes (30 September 2023) to the version 4.67:

- Boolean types handling was added to the Python bindings;
- Python library search path for OSX fixed.

Ada and Operating Systems

Ada Support on Arch Linux

From: Rod Kay <rodakay5@gmail.com>
Subject: Ada Support on Archlinux
Date: Mon, 10 Jul 2023 04:22:50 +1000
Newsgroups: comp.lang.ada

Unfortunately, the attempt to have gprbuild and xmlada added into the official Archlinux package repositories has not been successful. Nonetheless, thanks to those who voted for these packages. The gist of the problem from Arch's Trusted Users (who manage adding new packages into the official repos) was "I don't use Ada, so I won't sponsor it.".

Given this situation, I've created a custom repository for the (50 odd) Arch Ada packages, currently served by a Linode. Details (and the current package list) can be found here ...

https://wiki.archlinux.org/title/Ada

If anyone has any suggestions for additional Ada projects to add to the package list, please let me know.

MacOS: Best Not Upgrade to Xcode/CLT 15.0

From: Simon Wright <simon@pushface.org>
Subject: macOS: best not upgrade to Xcode/CLT 15.0
Date: Wed, 20 Sep 2023 10:35:29 +0100
Newsgroups: comp.lang.ada

If you accept the Xcode/Command Line Tools upgrade to 15.0, you'll get crashes in the linker (`ld`). If you can't resist the upgrade, gnatmake or gprbuild with `-largs -Wl,-ld_classic`.

You probably won't be offered the upgrade unless you're already running Ventura.

MacOS Ventura 13.6 Update Problem

From: Moi <findlaybill@blueyonder.co.uk>
Subject: macOS Ventura 13.6 update problem
Date: Fri, 22 Sep 2023 21:02:17 +0100
Newsgroups: comp.lang.ada

Installing the macOS Ventura 13.6 security update clobbers GNAT. Specifically, the link stage fails:

> -macosx_version_min has been renamed to -macos_version_min
> 0 0x104de0f43 __assert_rtn + 64
> 1 0x104ce2f43 ld::AtomPlacement::findAtom(unsigned char, unsigned long long, ld::AtomPlacement::AtomLoc const*&, long long&) const + 1411
> 2 0x104cff431 ld::InputFiles::SliceParser::parseObjectFile(mach_o::Header const*) const + 19745
> 3 0x104d0fb71 ld::InputFiles::parseAllFiles(void (ld::AtomFile const*) block_pointer)::$_7::operator()(unsigned long, ld::FileInfo const&) const + 657
> 4 0x7ff80b631066 _dispatch_client_callout2 + 8
> 5 0x7ff80b642e09 _dispatch_apply_invoke + 213
> 6 0x7ff80b631033 _dispatch_client_callout + 8
> 7 0x7ff80b6410f6 _dispatch_root_queue_drain + 683
> 8 0x7ff80b641768 _dispatch_worker_thread2 + 170
> 9 0x7ff80b7cec0f _pthread_wqthread + 257
> ld: Assertion failed: (resultIndex < sectData.atoms.size()), function findAtom, file Relocations.cpp, line 1336.
> collect2: error: ld returned 1 exit status
> gnatmake: *** link failed.

Simon's "magic formula", '-largs -Wl,-ld_classic' restores sanity. I guess the CLTs were updated without asking permission. 8-(

From: Simon Wright <simon@pushface.org>
Date: Sat, 23 Sep 2023 12:16:03 +0100

I managed to avoid this this morning (I've been resisting the attempted upgrade to CLT 15.0) by looking to see what was proposed, seeing that there were 2 upgrades (Ventura & CLT), and unchecking the CLT.

Iain Sandoe recommends [1] re-installing 14.3 (you should be able to download it from the developer.apple.com website, although you do need an apple ID to do that)

[1] https://github.com/iains/gcc-12-branch/issues/22#issuecomment-1730213294

From: Leo Brewin <leo.brewin@monash.edu>
Date: Sun, 24 Sep 2023 08:37:04 +1000

Hmm, this is odd. I'm having no problems. I'm running mac OS Ventura 13.6 (with the security update), CLT 15.0 and GNAT based on GCC 13.1.

From: Moi <findlaybill@blueyonder.co.uk>
Date: Sun, 24 Sep 2023 01:47:57 +0100

> Hmm, this is odd. I'm having no problems.

Ah! I'm still on GNAT 12.2.0.

I should have added that, although the link phase works, it produces this message, which is new:

> -macosx_version_min has been renamed to -macos_version_min

And Free Pascal Compiler version 3.2.2 [2021/05/16] for x86_64 also complains, thus;

> ld: warning: -multiply_defined is obsolete

But again, the linking succeeds.

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Sep 2023 12:55:56 +0100

> Hmm, this is odd. I'm having no problems. I'm running mac OS Ventura 13.6 (with the security update)

I was under the impression that 13.6 _was_ a security update!

Investigation so far shows that linking against the static Ada runtime (the default, -bargs -static) crashes, against the shared runtime (-bargs -shared) is OK.

With 12.2.0, 13.1.0, 14.0.0.

It occurs to me that it might be _any_ static library? ... later

From: Simon Wright <simon@pushface.org>
Date: Sun, 24 Sep 2023 16:38:36 +0100

> It occurs to me that it might be _any_ static library? ... later

Yes, so it is. Damn.

GNAT Linking and MacOS

From: Moi <findlaybill@blueyonder.co.uk>
Subject: GNAT linking and macOS
Date: Wed, 27 Sep 2023 20:30:17 +0100
Newsgroups: comp.lang.ada

I installed 14.0, Sonoma, on my M1 Mac last night.

The good news:

Using GNAT 12.2.0, it all just works, *so long as* I REMOVE '-largs -Wl,-ld_classic' from the linker options!

From: Simon Wright <simon@pushface.org>
Date: Thu, 28 Sep 2023 14:32:47 +0100

> Using GNAT 12.2.0, it all just works, *so long as* I REMOVE '-largs -Wl,-ld_classic' from the linker options!

Likewise, but that didn't work for me.

It turns out there's an environment variable DEFAULT_LINKER, which with the 15.0 CLT would be set to

export DEFAULT_LINKER=/Library/Developer/CommandLineTools/usr/bin/ld-classic

(the Xcode equivalent is much longer & more inscrutable)

I haven't tried this.

From: Simon Wright <simon@pushface.org>
Date: Thu, 28 Sep 2023 22:00:29 +0100

> Likewise, but that didn't work for me.

You can't reinstall the 14.3 CLT under Sonoma (it's "too old"). I reinstalled from a Time Machine backup, but if you don't have that set up I'd recommend taking a copy of /Library/Developer/CommandLineTools before updating to 15.0.

> It turns out there's an environment variable DEFAULT_LINKER [...]

> I haven't tried this.

This affects building the compiler, not using it.

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Date: Fri, 29 Sep 2023 19:30:27 -0700

[...] Are there any other workarounds to solve the inability to link? This does not only adversely affect Ada, but everything that uses a linker, BTW.

From: Simon Wright <simon@pushface.org>
Date: Sat, 30 Sep 2023 14:55:09 +0100

> Are there any other workarounds to solve the inability to link?

I have some evidence that the issue only arises with static libraries. Not much help.

We're hoping that the 15.1 release of Command Line Tools fixes this. In the meantime,

(1) using gnatmake, or gprbuild without changing the GPR:

```
$ gnatmake foo.adb -largs -Wl,-ld_classic
```

or

```
$ gprbuild -P foo -largs -Wl,-ld_classic
```

(2) modifying the GPR by adding a new package Linker:

```
package Linker is
   for Default_Switches ("ada") use
   ("-Wl,-ld_classic");
end Linker;
```

(3) if you already have a package Linker, modify it as above.

Ada Practice

Text vs Binary File Identification

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Subject: Using "pure" (?) Ada, how to determine whether a file is a "text" file, not a binary?
Date: Sat, 1 Jul 2023 10:15:25 -0700
Newsgroups: comp.lang.ada

Another very beginner question here...

Using "pure" (?) Ada, how to determine whether a file is a "text" file, not a binary?

Kind of like using the UNIX/Linux "file" command, but doesn't have to be comprehensive (yet). Something like the Perl "-T" feature.

On the other hand, if there already exists an Ada implementation of the UNIX "file" command as a library, could you point me to that?

As a side question, how does one read "binary" files in Ada?

A UNIX/Linux use case for the previous sentence is the concatenation of two (or more) "binary" files that were created using the UNIX/Linux "split" command.

So I'd be interested in emulating the UNIX "cat" command for "binary" files.

These are just personal experiments for learning how to do all kinds of Ada I/O...

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 1 Jul 2023 22:39:27 +0200

> Using "pure" (?) Ada, how to determine whether a file is a "text" file, not a binary?

That depends on the definition of a text file. Under Unix and Windows, all files are sequences of bytes, and so may be considered sequences of Characters, and so text files.

If you can define what distinguishes text files from binary files, then it should be fairly easy to write Ada to distinguish them.

For example, if a text file is one in which all the characters, except line terminators, are graphic characters, then it should be clear how to determine whether a file meets that definition of a text file.

> As a side question, how does one read "binary" files in Ada?

Ada has Direct_IO, Sequential_IO, and Stream_IO for reading binary files. Which you would use and how to use it depends on what's in the file and what you need to do with it.

Memoization in Ada

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Subject: memoization in Ada? Hash ADT?
Date: Fri, 21 Jul 2023 20:50:04 -0700
Newsgroups: comp.lang.ada

I'm working on the Rosetta Code task:

"Stirling numbers of the second kind"

I have a working recursive solution written in Ada but I'd like to memoize it to cut down on the redundant and duplicative calls (similar to a recursive solution to calculating the Fibonacci sequence).

So I think I need a hash ADT (which I've used in Perl) but I've never used in Ada.

So I want to preserve the calculation of the Stirling2 for each N and K so I can do a lookup. If this were based on a single unsigned integer, an array would suffice. Maybe a 2d array would suffice?

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Date: Fri, 21 Jul 2023 22:30:44 -0700

I solved the specific problem using a 2d array for caching. This is not memoization, per se, but this works very well. The recursive calls are now very fast as there is a maximum of one calculation per recursive call.

So, any resources on how to write Ada programs that take advantage of memoization?

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Mon, 24 Jul 2023 14:18:25 -0700

> So, any resources on how to write Ada programs that take advantage of memoization?

Look here https://forum.ada-lang.io/ for discussions about Advent of Code puzzles. Some solutions use (and need, for completing in a reasonable time) memoization.

You find with HAC (https://hacadacompiler.sourceforge.io/) a set of solutions (search "memoiz*" or "cache"), mostly compiling with the HAC subset.

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Date: Tue, 25 Jul 2023 21:38:02 -0700

Thanks for the pointer to the forum.

Regarding HAC, isn't that Windows-only? I'm on a Mac (M1 chip). I'll look again at HAC.

From: Simon Wright <simon@pushface.org>
Date: Wed, 26 Jul 2023 08:50:01 +0100

> Regarding HAC, isn't that Windows-only? I'm on a Mac (M1 chip). I'll look again at HAC.

It worked well enough for me to find a failing test case (now fixed!)

From: Gautier Write-Only Address <gautier_niouzes@hotmail.com>
Date: Wed, 26 Jul 2023 14:36:15 -0700

> Regarding HAC, isn't that Windows-only?

Not at all :-)

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Date: Wed, 26 Jul 2023 15:18:51 -0700

> > Regarding HAC, isn't that Windows-only?

> Not at all :-)

Downloaded and ran demo. Will experiment further as time permits. Nice!

Rosetta Proper Divisors Fails to Compile

From: Kenneth Wolcott kennethwolcott@gmail.com
Subject: Rosetta Code task Proper divisors fails to compile
Date: Tue, 25 Jul 2023 21:49:49 -0700
Newsgroups: comp.lang.ada

Trying to understand (and use) a Rosetta Code task (Proper divisors)... https://rosettacode.org/wiki/Proper_divisors#Ada

it fails to compile

```
gnatmake -vh ./proper_divisors.adb

GNATMAKE 13.1.0
Copyright (C) 1992-2023, Free Software Foundation, Inc.
"proper_divisors.ali" being checked ...
 -> "proper_divisors.ali" missing.
gcc -c -I./ -I- ./proper_divisors.adb
generic_divisors.ads:11:08: error: (Ada 2005) cannot copy object of a limited type (RM-2005 6.5(5.5/2))
generic_divisors.ads:11:08: error: return by reference not permitted in Ada 2005
End of compilation
gnatmake: "./proper_divisors.adb" compilation error
```

Why does this work for the submitter of the Rosetta Code task and not for me?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Wed, 26 Jul 2023 10:36:12 +0200

For some reason your gnatmake seems to be defaulting to -gnat05 mode. This code has an expression function, which is Ada 12, so try adding -gnat12 to the command.

You also should not need to put "./" in front of the file name, though I don't see how that would make a difference.

From: Kenneth Wolcott <kennethwolcott@gmail.com>
Date: Wed, 26 Jul 2023 11:30:12 -0700

Thank you for your suggestion. Doesn't seem to have any effect.

(*SIGH*)

```
gnatmake -vh -gnat2012 proper_divisors.adb

GNATMAKE 13.1.0
Copyright (C) 1992-2023, Free Software Foundation, Inc.
"proper_divisors.ali" being checked ...
-> "proper_divisors.ali" missing.
gcc -c -gnat2012 proper_divisors.adb
generic_divisors.ads:11:08: error: (Ada 2005) cannot copy object of a limited type (RM-2005 6.5(5.5/2))
generic_divisors.ads:11:08: error: return by reference not permitted in Ada 2005
End of compilation
gnatmake: "proper_divisors.adb" compilation error
```

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Wed, 26 Jul 2023 21:57:37 +0200

> Thank you for your suggestion. Doesn't seem to have any effect.

Interesting. I get the same results with GNAT 12.

Looking more closely at the code, I think the error, while its msg is misleading, is correct. A function that returns a limited type can only return an aggregate, a function call, or an object declared by an extended return statement. The generic formal object None is none of these.

Changing the generic parameter to a function

```
with function None return Result_Type;
```

makes the code correct. You need to change the definition of Empty in Proper_Divisors

```
function Empty return Pos_Arr is
(1 .. 0 => <>);
```

and create a function to supply for the 2nd instantiation

```
function None return Natural is (0);<
...
package Divisor_Count is new
Generic_Divisors (Result_Type => Natural,
    None => None,
    One => Cnt, Add =>  "+");
```

and then it compiles and runs.

Another possibility is to make Result_Type simply private, though that is slightly less general. That may be how the OP got the code to compile and run. It might be that limited was added later because the OP saw that it could be, and never tested the change.

Using 'Image with Alire

From: Seth Workman <saworkman1@gmail.com>
Subject: Using 'Image with Alire
Date: Sun, 6 Aug 2023 14:17:40 -0700
Newsgroups: comp.lang.ada

I have only started learning about Ada recently and have discovered the 'Image attribute that can be used on all types starting in Ada 2022.

I am using Alire and added the following to include the `-gnat2022` switch.

```
   for Default_Switches ("Ada") use
   Learning_Config.Ada_Compiler_Switches
   & ("-gnat2022");
```

The Alire documentation warns about switches ~"In general, this should be avoided to preserve consistency in the ecosystem"

Is this the correct way about adding this switch or is there a way to use a toolchain that already has it by default?

From: Simon Wright <simon@pushface.org>
Date: Sun, 06 Aug 2023 22:58:12 +0100

> Is this the correct way about adding this switch or is there a way to use a toolchain that already has it by default?

This works fine, but in your alire.toml you could say

```
[build-switches]
"*".ada_version = "ada2022"
```

or

```
[build-switches]
"*".ada_version = ["-gnat2022"]
```

See "Release Information" (near the end) and "Build Profiles and Switches" in the documentation.

From: Seth Workman <saworkman1@gmail.com>
Date: Sun, 6 Aug 2023 15:17:07 -0700

I see now, I think using
"*".ada_version = "ada2022"
is better for this case.

Parallel Loops in GNAT

From: Jerry <list_email@icloud.com>
Subject: Parallel loops in GNAT?
Date: Fri, 11 Aug 2023 17:44:38 -0700
Newsgroups: comp.lang.ada

Does GNAT such as Simon's GCC 13.1.0 for macOS aarch64 allow parallel loops and blocks?

From: Simon Wright <simon@pushface.org>
Date: Sun, 13 Aug 2023 11:39:59 +0100

No, sorry, that's one of the advanced features that AdaCore aren't working on yet.

https://blog.adacore.com/ada-202x-support-in-gnat

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sun, 13 Aug 2023 14:44:02 +0200

ISO/IEC 8652:2023 is the first Ada standard to be approved without a working implementation. Probably a bad sign.

From: Jerry <list_email@icloud.com>
Date: Sun, 13 Aug 2023 15:03:20 -0700

> https://blog.adacore.com/ada-202x-support-in-gnat

Thanks, Simon. I saw that blog post but since it's nearly three years old, I was hopeful. My O(N^4) radar simulations will have to remain slow. :-(

Unifont Statically Compiled and Stack Size

From: Micah Waddoups <micah.waddoups@gmail.com>
Subject: Unifont static compiled and stack size...
Date: Sun, 13 Aug 2023 09:16:27 -0700
Newsgroups: comp.lang.ada

I tried to compile the Unifont hex file, converted with a script into variable Ada code, but it went on forever, gradually blowing up my memory. I used an .ads file and aimed for a static build, but I suspect I would have hit the stack size limit for executables if I succeeded

My request for insight is:

(A) How do you recommend I compile the Unifont into a form that is usable within my Ada program. (I am thinking compiling C and importing, since C is so basic it might just work, but even better would be Assembly and I don't know how to import a large data variable compiled in Assembly or if I can even compile that much data using Assembly... It should work, but the compiler might complain and I still have to figure out the importing part.)

(B) Do you think this has a chance of succeeding if I compile the font as a shared library? That doesn't affect initial stack limits, right?

Just to be clear, I am trying to import values 0 .. 16#FFFFF#, way beyond the two byte limit that so many libraries are functionally bound by in one bottle neck or another. I want to support something similar to 'kmscon', but with some shortcuts since I don't want to redo everything that others have done well. I just want to finish my library to a point of usefulness and focus on other projects. I felt compelled to create this library because every other library I looked at was broken in some way and even the most common font systems fail to support Unicode's full character range, making much of it useless. I figured I could create the exact effects that I am trying to with Unifont both in graphical Windows of various OSs, and on the Linux terminal if I rewrite enough of the low level code that I don't have to rely on the less complete existing libraries. Admittedly, I have too little time to work on it, and am so far behind other people's wonderful work that I will certainly have many holes and omitted functionality that should eventually be added later. My goal is to both make my programming projects possible and free certain features from too restricted licensing.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 14 Aug 2023 11:07:14 +0300

> My request for insight is: (A) How do you recommend I compile the Unifont into a form that is usable within my Ada program.

Could you show a little of the script-generated Ada code, just for us to understand the approach you have taken to represent the Unifont file?

While waiting for that, it may help the compiler if you disable the more advanced compiler optimizations, because some of them are super-linear in complexity and probably (but depending on the exact form of the Ada code) do not help for this case, anyway.

> (I am thinking compiling C [...] but even better would be Assembly [...]

I'm not familiar with the structure of the Unifont file, but if it is something like a table with rows and columns, it should be rather easy to translate it into a list of assembly-language constant-data definitions.

Assemblers are typically linear in complexity and should be able to handle large data definitions, assuming there is not a myriad of assembler labels that make references between different parts of the data structure.

Exporting a data object from assembly to Ada is simple and does not depend on the size of the object (but I admit I don't know how this is done for dynamically linked libraries). The only part that needs thought is how to define the Ada type of the object, but if the Unifont file is a row-column table, in other words a list of "row" records, that should be straight-forward too.

So I think the assembly-language solution is highly likely to work; its drawback, of course, is non-portability. But the constant-data definitions of most assembly languages are very similar to each other, so the assembler-generating script should be easy to port to different assembly languages.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 14 Aug 2023 10:31:43 +0200

> I'm not familiar with the structure of the Unifont file [...]

A comparable case. I have XPM to Ada translator (for having built-in images in GTK). It simply creates packages with declarations of initialized arrays. No stack issues.

Doing something like that for bitmap fonts is just as simple. The only minor issue is creating an index map: code point to the bitmap image name (array), because a flat array would blow out.

P.S. I always wanted static functions in Ada for the purpose of all static initializations of objects like maps etc.

From: Kevin Chadwick
<kc-usenet@chadwicks.me.uk>
Date: Mon, 14 Aug 2023 09:25:07 -0000

> a flat array would blow out.

What does blow out mean in this context?

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 14 Aug 2023 11:39:45 +0200

> What does blow out mean in this context?

If you tried:

```
type Font_Type is array (Code_Point) of
   Bitmap_Ptr;
```

The range of code points is 0..16#10FFFF#. E.g. when I implemented Ada.Strings.Maps for Unicode, I could not use such arrays either as the native ASCII implementation does.

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Mon, 14 Aug 2023 12:06:35 +0200

> I tried to compile the Unifont hex file, converted with a script into variable Ada code, but it went on forever, gradually blowing up my memory. I used an .ads file and aimed for a static build, but I suspect I would have hit the stack size limit for executables if I succeeded

As I understand it, the file in question is for code points 0 .. 16#FFFF#, with a maximum of 32 bytes per code point. A straightforward representation of this is

```
package Unifont is
   type Byte is mod 2 ** 8 with Size => 8;
   type Line is array (1 .. 2) of Byte
      with Size => 16;
   type Bitmap is array (1 .. 16) of Line
      with Size => 256;
   function Width_8 (Map : in Bitmap)
   return Boolean is
      (for all L of Map => L (2) = 0);
   type Code_Point is mod 16#FFFF# + 1;
   type Font_Map is array (Code_Point)
      of Bitmap with Size => 2 ** 24;
Font : constant Font_Map :=
      (others => (others => (others => 0) ) );
end Unifont;
```

Font will occupy 2 MB.

We can test this with

```
with Ada.Text_IO;
with Unifont;
procedure Unifont_Test is
   -- Empty
begin -- Unifont_Test
   Ada.Text_IO.Put_Line (Item =>
   Unifont.Font (0) (1) (1)'Image);
end Unifont_Test;
```

and see what happens:

```
$ gnatmake -m -j0 -gnat12 -gnatan -gnato2 -
O2 -fstack-check unifont_test.adb
x86_64-linux-gnu-gcc-12 -c -gnat12 -gnatan -
gnato2 -O2 -fstack-check
unifont_test.adb
x86_64-linux-gnu-gcc-12 -c -gnat12 -gnatan -
gnato2 -O2 -fstack-check unifont.ads
x86_64-linux-gnu-gnatbind-12 -x
unifont_test.ali
x86_64-linux-gnu-gnatlink-12 unifont_test.ali
-O2 -fstack-check
$ ./unifont_test
0
```

so this representation seems to be workable. It should be trivial to write a program to read the file and produce the real array aggregate for Font.

From: Micah Waddoups
<micah.waddoups@gmail.com>
Date: Mon, 14 Aug 2023 08:10:01 -0700

Jeff, you missed a digit - it's five F's 16#FFFFF# because that is as high as Unifont goes in my copy of the hex file. [...]

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Mon, 14 Aug 2023 17:59:16 +0200

> Jeff, you missed a digit - it's five F's 16#FFFFF#

You're right. Sorry for misreading that. But increasing Code_Point to include 16#F_FFFF# still works for me.

Unicode defines code points up to 16#10_FFFF#, but perhaps those over 16#F_FFFF# are unused. Increasing Code_Point to include 16#10_FFFF# still works for me. That's 34 MB. It won't fit on the stack, but luckily library-level constants aren't allocated on the stack.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Mon, 14 Aug 2023 18:02:58 +0200

What are you going to do with this?

1. This is not a font usable in a GUI framework.
2. This is not a drawable in-memory image for a GUI framework either. Provided, you wanted to render obtained images manually. These images must be in a format supported by the corresponding engine.

E.g. GTK uses Pixbuf representation for drawable in-memory images. Which is

```
type Pixbuf_Image is array
   (Natural range 0..N*M-1) of GUChar;
pragma Convention (C, Pixbuf_Image);
```

containing 4 channels RGB + alpha, row-wise.

And, no, normally you cannot draw in an arbitrary OS window.

If you are so keen to use GNU Unifont, why do not you install it from its available formats like TrueType and be done with that? What is wrong with other fixed-size fonts?

Why do you want to render glyphs manually instead of using existing OS facilities and GUI libraries? You cannot get around these libraries without rewriting device drivers and who knows what else making the code highly non-portable.

From: Micah Waddoups
<micah.waddoups@gmail.com>
Date: Mon, 14 Aug 2023 21:48:03 -0700

> What are you going to do with this?

Dmitry, of course, you are correct. This was an attempt to take an image of the font source. [...]

The rendering is done with a small cache of per-need rendered Glyphs - each rendered glyph is at least eight times larger before any styling or transformation. Rendering all the glyphs at once takes up more memory than is needed and presupposes the destination format. So, in reality, more processor work is being done by doing it in stages, but less is being done in each stage, so there is more room for other processing at each stage. This is just the raw bitmap font that I want in the program without having to process and read the hex file every time it loads (that would be a very inefficient design).

In the first rendering, the glyphs that are actually used are rendered into a cache with only Alpha values. This is essentially gray-scale and is the value-map used for any transformations, glyph combining, and plotting on a per-line image plot map (which is another abstraction by line and index/column of just the first..last pixel boundaries). The plot map can very quickly be changed for insertions, deletions, changing text direction or flow, etc. and only at the final rendering of the View-able area is the Alpha transformed into full color (4-byte with alpha) to be sent to the GL, framebuffer, GTK, or other pixel-map handling system. Much like modern rendering systems, a lot of calculations happen behind the scenes at each key-press, only it is my attempt to combine it all into one library and scale it

into a project that can be plugged into whatever graphics system is being used. It is probably slower in response to input than GTK or any native text handling system because effects, layers of glyph-combinations, markups (squiggle line, underline, etc), and colorization all go into the font rendering before it hits the graphics buffer, but it feels to me more correct and avoids having to add post-rendering effects as a later stage as much as possible. Markups are done late, just before rotation (if any), but that is just as it must be, since they are effectively a rendered element of a different size than the individual glyphs they markup, yet still done before the rendered map is turned over to the graphics system.

The reason people rely on device drivers, native widgets, and less-portable combinations of libraries is that they incorporate the features and functionality desired into them, including how to handle input. I am attempting to take the lowest level of input (key-presses, clicks, touch, etc.) like a video game might, and outputting an image already rendered for display, thus replacing the normally convenient functionality provided by other systems. I am *not* trying to replace actual device drivers or rely on a particular operating system's device access scheme. That is why I am considering other well established libraries supporting GL or maybe SDL. Not all programs render text and input with the convenient systems, so this is nothing new. Also, since this scheme still tracks each character by line and index, even if I am only able to support text-mode terminal interface in one situation, that will only prevent using the full Unicode range of glyphs and combinations, not disable my ability to send/receive any given text with the display.

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Tue, 15 Aug 2023 10:40:45 +0200

> P.S. I always wanted static functions in Ada for the purpose of all static initializations of objects like maps etc.

If data form the equivalent of a static Ada array, thus a mapping from an index type to a value type, could you approximate the static initialization of maps using expression functions?

Simplifying example:

```
package sttc is
   type Key is range 1 .. 7;
   type Value is new Character;
   type Cursor is private;

   function lookup (K: Key) return Cursor;
   function element (C: Cursor) return Value;

private
   type Cursor is new Key;
end sttc;
```

```
package body sttc is
   function lookup (K: Key) return Cursor is
      (Cursor (K));
   function element (C: Cursor) return Value
      is
      (case C is
       when 1 => 'M',
       when 2 => 'M',
       when 3 => 'X',
       when 4 => 'X',
       when 5 => 'I',
       when 6 => 'I',
      when 7 => 'I'
      );
end sttc;
```

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 16 Aug 2023 08:17:50 +0200

> could you approximate the static initialization of maps using expression functions?

In general case no. Initialization cannot be decomposed into functions. E.g. when it requires global [yet static] data, many places to set etc.

P.S. Expression functions are evil. I wonder why there is no expression gotos and labels? If you sell your soul to the devil, get the whole package! (:-))

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 17 Aug 2023 22:04:18 -0500

> P.S. Expression functions are evil. I wonder why there is no expression gotos and labels? If you sell your soul to the devil, get the whole package! (:-))

Ada only allows expressions to be evaluated at elaboration time (outside of generic instantiations), and expressions have a well-defined and simple control flow (even accounting for conditional expressions and quantified expressions, both of which implicitly appear in aggregates even in Ada 83 - allowing programmers to write them explicitly makes the code more readable than the horrible work-arounds commonly used pre-Ada 2012). Gotos and labels have arbitrary control flow, which can be much harder to analyze. (Janus/Ada converts the majority of code into an expression form for optimization - essentially most id statements become if expressions, and so on. It simply punts when the control flow is too complex to convert, so the unrestricted use of gotos effectively prevents most optimization as well as static analysis.)

If it was up to me, I would have left out declare expressions and quantified expressions, so the capabilities of expression functions would have been much more limited. But it seems valuable to be able to abstract an expression without changing the semantics (as requiring a separate body does).

GCC Support for Ada 2022

From: philip...@gmail.com <philip.munts@gmail.com>
Subject: Which GCC releases have how much support for Ada 2022?
Date: Mon, 14 Aug 2023 13:30:06 -0700
Newsgroups: comp.lang.ada

I am interested in cross-compilers for Linux boards such as the Raspberry Pi. I have successfully built cross-compilers for GNAT/GCC 12.3.1 on Debian 12 using the latest Arm GNU manifest and Linaro ABE, and Debian 12 has system packages for GNAT/GCC 12.2.0, native and cross, as well.

It isn't clear to me from the GCC release notes how complete the support for Ada 2022 is in GNAT/GCC 12.2.0 or 12.3.1. Is there a document or table somewhere that keeps track of that? And how does GCC 12 compare with GCC 13 WRT Ada 2022?

From: Micah Waddoups <micah.waddoups@gmail.com>
Date: Tue, 15 Aug 2023 14:12:18 -0700

[...] You may be able to find most of what you are looking for with the link in his answer:
https://blog.adacore.com/ada-202x-support-in-gnat

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 21 Aug 2023 13:04:30 -0700

> And how does GCC 12 compare with GCC 13 WRT Ada 2022?

This post could be helpful:
https://forum.ada-lang.io/t/gcc-13-1-released/374/3

In https://learn.adacore.com/courses/whats-new-in-ada-2022/index.html course there are GCC versions per feature. Unfortunately the feature list is not complete.

Parameterised 'Image Attributes

From: Rod Kay <rodakay5@gmail.com>
Subject: Parameterised 'Image Attributes
Date: Fri, 18 Aug 2023 17:18:29 +1000
Newsgroups: comp.lang.ada

There has been some recent discussion on #ada irc regarding formatted output.

Would it be possible/desirable to allow the 'Image attribute to have formatting parameters ? Something along the lines of ...

```
put_Line (some_Integer'Image
   (Width => 5, Padding => '0'));
```

... and similar 'Image attribute parameters for other types.

If the parameters have defaults, then there should not be any backwards compatibility issues (I think).

From: Luke A. Guest <laguest@archeia.com>
Date: Fri, 18 Aug 2023 09:25:44 +0100

I wanted them for ages, but there was a conversation ages ago where someone on here said attributes were for "debugging only," yet that's not what the ARM says.

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Fri, 18 Aug 2023 11:53:44 -0700

TeleSoft's compiler (which I worked on) had 'Extended_Image and 'Extended_Value attributes that worked like that. I found them quite useful -- especially as an easy way to drop the leading space on Integer'Image.

One small problem was that we had different parameters for integer and enumeration types, which introduced an ambiguity for discrete formal types.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 19 Aug 2023 11:14:34 +0200

I wanted them for ages, but [...] someone on here said attributes were for "debugging only,"

The intent of the 'Image attribute is to have a quick representation, mainly for debugging purposes. If you want nice formatted output, use the Put procedure on String from Text_IO.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 19 Aug 2023 12:03:09 +0200

> The intent of the 'Image attribute is [...] mainly for debugging purposes.

It seems that for the vast majority of Ada users this intent was wrong...

> If you want nice formatted output, use the Put procedure on String from Text_IO.

Put does not supersede 'Image. Put is I/O. 'Image is pure string formatting. Put is generic and requires instantiation of some package with some difficult-to-guess name. 'Image is built-in [statically] dispatching and generated automatically by the compiler.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 19 Aug 2023 13:56:14 +0200

> It seems that for the vast majority of Ada users this intent was wrong...

The vast majority of Ada users ignore a number of useful features provided by the language, and keep asking for improvements that are already there...

> Put does not supersede 'Image. [...]

Yes, Put has nothing to do with 'Image. Yes, Put requires instantiation. So what? Ada is more verbose, in favor of stricter typing. Ease of reading over ease of writing has always been a major design principle of Ada - although I confess it had a bad effect on its popularity, people want to write fast and ignore long term maintenance issues.

If you want formatting on an integer type (with or without IO), you instantiate Integer_IO. I don't find it hard to guess the name... Maybe you had something else in mind?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 19 Aug 2023 15:01:36 +0200

> The vast majority of Ada users ignore a number of useful features [...]

Or these features are not that useful? Language users and designers have often different perspectives...

> Yes, put requires instantiation. So what? Ada is more verbose, in favor of stricter typing.

I don't see how instantiation is stricter typing. In fact instantiation introduces overloading (static ad-hoc polymorphism) which was always frowned upon at as less type safe than overriding.

> Ease of reading over ease of writing has always been a major design principle of Ada

I don't buy this either. It is

```
Put (X) vs. X'Image
```

equally readable and writable. If you refer to the instantiation noise or with/use clauses you would require to put somewhere far above in the package, that is not ease of reading. That is just meaningless noise.

> I don't find it hard to guess the name... Maybe you had something else in mind?

Yes, all other types that might require formatting. I doubt anybody, but a language lawyer could name the package appropriate for formatting a fixed-point type without looking into the RM. Which is absolutely unneeded as 'Image would be perfectly OK if it had the necessary parameters. All that generic text I/O packages are unnecessary as the stream I/O case perfectly illustrates. Ada 95 did stream I/O if not right, but far better making 'Read, 'Write etc attributes overridable. The problem of generic mess solved. We do not have and do not need any generics for stream I/O. Good riddance.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 19 Aug 2023 17:27:25 +0200

> The intent of the 'Image attribute is to have a quick representation, mainly for debugging purposes.

There is a common problem across many types and problem domains of having a function that returns a string of an appropriate length representing a value of the type with desired formatting. Common examples include numeric values, dates, and times. The resulting string is usually combined with other information into a message that may be stored in memory for a while, though it is rare for it not to be output eventually. As an example, the message may be put on a protected queue for later output by a logging task.

Ada 83 tended not to include anything that the developer could implement; there was no math library or image functions for dates or times. The 'Image attribute was provided, but is unsuited for most such uses.

The use of the Text_IO generic sub-pkg Put procedures that output to strings is not convenient because they are procedures, not functions.

Later versions of Ada included more support for such needs, but not for numeric values.

The obvious solution is to have a library containing appropriate functions, which can be built around the Put procedures while still being functions. Such functions would need to be generic, unlike attribute functions which are automatically available for all types.

The conflict between this common need and the minimal functionality provided by 'Image results in such requests. It seems desirable for the language to provide such functions, and extending the 'Image functions seems like a reasonable way for it to do so, regardless of the original intentions for the attribute.

One library with such functions is the PragmAda Reusable Components (https://github.com/jrcarter/PragmARC). The package PragmARC.Images (https://github.com/jrcarter/PragmARC/blob/Ada-12/pragmarc-images.ads) provides such functions for integer and floating-point types. Function PragmARC.Images.Image is an instantiation for Standard.Integer.

PragmARC.Date_Handler (https://github.com/jrcarter/PragmARC/blob/Ada-12/pragmarc-date_handler.ads) provides image functions for dates and times; although the language now provides a function for a date-time image, Date_Handler continues to be useful as it provides for customized formats rather than the single format provided by Ada.Calendar.Formatting. Many users also find the semantics of the latter's time-zone parameter to be confusing.

ISO/IEC 8652:2023 provides a date-time image function the returns the image for the local time zone, but as there are no compilers* for this version of the language, I don't consider that relevant.

(*A compiler for a version of the language implements the entire core language of that version of the ARM.)

From: Moi <findlaybill@blueyonder.co.uk>
Date: Sat, 19 Aug 2023 17:49:41 +0100

> The intent of the 'Image attribute is to have a quick representation, mainly for debugging purposes.

My code uses 'Image heavily, because it is usually the neatest and the clearest way to format many strings that mingle words and numbers.

I sometimes have to pass the result of 'Image to a function that implements the kind of functionality people are asking for, and it would be even neater and clearer if I could get that with parameters to 'Image itself.

None of that output has anything to do with debugging.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sun, 20 Aug 2023 02:25:20 -0500

The profile of the Image attribute is:

```
X'Image
```

where X is an object of any type (or value of most types). And the old, unnecessary form was:

```
S'Image(X)
```

where S and X are of any type.

If one tries to add parameters to this, one gives up this nice form for a mis-mash of profiles for various classes of types. Moreover, the result no longer composes in the obvious way (necessary to have Image for records and arrays).

Additionally, one ends up with a magic mechanism that only the compiler can use. That *never* is a good idea. Especially as there now is a way to allow Image to support user-defined types. It would seem necessary to also support user-defined formatting parameters (else one has magic only applicable to a handful of language defined types).

Attributes do not allow named parameters outside a few special cases, and *never* allow reordering of parameters. Does that need to change, too?

Float input/output in particular is very large, especially when all of the formatting options are included. Do you really want to drag that into *every* Ada program, whether it uses it or not??

'Image is convenient for integer and enumeration output, and one can format them in the rare case where that is necessary. But it is useless for float output -- manual reformatting the output of 'Image would round the results incorrectly.

Ada has few built-in facilities because its primary purpose is to support the development of proper ADTs. Ease of writing is not a goal at all, and in most cases, the extra text is valuable to compilers and tools (even if it is not so valuable to human readers). If it was up to me, I would eliminate most of the shortcuts from Ada and require everything to be written out. (IDEs could/should do most of that for you anyway, so the extra text is not adding much effort.)

Ergo, I hope this idea is dead-on-arrival. I certainly won't be involved in it, that's for sure.

From: G.B. <bauhaus@notmyhomepage.invalid>
Date: Sun, 20 Aug 2023 09:53:11 +0200

> The conflict between this common need and the minimal functionality provided by 'Image results in such requests.

So, also

- See how other languages address formats (good bits, bad bits).
- Consider use cases.
- I/O is the program(mer)'s raison d'être. Can we easily Put something into a stream without the help of a suitable library?

Could there be a language defined type F whose purpose is to support the description of formats? Objects of type F would "configure" what 'Image does when computing a representation of a date, a number, ...

```
My_Length'Image (Arg => diameter,
   Format => ___);
```

Some use cases:

- I18n of number formats (cf ARM F.3), CHF 1'234'000.–
- Handle ubiquitous ISO formats of date-time (as mentioned below; also cf. ARM 9.6.1)
- reporting,
- integrate own output with output of other system components (a site-wide monitoring system searches outputs, say)
- fill in templates when these do not support formatting
- 'Input an object of type F at run-time, so that program's use of 'Image can be changed according to customer's local expectations.
- support the formalized exchange of "numerical" data in heterogeneous systems, using text streams.

These use cases are about the O of I/O. By symmetry, it would be nice to have implementations of Ada that support the I part of this kind of I/O, I think, with work to be split between implementers and programmers.

```
My_Length'Value (Arg => diameter,
   Format => ___);
```

Or perhaps multimethods that take a stream and a format when they need to write a value?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 20 Aug 2023 11:27:47 +0200

> Could there be a language defined type F whose purpose is to support the description of formats?

Not without multiple dispatch support and classes:

```
Type x Format [ x Target ]
```

Otherwise you get an untyped mess as in C:

```
printf ("%s", 123);
```

In the case of 'Image the dispatch is hard-wired. The compiler generates it according to one of built-in classes like 'integer type'. So yes it would be no problem to add parameters specific to each of the classes as well as common parameters like padding or alignment inside a field. But it will never ever happen.

You seem suggesting a class-wide parameter type instead:

```
type Format_Type is tagged record
   Width: Natural:= 0;
   Alignment: Alignment_Type:= Left;
   Padding: Character:= ' ';
end record;
type Integer_Format is new Format_Type
   with record
      Plus_Sign : Boolean := False;
      Base : Base_Type := 10;
   end record;
X'Image (Format => Format_Type'Class)
```

This still requires a change that will be outright rejected on highest philosophical grounds. (:-))

However with a Format_Type you do not need 'Image. You can simply use a binary operation, e.g.

```
function "/" (Value : Integer; Format :
   Integer_Format) return String;
```

So would do

```
Put_Line ("X=" & X / (Width=>10,
   Padding=>'0', Alignment=>Right));
```

instead of

```
Put_Line ("X=" & X'Image (Width=>10,
   Padding=>'0', Alignment=>Right));
```

Of course it must be generic, which kills all fun.

Ergo

1. Compiler magic is necessary because the language type system is too weak to express things like formatting.
2. No proposal however useful and reasonable will survive ARG because of #1.
3. Use a library that does the stuff. E.g.

http://www.dmitry-kazakov.de/ada/strings_edit.htm#Integer_Edit

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 20 Aug 2023 11:43:01 +0200

> Additionally, one ends up with a magic mechanism that only the compiler can use. That *never* is a good idea.

A better idea would be to improve the language to remove need in magic, but that is *never* a good idea either! (:-))

> Attributes do not allow named parameters outside a few special cases, and *never* allow reordering of parameters. Does that need to change, too?

Elementary! Attribute is just an alternative syntactic form of a subroutine call. There is no reason why attribute should be limited to look like FORTRAN IV! (:-))

> 'Image is [...] useless for float output [...]

Which is why Float 'Image must have parameters!

> Ada has few built-in facilities because it's primary purpose is to support the development of proper ADTs. Ease of writing is not a goal at all [...]

How is this related to the attribute 'Image lacking necessary parameters? Why is a generic function having such parameters OK, while 'Image with same parameters is not?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 21 Aug 2023 18:11:19 -0500

Your #3 is the point of course. If a reasonable library can be written, you should use that. After all, the Ada philosophy is that it is suspicious to use any built-in types. Why then should it be less suspicious to use other things that are built-in??

The best approach for Ada going forward is to add things that make it easier to build good libraries (as in user-defined literals). And minimize magic.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 21 Aug 2023 18:34:55 -0500

> A better idea would be to improve the language to remove need in magic, but that is *never* a good idea either! (:-))

No, I generally agree with this. We probably disagree on what would constitute an improvement, however. ;-)

> Elementary! Attribute is just an alternative syntactic form of a subroutine call. There is no reason why attribute should be limited to look like FORTRAN IV! (:-))

That turns out to be a bad idea. The reason people love attributes so much is that they don't have to worry about visibility -- they're always visible. That is not and cannot be true for subprograms.

For example, the reason that we don't allow user-defined attributes is that they would compromise portability. Since they're always visible, they could hide/make illegal attributes that are used in units (like generic units) that don't know anything about the additions. Moreover, not all attributes can be described as subprograms given Ada's current rules (reduction attributes have a type parameter; some of the annex 13 attributes have "any type" parameters, etc.)

It certainly would be a very bad thing for Janus/Ada, which would have to have its resolution and subprogram definition mechanisms redesigned. (All subprograms are materialized in the Janus/Ada symboltable, in particular for visibility management reasons, and that would not be possible for attributes. Resolution only works on materialized subprogram definitions.)

> Which is why Float 'Image must have parameters!

Which is why one shouldn't use Float'Image! ;-)

> Why generic function having such parameters is OK, while 'Image with same parameters is not?

It's perfectly OK to overload functions however one wants, because you can keep anything that is problem from being considered by avoiding "use" (and "with").

'Image is not appropriate for an attribute in the first place; attributes are supposed to be simple compile-time defined properties of a type. String conversion is not that.

My preference for making Ada easier to use for this sort of thing is to allow class-wide elementary types. Then one could have non-generic subprograms that operate on all integer and float types. (Fixed and enumerations would still require generics, although I suspect most people would simply convert fixed to float for output rather than worrying about an instantiation.) That would make a library simple to use, and few people would think that something built-in is needed.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 21 Aug 2023 18:37:43 -0500

> Of course it must be generic, which kills all fun.

As noted in my other message, resurrecting the Ada 95 idea allowing class-wide types for elementary types would eliminate (or at least greatly reduce) this problem. I think that would be a more productive way to address this problem than hacking around with 'Image some more. (We've already proven that it is not a good way to define anything user-defined, thus the rather complex way to define such 'Image attributes.)

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 22 Aug 2023 09:38:38 +0200

> [...] allowing class-wide types for elementary types would eliminate (or at least greatly reduce) this problem.

Yes, but that would be a huge change.

> I think that would be a more productive way to address this problem than hacking around with 'Image some more.

One does not exclude another. If you allowed classes then there would be no reason not to have attributes [as] official primitive operations. E.g. an "imaginable" interface would provide "'Image" and the standard Integer would inherit from "imaginable"...

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 22 Aug 2023 10:13:47 +0200

> [...] not all attributes can be described as subprograms given Ada's current rules [...]

It is a primitive subprogram of some built-in class. The magic is not in the attribute, it is the class description. For magical classes overriding a primitive operation could look like

for <member-type>'<primitive-operation-name> use <subroutine-name>;

[...]

The problem is that whatever intention Ada designers had for attributes they also gave them the property of being a primitive operation where no user-defined class [is] allowed. This power steamrolls any "good" intentions.

Nobody loves the syntax T'Image (X) or X'Image! Give programmers X.Image and [<path-of-package-names-nobody-remembers>].Image (X) and they will forget about the attribute.

[...]

> Fixed and enumerations would still require generics

It would be interesting to play with the ways of constructing enumeration and fixed point classes. Both have static parameters, e.g. list of names in the case of enumeration. There might be a way to achieve static polymorphism without going full generic but also without turning the language into a C++ templates mess!

> That would make a library simple to use, and few people would think that something built-in is needed.

Absolutely. Ideally, everything must go into libraries.

From: Stephen Davies <joviangm@gmail.com>
Date: Wed, 23 Aug 2023 03:20:07 -0700

> Nobody loves the syntax T'Image (X) or X'Image!

I have no issue with the 'Image syntax.

Perhaps the formatting parameters could be restricted to T'Image(X) and not available for X'Image? Or, maybe the language should just add 'Trim_Image and 'Trim_Width and leave the advanced formatting to a library.

Actually, I think it might also be nice if Float'Trim_Image(X) returned a string that only used exponential notation for very large or very small values (which seems to be the default behaviour in Python). Different names would then be needed (Tidy_Image and Tidy_Width?).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 23 Aug 2023 18:16:10 +0200

> it might also be nice if Float'Trim_Image(X) returned a string that only used exponential notation for very large or very small values

To use the shortest representation for the given precision unless specified otherwise:

http://www.dmitry-kazakov.de/ada/strings_edit.htm#6

Ada 'Image attributes have "typographic quality" in plain contradiction to the claim being for debugging purposes. That is why the plus sign is always represented by a space and why floating-point representation is always selected even for exact zero and the way the exponent part is formatted. The typographic idea is to have *same looking* output. Note, even if the output is mathematically incorrect as in the case of floating-point numbers. 'Image considers precision and accuracy same, which is *always* wrong when dealing with floating-point numbers.

> Different names would then be needed (Tidy_Image and Tidy_Width?).

It takes several parameters to control the behavior in a reasonable way.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 28 Aug 2023 20:58:13 +0300

> There is no good reason why attributes should not have the same parameter syntax as subprograms and entry calls.

Yes in principle, but it is understandable that making this happen now could impact both the language definition and various implementations in non-trivial ways.

> Neither there is one why 'Image must be a non-overridable attribute.

In Ada 2022, 'Image is defined to call the new attribute 'Put_Image, which can be specified (ie. overridden) by the programmer for any type.

See http://www.ada-auth.org/standards/22rm/html/RM-4-10.html.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 28 Aug 2023 21:08:10 +0200

> it is understandable that making this happen now could impact both the language definition and various implementations

Compared to the useless and damaging sediments the language collects with each new release? (:-))

> See http://www.ada-auth.org/standards/22rm/html/RM-4-10.html.

Ah, thanks. I vaguely remembered that there was yet another ugly hack that does not really solve anything significant, but could not find it.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 6 Sep 2023 20:04:00 -0500

>Neither there is one why 'Image must be a non-overridable attribute

There actually is a good reason for this. Attributes have global visibility. So if you allowed overriding of attributes, then a with added or removed in a remote part of a program could silently change the behavior of code that has no knowledge of the change. That would be bad for "programming in the large". Note that Ada 95 was proven to have no such cases, and we've tried very hard to avoid them.

One could imagine adding rather severe restrictions to overriding of attributes to eliminate this problem (for instance, only allowing it for primitive operations of the type), but that would eliminate all real value of the feature (you can always use a primitive function and "use all" to get the same effect without any new features).

For 'Image specifically, the design of the attribute doesn't work well for composition (for Image for composite types), which is why Ada 2022 has a separate attribute that can be overridden similar to a stream attribute.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 7 Sep 2023 11:01:58 +0200

> [...] That would be bad for "programming in the large". [...]

Ah, but 'Image is for debugging only! (:-))

> One could imagine adding rather severe restrictions to overriding of attributes to eliminate this problem [...]

It must be a new type:

```
type My_Integer is new Integer;
for My_Integer'Image use Foo;
```

From: Rod Kay <rodakay5@gmail.com>
Date: Sat, 23 Sep 2023 20:00:26 +1000

I've been using 'Gnat.formatted_Output' which I've found quite useful. Unfortunately, it seems to be a little buggy with its formatting.

From: Vadim Godunko <vgodunko@gmail.com>
Date: Mon, 25 Sep 2023 22:47:17 -0700

You can take a look at VSS's Virtual_String_Templates and Formatters, see
https://github.com/AdaCore/VSS/blob/master/source/text/vss-strings-templates.ads

https://github.com/AdaCore/VSS/blob/master/source/text/vss-strings-formatters.ads

and an example of its use

https://github.com/AdaCore/gnatdoc/blob/3e94448ac57270caf4b4502f208f78e1d51da2b2/source/gnatdoc-messages.adb#L130

UNAS by TRW

From: Chris Sparks <mrada442@gmail.com>
Subject: UNAS by TRW
Date: Tue, 22 Aug 2023 05:58:37 -0700
Newsgroups: comp.lang.ada

[In the following: Universal Network Architecture Services (UNAS) is a product from the American TRW Inc. corporation. –arm]

Does anyone know how to get the complete UNAS package from TRW? I use it at work and I see it has an open usage clause in the source headers. Since I am not allowed to download it from my work, maybe I can find a source elsewhere to get it?

Also are there any tutorials out there on how to use it? I am in the process of upgrading the Ada (83 to 05) in my current project and I am getting stuck on the plethora of calls being made by UNAS for which I don't even know how to set it up so it can run happily.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 24 Aug 2023 22:24:59 +0200

> maybe I can find a source elsewhere to get it?

Doubtful.

> Also are there any tutorials out there on how to use it?

"Out there", definitely not. In a few closed places that still use UNAS, perhaps but doubtful. In the one place that I know still uses UNAS, no.

> I am in the process of upgrading the Ada (83 to 05) in my current project [...]

Do I divine correctly that "your current project" is not "at work" If so I would suggest you consider PolyORB as a replacement*. UNAS is long dead, unmaintained and unmaintainable, mostly because it is proprietary software without anyone getting a license for it other than in their current application. Also, apart from a couple of people I know, nobody understands UNAS anymore. The company that made it has abandoned it, perhaps even gone bankrupt, so UNAS is mostly technical debt. Sorry for the bad "news".

* Modern multi-core computers with lots of memory might even make it feasible to avoid distributing the software over multiple computers in the first place. Maybe a monolithic application would do the job just fine, nowadays.

From: Chris Sparks <mrada442@gmail.com>
Date: Thu, 24 Aug 2023 16:36:00 -0700

I suspected as much. If I could only find something that would show me how to install it and be operational so I can finish my upgrade project. Going to a new software is something that would bring on risk, unless I could narrow down what exactly the UNAS is being used for. This effort I am working on is for the contract I am working on.

What would really help is documentation. Installation, operation so that I can tell that it is working.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 25 Aug 2023 04:13:54 +0200

> What would really help is documentation. Installation, operation so that I can tell that it is working.

If your customer has UNAS, they probably have documentation, or what passes as documentation.

BTW, UNAS is a framework for distributed applications i.e. multiple programs doing remote procedure calls and message passing over the network.

From: Chris Sparks <mrada442@gmail.com>
Date: Fri, 25 Aug 2023 17:41:09 -0700

Unfortunately they don't have any documentation as it was set up very long ago.

If I had it on my home PC I would have more time to look at it.

Project Euler 26

From: Csyh (Qaq) <schen309@asu.edu>
Subject: project euler 26
Date: Mon, 4 Sep 2023 02:19:51 -0700
Newsgroups: comp.lang.ada

I am new to Ada, I know is there a good way to start this program? Thanks

https://projecteuler.net/problem=26

[The problem is: Find the value of d<1000 for which 1/d contains the longest recurring cycle in its decimal fraction part. –arm]

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Mon, 4 Sep 2023 14:06:13 +0300

First invent/discover the method (algorithm) for solving the problem, without thinking about the programming language.

I don't think any language has built-in features that would lead to a direct solution, although some functional language with lazy evaluation could come close, because such languages can manipulate unbounded (potentially infinite) sequences of values. Such sequences can be handled in Ada, too, but with more effort -- they are not "built in" to Ada.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 4 Sep 2023 14:39:17 +0200

Infinite division does not require big numbers, which Ada 22 has, but I i would not use them anyway because the performance would be abysmal.

BTW, Ada is perfect for numeric algorithms no need to resort to functional mess... (:-))

The problem itself requires as you said mathematical analysis, because a naive method of comparing a partial division result with itself is obviously wrong. E.g. let you have 0.12341234... you could not conclude that the period is (1234) because it could actually be (123412345).

From: Ben Bacarisse <ben.usenet@bsb.me.uk>
Date: Mon, 04 Sep 2023 17:01:04 +0100

> BTW, Ada is perfect for numeric algorithms no need to resort to functional mess... (:-))

Perfect? That's a bold claim!

Mind you, I don't think this problem is really a numerical one in that sense. It needs some simple integer arithmetic but then every language is perfect for that sort of arithmetic.

Using a functional mess (Haskell) a simple, native solution (i.e. using no modules) is only 9 lines long.

I don't want to start a language war. Ada is just more 'wordy' by deliberate design so a simple Ada solution is inevitably going to be longer in terms of lines. Rather my purpose in posting is to steer the OP away from thinking of this as a numerical problem in the classical sense. It really isn't.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 4 Sep 2023 21:20:56 +0200

[...]

> Using a functional mess (Haskell) a simple, native solution (i.e. using no modules) is only 9 lines long.

Apart from the fundamental inconsistency of functional paradigm: computing is about transition of states and nothing else; the imperative languages express solutions, i.e. an algorithm. Functional, and in general, declarative languages express puzzles.

They remind me of math examination tasks on studying a function. Here is a definition. Go figure out the properties and behavior...

Or, if you want, functional is like a chess composition: white to move and checkmate in 4 moves. Challenging, but Ada is about playing chess.

From: Ben Bacarisse <ben.usenet@bsb.me.uk>
Date: Mon, 04 Sep 2023 21:18:16 +0100

> Apart from the fundamental inconsistency of functional paradigm [...]

Rather than try to unpick that paragraph I'll just say that they can, none the less, give simple solutions to this sort of programming problem.

From: Francesc Rocher <francesc.rocher@gmail.com>
Date: Thu, 7 Sep 2023 00:31:09 -0700

> I am new to Ada, I know is there a good way to start this program?

Please take a look at my Euler tools repository, https://github.com/rocher/euler_tools (not the best math lib you'll find, I know).

I used this library tools to solve problem 26 here: https://github.com/rocher/alice-project_euler-rocher

Let me know what you think.

Equivalence between Named Anonymous Access

From: Blady <p.p11@orange.fr>
Subject: Equivalence between named access and anonymous access.
Date: Wed, 6 Sep 2023 16:37:08 +0200
Newsgroups: comp.lang.ada

I'm wondering about named access and anonymous access. In the following Ada code, are the writing of parameter P1 type of procedures PA and PB equivalent?

```
package C1 is
   type Inst is tagged null record;
   type Class is access all Inst'Class;
end C1;
```

```
with C1;
  package C2 is
    type Inst is tagged null record;
    type Class is access all Inst'Class;


    procedure PA (Self : Inst;
       P1 : C1.Class); -- named access
    procedure PB (Self : Inst; P1 : access
       C1.Inst'Class); -- anonymous access
  end C2;
```

Same with:

```
  function FA (Self : Inst) return C1.Class;
  -- named access
  function FB (Self : Inst) return access
     C1.Inst'Class; -- anonymous access
```

Are FA and FB writing equivalent?

If not, why?

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Wed, 6 Sep 2023 17:54:42 +0200

They are not equivalent from the access checks point of view:

```
   declare
     Y : C2.Inst;
     X : aliased C1.Inst;
   begin
     C2.PA (Y, X'Access);
     -- Non-local pointer error
     C2.PB (Y, X'Access); -- Fine
   end;
```

Furthermore, tagged anonymous access is controlling (dispatches) when not class-wide.

From: Gautier Write-Only Address
<gautier_niouzes@hotmail.com>
Date: Wed, 6 Sep 2023 13:55:02 -0700

> In the following Ada code, are the writing of parameter P1 type of procedures PA and PB equivalent?

They are not equivalent because the anonymous access opens more possibilities (example below), but you are certainly aware of that.

So I guess you have another question in mind...

```
with C1, C2;
procedure test is
  x2 : C2.Inst;
  type My_Reference_1 is access all
     C1.Inst'Class;
  r1 : My_Reference_1;
begin
  x2.PB (r1);
  x2.PA (r1);
  -- ^ expected type "Class" defined at
  -- c1.ads:3 found type "My_Reference_1"
  -- defined at line 6
end;
```

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 7 Sep 2023 02:20:02 +0200

> I'm wondering about named access and anonymous access.

The rules for using access-to-object types are:

1. Don't use access types
2. If you think you should use access types, see rule 1.
3. If you still think you should use access types, don't use anonymous access types
4. If you still think you should use anonymous access types, don't develop software

The semantics of named access types are well defined and easily understood. The semantics of anonymous access types are defined in ARM 3.10.2, of which the AARM says
"Subclause 3.10.2, home of the accessibility rules, is informally known as the 'Heart of Darkness' amongst the maintainers of Ada. Woe unto all who enter here (well, at least unto anyone that needs to understand any of these rules)."

The ARG freely admits that no one understands 3.10.2, which means that what you get when you use anonymous access types is whatever the compiler writer thinks it says. This may differ between compilers and between different versions of the same compiler, and from what you think it says.

So no sane person uses them.

From: Blady <p.p11@orange.fr>
Date: Thu, 7 Sep 2023 18:06:15 +0200

Thanks Dmitry, also Gautier and Jeff for your previous answers.

Well, I was questioning myself about the choice between named access and anonymous access in the old Ada port of Java library, for instance:

```
   type Typ;
   type Ref is access all Typ'Class;
   type Typ(LayoutManager2_I :
     Java.Awt.LayoutManager2.Ref;
     Serializable_I : Java.Io.Serializable.Ref)
    is new Java.Lang.Object.Typ
     with null record;
   ---------------------------------------
   -- Constructor Declarations --
   ---------------------------------------
   function New_BorderLayout
      (This : Ref := null) return Ref;
   function New_BorderLayout
      (P1_Int : Java.Int;
       P2_Int : Java.Int;
       This : Ref := null) return Ref;
   --------------------------------
   -- Method Declarations --
   --------------------------------
   procedure AddLayoutComponent
     (This : access Typ;
      P1_Component : access
         Standard.Java.Awt.
         Component.Typ'Class;
      P2_Object : access
          Standard.Java.Lang.
          Object.Typ'Class);
   function GetLayoutComponent
     (This : access Typ;
      P1_Object : access
         Standard.Java.Lang.
         Object.Typ'Class)
      return access Java.Awt.Component.
        Typ'Class;
```

Why choose named access for New_BorderLayout and anonymous access for AddLayoutComponent or GetLayoutComponent for the type of parameters P1_xxx and the return type?

Why not all named or all anonymous?

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>

Date: Thu, 7 Sep 2023 18:18:11 +0200

> Why choosing named access for New_BorderLayout and anonymous access for AddLayoutComponent

It's very poor design to have access types in the visible part of a non-private pkg spec.

From: Blady <p.p11@orange.fr>
Date: Thu, 7 Sep 2023 21:10:12 +0200

> It's very poor design to have access types in the visible part of a non-private pkg spec.

I got your point :-)

But, in this specific case, I was wondering why not writing all with named access or all with anonymous access?

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Thu, 7 Sep 2023 22:23:37 +0200

> Well, I was questioning myself about the choice between named access and anonymous access in the old Ada port of Java library [...]

Contravariance is unsafe. I gather that Typ is tagged. If you ever derive from it, it will "inherit" the broken construction function, because the function is class-wide. The safe choice here is anonymous access. The compiler will require to override the construction function. That is for the return value. The case for the argument depends. Again anonymous access type is handier but if you going to copy/store references, then named types are better.

> Why not all named or all anonymous?

My rough rule is like this:

Do not expose access types if you can.

If you successfully hide them either completely or by declaring them private, then named they go.

If you expose access types, then anonymous access is usually a better choice because it is easier to use, especially when access is merely to work around language limitations on argument/result passing (unconstrained object, access rules nightmare) AKA

closures. Then it is much safer in a hierarchy of types and it is more use-clause friendly.

Is Gnoga Dead?

From: Reinert <reinkor@gmail.com>
Subject: Is gnoga dead?
Date: Sat, 9 Sep 2023 02:02:46 -0700
Newsgroups: comp.lang.ada

I asked google bard:
"Is gnoga currently maintained?"

And it answered:

"No, Gnoga (Ada) is also no longer maintained. It was a fork of the original Gnoga project that was created in 2012. The last commit to the GitHub repository was in 2016.

Gnoga (Ada) was an attempt to improve the performance and scalability of Gnoga. It used a different statistical approach to NLG and was implemented in Ada.

There are a number of other NLG engines available, such as GPT-3 and LaMDA, which are more powerful and up-to-date than Gnoga (Ada)."

Is this correct?

From: Blady <p.p11@orange.fr>
Date: Sat, 9 Sep 2023 11:53:36 +0200

Do you mean this one:
https://sourceforge.net/projects/gnoga?

If so, the answer is yes [it is maintained –arm].

Could you please provide the link where you found the above quoted text?

From: Reinert <reinkor@gmail.com>
Date: Sat, 9 Sep 2023 04:47:41 -0700

I just logged into my google account and asked "bard" (https://bard.google.com/?hl=en).

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 9 Sep 2023 13:59:00 +0200

> I asked google bard:

> [...]

> Is this correct?

The presence of the word "also" in the first sentence should be enough to tell you that this is nonsense.

If you are asking about an "NLG engine" named gnoga, then maybe this is correct. If you are asking about the Ada web-application framework Gnoga that Blady maintains, then it's not correct.

From: Reinert <reinkor@gmail.com>
Date: Sat, 9 Sep 2023 09:46:26 -0700

Good to hear.

Yes, did mean the Ada web-application framework.

From: Stéphane Rivière <stef@genesix.org>
Date: Sat, 9 Sep 2023 19:55:56 +0200

Gnoga is not dead, being maintained by Pascal, Gautier and other individuals and companies.

Expect a pleasant surprise (imho) before the end of the year (with full web demos). v22 manual abstract:

1 About v22 framework
1.1 Ready to use in production
v22 is a general purpose, KISS oriented, modular Ada framework for GNU/Linux Debian/Ubuntu service, console and web programs.

v22 is composed of many packages in charge of UTF-8 strings, program and OS functions, HTTP(s)/WS(s) web framework, integrated cURL, console handling and text files, advanced network, MySQL and SQLite high level binding, logging and configuration files handling.

Although based on the v20 library, the v22 framework represents a major step forward in the following areas:

- UTF-8 compatibility;
- Simplified string processing (only one UTF-8 String type is used);
- Internationalization;
- New and extended database API;
- Extended database access to MySQL, in addition to SQLite, with schema on-the-fly update at table, index, and column level;
- Improved concurrent access and performance for SQLite;
- New LGPLv3 licensing instead of GPLv3;
- New FSF GNAT GCC Linux ready-to-use development environment for v22 (not tied anymore to GPLv2 license);
- And much more.

1.2 Cooperative and open

v22's native dependencies are Gnoga, Simple_Components, UXStrings and Zanyblue.

v22 is both a high-level framework and an extension to the lower level components cited above. v22 has been designed to:

- Use unmodified components;
- Not "reinvent the wheel". Component functions are to be used first;
- Offer higher-level functions or functions that do not exist in the components.

.../...

In short:

- UXStrings is used throughout v22. The v22.Uxs package extends UXStrings functionality. The v22.Sql package extends the functionality of Gnoga.Server.Database. The v22.Gui graphics framework is based on Gnoga.Gui;
- v22's architecture allows it to be open to additional packages, depending on the software development required.

From: Reinert <reinkor@gmail.com>
Date: Sat, 9 Sep 2023 23:51:12 -0700

Sounds like I can somehow trust that gnoga will be around for many years to come.

So my special issue: I work on making my (cancer) cellular behavior analysis program (https://korsnesbiocomputing.no) as a "cloud service". It's all programmed in Ada using GLOBE_3D. I am considering using guacamole apache. It's intensive about handling images. So what are the arguments for and against using guacamole (as compared to for example guacamole apache)?

From: Stéphane Rivière <stef@genesix.org>
Date: Mon, 11 Sep 2023 09:52:45 +0200

> Sounds like I can somehow trust that gnoga will be around many years to come.

I think so.

> So my special issue: I work on making my (cancer) cellular behavior analysis program (https://korsnesbiocomputing.no) as a "cloud service".

Very interesting indeed.

> [...] So what are the arguments for and against using guacamole (as compared to for example guacamole apache)?

No idea. My first concerns could be scaling.

From: Reinert <reinkor@gmail.com>
Date: Mon, 11 Sep 2023 23:00:45 -0700

Did mean ..... *gnoga* (as compared to, for example, guacamole apache) :-)

From: Stéphane Rivière <stef@genesix.org>
Date: Tue, 12 Sep 2023 09:16:52 +0200

> Did mean ..... *gnoga* (as compared to for example guacamole apache) :-)

No, sorry, I was thinking more of Guacamole. Not really fond of a remote desktop vs a true Web app... But maybe I'm wrong...

Aggregate with Derived Types

From: Blady <p.p11@orange.fr>
Subject: Aggregate with derived types.
Date: Thu, 14 Sep 2023 16:02:39 +0200
Newsgroups: comp.lang.ada

I want to extend a container type like Vectors, I've written:

```
type My_Float_List2 is new
   My_Float_Lists.Vector with null record;
```

But the initialization gives an error line 6:

```
1. with Ada.Containers.Vectors;
2. with Ada.Text_IO;
3. procedure
 test_20230914_derived_agg is
4.    package My_Float_Lists is new
Ada.Containers.Vectors (Positive, Float);
5.    subtype My_Float_List1 is
My_Float_Lists.Vector;
6.    type My_Float_List2 is new
My_Float_Lists.Vector with null record;
7.    ML1 : My_Float_List1 := [-3.1, -6.7,
3.3, -3.14, 0.0];
8.    ML2 : My_Float_List2 := ([-3.1, -6.7,
3.3, -3.14, 0.0] with null record);  |
>>> error: no unique type for this
aggregate
9. begin
10.    Ada.Text_IO.Put_Line
 (ML1.Element (3)'Image);
11.    Ada.Text_IO.Put_Line
(ML2.Element (3)'Image);
12. end test_20230914_derived_agg;
```

The RM says:

4.3.2 Extension Aggregates

1 [An extension_aggregate specifies a value for a type that is a record extension by specifying a value or subtype for an ancestor of the type, followed by associations for any components not determined by the ancestor_part.]

Language Design Principles

1.a The model underlying this syntax is that a record extension can also be viewed as a regular record type with an ancestor "prefix".

The record_component_association_list corresponds to exactly what would be needed if there were no ancestor/prefix type. The ancestor_part determines the value of the ancestor/prefix.

Syntax
2 extension_aggregate ::=
(ancestor_part with
record_component_association_list)
3 ancestor_part ::= expression |
subtype_mark

It is not so clear for me what a unique type could be? Any clue?

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 14 Sep 2023 17:31:47 +0200

IIUC, you have to qualify the value:

```
(My_Float_List1'[-3.1, -6.7, 3.3, -3.14, 0.0]
   with null record)
```

or

```
(My_Float_Lists.Vector'[-3.1, -6.7, 3.3,
    -3.14, 0.0] with null record)
```

(not tested)

From: Blady <p.p11@orange.fr>
Date: Thu, 14 Sep 2023 22:00:19 +0200

Thanks Jeff, both proposals are compiled ok by GNAT.

I wonder why the float list aggregate isn't inferred by the compiler and need some help with a qualification.

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Thu, 14 Sep 2023 23:37:47 +0200

I'm not sure. But can't you simply write

```
ML2 : My_Float_List2 := [-3.1, -6.7, 3.3,
  -3.14, 0.0];
```

? I presume that My_Float_List2 inherits its aggregate definition from My_Float_List1.

From: Blady <p.p11@orange.fr>
Date: Fri, 15 Sep 2023 09:27:57 +0200

Unfortunately not directly:

```
10.    ML2c : My_Float_List2 := [-3.1, -6.7,
3.3, -3.14, 0.0];                  |
>>> error: type of aggregate has private
ancestor "Vector"
>>> error: must use extension aggregate
```

Shouldn't it inherit them?

Indeed you have it if you defined a private extension with explicit aspects:

```
package PA is
   type My_Float_List3 is new
      My_Float_Lists.Vector with private
      with
         Constant_Indexing =>
            Constant_Reference,
         Variable_Indexing => Reference,
         Default_Iterator => Iterate,
         Iterator_Element  => Float,
         Aggregate =>
            (Empty => Empty,
             Add_Unnamed => Append,
             New_Indexed => New_Vector,
             Assign_Indexed =>
                Replace_Element);
   function Constant_Reference
     (Container : aliased My_Float_List3;
      Index : Positive) return
        My_Float_Lists.
        Constant_Reference_Type is
       (My_Float_Lists.Constant_Reference
          (My_Float_Lists.Vector
             (Container), Index));
   function Reference (Container : aliased
      in out My_Float_List3;
      Index : Positive) return
         My_Float_Lists.Reference_Type is
     (My_Float_Lists.Reference
        (My_Float_Lists.Vector (Container),
         Index));
private
   type My_Float_List3 is new
      My_Float_Lists.Vector with
      null record;
end PA;
ML3 : PA.My_Float_List3 := [-3.1, -6.7,
   3.3, -3.14, 0.0];
```

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Sat, 16 Sep 2023 01:39:57 -0500

> I wonder why the float list aggregate isn't inferred by the compiler and need some help with a qualification.

The language rule is that the ancestor_part of an extension aggregate is expected to be of "any tagged type" (see 4.3.2(4/2)). An aggregate needs to have a single specific type, and "any tagged type" is not that.

The reason that the ancestor is "any tagged type" is that the type of the ancestor determines the extension components needed along with other legality rules. One could imagine a language where all of these things are decided simultaneously, but people worried that the complexity would make it difficult/impossible to implement. So aggregates are essentially black boxes whose type has to be determinable from the outside, and similar rules exist for parts inside the aggregate.

Project Euler 29

From: Csyh (Qaq) <schen309@asu.edu>
Subject: project euler 29
Date: Fri, 15 Sep 2023 02:03:16 -0700
Newsgroups: comp.lang.ada

Now this time, I am facing trouble for problem #29.

[How many *distinct* terms are in the sequence (for a in 2 .. 100 => (for b in 2 .. 100 => a**b))? –arm]

As I know integer type is for 32 bits. but for this problem as me to find out the 2 ** 100 and even 100 ** 100.

I used Python to get the answer correctly in 5 minutes.

```
context = []
for a in range(2,101):
    for b in range(2,101):
        context.append(a**b)
len(list(set(context)))
```

I know the algorithm is easy, but I am pretty interested in how to calculate a large [?] like it. And thanks for the help from problem 26, your discussions come to me every working hour.

For this problem I want to know how to know is there an easy way to store a large number like 100 ** 100, and how do you make a similar function like "set(context)" to delete the duplicate value in a vector.

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Fri, 15 Sep 2023 11:50:42 +0200

> for this problem I want to know how to know is there an easy way to store a large number like 100 ** 100, and how do U make a similar function like "set(context)" to delete the duplicated value in a vector.

You will need an unbounded-integer pkg. If you want to write portable code in a standard language, then you can write Ada 12 using a library such as PragmARC.Unbounded_Numbers. Integers

(https://github.com/jrcarter/PragmARC/blob/Ada-12/pragmarc-unbounded_numbers-integers.ads). This will compile with both GNAT and ObjectAda.

If you want to write non-portable code in a non-standard, Ada-like language, then you can use the GNAT language, which is mostly Ada 12 with some Ada 23 features, one of which is the Ada-23 standard package Ada.Numerics.Big_Numbers.Big_Integers (http://www.ada-auth.org/standards/22aarm/html/AA-A-5-6.html). This can only be compiled with GNAT. Note that, unlike PragmARC.Unbounded_Numbers.Integers, GNAT's implementation of Ada.Numerics.Big_Numbers.Big_Integers is not truly unbounded. I don't know if it will hold 101 ** 101 without modification.

You can store the results directly in a set from the standard library to avoid duplicate values. If I understand your Python (probably not), you would want to output the result of Length for the resulting set.

From: Ben Bacarisse <ben.usenet@bsb.me.uk>
Date: Fri, 15 Sep 2023 16:42:38 +0100

> I know the algorithm is easy [...]

Most of the Project Euler problems have solutions that are not always the obvious one (though sometimes the obvious one is the best). You can, of course, just use a big number type (or write your own!) but this problem can be solved without having to use any large numbers at all.

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Fri, 15 Sep 2023 18:34:21 +0200

> As I know integer type is for 32 bits [...]

I missed this the first time.

No, you don't know that Integer is 32 bits. ARM 3.5.4 (21) [http://www.ada-auth.org/standards/aarm12_w_tc1/html/AA-3-5-4.html] requires "In an implementation, the range of Integer shall include the range –2**15+1 .. +2**15–1."

There are compilers for which Integer is less than 32 bits, so assuming otherwise is not portable. I know a lot of people don't care about portability, but I've also seen projects that spent large sums porting code that they thought didn't have to be portable. The cost of writing portable code is usually much smaller than the cost of porting non-portable code.

Of course, you can always declare your own integer type with whatever range is appropriate for your problem, though the compiler doesn't always have to accept it. I don't know of any compiler that doesn't accept 32-bit integer declarations, nor any targeting 64-bit platforms that doesn't accept 64-bit integers. But you're unlikely to find a compiler that will accept range 2 .. 101 ** 101

In King (https://github.com/jrcarter/King) the compiler must accept all integer type declarations.

From: Keith Thompson <keith.s.thompson+u@gmail.com>
Date: Fri, 15 Sep 2023 11:04:27 -0700

> I don't know if it will hold 101 ** 101 without modification.

It only has to hold 100 ** 100. The Python code in the parent uses the expression `range(2,101)`. Python's range() function yields a range that includes the first bound and excludes the second bound.

From: Francesc Rocher <francesc.rocher@gmail.com>
Date: Sat, 16 Sep 2023 03:07:06 -0700

Please take a look at this solution:

https://github.com/rocher/alice-project_euler-rocher/blob/main/src/0001-0100/p0029_distinct_powers.adb

It's not using any big numbers library.

From: Paul Rubin <no.email@nospam.invalid>
Date: Sun, 17 Sep 2023 15:54:12 -0700

> Also, do you have a different approach to solve this 29th problem?

I see two natural approaches: 1) use bignums--it didn't occur to me to not use them until this discussion. 2) Notice that a**b == c**d exactly when the two sides have the same prime factorization, and the factors of a**b are just the factors of a repeated b times, so you can count up the distinct tuples of factors.

Method #2 is efficient (since a,b,c,d are all < 100) and doesn't use bignums, but it is a fair amount of code to write unless you have convenient libraries at hand for factorization and can easily count sets of distinct tuples. I guess there are fancier approaches possible too, that avoid searching 100**2 combinations, but 100**2 is just 10000 which is small.

Certainly both are easier to do if your language or libraries has convenient features for dealing with variable sized objects like bignums, or sets of tuples. The bignum approach is less efficient but it is much easier to code. The Python expression

```
len(set(a**b for a in range(2,101) for b in range(2,101)))
```

takes around 25 msec to compute on my old slow laptop.

I will look at your Ada solution!

From: Paul Rubin <no.email@nospam.invalid>
Date: Sun, 17 Sep 2023 17:09:38 -0700

>> Also, do you have a different approach to solve this 29th problem?

> Yes, but it's not in Ada. I implemented an equality test for a^b == c^d.

Oh interesting, based on a comment in Francesc's code, I think I see a method to do it without the auxiliary array, at a small increase in runtime cost. Basically given a and b, you can find their prime factors and easily enumerate the combinations x,y with a**b==x**y and 1 <= x,y <= 100. You can label each "equivalence class" by the (a,b) with the smallest possible a.

So you just loop through 1 <= a,b <= 100 and count only the a,b pairs where a is the smallest a for its equivalence class. I might see if I can code this, which should also let me describe it more concisely.

From: Ben Bacarisse <ben.usenet@bsb.me.uk>
Date: Mon, 18 Sep 2023 01:16:19 +0100

> So you just loop through 1 <= a,b <= 100 and count only the a,b pairs where a is the smallest a for its equivalence class.

This is likely to be fast which is why I wanted to compile Francesc's to try it out. Mind you, a naive a^b == c^d test gives pretty good performance for the kind of range requested.

Get Character and Trailing New Lines

From: Blady <p.p11@orange.fr>
Subject: Weird behavior of Get character with trailing new lines.
Date: Fri, 22 Sep 2023 21:30:15 +0200
Newsgroups: comp.lang.ada

I'm reading a text file with Get character from Text_IO with a while loop controlled by End_Of_File.

% cat test_20230922_get_char.adb

```
with Ada.Text_IO; use Ada.Text_IO;
procedure test_20230922_get_char is
   procedure Get is
      F : File_Type;
      Ch : Character;
   begin
      Open (F, In_File,
         "test_20230922_get_char.adb");
      while not End_Of_File(F) loop
        Get (F, Ch);
        Put (Ch);
      end loop;
      Close (F);
      Put_Line ("File read with get.");
   end;
begin
Get;
end;
```

All will be well, unfortunately not!

Despite the End_Of_File, I got an END_ERROR exception when there are

several trailing new lines at the end of the text:

```
[...] Execution of
../bin/test_20230922_get_char terminated by
unhandled exception
raised ADA.IO_EXCEPTIONS.END_ERROR
: a-textio.adb:517
```

The code is compiled with GNAT, does it comply with the standard?

A.10.7 Input-Output of Characters and Strings

For an item of type Character the following procedures are provided:

```
procedure Get(File : in File_Type;
   Item : out Character);
procedure Get(Item : out Character);
```

After skipping any line terminators and any page terminators, reads the next character from the specified input file and returns the value of this character in the out parameter Item. The exception End_Error is propagated if an attempt is made to skip a file terminator.

This seems to be the case, then how to avoid the exception?

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Fri, 22 Sep 2023 22:52:21 +0300

In Text_IO, a line terminator is not an ordinary character, so you must handle it separately, for example like this:

```
while not End_Of_File(F) loop
   if End_Of_Line(F) then
      New_Line;
      Skip_Line(F);
   else
      Get (F, Ch);
      Put (Ch);
   end if;
```

From: Jeffrey R.Carter
<spam.jrcarter.not@spam.acm.org.not>
Date: Fri, 22 Sep 2023 22:05:55 +0200

As you have quoted, Get (Character) skips line terminators. End_Of_File returns True if there is a single line terminator before the file terminator, but False if there are multiple line terminators before the file terminator. So you either have to explicitly skip line terminators, or handle End_Error.

From: J-P. Rosen <rosen@adalog.fr>
Date: Sat, 23 Sep 2023 09:02:37 +0200

And this works only if the input file is "well formed", i.e. if it has line terminators as the compiler expects them to be (f.e., you will be in trouble if the last line has no LF). That's why I never check End_Of_File, but handle the End_Error exception. It always works.

From: Niklas Holsti
<niklas.holsti@tidorum.invalid>
Date: Sat, 23 Sep 2023 11:39:25 +0300

Hm. The code I suggested, which handles line terminators separately, does work without raising End_Error even if the last line has no line terminator, at least in the context of the OP's program.

> That's why I never check End_Of_File, but handle the End_Error exception. It always works.

True, but it may not be convenient for the overall logic of the program that reads the file. That program often wants to do something with the contents, after reading the whole file, and having to enter that part of the program through an exception does complicate the code a little.

On the other hand, past posts on this issue say that using End_Error instead of the End_Of_File function is faster, probably because the Text_IO code that implements Get cannot know that the program has already checked for End_Of_File, so Get has to check for that case anyway, redundantly.

My usual method for reading text files is to use Text_IO.Get_Line, and (I admit) usually with End_Error termination.

From: Dmitry A. Kazakov
<mailbox@dmitry-kazakov.de>
Date: Sat, 23 Sep 2023 11:25:05 +0200

> [...] having to enter that part of the program through an exception does complicate the code a little.

It rather simplifies the code. You exit the loop and do whatever is necessary there.

Testing for the file end is unreliable and non-portable. Many types of files simply do not support that test. In other cases the test is not file immutable with the side effects that can change the program logic.

It is well advised to never ever use it.

From: Blady <p.p11@orange.fr>
Date: Mon, 25 Sep 2023 21:55:56 +0200

Thanks all for your helpful answers.It actually helps.

Especially, I was not aware of the particular behavior of End_Of_File with a single line terminator before the file terminator.

In my case, I prefer to reserve exceptions for exceptional situations :-) so I've taken the code from Niklas' example.

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Tue, 26 Sep 2023 00:53:53 -0500

> And this works only if the input file is "well formed"

Agreed. And if the file might contain a page terminator, things get even worse because you would have to mess around with End_of_Page in order to avoid hitting a combination that still will raise End_Error. It's not worth the mental energy to avoid it, especially in a program that will be used by others. (I've sometimes used the simplest possible way to writing a "quick&dirty" program for my own use; for such programs I skip the error handling as I figure I can figure out what I did wrong by looking at the exception raised. But that's often a bad idea even in that case as such programs have a tendency to get reused years later and then the intended usage often isn't clear.)

'Valid Attribute and Input Operations

From: Maciej Sobczak
<see.my.homepage@gmail.com>
Subject: Valid attribute and input operations
Date: Sat, 23 Sep 2023 13:22:09 -0700
Newsgroups: comp.lang.ada

I am in the middle of a heated debate with Richard Riehle on LinkedIn, where we cannot get to terms with regard to the exact semantics of X'Valid in the context of input operations performed by standard Get procedure.

In short, consider the following example:

```
with Ada.Text_IO;
with Ada.Integer_Text_IO;
procedure Is_Valid_Test is
   X : Integer range 0..200;
begin
   Ada.Text_IO.Put("Get an Integer: ");
   Ada.Integer_Text_IO.Get(X);
   if X'Valid then
      Ada.Text_IO.Put_Line
         ("The Input is Valid ");
   else
      Ada.Text_IO.Put_Line
         ("The Input is not Valid ");
   end if;
end Is_Valid_Test;
```

When the input is 500, what should be the behavior of this program?

There are two interpretations:

1. Get is an input operation and can create invalid representations (as stated in 13.9.2, p.7). Then, the X'Valid test that follows Get(X) can be used to safely recognize whether the value is in the range or not. The program should print the second string (from the else branch), but should not raise any exceptions for this input (500).
2. Get is not an input operation in the meaning referred to in 13.9.2/7, or is an input, but only for type Integer (and it cannot create invalid integer representations on typical computers anyway). The X variable is an actual parameter that has a subtype that is different from the formal parameter and is subject to conversions when the Get subprogram exits normally (6.4.1/17,17a). This conversion should raise Constraint_Error for this input (500).

I have checked the above program on several on-line compilers, all of them behave according to interpretation 2 above.

Richard claims to get behavior 1 on his compiler.

What is your take on this? Any language lawyers?

From: Jeffrey R.Carter <spam.jrcarter.not@spam.acm.org.not>
Date: Sat, 23 Sep 2023 23:48:49 +0200

> What is your take on this? Any language lawyers?

The important thing is the definition of Ada.Text_IO.Integer_IO.Get [ARM A.10.8(7-10)]:

"... skips any leading blanks, line terminators, or page terminators, then reads a plus sign if present or (for a signed type only) a minus sign if present, then reads the longest possible sequence of characters matching the syntax of a numeric literal without a point. ...

"Returns, in the parameter Item, the value of type Num that corresponds to the sequence input.

"The exception Data_Error is propagated if the sequence of characters read does not form a legal integer literal or if the value obtained is not of the subtype Num."

So a call to Get can only return a valid value of type Num (Integer for your case) or raise Data_Error.

If Get is reading "500" then that certainly represents a valid value of type Integer, and Get should copy that back to the actual parameter.

If you are using Ada (a language with run-time checks), then a check should be made that the value is in the range of the actual parameter's subtype, here Integer range 0 .. 200. That should fail and Constraint_Error should be raised.

However, if you are not using Ada because that check has been suppressed, then the actual parameter will be left with the invalid value 500 and Constraint_Error will not be raised.

If I build your program with checks enabled, I get Constraint_Error. If I build it with checks suppressed, I get the not-valid message (GNAT 12.3).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 26 Sep 2023 01:13:53 -0500

I believe Jeffrey's analysis is correct.

Note that there are some special cases for validity that are intended to make it easier to write code like that you have. But they only make sense for base subtypes (and the type you have is not that). Moreover, they are not foolproof -- execution is not erroneous in these cases, but they still are a bounded error, and it is always correct for a bounded error to be detected and raise Program_Error.

This can happen in practice, too. For instance, for Janus/Ada, enumeration types with specified representations operate internally on the position numbers, and thus reading an enumeration variable will convert the representation to a position number with a table lookup. If the lookup fails, Program_Error is raised, and that happens before the value ever can be assigned to a named variable (and thus before any possible test of validity). I believe that we identified other similar cases back in the day. Probably one of them is the signalling NaN. Some bit patterns for float values represent signalling NaNs, which trap instantly if read. That's at the hardware level on most processors, so the only hope is to handle the resulting exception. It's too late by the time you get to 'Valid.

Moral: to make truly bulletproof code, you have to handle exceptions AND use 'Valid. You probably can skip the exceptions if everything is typed with integer basetypes, but if any other kinds of types are involved, they are necessary.

From: Niklas Holsti <niklas.holsti@tidorum.invalid>
Date: Tue, 26 Sep 2023 10:22:14 +0300

> ... for Janus/Ada, enumeration types with specified representations operate internally on the position numbers

Hm, that's interesting. Is that also the representation for record components of such an enumerated type?

For example, if I have:

```
type Command is (Off, On) with Size
   => 4;
for Command use (Off => 2, On => 5);
type Two_Commands is record
   C1, C2: Command;
end record
with Pack, Size => 8;
TwoC : Two_Commands :=
   (C1 => On, C2 => Off);
```

will the record components (in memory) have the values C1 = 1 and C2 = 0 (position numbers) or C1 = 5, C2 = 2 (specified representation)?

if they are represented by position numbers in the record, many if not most of my embedded Ada programs would fail if compiled with Janus/Ada, because the record values stored in I/O control registers or accessed via DMA would be wrong.

Damn, I thought those programs were not so compiler-dependent :-(

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 27 Sep 2023 22:27:41 -0500

No, the specified representation is always used when storing to memory (with the single exception of loop parameters, which cannot have address clauses or other representation specifications). I think even enum parameters are written in the representation. However, any time an enumeration value is read into a register it is converted to a position number. Usually, such values are used in indexing, comparing, or an attribute like 'Pos or 'Succ, all of which are defined to work on position numbers. But if you simply assign the value out again, it will get converted both ways. We do have an optimization to remove pairs of TOREP/DEREP, but not the reverse since Program_Error is a possibility from DEREP. (Well, unless unsafe optimizations are on, but I don't recommend using those for the obvious reasons.)

Should Light Runtimes Get More Consideration?

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Subject: Should light runtimes get more consideration?
Date: Tue, 26 Sep 2023 11:44:02 -0000
Newsgroups: comp.lang.ada

I created the issue below a little while ago. Today I wonder whether Ada 2022s 'Image attribute on records use of Unbounded strings is for good reason. Is it an oversight that Bounded String would work with the new light runtime or String with all runtimes including the older zero footprint runtimes?

Perhaps it was decided that a light runtime would not use this feature? And I can certainly avoid it. However I use a light runtime with 100s of kilobytes or RAM and many gigabytes of flash.

Ada is a much nicer language than Rust which uses unsafe all over for embedded but one thing that is interesting is that I believe all Rust code can be run easily on any target. Should Ada aspire to that?

On the other hand, micros are becoming multiprocessors bringing more demand for tasking (protected types are not compatible with a light runtime) but personally I believe multi chip single core designs are far better than multicore and not only due to the impossibility of side channel attacks like Spectre.

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/67

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 27 Sep 2023 22:48:16 -0500

As noted on the ARG Github, you confused the Unbounded version of Text_Buffers with an unbounded string (completely unrelated things), and

moreover, failed to notice that the language provides multiple ways to use a Bounded Text_Buffer instead. So the language addresses this particular concern.

I don't know if GNAT implements all of those ways (in particular, the restriction Max_Image_Length), but that is hardly the fault of the language!

For anyone else interested in this particular discussion, I recommend reading and following up on the ARG Githib issue rather than here (https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/67).

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Thu, 28 Sep 2023 09:46:23 -0000

> Bounded Text_Buffer instead. So the language addresses this particular concern.

> I don't know if GNAT implements all of those ways [...]

I see. I guess the error message could suggest those options, too. Perhaps after the 2022 GNAT support work is completed.

That buffer support is pretty neat but my main concern, which GNAT may (it may not) address more than the current language by providing a cortex runtime, is that such demanding runtimes are brilliant but I am not sure if even Ravenscar is scalable to so many microchips such as Rust is trying to support. That isn't a huge issue but barriers to entry like having to work out your own exception replacement might be turning users away. Which is unfortunate when Ada is the best language out there by a significant margin for embedded development or frankly any protocol or hardware register use.

Of course others will rightly argue Ada is the best due to many of the more complex runtime features but that doesn't help with the issue of ease of adoption on an unsupported microchip that I have raised above.

From: Simon Wright <simon@pushface.org>
Date: Thu, 28 Sep 2023 14:25:18 +0100

When I started on Cortex GNAT RTS [1], a large part of the motivation (aside from the fun element) was that AdaCore's bare-board RTSs were GPL'd (they still are). Not that I cared about that, but other people did.

I took the approach of AdaCore's SFP (small footprint) profile, now renamed to light tasking, which implemented Ravenscar tasking but not exception propagation or finalization.

The tasking part wasn't too hard, though I think exception handling and finalization might have made things more difficult.

Basing the tasking on FreeRTOS saved a lot of grief (there are a couple of areas when the resulting semantic isn't _quite_ Ada's).

I did some work on finalization, not merged.

Exception handling, without finalization, seemed a daunting prospect, specially since the last project I worked on before retirement regarded an unhandled exception as requiring a reboot (and ditching any missiles in flight).

The current implementation has about 17 files (1 .h, 1 .s, 9 .ads, 4 .adb) to customise to the chip (setting up interrupt vectors, the clock, and memory). There are about 200 Ada sources that are common.

AdaCore currently has 68 RTS packages in the Alire gnat_arm_elf toolchain. 18 of these are 'embedded' packages (full Ada, but with Jorvik tasking). I'd be surprised if they had a higher proportion of chip dependency than my efforts. Most if not all of the exception handling will be chip-independent. I'm not sure how many of the 90 or so Ada sources in the STM32F4 gnarl/ directory are actually chip-dependent, I get the impression it's not high.

So, unless you're going to use some target that AdaCore haven't released support for, your best bet must be to either use or at worst start from the available RTS packages.

[1] https://github.com/simonjwright/cortex-gnat-rts

From: Drpi <314@drpi.fr>
Date: Thu, 28 Sep 2023 19:51:57 +0200

> I'm not sure how many of the 90 or so Ada sources in the STM32F4 gnarl/ directory are actually chip-dependent, I get the impression it's not high.

Right, not high.

I've created 2 of them based on one of the AdaCore RTS. I can't say it has been easy since you first have to understand how it works (and things change at each new release). One important point is that some critical parameters are hard coded in the source code. Like the core frequency. You MUST use a fixed clock frequency to get correct time management (delays, ...). This is why in their last version, you run a script to generate part of the RTS source code (frequency and other parameters are injected in the source code). When you change the core frequency you have to regenerate the RTS binary.

I created the RTS to evaluate the potential use of Ada on embedded targets. I have never used them except for testing. The main reason is that AdaCore RTS are made for specific use (avionics, spatial...). The code using these RTS must be provable (or as provable as possible). This induces big limitations. Tasking is very limited. For example you can't use timeouts. Never. They propose a workaround but it is complex and not equivalent to a real timeout management. I'd like to have a full Ada RTS for embedded targets, like on desktop. I don't need to certify/prove my hardware/software. Some people say micro-controllers are too limited for this. That's true for some of them. I use micro-controllers with megabytes of FLASH memory and hundreds of kilobytes of RAM. Is this not enough?

From: Simon Wright <simon@pushface.org>
Date: Thu, 28 Sep 2023 21:53:14 +0100

> I'd like to have a full Ada RTS for embedded targets, like on desktop.

Have you considered using something like a Raspberry Pi?

From: Drpi <314@drpi.fr>
Date: Thu, 28 Sep 2023 23:18:15 +0200

> Have you considered using something like a Raspberry Pi?

A RaspberryPi is a computer (based on a microprocessor with an OS), not a microcontroller. It consumes a lot of electrical power. The OS (linux) is not real time. It uses a lot of board space. The processor is a proprietary black box...

From: Chris Townley <news@cct-net.co.uk>
Date: Fri, 29 Sep 2023 00:51:11 +0100

> A RaspberryPi is a computer [...]

Plenty use the Raspberry Pi as a microcontroller.

From: Kevin Chadwick <kc-usenet@chadwicks.me.uk>
Date: Fri, 29 Sep 2023 09:59:37 -0000

>Plenty use the Raspberry Pi as a microcontroller.

I think Simons point was that ARM/Linux has a working full runtime. I guess bare Raspberry Pi would not and I guess it would be a rather large module or board or single board computer depending on the model.

WRT energy use. I use a low power run feature on the STM32L4 which means the system clock speed can change at any time. That seems to be incompatible with any runtime that I have seen except the minimal light-cortex-m4 one. I assume working with clocks is more scalable than working with runtimes but I do not know for sure.

From: Chris Townley <news@cct-net.co.uk>
Date: Fri, 29 Sep 2023 11:42:08 +0100

> I think Simons point was that Arm/Linux has a working full runtime. [...]

Agreed, but in addition to the mainline Pis there is the Zero, and the Pico, which has a 'RP2040' made by Raspberry Pi and is a dual-core ARM Cortex M0+ processor, with a flexible clock running up to 133MHz.

From: Drpi <314@drpi.fr>
Date: Fri, 29 Sep 2023 15:42:17 +0200

> WRT energy use. I use a low power run feature on the STM32L4 which means the system clock speed can change at any time. [...]

The fact that the clock speed is hard coded is a design choice. It simplifies the time management. It makes the runtime more "hard real time" compliant since there are less computations to be done at execution.

From: Drpi <314@drpi.fr>
Date: Fri, 29 Sep 2023 15:44:31 +0200

> [...] which has a 'RP2040' made by Raspberry Pi and is a dual-core ARM Cortex M0+ processor, with a flexible clock running up to 133MHz

A runtime for the RP2040 already exists. It is based on the AdaCore ARM runtimes so it has the same limitations.