Ada for teaching, learning and doing crystallography

Saulius Gražulis

Poznań, 2025

Vilnius University, Life Sciences Centre, Institute of Biotechnology
Vilnius University, Faculty of Mathematics and Informatics, Institute of Informatics

https://www.crystallography.net/cod/archives/2025/slides/2025-Poznan-CCF/slides.pdf

Id: slides.tex 3078 2025-08-23 05:42:55Z saulius August 23, 2025

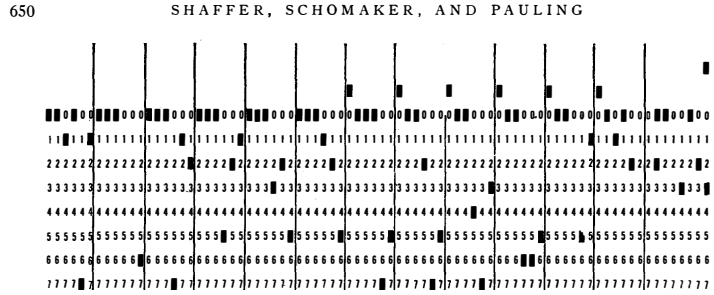
Talk plan

1. A brief historical note
2. Current problems with scientific and crystallographic software
3. A possible way to a solution
4. 2-3 code examples
5. Conclusions and plans

Computing in crystallography

History, background

- Crystallography – one of the earliest applications of computers in science :) (Shaffer et al. 1946);



- Well-defined theory with a lot of computable results;

Readability is a king

Software is a means to describe our science

- Software is more often read than written...
- We need an understandable, durable software;

Readability is a king

Software is a means to describe our science

- Software is more often read than written...
- We need an understandable, durable software;

"... it may equally well lead to an increase of errors if too much faith is put in results obtained with magical black boxes" (Kleywegt et al. 2002)

Readability is a king

Software is a means to describe our science

- Software is more often read than written...
- We need an understandable, durable software;

"... it may equally well lead to an increase of errors if too much faith is put in results obtained with magical black boxes" (Kleywegt et al. 2002)

"When the code is a black box, [software] may easily produce outputs that are simply accepted at face value" (Soergel 2015)

Software bugs

Mission critical?

- Retractions...
  - Retraction because of the F+/F- confusion (Miller 2006; Chang et al. 2006);
  - Retraction because of the buggy Perl script (B. G. Hall et al. 2007);
  - Retraction because of the buggy statistics script (Herzog et al. 2022);

Software bugs

Mission critical?

- Retractions...
  - Retraction because of the F+/F- confusion (Miller 2006; Chang et al. 2006);
  - Retraction because of the buggy Perl script (B. G. Hall et al. 2007);
  - Retraction because of the buggy statistics script (Herzog et al. 2022);

"Confusion around the implementation of the software's algorithms is common (24), even though the algorithms have been published in peer-reviewed literature (23)." (Joppa et al. 2013)

Software bugs

Mission critical?

- Retractions...
  - Retraction because of the F+/F- confusion (Miller 2006; Chang et al. 2006);
  - Retraction because of the buggy Perl script (B. G. Hall et al. 2007);
  - Retraction because of the buggy statistics script (Herzog et al. 2022);

"Confusion around the implementation of the software's algorithms is common (24), even though the algorithms have been published in peer-reviewed literature (23)." (Joppa et al. 2013)

"Scientific software code needs to be not only published and made available (6, 7) but also peer-reviewed." (ibid.)

Re-runnable understandable workflows

Make sure it works

“Our analysis showed that nearly 80% of the tested workflows failed to be either executed or produce the same results (if testable), and those from earlier years (2007-2009) had more than 80% failure rate” (Zhao et al. 2012)

Re-runnable understandable workflows

Make sure it works

“Our analysis showed that nearly 80% of the tested workflows failed to be either executed or produce the same results (if testable), and those from earlier years (2007-2009) had more than 80% failure rate” (Zhao et al. 2012)

Reproducibility killer:

```
-- Found Python: /home/saulius/src/pytorch/build/venv/bin/python3 (found version "3.8
CMake Error at cmake/Dependencies.cmake:832 (message):
  Found Python libraries version 3.8.10.  Python < 3.9 is no longer supported
  by PyTorch.
Call Stack (most recent call first):
  CMakeLists.txt:865 (include)
```

How to solve that?

A pseudo-code?

```
type Hash_Entry is record
   Key   : Key_Type;
   Value : Value_Type;
end record;

type Hash_Table is array (Positive range <>) of Hash_Entry;

function Perl_Hash (Key : Key_Type; Max_Size : Natural) return Natural is
   Hash_Value : Natural := 0;
begin
   for I in 1 .. Length (Key) loop
      Hash_Value := (Hash_Value * 31 + Character'Pos (Element (Key, I)))
        mod Max_Size;
   end loop;
   return Hash_Value + 1; -- Adjusting to 1-based index
end Perl_Hash;
```

Why Ada/SPARK?

1. Durable design – first designed in 1983!
2. Modern language – latest standard is Ada 2022;
3. Mostly backwards compatible;
4. Good F/LOSS compiler available – GNAT;
5. Ada is statically very strictly typed;
6. Programs are easy to read (Level (Ada) > Level (C));
7. Ada & SPARK has a rich type system;
8. Language level concurrent programming;
9. Produces fast optimised native code, links with any language;
10. SPARK subset takes computer arithmetic into account;
11. Not controlled by any private company;

Why is Ada not popular (yet)?

1. The language is complex and difficult to implement;
2. No good compilers in the 1990's;
3. Procured by the DOD, used for "war fighting software";
4. Poor academic outreach in the 20th century;

Why is Ada not popular (yet)?

1. ~~The language is complex and difficult to implement~~;
2. The language is rich and convenient to program/design in;
3. ~~No good compilers in the 2020's;~~
4. Very nice compiler and dev. system available: `gnat`;
5. Procured by the DOD, used for "war fighting software";
6. Used for mission-critical software (avionics, spacecraft ctrl., railways, plant ctrl...)
7. Poor academic outreach in the 21st century;
8. SPARK allows formal verification of the code (!);

Why is Ada not popular (yet)?

1. ~~The language is complex and difficult to implement~~;
2. The language is rich and convenient to program/design in;
3. ~~No good compilers in the 2020's;~~
4. Very nice compiler and dev. system available: `gnat`;
5. Procured by the DOD, used for "war fighting software";
6. Used for mission-critical software (avionics, spacecraft ctrl., railways, plant ctrl...)
7. Poor academic outreach in the 21st century;
8. SPARK allows formal verification of the code (!);

Readability of Ada

Student's responses

After 1 semester Ada intro during bioinformatics (analysis of 3D structures):

Q: "Name three strengths and three weaknesses of Ada"

Top pro-answer: "Ada is easy to read"

In the same vein: "Not too many ways to write code, its clear how to proceed"

Other pros: compilable, fast, F/LOSS, ...

Example task

Students' assignment

```
Write a Perl*) program that calculates torsion and/or bond angles for
macromolecules described in PDB and PDBx files and outputs them to
STDOUT. Particular angles will be assigned by the professor.

*) Note: Ada programs are acceptable; in that case all native Perl
   features mentioned in this specification SHOULD be replaced by the
   corresponding Ada features.

NB: for this particular assignment, you MAY NOT use external libraries
to read PDB files or calculate dihedral angles; these functions MUST
be implemented in the program.
```

Essentially, reproduce results from (Ramachandran et al. 1963) or the μ and θ angle computations for C_α atoms from (Škrbić et al. 2021).

Types with full operator overloading

High level abstractions

```
type Vector_3D is array (1 .. 3) of Long_Float;

-- a scalar product:
function "*" (V, W : Vector_3D) return Long_Float is
   VX : Long_Float := V(1);
   VY : Long_Float := V(2);
   VZ : Long_Float := V(3);

   WX : Long_Float := W(1);
   WY : Long_Float := W(2);
   WZ : Long_Float := W(3);
begin
   return VX * WX + VY * WY + VZ * WZ;
end;
```

Programming in large and in small

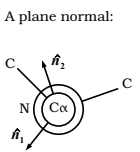
Explicit interfaces

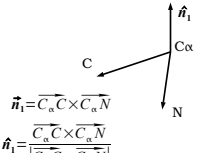
```
subtype Dihedral_Angle_Type is Long_Float range -180.0 .. 180.0;

function Dihedral (A : Chemical_Atom_Set) return Dihedral_Angle_Type is
   R1 : Vector_3D := To_Vector (A.Atoms (1));
   R2 : Vector_3D := To_Vector (A.Atoms (2));
   R3 : Vector_3D := To_Vector (A.Atoms (3));
   R4 : Vector_3D := To_Vector (A.Atoms (4));

   Axis : Vector_3D := R3 - R2;
   V1 : Vector_3D := R1 - R2;
   V2 : Vector_3D := R4 - R3;

   N1 : Vector_3D := Cross (Axis, V1);
   N2 : Vector_3D := Cross (Axis, V2);

   NX : Vector_3D := Cross (N1, N2);

   Sign : Long_Float := (if Axis * NX >= 0.0 then 1.0 else -1.0);
begin
   pragma Assert (A.N_Atoms = 4);
   return Sign * Angle(N1, N2);
end;
```

```
subtype Dihedral_Angle_Type is Long_Float range -180.0 .. 180.0;

function Dihedral (A : Chemical_Atom_Set) return Dihedral_Angle_Type is
   R1 : Vector_3D := To_Vector (A.Atoms (1));
   R2 : Vector_3D := To_Vector (A.Atoms (2));
   R3 : Vector_3D := To_Vector (A.Atoms (3));
   R4 : Vector_3D := To_Vector (A.Atoms (4));

   Axis : Vector_3D := R3 - R2;
   V1 : Vector_3D := R1 - R2;
   V2 : Vector_3D := R4 - R3;

   N1 : Vector_3D := Cross (Axis, V1);
   N2 : Vector_3D := Cross (Axis, V2);

   NX : Vector_3D := Cross (N1, N2);

   Sign : Long_Float := (if Axis * NX >= 0.0 t
begin
   pragma Assert (A.N_Atoms = 4);
   return Sign * Angle(N1, N2);
end;
```

Calculation of φ and ψ angles

A plane normal:





The angle between two planes:

$$\cos\varphi=(\hat{n}_1\cdot\hat{n}_2)=\frac{(\vec{n}_1\cdot\vec{n}_2)}{|\vec{n}_1||\vec{n}_2|} \qquad sign\,\varphi=sign([\hat{n}_1\times\hat{n}_2]\cdot\overrightarrow{C_\alpha C})$$

Dihedral angle survey in the PDB

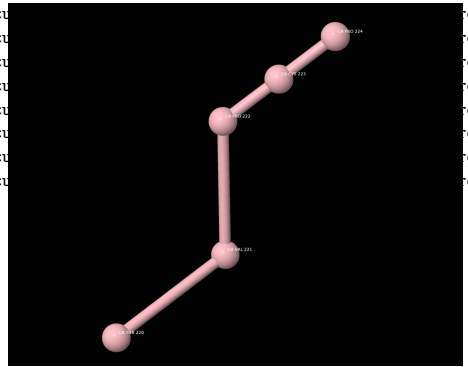
Funny angles in the PDB :)

```
+ find /usr/data/databases/PDB/current/mmCIF/ -name '*.cif.gz'
+ sort
+ xargs -r zcat
+ ./pdb_pep_torsions_tm
./pdb_pep_torsions_tm: could not calculate angle for the atom G   'CA  ' VAL  221 from '2QTJ' --
    Ada.Numerics.Argument_Error raised with invalid floating point dot product for Arccos
./pdb_pep_torsions_tm: could not calculate angle for the atom G   'CA  ' PRO  222 from '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom G   'CA  ' VAL  221 from '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom G   'CA  ' PRO  222 from '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom C   'CA  ' VAL  221 from '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom C   'CA  ' PRO  222 from '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom E   'CA  ' PHE  172 from '1X18' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom E   'CA  ' ILE  173 from '1X18' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calculate angle for the atom E   'CA  ' PRO  174 from '1X18' -- Ada.Numerics....
+ date
2025-08-18T19:55:48 EEST
```

Dihedral angle survey in the PDB

Funny angles in the PDB :)

```
+ find /usr/data/databases/PDB/current/mmCIF/ -name '*.cif.gz'
+ sort
+ xargs -r zcat
+ ./pdb_pep_torsions_tm
./pdb_pep_torsions_tm: could not calculate angle for the atom G   'CA  ' VAL  221 from '2QTJ' --
    Ada.Numerics.Argument_Error raised with invalid floating point dot product for Arccos
./pdb_pep_torsions_tm: could not calcu                                    om '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '2QTJ' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '1X18' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '1X18' -- Ada.Numerics....
./pdb_pep_torsions_tm: could not calcu                                    om '1X18' -- Ada.Numerics....
+ date
2025-08-18T19:55:48 EEST
```



Code example

Space group symbol decoder

Decode Hall and H-M symbols with explicit change-of-basis; e.g:

`C121 (x-y, x+y, z),  C121 (1/2*a - 1/2*b, 1/2*a + 1/2*b, c)`

(Shmueli et al. 2001; Zwart et al. 2007)

```
type Axis_Direction_Type is
  (X_AXIS, Y_AXIS, Z_AXIS, UNKNOWN);

subtype Known_Axis_Direction is
  Axis_Direction_Type range X_AXIS .. Z_AXIS;

type Axis_Order_Type is
  (IDENTITY, TWOFOLD, THREEFOLD, FOURFOLD, SIXFOLD, UNKNOWN);

subtype Known_Axis_Order is
  Axis_Order_Type range IDENTITY .. SIXFOLD;
```

(S. R. Hall 1981a; S. R. Hall 1981b)

Code example

Space group symbol decoder

```
-- Rotation matrices from Hall 1981 [1], Table 3:

Principal_Rotations : constant array
  (Known_Axis_Direction, Known_Axis_Order) of Symmetry_Operator :=
  (
   X_AXIS =>
     ( -- axis x (a)
       IDENTITY =>
         (
          (1.0,  0.0,  0.0,  0.0),
          (0.0,  1.0,  0.0,  0.0),
          (0.0,  0.0,  1.0,  0.0),
          (0.0,  0.0,  0.0,  1.0)
         ),
       TWOFOLD =>
         (
          (1.0,  0.0,  0.0,  0.0),
          (0.0, -1.0,  0.0,  0.0),
          (0.0,  0.0, -1.0,  0.0),
          (0.0,  0.0,  0.0,  1.0)
         ),
```

Code example

Space group symbol decoder

```
Get_Hall_Symbol_Inversions (Symbol, Pos, N_Inversions);
Get_Hall_Symbol_Centerings (Symbol, Pos, Centering, N_Centering);

for Axis_Number in 1..4 loop
   Get_Hall_Symbol_Rotation (Symbol, Pos, Symmetry_Operators,
                             N_Symmetry_Operators,
                             Preceeding_Axis_Direction,
                             Preceeding_Axis_Order, Axis_Number);
end loop;

Get_Change_Of_Basis (Symbol, Pos, Change_Of_Basis);

Apply_Change_Of_Basis
  (
   Symmetry_Operators, N_Symmetry_Operators,
   Centering, N_Centering,
   Inversions, N_Inversions,
   Change_Of_Basis
  );

Build_Group (Symmetry_Operators, N_Symmetry_Operators);
```

Code example

The interface

```
with Symmetry_Operations; use Symmetry_Operations;

package Hall_Symbol_Parser is

   Debug_Print_Matrices : Boolean := False;

   function Decode_Hall_Symbol (Symbol : in String)
                                return Symmetry_Operator_Array;

end Hall_Symbol_Parser;
```

Run the compiled program

Just set the PATH to it...

```
saulius@pterodaktilis $ decode_hm 'C121 (x-y, x+y, z)'
X,Y,Z
-Y,-X,-Z

saulius@pterodaktilis $ decode_hm 'C121 (1/2*a - 1/2*b, 1/2*a + 1/2*b, c)'
X,Y,Z
-Y,-X,-Z



saulius@pterodaktilis $ decode_hall 'C 2y (x-y, x+y, z)'
X,Y,Z
-Y,-X,-Z

saulius@pterodaktilis $ decode_hall 'C 2y (1/2*a - 1/2*b, 1/2*a + 1/2*b, c)'
X,Y,Z
-Y,-X,-Z
```

Cool things about Ada

Unit control (GNAT Ada)

https://docs.adacore.com/gnat_ugn-docs/html/gnat_ugn/gnat_ugn/gnat_and_program_execution.html#performing-dimensionality-analysis-in-gnat

```
with Ada.Text_IO;  use Ada.Text_IO;

with System.Dim.MKS;     use System.Dim.Mks;
with System.Dim.Mks_IO;  use System.Dim.Mks_IO;

procedure Angstroem is
   Angstroems : constant Length := 1.0E-10 * m;
   H_R        : Length := 1.1 * Angstroems;
   H_R_cm     : Length := 1.1E-8 * cm;
   One_cm     : Length := 1.0 * cm;
   V          : Volume := (4.0 / 3.0) * Pi * H_R ** 3;
```

Cool things about Ada

Unit control (GNAT Ada)

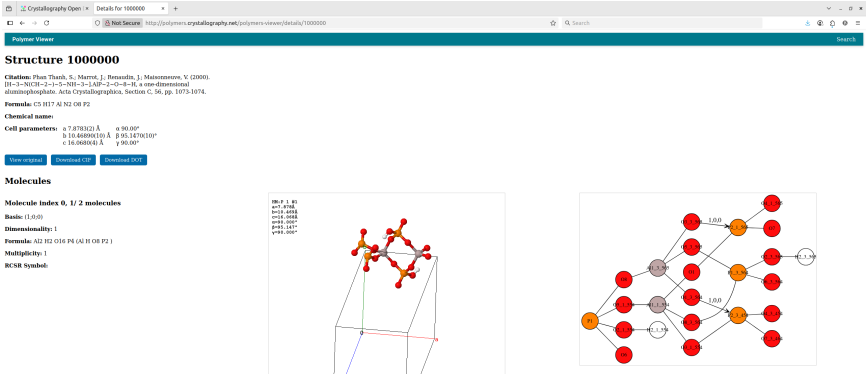
https://docs.adacore.com/gnat_ugn-docs/html/gnat_ugn/gnat_ugn/gnat_and_program_execution.html#performing-dimensionality-analysis-in-gnat

```
begin
   Put ("Hydrogen atom diameter is:");
   New_Line;
   Put (H_R, 0, 1, 2);
   Put ("Atom volume is:");
   New_Line;
   Put (V, 0, 1, 2);
   New_Line;
   -- V := H_R; -- ERROR:
   --
   -- angstroem.adb:32:06: error: dimensions mismatch in assignment
   -- angstroem.adb:32:06: error: left-hand side has dimension [L**3]
   -- angstroem.adb:32:06: error: right-hand side has dimension [L]
end Angstroem;
```

Web applications

AWS (Ada Web Server)

http://polymers.crystallography.net/polymers-viewer



Created by Yaroslav Rozdobudko

Server side implementation

Using AWS

```
with Search_Callbacks;

...
procedure Cif_Explorer is

...
begin

...
   --
   AWS.Services.Dispatchers.URI.Register (Root, "/", Search_Callbacks.Main'Access);
   AWS.Services.Dispatchers.URI.Register (Root, "/index.html", Search_Callbacks.Main'Access);
   AWS.Services.Dispatchers.URI.Register (Root, "/search", Search_Callbacks.Main'Access);

...
end Cif_Explorer;
```

Automatic compilation of proven code

Ada and SPARK

examples/make_group.ads

```
 8    type Ring_Element is mod 37;
```

```
29    function Build_Group (E : Ring_Element) return Group
30      with
31        Post => Is_Group (Build_Group'Result);
```

gnatprove -P main.gpr --report=all make_group.adb

```
make_group.ads:23:14: info: postcondition proved
make_group.ads:27:14: info: postcondition proved
make_group.ads:31:14: info: postcondition proved
group_theory.ads:16:15: info: postcondition proved, in instantiation at make_group.ads:16
```

```
saulius@tasmanijos-velnias spacegroups/ $ ./run_make_group 8
(1, 8, 27, 31, 26, 23, 36, 29, 10, 6, 11, 14)

saulius@tasmanijos-velnias spacegroups/ $ ./run_make_group 7
(1, 7, 12, 10, 33, 9, 26, 34, 16)
```

Ada Alire ecosystem

Canned software packages

Over 500 libraries and packages available (2025-08-19)

```
saulius@starta 2025-Poznan-Computing-in-Crystallography-Forum/ $ alr search OpenGL
warn: Spent 0.07 seconds exploring complete solutions

NAME                STATUS  VERSION     DESCRIPTION
adagl_gtk3                  0.0.1       OpenGL/Gtk3 binding
asfml                 X     2.6.1       Ada binding to SFML, the Simple and Fast Multimedia
globe_3d              U     2023.11.12  GLOBE_3D: GL Object Based Engine for 3D
lace_opengl                 0.1.0       Provides an openGL engine.
openglada                   0.9.0       Thick Ada binding for OpenGL
openglada_glfw              0.9.0       GLFW binding for use with OpenGLAda
openglada_images            0.9.0       Image loading library for OpenGLAda
openglada_text              0.9.0       Text rendering library for OpenGLAda
orka                        1.0.0       OpenGL 4.6 rendering kernel written in Ada 2012
orka_awt                    1.0.0       Ada Window Toolkit
orka_egl                    1.0.0       Ada 2012 bindings for EGL
...
```

Bad things about Ada...

Not all is roses

“A lot of boiler-plate..”
“you must get it right everywhere before you try one thing”
Need to know type compatibility and visibility rules to use the type system efficiently...
Certain usual techniques require low level (unsafe) features...

Conclusions

Hopefully defensible

- Ada code is readable;
- Ada compiler (GNAT, F/LOSS) is here and usable;
- Number of libraries is growing;
- Static typing helps to document code;
- Static typing helps to catch bugs;
- Formal verification is on the horizon;
- …so why not use it?

Plans for the future

Collaboration

- Involve students into long-term projects (give a student a `package`, ask to write a `package body`…);
- Build a memory safe, flexible crystallographic library;
- Build a *validated* crystallographic library;
- Have it available in Alire;
- Have fun with crystallographic computing :)

Acknowledgements

VU LSC IBT (KICIS)

Andrius Merkys
Antanas Vaitkus
Algirdas Grybauskas

VU MIF II (FMG)

Linas Laibinis
Karolis Petrauskas
Irus Grinis
Haroldas Giedra

QM community

Linas Vilčiauskas
Vytautas Žalandauskas
Lukas Razinkovas
Nicola Marzari
Giovanni Pizzi
Lubomir Smrcok
Rickard Armiento

COD Advisory board

Daniel Chateigner
Robert T. Downs
Werner Kaminsky
Armel Le Bail
Luca Lutterotti
Peter Moeck
Peter Murray-Rust
Miguel Quirós

Funding:
Data Center for Machine Learning and Quantum Computing in Natural and Biomedical Sciences, **Research Council of Lithuania Project No.: S-A-UEI-23-11**

Useful links

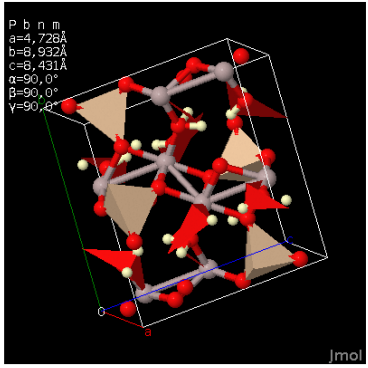
Ada code and Ada books

Code: https://github.com/sauliusg/

Ada books: https://bookauthority.org/books/best-ada-books

Open-access Ada books from AdaCore: https://learn.adacore.com/

Thank you!



Coordinates	2207377.cif
Original IUCr paper	HTML

http://en.wikipedia.org/wiki/Topaz

http://www.crystallography.net/2207377.html

References I

Chang, Geoffrey et al. (Dec. 2006). “Retraction”. In: *Science* 314.5807, pp. 1875–1875. ISSN: 1095-9203. DOI: 10.1126/science.314.5807.1875b.

Gražulis, Saulius et al. (2009). “Crystallography Open Database – an open-access collection of crystal structures”. In: *Journal of Applied Crystallography* 42, pp. 726–729. DOI: 10.1107/S0021889809016690. URL: http://dx.doi.org/10.1107/S0021889809016690.

Hall, Barry G. et al. (July 2007). “Retraction: Measures of clade confidence do not correlate with accuracy of phylogenetic trees.”. In: *PLoS computational biology* 3.7 (7), e158. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.0030158. ppublish.

Hall, S. R. (July 1981a). “Space-group notation with an explicit origin”. In: *Acta Crystallographica Section A* 37.4, pp. 517–525. DOI: 10.1107/s0567739481001228.

— (Nov. 1981b). “Space-group notation with an explicit origin; erratum”. In: *Acta Crystallographica Section A* 37.6, pp. 921–921. DOI: 10.1107/s0567739481001976.

Herzog, Rubén et al. (Sept. 2022). “Retraction note: A mechanistic model of the neural entropy increase elicited by psychedelic drugs”. In: *Scientific Reports* 12.1, p. 15500. ISSN: 2045-2322. DOI: 10.1038/s41598-022-20093-y.

Joppa, Lucas N. et al. (May 2013). “Troubling trends in scientific software use”. In: *Science* 340.6134, pp. 814–815. ISSN: 1095-9203. DOI: 10.1126/science.1231535.

References II

Kleywegt, Gerard J. et al. (2002). “Homo crystallographicus–quo vadis?” In: *Structure (London, England : 1993)* 10.4, pp. 465–472. DOI: 10.1016/S0969-2126(02)00743-8.

Miller, Greg (Dec. 2006). “A scientist’s nightmare: software problem leads to five retractions”. In: *Science* 314.5807, pp. 1856–1857. ISSN: 1095-9203. DOI: 10.1126/science.314.5807.1856.

Ramachandran, G. N. et al. (July 1963). “Stereochemistry of polypeptide chain configurations”. In: *Journal of Molecular Biology* 7.1, pp. 95–99. DOI: 10.1016/s0022-2836(63)80023-6.

Shaffer, P. A. et al. (Nov. 1946). “The use of punched cards in molecular structure determinations I. Crystal structure calculations”. In: *The Journal of Chemical Physics* 14.11, pp. 648–658. DOI: 10.1063/1.1724081. URL: https://authors.library.caltech.edu/45530/1/SCHjcp46a.pdf.

Shmueli, U. et al. (Oct. 2001). “Space-group symbols for numeric and symbolic computations”. In: *International Tables for Crystallography, vol. B*. Ed. by U. Shmueli. Vol. 1. IUCr. Chap. Appendix 1.4.2, pp. 107–119. ISBN: 07-9236-592-5.

Škrbić, Tatjana et al. (Feb. 2021). “Local sequence-structure relationships in proteins”. In: *Protein Science* 30.4, pp. 818–829. ISSN: 1469-896X. DOI: 10.1002/pro.4032.

Soergel, David A. W. (July 2015). “Rampant software errors may undermine scientific results”. In: *F1000Research* 3, p. 303. ISSN: 2046-1402. DOI: 10.12688/f1000research.5930.2.

References III

Zhao, Jun et al. (Oct. 2012). "Why workflows break — Understanding and combating decay in Taverna workflows". In: *2012 IEEE 8th International Conference on E-Science*. IEEE, pp. 1–9. DOI: 10.1109/escience.2012.6404482.

Zwart, Peter H. et al. (Dec. 2007). "Surprises and pitfalls arising from (pseudo)symmetry". In: *Acta Crystallographica Section D Biological Crystallography* 64.1, pp. 99–107. DOI: 10.1107/s090744490705531x.

Spare slides

Just in case someone asks...

Ada CIF parser

Multi language programming

`cod-tools` CIF parser: svn://www.crystallography.net/cod-tools

```
procedure Parse_Cif_From_File_With_Error_Code (Filename : char_array;
                                               CO : Cif_Option_T)
      with Import => True,
      Convention => C,
      External_Name => "parse_cif_from_file_with_error_code";


task Cif_Parser_Task is
   entry Begin_Parsing (Cif_Filename : Unbounded_String;
                        Cif_Options : Cif_Option_T);
end Cif_Parser_Task;
```

Ada popularity

Growing ?!

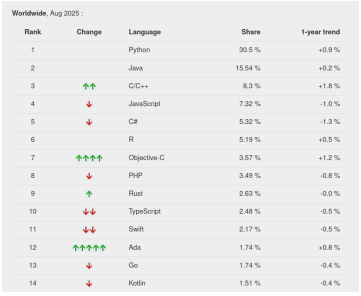
(All URLs accessed 2025-08-21)

“Why is 40-year-old programming language Ada hot again?”
https://www.developer-tech.com/news/why-is-40-year-old-programming-language-ada-hot-again/

“2025: ‘Golden Oldie’ Ada Hits Popularity Milestone”
https://www.techrepublic.com/article/news-tiobe-analysis-july-2025/

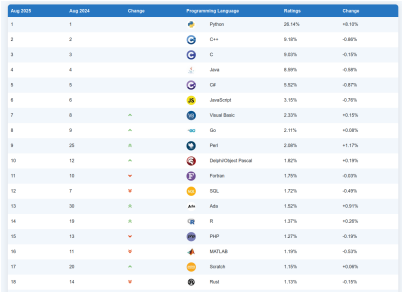
https://pypl.github.io/PYPL.html

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



Worldwide, Aug 2025 :

Rank	Change	Language	Share	1-year trend
1		Python	30.5 %	+0.9 %
2		Java	15.54 %	+0.2 %
3	↑↑	C/C++	8.3 %	+1.8 %
4	↓	JavaScript	7.32 %	-1.0 %
5	↓	C#	5.32 %	-1.3 %
6		R	5.19 %	+0.5 %
7	↑↑↑↑	Objective-C	3.57 %	+1.2 %
8	↓	PHP	3.49 %	-0.8 %
9	↑	Rust	2.63 %	-0.0 %
10	↓↓	TypeScript	2.48 %	-0.5 %
11	↓↓	Swift	2.17 %	-0.5 %
12	↑↑↑↑↑	Ada	1.74 %	+0.8 %
13	↓	Go	1.74 %	-0.4 %
14	↓	Kotlin	1.51 %	-0.4 %



Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.13%
2	2		C++	9.18%	-0.88%
3	3		C	9.03%	-0.15%
4	4		Java	8.89%	-0.68%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	↑	Visual Basic	2.33%	+0.15%
8	9	↑	Go	2.11%	+0.08%
9	25	⇑	Perl	2.08%	+1.17%
10	12	↑	Delphi/Object Pascal	1.82%	+0.19%
11	10	↓	Fortran	1.79%	-0.03%
12	7	⇓	SQL	1.72%	-0.49%
13	38	⇑	Ada	1.52%	+0.91%
14	19	⇑	R	1.37%	+0.28%
15	13	↓	PHP	1.27%	-0.19%
16	11	⇓	MATLAB	1.19%	-0.53%
17	28	↑	Scratch	1.15%	+0.06%
18	14	⇓	Rust	1.13%	-0.15%

Example: dihedral angle calculation

The assignment

```
Write a Perl*) program that calculates torsion angles for DNA and RNA
macromolecules. Compute the following angles: alpha, beta.

*) Note: Ada programs are acceptable; in that case all native Perl
   features mentioned in this specification SHOULD be replaced by the
   corresponding Ada features.

NB: for this particular assignment, you MAY NOT use external libraries
to read PDB files or calculate dihedral angles; these functions MUST
be implemented in the program.
```

NOTE: Students could chose a set of atoms for their individual assignment

Full assignment text: https://tinyurl.com/yp24aad4

Example: dihedral angle calculation

The vector algebra code

```
type Vector_3D is array (1 .. 3) of Long_Float;

function "-" (V, W : Vector_3D) return Vector_3D is
begin
   return (V(1) - W(1),  V(2) - W(2), V(3) - W(3));
end;
```

Example given to the students

```
type Vector_3D is record
   X, Y, Z : Float;
end record;

function "-" (V1, V2 : Vector_3D) return Vector_3D is
begin
    return (V1.X - V2.X, V1.Y - V2.Y, V1.Z - V2.Z);
end;
```

Version designed by the students (Martynas Mažuolis)

Example: dihedral angle calculation

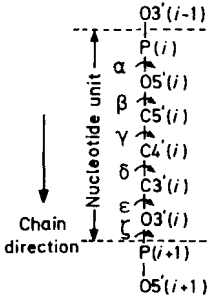
Angle definition and calculation

```
Function Dihedral_angle (Atoms: Chemical_Atom_Set) return float is
        Vector1 : Vector_3D := to_vector(Atoms.Atom_Array(1)) - to_vec..
        Vector2 : Vector_3D := to_vector(Atoms.Atom_Array(2)) - to_vec..
        Vector3 : Vector_3D := to_vector(Atoms.Atom_Array(3)) - to_vec..
        Angle1 : Bond_Angle := Angle(Vector1, Vector2);
        Angle2 : Bond_Angle := Angle(Vector2, Vector3);
        Normal1 : Vector_3D := Cross(Vector1, Vector2);
        Normal2 : Vector_3D := Cross(Vector2, Vector3);
        NX: Vector_3D := Cross(Normal1, Normal2);
        Sign : float := (if Vector2*NX >= 0.0 then 1.0 else -1.0);
        Dihedral : Float := Angle(Normal1, Normal2)*Sign;

        begin
        pragma Assert(Atoms.N>=4);
    return Dihedral;

        end;
```

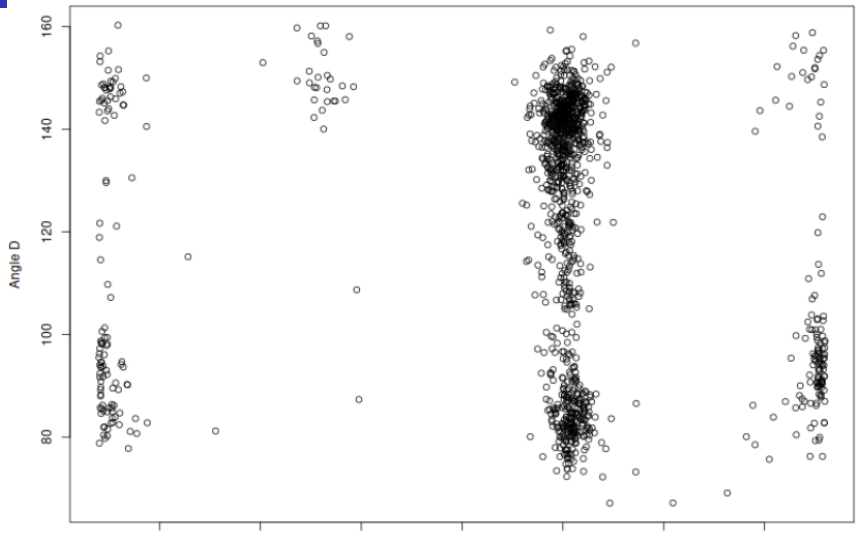
Version designed by the students (Viktorija Jugai)

Example: dihedral angle calculation

Results on the whole PDB



(IUPAC-IUB, **IIJCBN1983**)



Result computed by Aušrinė Zeleckytė