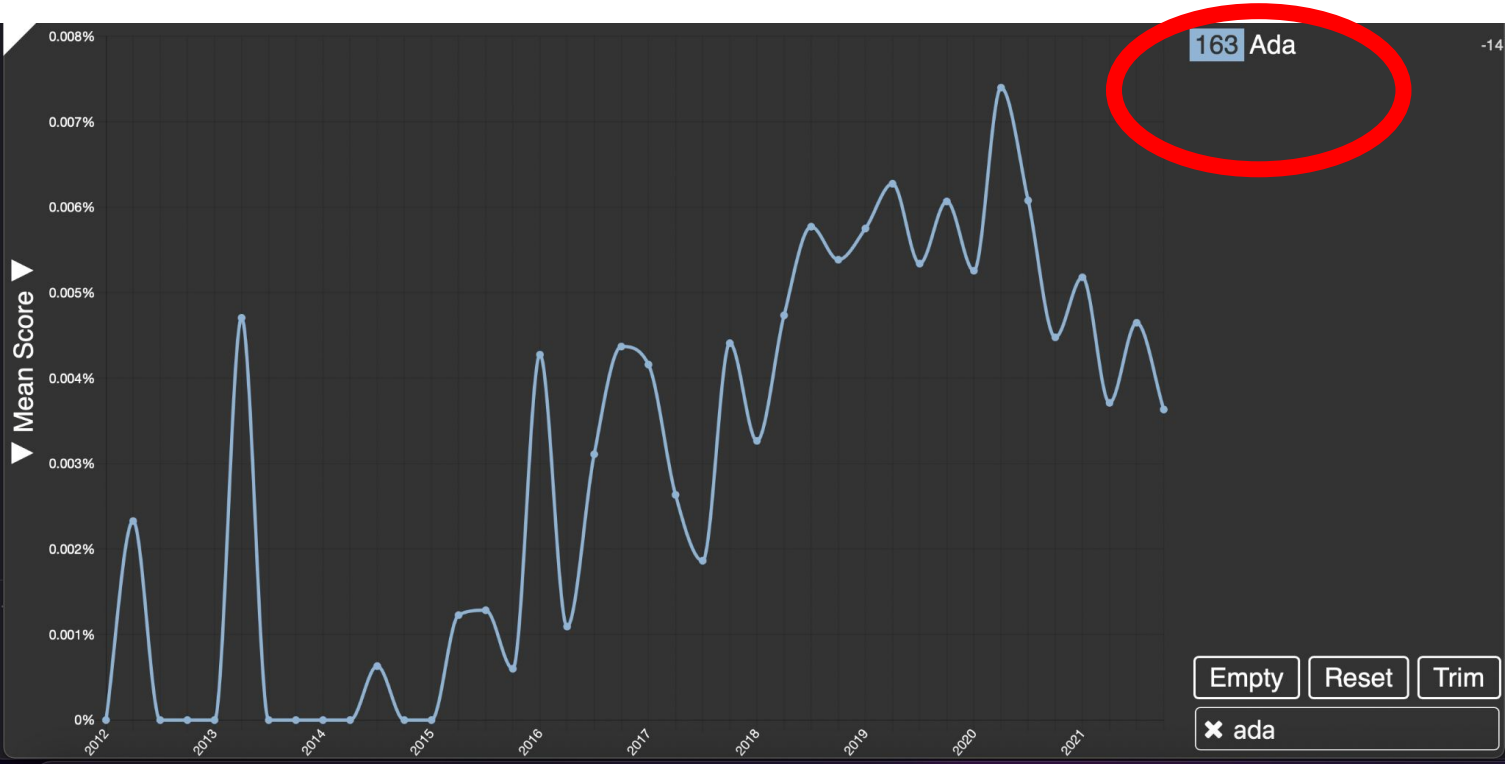
The Outsider's Guide to Ada

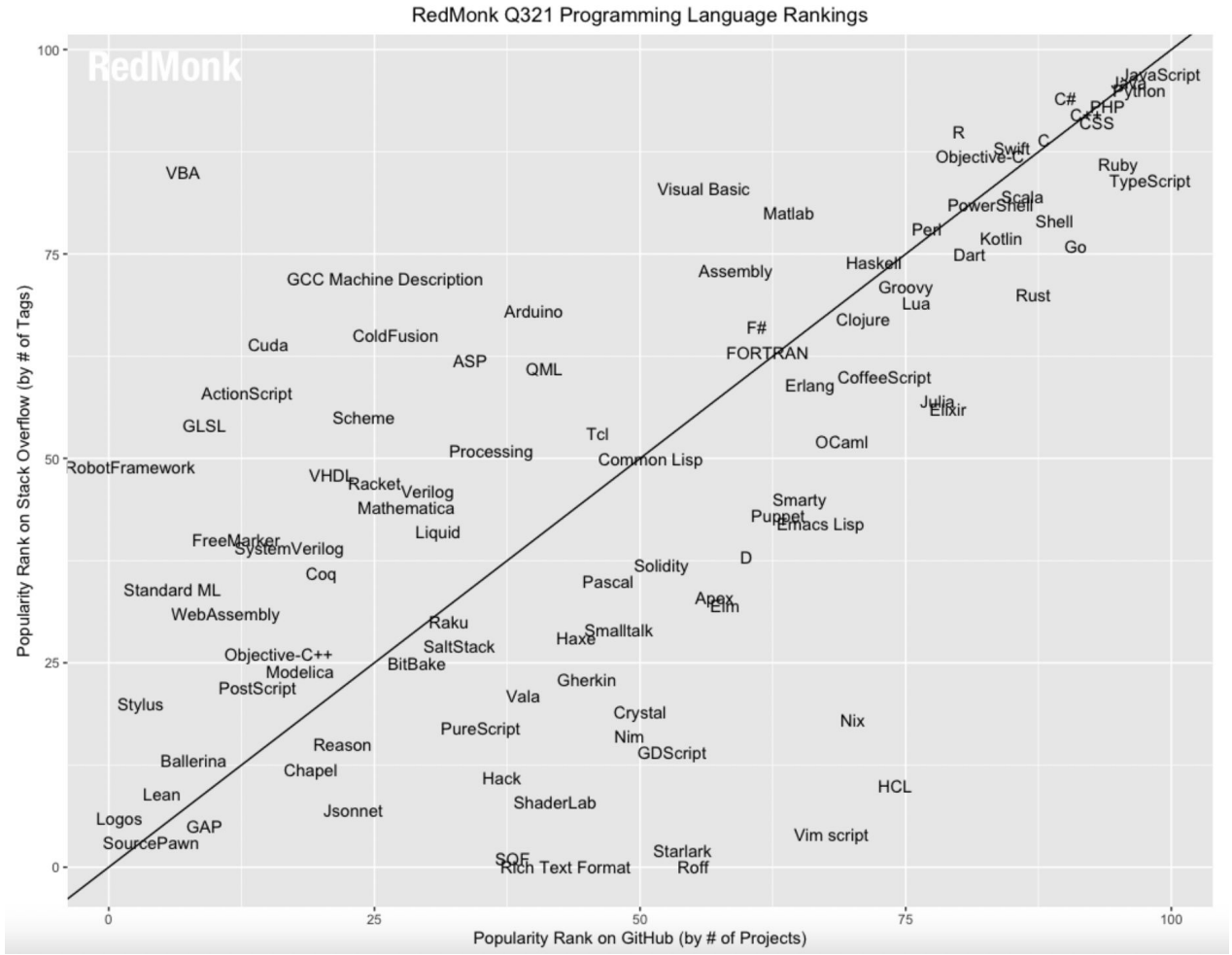
FOSDEM 2022
Ada Devroom

Ada in 2022

- Still in use (Ada 83, Ada 95, Ada 2005, Ada 2012, Ada 202x)
  - :=, =, /=, **begin**, **end**
- Free to use (FSF GNAT, part of GCC)
  - Code doesn’t have to be GPL
- Formal verification with SPARK subset
  - Mixes with Ada code
- Package manager (Alire)
  - Can do GNAT toolchain install
  - Wraps build/run commands
  - Contains both Ada and SPARK crates



https://tjpalmer.github.io/languish/



https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/

Ada not listed!

SYNOPSYS® | Black Duck Open Hub

Follow @ OH | Sign In | Join Now

Projects | People | Organizations | Tools | Blog | Projects ▾ | Search...

Language Statistics

Ada

Earliest usage tracked by Open Hub: October 2001

Total Lines
43,195,123

Code
24,593,246

Comments
10,886,795 (30.7%)

Projects 784

Contributors 1,601

Commits 143,334

SYNOPSYS® | Black Duck Open Hub

Follow @ OH | Sign In | Join Now

Projects | People | Organizations | Tools | Blog | Projects ▾ | Search...

Language Statistics

Rust

Earliest usage tracked by Open Hub: March 2002

Total Lines
32,289,029

Code
24,852,859

Comments
4,477,052 (15.3%)

Projects 1,162

Contributors 17,808

Commits 951,305

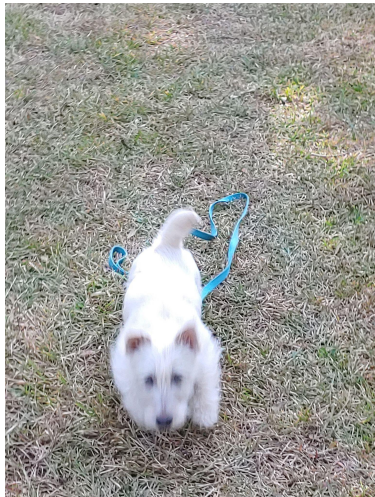
https://www.openhub.net

PYPL PopularitY of Programming Language

17	↑↑↑	Ada	0.75 %	+0.1 %

http://pypl.github.io

Cardano?



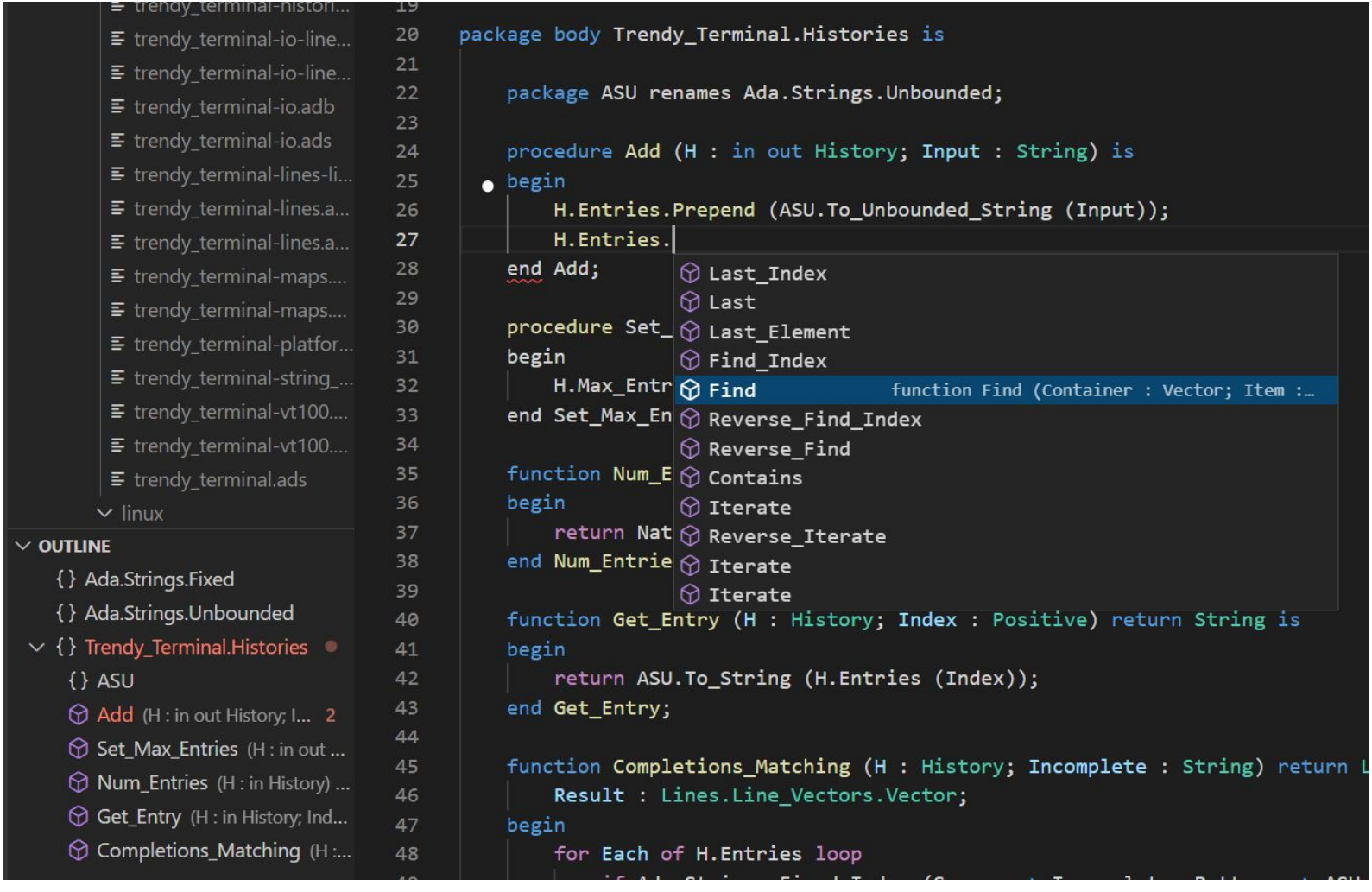
How I Use Ada

- Alire
  - Toolchain install
  - `alr build`
  - `alr edit`
  - `alr run`
- FSF GNAT
- GNAT Studio
- Visual Studio Code
  - Language Support for Ada (plugin)

```
{
  "folders": [
  {
    "path": "."
  }
  ],
  "settings": {
    "ada.projectFile": "septum.gpr"
  }
}
```

workspace.code-workspace

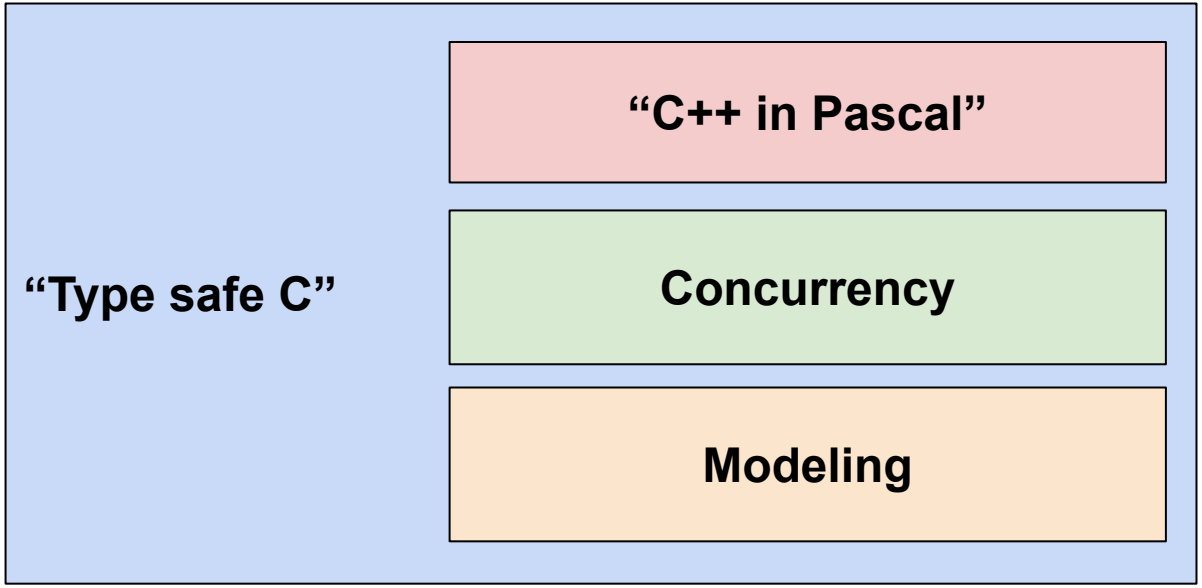
```
alr config --set editor.cmd “code workspace.code-workspace”
```


trendy_terminal-io-line...
trendy_terminal-io-line...
trendy_terminal-io.adb
trendy_terminal-io.ads
trendy_terminal-lines-li...
trendy_terminal-lines.a...
trendy_terminal-lines.a...
trendy_terminal-maps....
trendy_terminal-maps....
trendy_terminal-platfor...
trendy_terminal-string_...
trendy_terminal-vt100....
trendy_terminal-vt100....
trendy_terminal.ads
linux
OUTLINE
{} Ada.Strings.Fixed
{} Ada.Strings.Unbounded
{} Trendy_Terminal.Histories
{} ASU
Add (H : in out History; I... 2
Set_Max_Entries (H : in out ...
Num_Entries (H : in History) ...
Get_Entry (H : in History; Ind...
Completions_Matching (H :...
20 package body Trendy_Terminal.Histories is
22 package ASU renames Ada.Strings.Unbounded;
24 procedure Add (H : in out History; Input : String) is
25 begin
26 H.Entries.Prepend (ASU.To_Unbounded_String (Input));
27 H.Entries.
28 end Add;
30 procedure Set_
31 begin
32 H.Max_Entr
33 end Set_Max_En
35 function Num_E
36 begin
37 return Nat
38 end Num_Entrie
40 function Get_Entry (H : History; Index : Positive) return String is
41 begin
42 return ASU.To_String (H.Entries (Index));
43 end Get_Entry;
45 function Completions_Matching (H : History; Incomplete : String) return L
46 Result : Lines.Line_Vectors.Vector;
47 begin
48 for Each of H.Entries loop
Last_Index
Last
Last_Element
Find_Index
Find function Find (Container : Vector; Item :...
Reverse_Find_Index
Reverse_Find
Contains
Iterate
Reverse_Iterate
Iterate
Iterate

Decoding Ada

Ada Term	Related, might not be an equivalent term
package	namespace, module
subprogram, procedure, function	function
record	struct
aggregate	designated initializer
access type, accessibility	pointer, lifetime reference
natural, positive, modular type	unsigned integer
limited type	immovable, uncopyable
tagged type	class
type extension	subclass
class-wide type	base class reference
dynamic dispatch ‘Class, redispatching	polymorphism, virtual
controlled type	RAII, constructor, destructor
subprogram, primitive operation	method, member function
view conversion qualification	cast
task	thread
protected object	monitor
entry, guard, barrier rendezvous	semaphore, lock, async, await, promise, future
generic	template
storage pool	allocator
elaboration	initialization

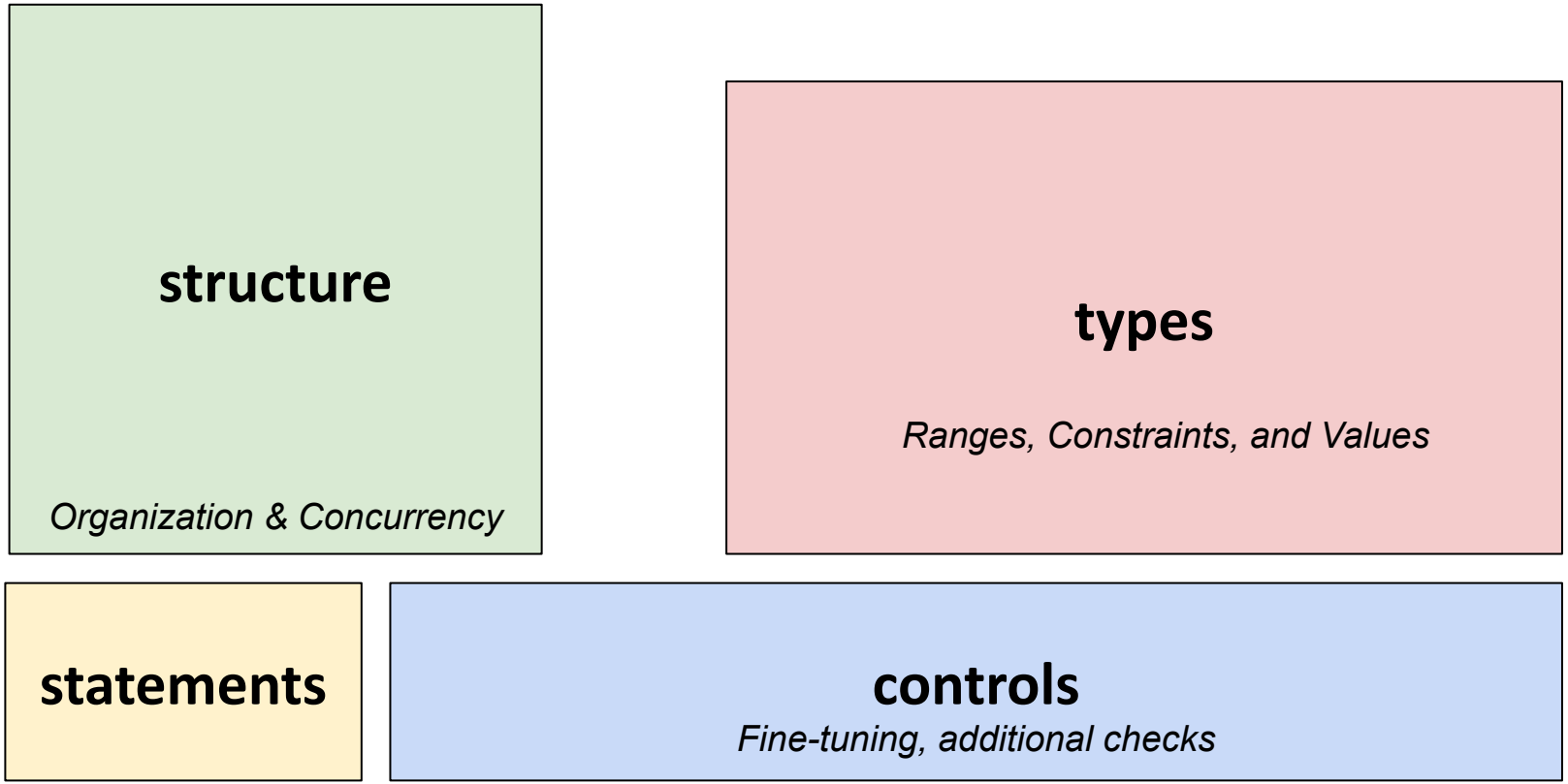
Ada, opt-in features, the *a la carte* language



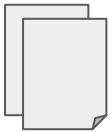
Missing: Variadic templates, move semantics, macros, robust compile-time computation

Choose and use only the features you need

High Level Syntax Overview



Physical Structure

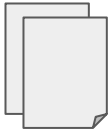


Specifications (.ads)
"What is available?"

```
-- p.ads
-- nothing needed here

package P is
   -- Declare things to exist
   procedure Hello_World;
end P;
```

Context Clause



Bodies (.adb)
"What to do?"

```
-- p.adb
with Ada.Text_IO;

package body P is
   procedure Hello_World is
   begin
      Put_Line ("Hello, World!");
   end Hello_World;
begin
   -- Pre-Main setup
end P;
```

```
-- main.adb
with P;

procedure Hello_World_Main is
   -- Nothing to declare
begin
   P.Hello_World;
end Hello_World_Main;
```

Structural Self-Similarity

Declare anything, anywhere

```
function Foo(B : Bar)
    return Value
is
    Declare things
begin
    Do these steps
end Foo;
```

```
procedure Foo(B : Bar)
is
    Declare things
begin
    Do these steps
end Foo;
```

```
declare
    Declare things
begin
    Do these steps
end;
```

```
task type T is
   Declare things
end P;

task body T is
   Declare things
begin
   Do this in parallel
end T;
```

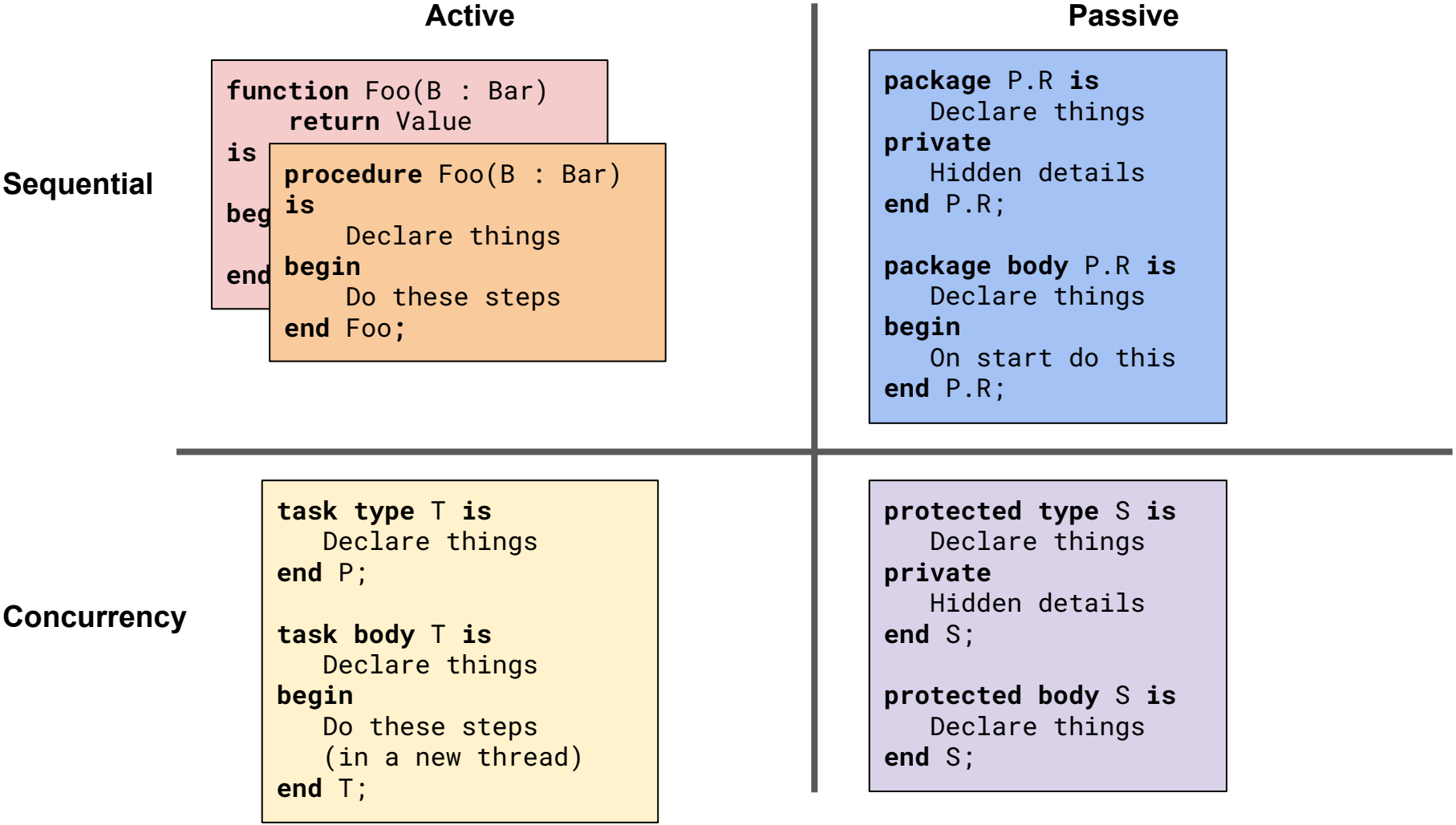
Types are not modules!

```
package P is
   Declare things
private
   Hidden details
end P;

package body P is
   Declare things
begin
   On start do this
end P;
```

```
protected type R is
   Declare things
private
   Hidden details
end R;

protected body P is
   Declare things
end R;
```


Active
Passive
Sequential
Concurrency
function Foo(B : Bar)
return Value
is
beg
end
procedure Foo(B : Bar)
is
Declare things
begin
Do these steps
end Foo;
package P.R is
Declare things
private
Hidden details
end P.R;
package body P.R is
Declare things
begin
On start do this
end P.R;
task type T is
Declare things
end P;
task body T is
Declare things
begin
Do these steps
(in a new thread)
end T;
protected type S is
Declare things
private
Hidden details
end S;
protected body S is
Declare things
end S;

Types

```
type Percentage is range 0 .. 100;
```

```
type Reference_Count is new Integer;
```

```
type Cursor_Position is record
    Row : Integer;
    Col : Integer;
end record;
```

struct

```
type cc_array is array (Natural range 0 .. NCCS - 1) of cc_t;
```

uncopyable opaque type

```
type Search is limited private;
```

```
type Operation is limited interface;
```

enum

```
type Filter_Action is (Keep, Exclude);
```

Abstract base class

```
type Filter is abstract tagged record
    Action: Filter_Action;
end record;
```

Subclass

```
type Case_Sensitive_Match_Filter is new Filter with record
   Text : Ada.Strings.Unbounded.Unbounded_String;
end record;
```

Access Types

```
type Apple is access Foo;
type Orange is access Foo;
type Any_Foo is access all Foo;

for Apple'Storage_Pool use Special_Pool;
A : Apple := new (Sub_Pool) Apple'(...);
```

```
type IOStream is record
    File            : Linux.FILE_Ptr;
    File_Descriptor : Linux.FD;
    Settings        : aliased Linux.Termios;
end record;

function Load_Terminal (
   File_Descriptor : Linux.FD;
   Terminal        : not null access Linux.Termios)
return Boolean;
```

Overloading

```
with Ada.Strings.Unbounded;
with Trendy_Terminal.Platform;

package Trendy_Terminal.IO is
    package ASU renames Ada.Strings.Unbounded;

    procedure Put (C : Character) renames Trendy_Terminal.Platform.Put;
    procedure Put (S : String)    renames Trendy_Terminal.Platform.Put;
    procedure Put (S : ASU.Unbounded_String);

    procedure Put_Line (S : String);
    procedure Put_Line (S : ASU.Unbounded_String);

end Trendy_Terminal.IO;
```

A Function Taking a Parameter

```
public class Foo // Java
{
  public static Value strawberry(Foo f) {}
  public        Value dewberry  () {}
  private       Value plum      () {}
}
```

```
impl Foo // Rust
{
    fn cherry    (&self)     -> Value {}
    fn dewberry  (&mut self) -> Value {}
    fn elderberry(self)      -> Value {}
}

fn fig     (f : &mut Foo) -> Value {}
fn grape   (f : &Foo)     -> Value {}
fn honeydew(f : Foo)      -> Value {}
```

```
class Foo { // C++
public:
  Value dewberry  ();
  Value cherry    () const;

  static Value orange    (const Foo*);
  static Value pineapple (Foo*);
  static Value quince    (Foo&&);
  static Value raspberry (Foo);
  static Value strawberry(Foo&);
  static Value tomato(const Foo&);

private:
    Value plum();
};

Value fig     (Foo&);
Value grape   (const Foo&);
Value honeydew(Foo);
Value kiwi    (Foo&&);
Value lemon   (Foo*);
```

Not a complete list of alternatives

A Function Accepting a Parameter

```
function Cherry    (Self : in     Foo) return Value;
function Dewberry  (Self :    out Foo) return Value;
function Olive     (Self : in out Foo) return Value;
function Pineapple (Self : access Foo) return Value;
function Strawberry(Self : not null access Foo) return Value;
```

Any type in class hierarchy

```
function Vanilla    (Self : in     Foo'Class) return Value;
function Watermelon (Self : in out Foo'Class) return Value;
```

Operator overload

```
function "+"(Left, Right : Foo) return Value;
```

Allow access to parameter

```
procedure Apricot(Value : aliased in Foo) return Foo;
```

Subprograms (functions and procedures)

```
type Line is private;
```

Is this a constructor or just a function?

```
function Make (S : String) return Line;
```

Are these just functions or const member functions?

```
function Length           (Self : in Line) return Natural;
function Get_Cursor_Index (Self : in Line) return Positive;
```

Does it matter if this is a member function (method)?

```
procedure Set_Cursor_Index (Self : in out Line; Cursor_Index : Positive);
```

```
declare
  L : Line := Make("Line");
begin
  Set_Cursor_Index (L, 4);

  -- tagged
  L.Set_Cursor_Index (4);
end
```

Controlled Types (RAII)

```
-- Atomic reference counting pointer.
type Arc is new Ada.Finalization.Controlled with private;

overriding
procedure Initialize (Self : in out Arc);

overriding
procedure Adjust     (Self : in out Arc);

overriding
procedure Finalize   (Self : in out Arc);
```

```
-- RAII-based type to restore environment on exit.
type Environment is tagged limited private;

type Environment is new Ada.Finalization.Limited_Controlled with record
    Initialized : Boolean := False;
end record;

overriding procedure Initialize (Self : in out Environment);
overriding procedure Finalize   (Self : in out Environment);
```

Attributes

```
Trendy_Terminal.IO.Line_Editors.Get_Line (
   Format_Fn     => Format_Input'Access,
   Completion_Fn => Complete_Input'Access,
   Line_History  => Line_History'Unchecked_Access
);
```

```
function Matches_File (
   F     : Filter'Class;
   Lines : String_Vectors.Vector) return Boolean;
```

```
Value := Positive'Value (Str);
```

- ‘Size instead of sizeof
- ‘Alignment instead of alignof
- ‘Access or ‘Address instead of &
- ‘Class, ‘Tag
- ‘Length for array size
- ‘Image for string conversion
- ‘Value for parsing from string
- Discrete types
  - ‘Image, ‘Value
  - ‘Pred, ‘Succ
  - ‘First, ‘Last
  - ‘Range
- Enums
  - ‘Val <-> ‘Pos

```
for I in All_Searches'Range loop
   System.Multiprocessors.Dispatching_Domains.Set_CPU (I, All_Searches (I)'Identity);
   All_Searches (I).Start;
end loop;
```

Aspects

```
type Line is private
   with Type_Invariant =>
      Get_Cursor_Index (Line) in 1 .. Length (Line) + 1;
```

```
type Local_Flags is array (c_lflag_t) of Boolean
   with Pack, Size => 32;
```

Aspects don't interfere with the grammar of the type or structural element.

```
procedure Insert (Self : in out Line; S : String)
   with Inline,
        Pre  => S'Length >= 0,
        Post => Num_Cursor_Positions (Self'Old) + S'Length = Length(Self)
                and then Get_Cursor_Index (Self'Old) + Num_Cursor_Positions (S)
                         = Get_Cursor_Index(Self);
```

```
function tcgetattr (File_Descriptor : FD;
                    Terminal : System.Address)
   return BOOL
   with Import     => True,
        Convention => C;
```

Generics

```
generic
   type Key_Type is private;
   type Element_Type is private;
   with function "<" (Left, Right : Key_Type) return Boolean is <>;
   with function "=" (Left, Right : Element_Type) return Boolean is <>;
package Ada.Containers.Ordered_Maps is
   function Equivalent_Keys (Left, Right : Key_Type) return Boolean;
-- ...
```

```
package File_Maps is new Ada.Containers.Ordered_Maps (
   Key_Type     => Ada.Strings.Unbounded.Unbounded_String,
   Element_Type => String_Vectors.Vector,
   "<"          => Ada.Strings.Unbounded."<",
   "="          => String_Vectors."=");

-- ... Later usage ...

Contents : File_Maps.Map;
```

Structure

- **generic**
  - **procedure**
  - **function**
  - **package**
- “threads” — **task**
- Shared concurrent resources — **protected**

Types

- **type**
- **subtype**
- **record**
- **limited** — “uncopyable”
- **tagged** — “classes”
- *numeric*
- *enum*
- **access**
- **array**
- **exception**

Controls

pragma	*attributes*	*aspect*
pragma Identifier	‘Attribute	**with** Aspect =>

Strings

```
type String is array (Positive range <>) of Character with Pack;
```

```
overriding function Image (F : Regex_Filter) return String is
begin
   return "Regex " & Ada.Strings.Unbounded.To_String (F.Source);
end Image;
```

```
-- Example from Alire project
type String_Access is access constant String;
Replacings : constant array (Character) of String_Access :=
             ('&'    => new String'("&amp;"),
              '''    => new String'("&apos;"),
              '"'    => new String'("&quot;"),
              '<'    => new String'("&lt;"),
              '>'    => new String'("&gt;"),
              others => new String'(""));
```

```
package Command_Maps is new Ada.Containers.Ordered_Maps(
   Key_Type     => Unbounded_String,
   Element_Type => Executable_Command);
```

Tasking

Block and wait for entry call

Stop task if the thing holding this task is waiting to leave scope

Block and wait for a file to appear on queue

Timeout after one second

```
task body File_Loader_Task is
    Elem : Ada.Strings.Unbounded.Unbounded_String;
begin
    loop
        select
            accept Wake;
        or
            terminate;
        end select;

        loop
            select
                File_Queue.Dequeue (Elem);
            or
                delay 1.0;
                exit;
            end select;

            if Is_Text (To_String (Elem)) then
                Cache_File (File_Cache, Elem);
            end if;
            Progress.Finish_Work (1);
        end loop;
    end loop;
end File_Loader_Task;
```

```
task type File_Loader_Task is
   entry Wake;
end File_Loader_Task;
```

Low Level Controls

Importing C functions

```
function tcgetattr (
   File_Descriptor : FD;
   Terminal : System.Address)
return BOOL
   with Import     => True,
        Convention => C;
```

Call assembly

```
procedure Breakpoint is
begin
   System.Machine_Code.Asm("int3", Volatile => True);
end Breakpoint;
```

Control binary layouts

```
type Bitmap_File_Header is record
    Identifier      : Integer_16;
    File_Size_Bytes : Integer_32;
    Reserved1       : Integer_16;
    Reserved2       : Integer_16;
    Offset          : Integer_32;
end record with Size => Byte * 14;

for Bitmap_File_Header use record
    Identifier      at  0 range 0 .. 15;
    File_Size_Bytes at  2 range 0 .. 31;
    Reserved        at  6 range 0 .. 15;
    Reserved2       at  8 range 0 .. 15;
    Offset          at 10 range 0 .. 31;
end record;
```

Intrinsics

```
-- Example from Atomic project
function Intrinsic_Sync_Sub_And_Fetch
     (Ptr   : access Atomic_Counter;
      Value : Atomic_Counter) return Atomic_Counter;
pragma Import (Intrinsic,
   Intrinsic_Sync_Sub_And_Fetch,
   External_Name => "__sync_sub_and_fetch_4");
```

```
type Local_Flags is array (c_lflag_t)
   of Boolean
with Pack, Size => 32;
```

Flags stored in a 32-bit integer

Ada, opt-in features, the *a la carte* language

“Type safe C”

- Procedures
- Functions
- Packages
- Integers
- Modular Types
- Floats
- Enums
- Arrays
- Strings
- Typed Pointers (access types)
- Function pointers
- Interface w/ C
- Intrinsics

“C++ in Pascal”

- Tagged types (classes)
  - Inheritance
    - Controlled types (RAII)
- Generics
  - Generic Packages
  - Generic Functions
- Operator overloading

Missing: Variadic templates, move semantics, macros, robust compile-time computation

Focused Concurrency

- Tasks
  - Protected Objects

Modeling

- Derived (semantic) Types
  - Ranges
- Design by Contract
  - Pre/Post conditions
  - Invariants
  - Subtype Predicates
- SPARK

Choose and use only the features you need

Ada : Readable, Correct, Intent

- That **package** must be provided **with** this compilation unit.
- These **type**s and subprograms are related, lets **package** them together.
- The details of this **type** are **private**.
- This **type** should be **tagged** for dynamic dispatching.
- This **type** should be **limited** in its ability to copy itself.
- This **type** is a description of an **interface**.
- This **type** is a description of **synchronized** behavior.
- This **type** describes values **protected** from simultaneous writes.
- This integer **type** should only have values in this **range**.
- This **package renames** another **package**.
- This **package** instantiates a **new** version of a **generic package**.
- I want to **use** the contents of this **package** here.
- Do this **task** separately.
- The index into this **array** are these enumeration values.
- This subprogram produces a value, so it's a **function**.
- This subprogram doesn't produce a value, so it's just a **procedure**.
- This parameter is an **in**put.
- This parameter is an **ou**tput.
- This parameter is both an **in**put and an **ou**tput.
- This parameter might be **aliased** by another name.
- This variable might be **aliased** by another name.
- I want to be able to **access** this type indirectly.
- I want to be able to **access all** forms of this type indirectly.