Ada Programming Language

Gianlucca Assunção Leoncini

Luiza Torello Vieira

Victor Azadinho Miranda

História da Linguagem

História da Linguagem

- Desenvolvida pelo Dr. Jean Ichbiah, na França.
- Departamento de Defesa dos Estados Unidos usava cerca de 450 linguagens de programação diferentes na década de 1970.
- Ada foi a vencedora dentre 17 linguagens analisadas.
- Oficialmente publicada em 1983, conhecida como Ada 83.

História da Linguagem

- O nome foi uma homenagem à Condessa de Lovelace, Ada Augusta Byron King.
- Matemática que desenvolveu o primeiro algoritmo legível por um computador.

História da Linguagem

- Linguagem orientada a objetos.
- Uma das primeiras a implementar esse paradigma.
- Não estavam bem definidos os pilares da programação orientada a objetos.

Usos de Ada

Usos de Ada

- Altamente precisa e confiável.
- Extremamente forte tipagem, verificação de tipos e verificação minuciosa de erros na compilação.
- Usada em sistemas críticos e de tempo real.
- Linguagem de nicho.

Orientação a Objetos em Ada

Orientação a Objetos em Ada

- Não possuía total suporte ao paradigma orientado a objetos.
- Com a versão Ada 95, que foram incorporadas:
  - Polimorfismo.
  - Extensões de tipos.
- Não obriga seus programas a serem orientados a objeto.
- Sem penalidade de tempo de execução.

Orientação a Objetos em Ada

Tipos Derivados

- Fazendo parte de Ada desde o começo, os tipos derivados permitem o uso da herança.

Orientação a Objetos em Ada

Tipos Derivados

O exemplo ao lado cria um tipo T que contém informações e comportamento baseados em um subprograma.

```
package P is
  type T is private;
  function Create (Data: Boolean) return T;
  procedure Work (Object : in out T);
  procedure Work (Pointer: access T);
  type Acc_T is access T;
  procedure Proc (Pointer: Acc_T);
private
  type T is record
    Data: Boolean;
  end record;
end P;
```

Orientação a Objetos em Ada

Tipos Derivados

```
with P;
package Q is
  type Derived is new P.T;
end Q;
```

O tipo Q.Derived possui as mesmas informações e comportamento que P.T; ele herda tanto dados como subprogramas.

Orientação a Objetos em Ada

Extensões de Tipos

- Adicionados em Ada 95, os tipos tagged permitem o uso de polimorfismo.
- Polimorfismo usado em Ada é do tipo de sobrescrita.

Orientação a Objetos em Ada

Classes e Objetos

- Tanto o package quanto o record devem ser nomeados.
- Em Ada usa-se três tipos de nomeação:
  - Um nome no plural para o package e sua forma no singular para o record.
  - Um nome para o package e Object para o record.
- E um nome para o package usando Type após o nome(Class Type) para record.

Orientação a Objetos em Ada

Classes e Objetos

A criação de objetos em Ada é feita em tempo de execução e é de um dado tipo.

```
package Person is
   type Object is tagged private;
   procedure Put (O : Object);
private
   type Object is tagged
     record
         Name   : String (1 .. 10);
         Gender : Gender_Type;
     end record;
end Person;
```

Orientação a Objetos em Ada

Classes e Objetos

A criação pode ser feita por declaração, dando dados pré-definidos(aggregate) ou por chamada de função.

```
declare
   P: Person;
begin
   Text_IO.Put_Line("The name is " & P.Name);
end;
```

```
declare
   P: Person := (Name => "Scorsese", Gender => Male);
begin
   Text_IO.Put_Line("The name is " & P.Name);
end;
```

Orientação a Objetos em Ada

Novidades de Ada 2005

Algumas funcionaidades foram adicionadas, como indicadores de sobrescrita (overriding e not overriding).

```
package Persons is

   type Person is tagged private;

   function Make (Name: String; Sex: Gender_Type) return Person;

   function Name (P: Person) return String;
```

```
package X is

   type Object is abstract tagged …;

   procedure One_Class_Member      (This : in     Object);
   procedure Another_Class_Member  (This : in out Object);
   function  Abstract_Class_Member return Object  is abstract;

end X;
```

Legibilidade

Legibilidade

- Ada foi criada para favorecer mais ao leitor que ao escritor.
- Escrito apenas uma vez para ser lido muitas.
- Facilitando assim, a manutenibilidade da linguagem.
- Bane todo o tipo de construção ambígua.

Legibilidade

Sua sintaxe é simples, contendo um total de 73 comandos básicos e 21 operadores.

Ada Keywords			
abort	else	new	return
abs	elsif	not	reverse
abstract (Ada 95)	end	null	
accept	entry		select
access	exception	of	separate
aliased (Ada 95)	exit	or	some (Ada 2012)
all		others	subtype
and	for	out	synchronized (Ada 2005)
array	function	overriding (Ada 2005)	
at			tagged (Ada 95)
	generic	package	task
begin	goto	pragma	terminate
body		private	then
	if	procedure	type
case	in	protected (Ada 95)	
constant	interface (Ada 2005)		until (Ada 95)
	is	raise	use
declare		range	
delay	limited	record	when
delta	loop	rem	while
digits		renames	with
do	mod	requeue (Ada 95)	xor

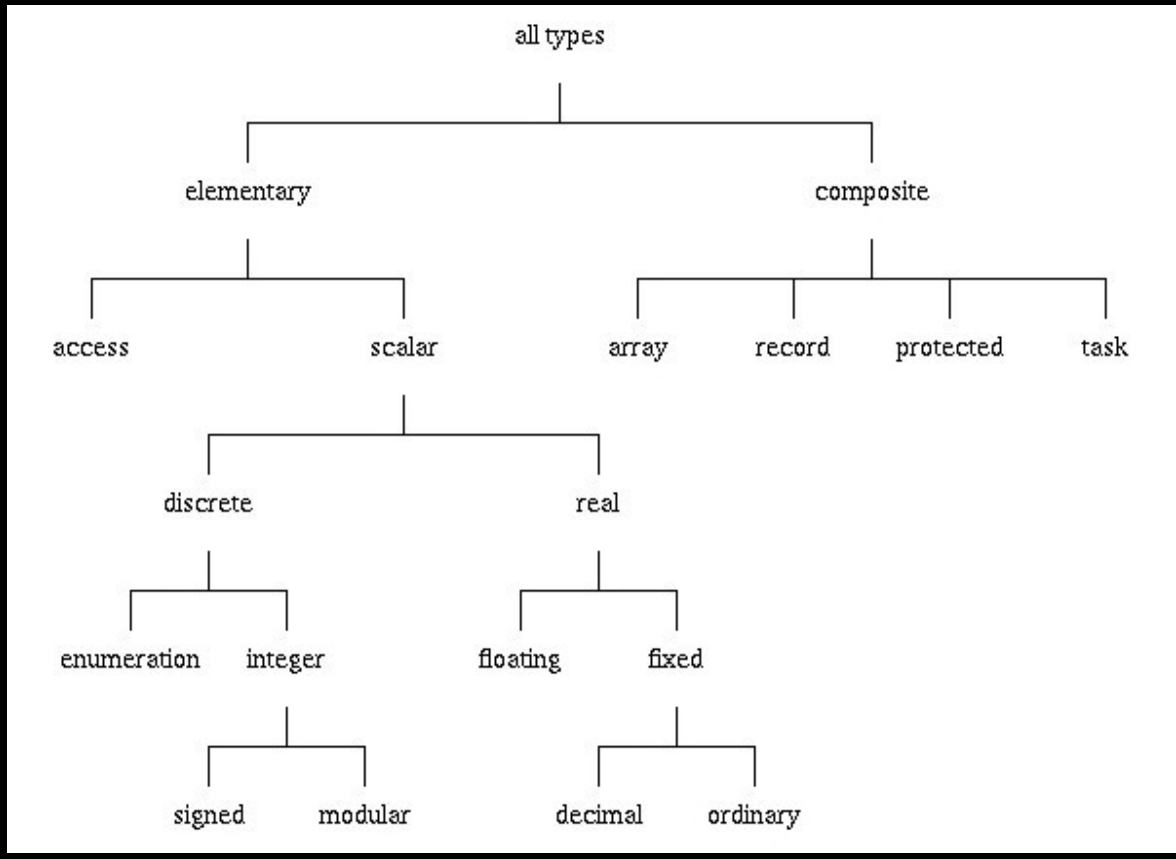
Tipos de Operadores					
Lógicos	Relacionais	Adição Unária	Adição Binária	Multiplicação	Maior Precedência
and	/=	+	+	*	**
or	=	-	-	/	not
xor	<		&	mod	abs
	<=			rem	
	>				
	>=				

Legibilidade

Existem também 11 tipos de dados diferentes, os quais estão esquematizados na imagem abaixo.

Pode-se perceber também a ortogonalidade de Ada.

Tal característica contribui para a melhor legibilidade da linguagem.



Facilidade de Escrita

Facilidade de Escrita

Suporte a Abstração

Keyword overriding.

Essa keyword é opcional.

Abstrair tipos de dados.

```
subtype Day_Subtype is Integer range 1 .. 31;
type Month_Type is (Jan,Feb,Mar,Apr,May,Jun,
   Jul,Aug,Sep,Oct,Nov,Dec);
type Day_Of_Week_Type is (Sun,Mon,Tue,Wed,Thu,Fri,Sat);
type Date is tagged
  record
    Day   : Day_Subtype;
    Month : Month_Type;
    Year  : Integer;
  end record;
type Complete_Date is new Date with
  record
    Day_Of_Week : Day_Of_Week_Type;
  end record;
```

Facilidade de Escrita

- Sintaxe bem próxima à linguagem humana.
- Pode-se escrever código com poucos comandos.

Expressividade

- Reduz o esforço necessário para se efetuar tarefas simples.

Facilidade de Escrita

Expressividade

Em ambos os casos o resultado é o mesmo.

Ada possui uma sintaxe mais amigável e expressiva.

```
for (variable declarations; condition; iteration statement) {
   statement_1
   statement_2
   ...
   statement_n
}
```

```
for variable in range loop
   statements
end loop;
```

Confiabilidade

Confiabilidade

- Verificação de tipos.
- Tratamento de excessões.
- Não pode apresentar vazamentos de memória, corrupção de dados ou saídas inválidas.

Confiabilidade

Uma exceção pode ser iniciada por uma declaração especial raise.

```
package body Directory_Enquiries is

   procedure Insert (New_Name   : in Name;
                     New_Number : in Number)
   is
      …
   begin
      …
      if New_Name = Old_Entry.A_Name then
         raise Name_Duplicated;
      end if;
      …
      New_Entry :=  new Dir_Node'(New_Name, New_Number,…);
      …
   exception
      when Storage_Error => raise Directory_Full;
   end Insert;

   procedure Lookup (Given_Name  : in  Name;
                     Corr_Number : out Number)
   is
      …
   begin
      …
      if not Found then
         raise Name_Absent;
      end if;
      …
   end Lookup;

end Directory_Enquiries;
```

Confiabilidade

Outra forma de tratamento de exceções também bastante utilizado é a partir do exception when.

```
declare
   A : Matrix (1 .. M, 1 .. N);
begin
   for I in 1 .. M loop
      for J in 1 .. N loop
         begin
            Get (A(I,J));
         exception
            when Data_Error =>
               Put ("Ill-formed matrix element");
               A(I,J) := 0.0;
         end;
       end loop;
   end loop;
exception
   when End_Error =>
      Put ("Matrix element(s) missing");
end;
```

Confiabilidade

É possível também declarar exceções próprias e exibir mensagens através do procedimento Raise Exception.

```
declare
   Valve_Failure : exception;
begin
  ...
  raise Valve_Failure with "Failure while opening";
  ...
  raise Valve_Failure with "Failure while closing";
  ...
exception
  when Fail: Valve_Failure =>
    Put (Exception_Message (Fail));
end;
```

Confiabilidade

- Outro fator influente na confiabilidade são os apelidos restritos, também conhecidos como alias.
- Tal recurso é considerado perigoso.
- Em Ada, tem-se os chamados access types, denominados ponteiros em muitas linguagens.
- Existem quatro tipos diferentes de access types:
  - Pool access types.
  - General access types.
  - Anonymous access types.
  - Access to subprogram types.

Trade-Off

Trade-Off

- Ada foi desenvolvida com três pontos em foco:
- Confiabilidade e manutenibilidade de programas.
- Programação como uma atividade humana.
- Eficiência.
- “Curva de aprendizagem” que desfavoreça os novatos ao Ada.
- Programas simples se tornem desnecessariamente grandes.

Referências

- https://www.quora.com/Is-the-Ada-programming-language-still-used
- http://homepages.dcc.ufmg.br/rimsa/documents/decom009/lessons/Aula05.pdf
- https://en.wikibooks.org/wiki/Ada Programming
- http://lpunb.wikia.com/wiki/Semin%C3%A1rio sobre Ada - 2012/2 -Grupo1
- https://en.wikibooks.org/wiki/Ada Programming/Operators
- https://pt.linkedin.com/pulse/elabora%C3%A7%C3%A3o-de-algoritmos-tratamento-exce%C3%A7%C3%B5es-carlos-eduardo

References

- https://en.wikibooks.org/wiki/Ada Programming/Types/access
- https://books.google.com.br/books?id=J3RZDwAAQBAJpg=PA7lpg=PA7dq=como+os+apelidos+restritos+afetam+a+confiabilidade+de+uma+linguagemsource=blots=IKGk1gN99sig=LDhCI8d923qW9iPyyJEsEGiFdoshl=ptBRsa=Xved=0ahUKEwi3undttLbAhVBEJAKHfytDb4Q6AEIKDAAv=snippetq=confiabilidadef=false
- https://en.wikibooks.org/wiki/Ada Programming/Object OrientationCreating Objects
- MAYOH, Brian. Problem Solving with ADA. Aarhus University: John Wiley Sons, 1982.

References

- https://www.adacore.com/about-ada
- https://pt.wikipedia.org/wiki/Ada Lovelace
- https://en.wikipedia.org/wiki/Ada (programming language)
- https://pt.wikipedia.org/wiki/Ada (linguagem de programa%C3%A7%C3%A3o)
- https://groups.google.com/forum/!topic/comp.lang.ada/1NnCRVaIVxg
- http://www.electronicdesign.com/iot/comparing-ada-and-c