

Université de Monastir
Institut Supérieur d'Informatique et de Mathématiques

Cours: Programmation Temps Réel et Concurrente

Filière: MP2-GL

Chapitre 1: Introduction à la programmation concurrente Ada

Réalisé par:
Dr. Sakka Rouis Taoufik

1

Ch 1: Introduction à la programmation concurrente Ada

I. Introduction

Ada est un langage de programmation fortement recommandé pour le développement des systèmes distribués.

En effet, le langage Ada offre des possibilités importantes vis-à-vis :

- **De la programmation structurée** : structuration de données (types prédéfinis et constructeurs de types simples et structurés) et structuration de traitements (structures de contrôle et les sous-programmes avec une distinction nette entre procédure et fonction). De plus, Ada est un langage fortement typé : il est plus sévère que Pascal!
- **De la programmation modulaire** en offrant un concept puissant, appelé package, doté de deux parties : interface (package) et implémentation (package body).

2

Ch 1: Introduction à la programmation concurrente Ada

I. Introduction

- **De la programmation générique** en offrant la possibilité de concevoir et de réaliser des unités génériques : sous-programmes et paquetages. Ces unités peuvent être paramétrées sur de types, variables et sous-programmes.
- **De la gestion des exceptions**: ceci permet d'écrire des logiciels robustes dans divers domaines critiques : temps réel, systèmes embarqués.
- **De la traitement de la concurrence**: le langage Ada offre des constructions intéressantes permettant d'écrire des programmes concurrents fiables.
- **De l'analyse statique**: Ada est supporté par plusieurs outils permettant l'analyse statique d'un programme concurrent Ada tels que SPIN, SMV, INCA et FLAVERS. Sachant que l'analyse statique est une technique qui permet d'analyser un programme sans toutefois l'exécuter.

3

Ch 1: Introduction à la programmation concurrente Ada

I. Introduction

- **Domaines d'application** : transport (avionique et ferroviaire), spatial, militaire.
- **Exemples** : Airbus (320, 380), Boeing (777), Fokker, Tupolev, Eurostar, Metro (14 Paris), TGV, Ariane (4 et 5), Satellites (Intersat), spatial (sonde Cassini, Huygens, Soho, Mars Express), militaire (Tigre, Apache, Patriot)
- **Exemples de projets Ada**
<https://www2.seas.gwu.edu/~mfeldman/ada-project-summary.html>

4

Ch 1: Introduction à la programmation concurrente Ada

II. Notre premier programme en Ada

```
with ada.text_io ;
use ada.text_io ;
procedure Hello is
    --partie réservée aux déclarations
begin
    Put ("Salut tout le monde !") ; --on affiche le message
end Hello ;
```

5

Ch 1: Introduction à la programmation concurrente Ada

II. Notre premier programme en Ada

- **Ada.Text_IO** est un package créé pour gérer le texte comme son nom l'indique (Text = texte et IO = In Out, entrée et sortie).
- Pour afficher un nombre entier, il faut utiliser le package **Ada.Integer_Text_IO**.
- Pour afficher un nombre reel, il faut utiliser le package **Ada.Float_Text_IO**
- La fonction **New_line** permet de retourner à la ligne.
- La fonction **Put_line()** fonctionne comme Put(), sauf qu'elle crée automatiquement un retour à la ligne à la fin ₆

Ch 1: Introduction à la programmation concurrente Ada

III. Variables I : Typage et affectation

Exemples:

X: Integer; -- val dans Z

A: Natural; -- val dans N

C1, C2: Character;

F1 : Float :=3.5;

7

Ch 1: Introduction à la programmation concurrente Ada

III. Variables I : Typage et affectation

```

WITH Ada.Text_IO ; USE Ada.Text_IO ;
WITH Ada.Integer_Text_IO ; USE Ada.Integer_Text_IO ;
PROCEDURE VotreAge IS
    age : Integer ;
BEGIN
    age := 27 ; --affectation simple
    Put("Vous avez ") ;
    Put(age) ;
    Put(" ans.") ;
END VotreAge ;

```

8

Ch 1: Introduction à la programmation concurrente Ada

III. Variables I : Typage et affectation

```

with Ada.Text_IO, Ada.Float_Text_IO ;
use Ada.Text_IO, Ada.Float_Text_IO ;
Procedure TaillePoids is
    Taille, Poids : float ;
Begin
    Put("Quelle est votre taille ?") ;
    Get(Taille) ; --affectation par l'utilisateur (saisie)
    Skip_line ; --Skip_line permet de vider la mémoire tampon
    Put("Vous mesurez ") ;
    Put(Taille) ;
    Put(" m.") ;
    Put("Quel est votre poids ?") ;
    Get(Poids) ; Skip_line ;
    Put("Vous pesez ") ;
    Put(Poids) ;
    Put(" kg.") ;
End TaillePoids ;

```

9

Ch 1: Introduction à la programmation concurrente Ada

IV. Constantes

Syntaxe:

MaVariable : **Constant** Integer := 15 ;

Exemple:

PI : **Constant** Float := 3.1415926535 ;

10

Chapitre 4: la concurrence en Ada

V. Attributs

Comment obtenir le plus petit et le plus grand Integer ?
Comment savoir quel est le 93ème character ? Grâce aux attributs, bien sûr. Ce sont des sortes d'instructions qui s'appliquent aux types.

```
N := integer'first ; --N sera le premier des integer
M := integer'last  ; --M sera le dernier des integer
C := character'val(93) ; --C sera la 93ème valeur des char
P := character'pos('f') ; --P aura pour valeur la position du
                        --character 'f'
```

11

Ch 1: Introduction à la programmation concurrente Ada

VI. Les opérations

1) Opérations sur les Integer et les Natural

Symbole	Opération	Exemple
+	Addition	N := 3 + 4 (résultat 7)
-	Soustraction	N := 5 - 4 (résultat 1)
*	Multiplication	N := 3 * 4 (résultat 12)
/	Division euclidienne (sans nombre à virgule)	N := 7 / 2 (résultat 3 et pas 3.5)
mod	Modulo ("reste" de la division euclidienne)	N := 7 mod 2 (résultat 1)
**	Puissance	N := 5**2 (résultat 25 car 5*5 = 25)

12

Ch 1: Introduction à la programmation concurrente Ada

VI. Les opérations

2) Opérations sur les Float

Symbole	Opération	Exemple
+	Addition	X := 3.0 + 4.1 (résultat 7.1)
-	Soustraction	X := 5.8 - 4.3 (résultat 1.5)
*	Multiplication	X := 3.5 * 4.0 (résultat 14)
/	Division décimale	X := 7.0 / 2.0 (résultat 3.5)
**	Puissance	X := 36.0**0.5 (résultat 6, la puissance 0.5 équivaut à la racine carre)
ABS	Valeur absolue	X := ABS(-8.7) (résultat 8.7)

13

Ch 1: Introduction à la programmation concurrente Ada

VI. Les opérations

3) Opérations sur les Character

Attribut	Explication	Exemple
'first	Renvoie le premier de tous les caractères.	c:=character'first ;
'last	Renvoie le dernier de tous les caractères.	c:=character'last ;
'pos(#)	Renvoie la position du caractère remplaçant #. Attention, le résultat est un Integer.	n := character'pos('z') ;
'val(#)	Renvoie le #ème caractère. Attention, le symbole # doit être remplacé par un Integer	c:=character'val(165) ;
'succ(#)	Renvoie le character suivant	c:=character'succ('c') ;
'pred(#)	Renvoie le character précédent	c:=character'pred('c') ;

14

Ch 1: Introduction à la programmation concurrente Ada

VI. Les opérations

3) Opérations sur les Character

Attribut	Explication	Exemple
'first	Renvoie le premier de tous les caractères.	c:=character'first ;
'last	Renvoie le dernier de tous les caractères.	c:=character'last ;
'pos(#)	Renvoie la position du caractère remplaçant #. Attention, le résultat est un Integer.	n := character'pos('z') ;
'val(#)	Renvoie le #ème caractère. Attention, le symbole # doit être remplacé par un Integer	c:=character'val(165) ;
'succ(#)	Renvoie le character suivant	c:=character'succ(c) ;
'pred(#)	Renvoie le character précédent	c:=character'pred(c) ;

15

Ch 1: Introduction à la programmation concurrente Ada

VII. Exercices

Exercice 1

Rédigez un programme appelé Multiple qui demande à l'utilisateur de saisir un nombre entier et lui retourne son double et son triple

Exercice 2

Rédigez un programme appelé Euclide qui demande deux nombres entiers à l'utilisateur puis renvoie le quotient et le reste de leur division euclidienne (c'est-à-dire la division entière telle que vous l'avez apprise en primaire).

16

Ch 1: Introduction à la programmation concurrente Ada

VII. Exercices

Exercice 3

Rédigez un programme appelé Cercle qui demande à l'utilisateur de saisir la longueur d'un rayon d'un cercle et qui affichera le périmètre et l'aire de ce cercle. Contrainte : le rayon du cercle sera un nombre **entier**

Exercice 4

Rédigez un programme appelé Lettres qui demande à l'utilisateur de saisir deux lettres minuscules et qui affichera leur majuscule ainsi que la lettre se trouvant «au milieu» des deux. À défaut, s'il y a deux lettres, le programme choisira la «plus petite».

NB. Pour obtenir la lettre majuscule, il suffit de lui «soustraire 32» (avec les précautions d'usage bien sûr)

17

Ch 1: Introduction à la programmation concurrente Ada

VII. Exercices

Correction Exercice 4:

```
WITH Ada.Text_IO ;
USE Ada.Text_IO ;

PROCEDURE Lettres IS
  C1,C2,C3 : character ;

BEGIN
  Put("Saisissez une premiere lettre minuscule : ") ;
  Get(C1) ; Skip_Line ;
  Put("Sa majuscule est ") ; Put(Character'Val(Character'Pos(C1)-32)) ; New_Line ;


  Put("Saisissez une deuxieme lettre minuscule : ") ;
  Get(C2) ; Skip_Line ;
  Put("Sa majuscule est ") ; Put(Character'Val(Character'Pos(C2)-32)) ; New_Line ;

  C3 := character'val((character'pos(C1) + character'pos(C2))/2) ;

  Put("La lettre du milieu est ") ; Put(C3) ;

END Lettres ;
```

18



<https://github.com/srtaoufik/CoursAdaMP2>

19