

PIM - Résumé

October 20, 2023

THEVENET Louis

Table des matières

1. Language algorithmique	1
1.1. Procédures et fonctions	1
2. Types de données	2
2.1. Algorithmique	2
2.2. Ada	3
3. Modules (Ada)	4
3.1. Spécification	4
3.2. Utilisation	4
4. Généricité	4
5. Exceptions (Ada)	5
6. Structure de données dynamiques - Pointeurs	5
6.1. Algorithmique	5
6.2. Ada	5

1. Language algorithmique

1.1. Procédures et fonctions

Définition 1.1.1: Une procédure est une **abstraction de contrôle**, qui définit une **action** ou une **instruction**.

Spécification :

1. identificateur (*un verbe*)
2. Paramètres
3. Pré-condition (*assurée par le contrat*)
4. Post-condition (*assurée par le contrat*)
5. Tests/exemples
6. Exceptions

Exemple : Spécification d'une procédure

```
1 nom : nom de la procédure
2 sémantique: décrire ce que réalise la procédure
3 paramètres:
4     F_Param_1 : Mode (In, In/Out, Out) Type; --Rôle du paramètre
5     ...
6     F_Param_n : Mode (In, In/Out, Out) Type; --Rôle du paramètre
7 pré-condition: Conditions sur les paramètres en entrée (in)
8 post-condition: Conditions sur les paramètres en sortie (out)
```

Définition 1.1.2: Une fonction est une **abstraction de données**, elle définit une **expression**, un **calcul**.

Elle n'a pas d'effet de bord et **retourne toujours un résultat**. La différence est que les flots de données sont limités à in.

Spécification :

1. Identificateur (*un nom commun*)
2. Paramètres
3. Type de retour
4. Pré-condition (*assurée par le contrat*)
5. Post-condition (*assurée par le contrat*)
6. Tests
7. Exceptions

Exemple : Spécification d'une fonction

```
1 Nom : nom de la fonction
2 Sémantique : sémantique de la fonction
3 Paramètres
4     F_Param_1 : mode (in) Type -- rôle du paramètre
5     ...
6     F_Param_n : mode (in) Type -- rôle du paramètre
7 Type de retour : Type du résultat retourné
8 Pré-condition : Conditions sur les paramètres en entrée
9 Post-condition : Conditions sur le résultat retourné
```

2. Types de données

2.1. Algorithmique

Définition 2.1.1: Énumération

C'est une liste de **valeurs possibles**, avec une relation d'ordre définie par l'ordre des éléments dans la définition.

Définition :

```
1 TYPE T_COULEUR EST ENUMERATION (BLANC, BLEU, ROUGE)
```

Ainsi on a $BLANC < BLEU < ROUGE$.

Définition 2.1.2: Enregistrement

Produit cartésien de plusieurs domaines.

Définition :

```
1 TYPE Date EST ENREGISTREMENT
2   Jour : entier {1 <= Jour ET Jour <= 31}
3   Mois : entier {1 <= Mois ET Mois <= 12}
4   Annee : entier
5 FIN ENREGISTREMENT
```

On peut ajouter des assertions comme ci-dessus.

Définition 2.1.3: Tableau

Indexation de valeurs de même type.

Définition :

```
1 MAX_1 : CONSTANCE Entier <-- 4
2 MAX_2 : CONSTANCE Entier <-- 6
3
4 TYPE T_Matrice EST TABLEAU (1..MAX_1, 1..MAX_2) DE Entier
5
6 T: T_Matrice
7 T(2,3) <-- 2017
```

2.2. Ada

```
1 type T_Couleur is (BLANC, BLEU, ROUGE);
2
3 type T_Date is record
```

```

4     Jour : Integer; --{1 <= Jour ET Jour <= 31}
5     Mois : Integer;
6     Annee : Integer;
7 end record;
8
9 type T_vecteur is array(1..10) of Integer;

```

3. Modules (Ada)

3.1. Spécification

```

1  -- Spécification d'un module Dates très simplifié.
2  package Dates is
3      type T_Mois is (JANVIER, FEVRIER, MARS, AVRIL, MAI, JUIN, JUILLET, AOUT,
4          SEPTEMBRE, OCTOBRE, NOVEMBRE, DECEMBRE);
5      type T_Date is record
6          Jour : Integer;
7          Mois : T_Mois;
8          Annee : Integer;
9      end record;
10
11     -- spécifications omises
12     procedure Initialiser (Date : out T_Date; Jour : in Integer; Mois : in
13         T_Mois; Annee : in integer )
14         with
15             Pre => Annee >= 0 and Jour >= 1 and Jour <= 31,
16             Post => Le_Jour (Date) = Jour and Le_Mois (Date) = Mois and L_Annee
17                 (Date) = Annee;
18
19     procedure Afficher (Date : in T_Date);
20     function Le_Mois (Date : in T_Date) return T_Mois;
21     function Le_Jour (Date : in T_Date) return Integer;
22     function L_Annee (Date : in T_Date) return Integer;
23 end Dates;

```

3.2. Utilisation

```

1  with Ada.Text_IO; use Ada.Text_IO;
2  with Dates; use Dates;
3  procedure Exemple_Dates is
4      Une_Date : T_Date;
5  begin
6      -- Initialiser une date
7      Initialiser (Une_Date, 2, OCTOBRE, 2020);
8      Afficher (Une_Date);
9      New_Line;
10 end Exemple_Dates;

```

4. Généricité

- Spécifier l'unité avec des paramètres de généricité

```

1  generic
2      type Un_Type is private;
3  procedure Permuter_Generique (X, Y : in out Un_Type);

```

- Implanter l'unité

```

1  procedure Permuter_Generique (X,Y : in out Un_Type) is
2  Memoire: Un_Type;
3  begin
4  Memoire := X;
5  X:=Y;
6  Y:=Memoire;
7  end Permuter_Generique;

```

- Instancier avec une valeur pour les paramètres de généricité

```

1  procedure Permuter_Entiers is new Permuter_Generique(Un_Type => Integer);

```

- Utiliser l'unité

```

1  A, B: Integer;
2  Permuter(A,B);

```

5. Exceptions (Ada)

```

1  procedure Fonction() is
2  begin
3  -- du code
4
5  exception
6      when Exception_name =>
7      -- bloc à exec
8  end Fonction

```

6. Structure de données dynamiques - Pointeurs

6.1. Algorithmique

```

1  TYPE T_Ptr_Sur_T_Nom_Type EST POINTEUR SUR T_Nom_Type
2  Ptr_T : T_Ptr_Sur_T_Nom_Type
3
4  Ptr_T <-- Null          -- initialisation
5  Ptr_T <-- NEW ENTIER    -- allocation
6  Ptr_T <-- Ptr_T_1       -- affectation
7  Ptr_T^                  -- déréférencement

```

6.2. Ada

```

1  type T_Cellule_Caractere; --! pour annoncer la référence en avant
2
3  type T_Liste_Caractere is access T_Cellule_Caractere; --! Type pointeur

```

```

4
5  type T_Cellule_Caractere is record
6      Element : Character;
7      Suivant : T_Liste_Caractere; -- Accès à la cellule suivante
8  end record;
9
10 -- ...
11
12 Liste : T_Liste_Caractere;
13 Liste.All --! déréférencement
14 Liste.All.Element --! accès à Element

```