Algorithmique & programmation

Type structuré

Article, Enregistrement, Structure

Définition de nouveaux types

- On a vu les types simples
  - entier, booléen, caractère, chaîne de caractères
- Comment gérer des types plus complexes ?
  - Date
    - jour, mois, année
  - Personne
    - nom, prénom, date de naissance, situation maritale, …
  - Informations sur un étudiant
    - nom, prénom, âge, classement
- Avec des types structurés
  - Encapsulation de différentes informations

Définitions

- Un **`article`** (une **structure**) est un objet
  - composé d'autres objets (de types éventuellement différents) appelés
    - champs
    - composants
    - attributs
- Un **`article`** (une **structure**) est décrit par son type
  - `type d'article`
  - type de structure

Modèle de déclaration

en ada	Dans un algorithme
type *<nom du type>* **is record** *<liste de composants>* **end record;**	**type** *<nom du type>* **= structure** *<liste de composants>* **fin;**

- En **ada** on convient de faire précéder tous les noms de type d'article par **`TR_`**

Exemples (ada)

```
type T_Mois is
 (Jan,Fev,Mar,Avr,Mai,Jui,Jul,Aou,Sep,Oct,Nov,Dec);
subtype T_Jour  is Integer range 1..31;
subtype T_Année is Integer range 1900..2099;

  type TR_Date is record
    Jour   : T_Jour;
    Mois   : T_Mois;
    Annee  : T_Année ;
  end record;

  type TR_Point is record
    x,y : integer;
  end record;
```


Exemples (ada)

```
subtype TV_Nom         is String(1..9);
type    T_Genre        is (Masculin,Feminin);
type    T_SitMaritale is
                  (Célibataire, Marié, Divorcé, Veuf);


type TR_Personne is record
  Nom,Prénom    : T_Nom;
  DateNaissance : T_Date;
  Genre         : T_Genre;
  Statut        : T_SitMaritale;
end record;
```

Exemples (algorithme)

```
type    date = structure
            jour, mois, an : entier ;
        fin ;

type    personne = structure
            nom : chaîne20 ;
            adresse : chaîne40 ;
            naissance : date ;
            marié : booléen ;
            enfants : entier ;
        fin ;
```


Variable de type article, structure

- Lorsqu'un type article a été déclaré, il est possible de déclarer des variables de ce type
- Exemples `ada` :
  - `Aujourdhui,Demain : TR_Date;`
  - `Moi : TR_Personne;`
- Exemples algo :
  - aujourdhui : date;
  - moi : personne;

Accès aux attributs, composants

- En ada et dans un algorithme on utilise la même notation
  - Si **pers** est une variable de type **personne**
  - Accès aux attributs de **pers** avec l'opérateur .

- Ada
  - `lejour:=demain.jour;  --de type T_jour`
  - `lemoi:=demain.mois;  --de type T_mois`
  - `lannée:=demain.annee;  --de type T_annee`

- Algo
  - nom_pers := pers.nom ; *{de type chaîne20}*
  - nb_enfants := pers.enfants ; *{de type entier}*

Agrégats (ada)

- ada fournit une notation pratique, appelée **agrégat**, pour initialiser les valeurs des structures de données de base (tableaux ou articles)

- Un agrégat est capable de fournir la description complète de la valeur d'une variable article ou tableau

- Cette initialisation consiste simplement à énumérer les valeurs de chaque composant

Agrégat avec une variable tableau

```
type T_Mois is
             (Jan,Fev,Mar,Avr,Mai,Jui,Jul,Aou,Sep,Oct,Nov,Dec);

type TV_DuréeMois is array(T_Mois) of Positive range 28..31;

DuréeMois : TV_DuréeMois;

...
--les deux instructions suivantes produisent le même effet
DuréeMois := (31,28,31,30,31,30,31,31,30,31,30,31);

DuréeMois := (fev=>28, avr=>30, jui=>30, sep=>30, nov=>30,
           others => 31);
```


Agrégat avec une variable article

- L'utilisation d'un agrégat avec une variable article est similaire à celle des tableaux
- Les deux systèmes de notation par position et par nom sont disponibles
- En notation nommée on utilise les noms des champs (au lieu des index)

Exemples

```
Aujourdhui := (25,oct,1995);

Demain := (Mois=>oct, Annnée=>1995,
Jour=>25);

Lui : TR_Personne :=
("dupont    ", "jacques  ",
 (10,oct,1970), masculin, marié);
```


Opérations sur les articles

- La sélection d'un composant est réalisée par l'expression suivante :

  `nom_de_la_variable.nom_du_champ_désiré`

- `Aujourdhui.Mois` désigne donc le champ `Mois` de l'article `Aujourdhui`
- Un nom de champ n'est jamais utilisé isolément
  - (sauf dans un agrégat avec notation nommée) ...
- ... pour des raisons évidentes d'ambiguïté
  - les noms de champ sont partagés par toutes les variables d'un même type article.

Exemple

- Calcul de la date de demain en utilisant des agrégats et des sélections

```
subtype T_Jour is Positive range 1..31;

type    T_Mois is
            (Jan,Fev,Mar,Avr,Mai,Jui,Jul,Aou,Sep,Oct,Nov,Dec);

type TR_Date is record
  Jour : T_Jour;
  Mois : T_Mois;
  Année: Integer range 1900..2099;
end record;

type TV_DuréeMois is array(Tmois) of Positive range 28..31;

DUREEMOIS : constant T_DuréeMois :=
                        (31,28,31,30,31,30,31,31,30,31,30,31);
```



```
function Demain(Now : in TR_Date) returns TR_Date is
begin
   --pas le dernier jour du mois
   if Now.Jour<DUREEMOIS(Now.Mois) then
     --un jour de plus
     return (Now.Jour+1, Now.Mois, Now.Année);
   --dernier jour d'un mois qui n'est pas le dernier mois
   elsif Now.Mois<T_Mois'Last then
     --premier du mois suivant
     return (1, T_Mois'Succ(Now.Mois),Now.Année);
   --dernier jour du dernier mois
   else
     --premier du premier mois de l'année suivante
     return (1, T_Mois'First, Now.Année+1);
   end if;
end Demain;
```

Comparaison de deux articles

- Pour des raisons d'ambiguïté …
  - il y a de nombreuses façons d'ordonner des articles en fonction des différents attributs
- … on ne peut donc tester que l'**égalité** ou la **non égalité** de deux articles

```
if Anniversaire=Aujourdhui then
  Ecrire("Bon anniversaire");
end if;
```


Affectation à un article

- On peut affecter globalement deux articles en ada et dans un algorithme ainsi :

DateNaissance := Aujourdhui;

- revient à écrire

DateNaissance.Année := Aujourdhui.Année;
DateNaissance.Mois := Aujourdhui.Mois;
DateNaissance.Jour := Aujourdhui.Jour;