
Algorithmique & programmation

Ada

paquetage, module, bibliothèque

Paquetages (*modules*)

- Répondre à des objectifs importants en programmation
 - réutiliser des portions de programmes existant afin de réduire le coût du développement
 - regrouper un ensemble de ressources (déclarations, procédures ou fonctions) liées logiquement entre elles

- Le concept de module a été introduit comme unité de structuration pour les programmes de taille importante mettant en œuvre plusieurs programmeurs

- Les programmeurs développent séparément des modules et communiquent entre eux par échange d'interface

- Les **paquetages** sont l'une des formes d'**unités de programmes** (comme les sous-programmes – **fonctions & procédures**) servant à (dé)composer les programmes

Paquetages (modules)

- Un paquetage est une unité de compilation regroupant :
 - des déclarations de données (types et objets) **et/ou**
 - des sous-programmes pour le leur traitement
- Un **paquetage** est composé de 2 parties, enregistrées dans **2 fichiers** séparés :
 - une **spécification** (package)
 - contenant l'information qui doit être visible à l'extérieur
 - fichier avec l'extension **.ads** (**ada specification**)
 - un **corps** (package body)
 - contenant les détails d'implémentation qui n'ont pas besoin d'être visibles des autres unités
 - fichier avec l'extension **.adb** (**ada body**)

Paquetages (modules)

- Un paquetage est réutilisable dans un autre contexte,
 - son utilisation dans une autre unité est indiquée par la clause **with**

```
with P_Esiut; use P_Esiut;
```

- L'ordre de compilation est important



- les unités de programmes doivent être compilées après celles qu'elles utilisent

- Convention

- Tout nom de paquetage sera préfixé par **P_**

Exemple de Spécification

□ Contenu du fichier `P_Calcul.ads`

```
package P_Calcul is
  subtype T_Note is Integer range 0..20;
  function Maximum (X,Y : in T_Note) return T_Note;
  --spécification de la fonction Maximum
  procedure Afficher (N : in T_Note);
  --spécification de la procédure Afficher
end P_Calcul ;
```

□ Le fichier de spécification contient :

- les déclarations de type ou sous-type
- les entêtes des fonctions et procédures du paquetage

Exemple de Corps (*body*)

□ Contenu du fichier `P_Calcul.adb`

```
with P_Esiut ; use P_Esiut ;
package body P_Calcul is
  function Maximum (X,Y: in T_Note) return T_Note is
  begin
    if X>Y then
      return X;
    else
      return Y;
    end if;
  end Maximum ;
  procedure Afficher (N: in T_Note) is
  begin
    Ecrire(" note :") ; Ecrire(N) ;
  end Afficher ;
end P_Calcul;
```

```

with P_Esiut ; use P_Esiut ;
  --paquetage nécessaire à l'exécution des
  fonctions et procédures du paquetage P_Calcul

package body P_Calcul is  --début du corps du paquetage

  function Maximum (X,Y: in T_Note) return T_Note is
  --définition de la fonction Maximum
  begin
    if X>Y then
      return X;
    else
      return Y;
    end if;
  end Maximum ;

  procedure Afficher (N: in T_Note) is
  --définition de la procédure Afficher
  begin
    Ecrire(" note :") ; Ecrire(N) ;
  end Afficher ;

end P_Calcul;  --fin du corps du paquetage

```

Paquetage : corps

- ❑ Toute fonction ou procédure déclarée dans la spécification (fichier `.ads`) doit apparaître – éventuellement réduite à l'instruction `null` – dans le corps correspondant.

- ❑ Un corps peut aussi contenir des déclarations de
 - types,
 - constantes,
 - variables,
 - et aussi d'autres fonctions et procédures :
 - ❑ elles sont alors locales au corps et utilisables dans toutes les procédures du corps.

Procédure principale utilisant P_Calcul

□ Contenu du fichier demo.adb

```
with P_Esiut; use P_Esiut;
with P_Calcul;
procedure Demo is
  A,B,M : T_Note;
begin
  Ecrire("première note : ");
  Lire(A) ;
  Ecrire("deuxième note : ");
  Lire(B) ;
  M := P_Calcul.Maximum(A,B);
  P_Calcul.Afficher(M) ;
end Demo ;
```

```
with P_Esiut; use P_Esiut;
with P_Calcul;
procedure Demo is
  A,B,M : T_Note;
begin
  Ecrire("première note : ");
  Lire(A) ;
  Ecrire("deuxième note : ");
  Lire(B) ;
  M := P_Calcul.Maximum(A,B);
  P_Calcul.Afficher(M) ;
end Demo ;
```

with + use

le compilateur sait où trouver `ecrire` et `lire` et peut faire l'édition de lien

with

le compilateur autorise l'appel à des fonctions et procédures de `P_Calcul`; pour faire l'édition de lien, il faut **préfixer les procédures et fonctions par le nom du paquetage**

Procédure principale utilisant P_Calcul

- Si on insère la clause `use P_Calcul`; alors on peut supprimer le préfixe `P_Calcul.` devant les appels de sous-programmes de `P_Calcul`

```
with P_Esiut; use P_Esiut;
with P_Calcul; use P_Calcul;
procedure Demo is
  A,B,M : T_Note;
begin
  Ecrire("première note : ");
  Lire(A) ;
  Ecrire("deuxième note : ");
  Lire(B) ;
  M := Maximum(A,B);
  Afficher(M) ;
end Demo ;
```

```
with P_Esiut; use P_Esiut;
with P_Calcul; use P_Calcul;

procedure Demo is

  A,B,M : T_Note;

begin

  Ecrire("première note : ");
  Lire(A) ;
  Ecrire("deuxième note : ");
  Lire(B) ;
  M := Maximum(A,B);
  Afficher(M) ;

end Demo ;
```

with + use

le compilateur sait où trouver `ecrire` et `lire` et peut faire l'édition de lien

with + use

le compilateur sait où trouver `Maximum` et `Afficher` et peut faire l'édition de lien **il n'est plus nécessaire de préfixer les procédures et fonctions par le nom du paquetage**



Si des procédures ou fonctions ont le même nom dans 2 paquetages différents, **il faut préfixer** pour que le compilateurs sache laquelle utiliser !