Cours de Programmation

Stéphanie Combettes

Présentation du cours de PROG

- **Enseignement de prog. divisé en**
  - Cours - TD : début de semaine
  - TP (sur machine) : fin de semaine
  - Tests hebdomadaires à faire

- **Equipe enseignante**
  - C-TD : Stéphanie Combettes, Samir Medjiah, Philippe Latu et Régis Nohé
  - TP : Les mêmes + Claude Cousturian, Isabelle Silvain et Luca Sartori

Présentation du cours de PROG -

- **Tests sous « Moodle »**
  - Aide au travail personnel
  - 1 test à faire par semaine ouvert pour 15 jours

     Ne pas attendre le dernier jour !!!
- **Examens – notes :**
  - 1 partiel écrit
  - 1 examen de TP
  - 3 examens de TD
  - 1 note de suivi de TP
  - 1 note des tests Moodle

  ➢ 3 Notes finales :
  - 1 note d'Examen écrit
  - 1 note de TD
  - 1 note de TP
- **Projet de fin de semestre** : 1 semaine pour tout changer en prog
- **Planning du semestre** disponible sous Moodle

Chapitre 4
Sous-programmes

- Contexte
- Définition d'un sous-programme
- Généralités sur les sous-programmes
  - Création d'un SP
  - Structure d'un SP
  - Catégories de SP : fonction et procédure
- Fonction
- Procédure

Contexte

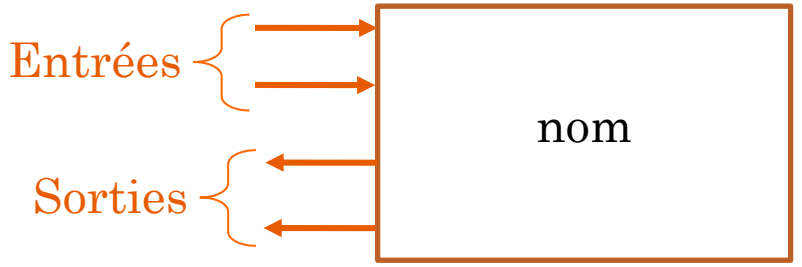
- Si problèmes à résoudre complexes
  - Découper le problème en modules plus simples
    - Un module peut être lui-aussi raffiné en modules plus simples, etc…
  - Faire la mise au point facilitée
    - Module par module
  - Réutiliser plus facilement un module

- Même démarche en programmation
  - Découper un programme en sous-programmes
  - Mettre au point chaque sous-programme indépendamment les uns des autres
  - Pouvoir réutiliser les sous-programmes
    - Dans le programme
    - Dans plusieurs programmes (notion de bibliothèque)

Définition

Un sous-programme est un programme particulier qui réalise un service spécifique

- il possède :
  - un identificateur,
  - un début et une fin,
  - des données propres,
  - une séquence d'actions
- mais il est exécuté
  - à la demande d'un programme (ou sous-prog) dit **Programme Appelant**
- il peut
  - recevoir des informations du programme appelant : des **Entrées**
  - restituer des résultats au programme appelant : des **Sorties**
- il peut être
  - créé par un programmeur
  - déjà défini dans des bibliothèques existantes



SOUS-PROGRAMME

GÉNÉRALITÉS

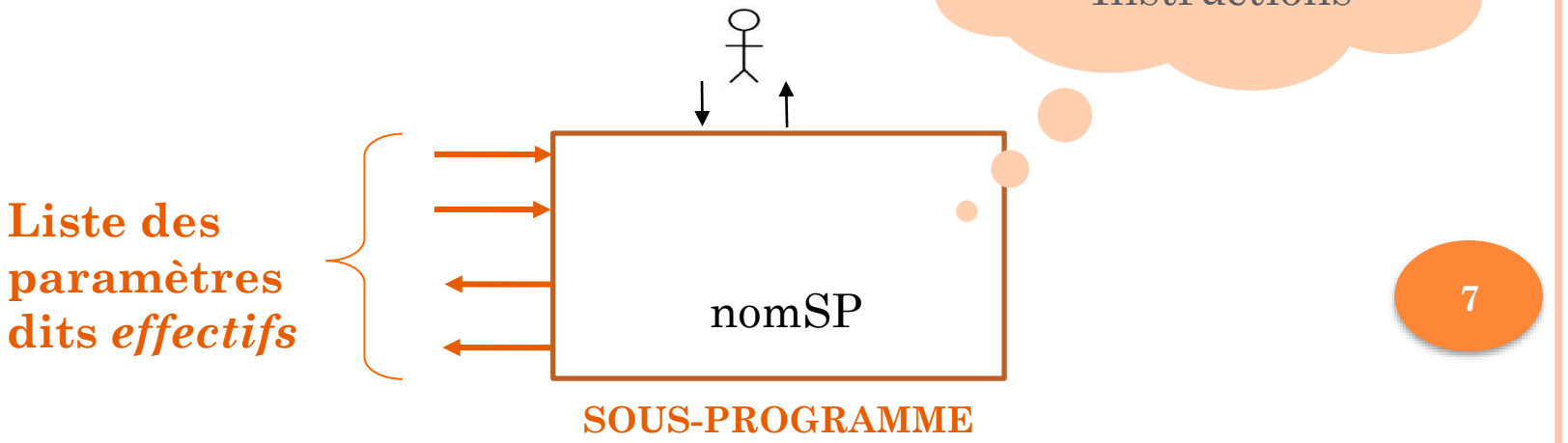
- **Données locales** (ou propres) sont les données utilisées dans le sous-programme (SP).
  - Exemple : la variable *indice* d'une boucle POUR
- **Suite d'instructions** ou d'actions permet de réaliser l'objectif du SP.

Vue interne du SP : contenu non visibles par les programmes appelants et les autres SPs

- **Paramètres** : ce sont les entrées et sorties échangées entre le sous-programme appelé et le (sous-)programme appelant
- Vue externe

d'un sous-programme

Vue interne : Données locales, Instructions



CRÉATION D'UN SP : ETAPE 1

1) Déclaration ou définition

- Description du SP dans son entièreté

Entête (vue externe)

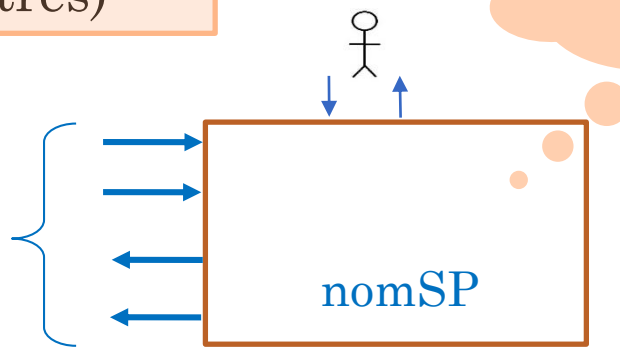
- Identificateur : le nom du SP
- Liste des paramètres *formels* (Entrées/sorties)

Corps (vue interne)

- Données locales (propres) au SP
- Instructions du SP (c'est-à-dire de la séquence d'actions qu'il réalise sur les données locales et les paramètres)

Vue interne :
Données locales,
Instructions

Liste des paramètres dits *effectifs*

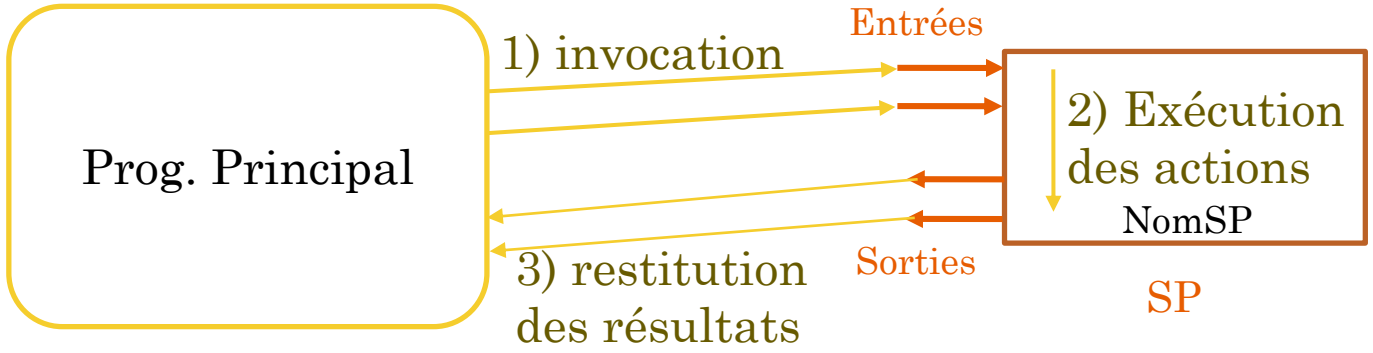


Sous-Programme

CRÉATION D'UN SP : ETAPE 2

2) Appel ou Invocation

- Il s'agit de l'utilisation du SP par le programme appelant
- Le programme appelant (programme principal par ex.) appelle ou invoque le SP pour que ce SP lui fournisse le résultat de son service
  1. Lors de l'appel, le programme appelant fournit au SP les valeurs des paramètres d'entrées
  2. Le SP s'exécute en utilisant les valeurs fournies par le SP et calcule les valeurs résultats
  3. A la fin de l'exécution de son exécution le SP fournit au programme appelant les valeurs des paramètres de sorties



- Une SEULE déclaration de SP, mais PLUSIEURS utilisations possibles

STRUCTURE D'UN PROGRAMME CONTENANT DES SP

DECLARATIONS GLOBALES (du Programme principal)

SOUS-PROGRAMMES

% déclaration des SP ➔ étape 1

DEBUT PP (programme principal)

DECLARATIONS (du programme principal)

ACTIONS(du programme principal)

% Appel ou invocation des SP ➔ étape2

FIN PGME

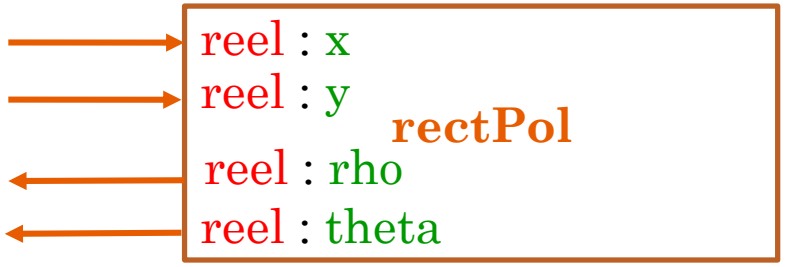
Catégories de SP

- 2 catégories de sous-programmes qui dépendent du nombre de résultats retournés

  - La **fonction** qui fournit **un et un seul résultat**

FONCTION

  - La **procédure** qui possède **plusieurs sorties** ou **aucune**
    - Résultat multiforme



PROCEDURE

La fonction

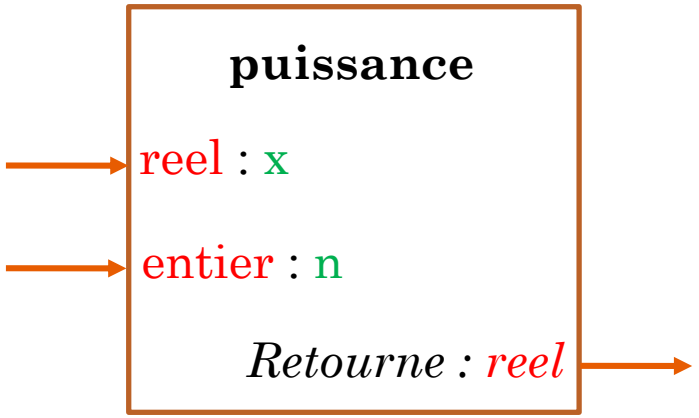
- Une **fonction** décrit une relation entre une ou plusieurs entrées et UNE SEULE sortie
- Elle renvoie 1 seul résultat (numérique ou booléen ou caractère)

- Structure de l'entête

  Fonction nomFonction : type (liste des paramètres)
- Corps :
  - Déclaration des variables locales (visibles uniquement dans le SP)
  - Description des actions à réaliser

Structure de la déclaration d'une fonction

- toute fonction est **TYPEE**
  - **Le type de la fonction** correspond au **type du résultat retourné**
- Elle renvoie UN SEUL résultat via « ***Retourne*** » (ou « `return` » en C/C++)



Structure générale de la déclaration d'une fonction

```
FONCTION type : nom_Fonction ( liste des paramètres formels et de leur type)
    |       DECLARATIONS_LOCALES
    |       |  % Les variables déclarées ici sont visibles
    |       |   % UNIQUEMENT dans ce SP
    |       ACTIONS
    |       |    % Description des actions à faire pour atteindre
    |       |    % son objectif
    |       |   retourne variableResultat
FIN_FCT
```

```
FONCTION reel : puissance (reel : x , entier : n)
    |       DECLARATIONS_LOCALES
    |       |   reel : resultat
    |       |   entier : i
    |       ACTIONS
    |       |   resultat ← 1
    |       |  POUR (i←0) QUAND (i<n ) AVEC (i←i+1)
    |       |  |     resultat ←resultat*x
    |       |  FPR
    |       |  retourne resultat
FIN_FCT
```

APPEL OU INVOCATION D'UNE FONCTION

Resultat ← nomFonction (liste des paramètres *effectifs*)

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  FONCTION reel : puissance (reel : x, entier : n)
   |     |  ....
   |  FIN_FCT
DEBUT_PGME PRINCIPAL
    |     DECLARATIONS
    |      |  entier : i
    |      |  reel : a, aCarre, a_i , cCube
    |     ACTIONS
    |      |  lire_clavier a , i
    |      |  aCarre ← puissance (a, 2)
    |      |  a_i ← puissance ( a, i )
    |      |  c_cube ← puissance((a*i)+10 , 3)
    |     |
FIN_PGME
```

Paramètres FORMELS

Paramètres EFFECTIFS : même ordre et correspondance des types

- Correspondance des types et de l'**ordre** des paramètres entre les paramètres effectifs et formels
- La valeur du résultat calculé dans la fonction est stockée (via « Retourne ») dans la variable située à gauche de ←

FOCUS : PARAMÈTRES FORMELS - PARAMÈTRES EFFECTIFS

- Paramètres **formels**
  - **Entrées** (c-à-d données et types) nécessaires au SP pour réaliser son objectif et **sorties** servant à stocker les résultats
  - Représentent le cas général d'utilisation du SP
  - Permettent de décrire les actions à réaliser sur les entrées et les résultats affectés aux sorties

- Paramètres **effectifs**
  - Sont les données (valeur ou variable) réellement utilisées lors de l'exécution du SP
  - Représente un cas particulier d'utilisation du SP
  - Ont le même type que les paramètres formels
  - Sont écrits dans le **même ordre**
  - Peuvent être une variable, une valeur ou une expression

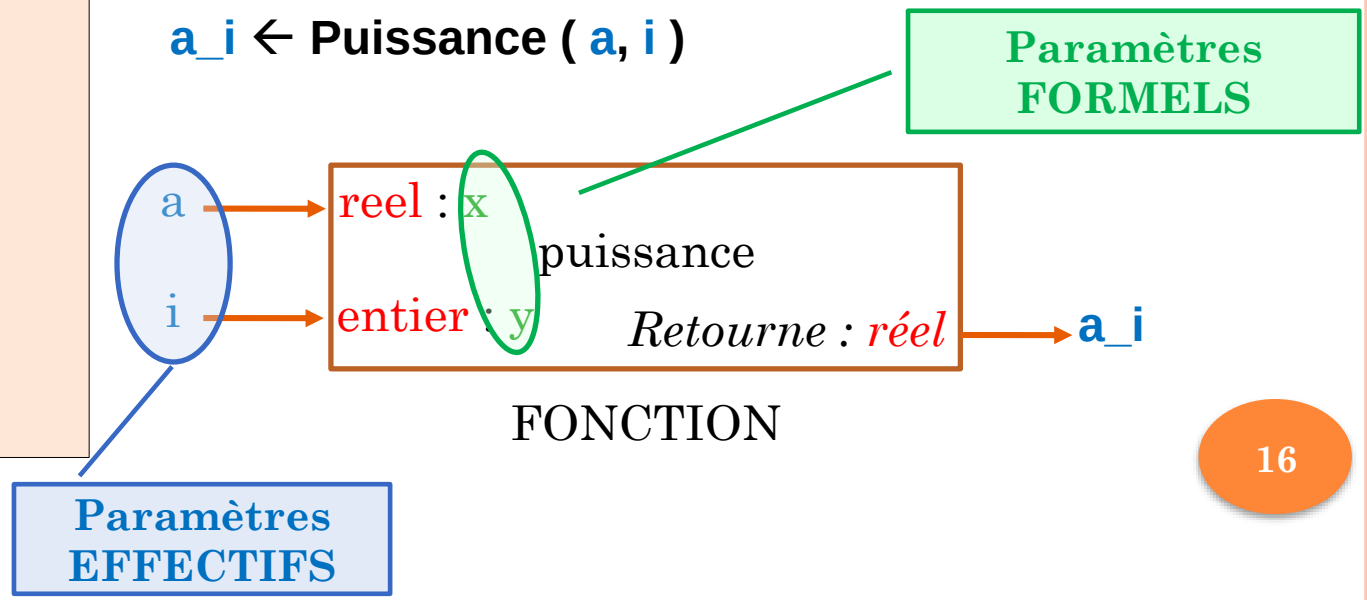
FOCUS : PARAMÈTRES FORMELS – PARAMÈTRES EFFECTIFS

```
SOUS-PROGRAMMES
   |  % définitions des SP
   |  FONCTION reel : puissance(reel : x, entier : n)
   |     | ....
   |  FIN_FCT
DEBUT_PGME PRINCIPAL
    |      DECLARATIONS
    |      | entier : i
    |      | reel : a, aCarre, a_i , cCube
    |      ACTIONS
    |      | lire_clavier a , i
    |      |aCarre ← puissance (a, 2)
    |      | a_i ← puissance ( a, i )
    |      | cCube ← puissance((a*i)+10 , 3)
    |      |
FIN_PGME
```

- Exemple de **paramètres formels**

reel : puissance (reel : x, entier : n) ☜ Cas général

- Exemples de **paramètres effectifs**
  - puissance (3.5, 2) ☜ 1 cas particulier
  - puissance (a, i)
  - puissance ((a*i)+10 , 3)



La fonction : synthèse

SOUS-PROGRAMMES

| % **Définitions des SP**

DEBUT PGME PRINCIPAL

| **DECLARATIONS**

| **ACTIONS**

| % Appel ou invocation des SP

FIN PGME

FONCTION type : nomFonction (liste des paramètres *formels* et de leur type)

| **DECLARATIONS LOCALES**

| % Les variables déclarées ici sont visibles

| % UNIQUEMENT dans ce SP

| **ACTIONS**

| % Description des actions à faire pour atteindre

| % son objectif

| **retourne resultat**

FIN FCT

Ne Pas Oublier !

leResultat ← Nom_Fonction (liste des paramètres *effectifs*)

La Fonction en C++

- La déclaration commence par le type de la fonction
- Paramètres d'entrée **passés par valeur**
- Le résultat restitué via ***return***

```
float puissance ( float x, int n)
{
    ....
    return resultat ;
}
```

- Appel dans le Prog. appelant

```
carre_a = puissance(a,2);
```

Pas de type lors de l'appel

Passage par Valeur

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  FONCTION reel : puissance (reel : x,
                                 entier : n)
   |     | ....
   |  FIN_FCT
DEBUT_PGME PRINCIPAL
   |     DECLARATIONS
   |     |  entier : i
   |     |  reel : a, a_i
   |     ACTIONS
   |     |  a ← 3,0
   |     |  i ← 4
   |     |  a_i ← puissance(a,i)
   |     |  ecrire-ecran a_i
FIN_PGME
```

Passage par valeur

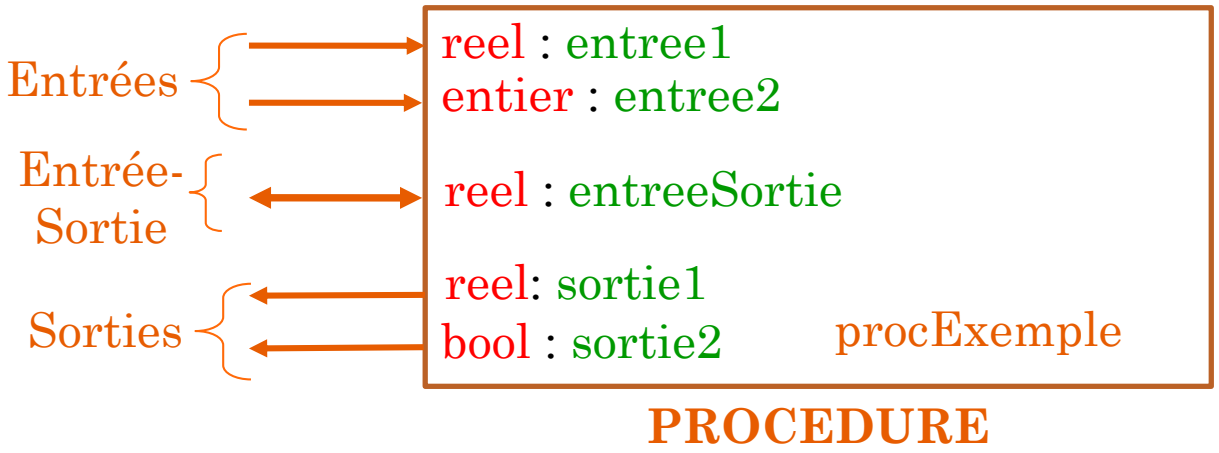
- La valeur du paramètre effectif (lors de l'appel de la fonction) est **copiée** dans la variable locale représentée par le paramètre formel (lors de la déclaration de la fonction).
  - Le **paramètre effectif ne peut pas être modifié**.
- Les modifications des paramètres formels à l'intérieur de la fonction sont perdues à la sortie de la fonction.

Exemple ci-contre : a_i ← Puissance(a, i)

1. La valeur de a **est copiée** dans le paramètre formel x.
2. La valeur de i **est copiée** dans le paramètre formel n.
3. La fonction calcule le résultat à partie des valeurs précédemment copiées de x et n et stocke la valeur du résultat dans la variable a_i.
4. Une fois le résultat stocké, les valeurs de x et n ne sont pas conservées.

NB. Les variables a et i ne sont pas modifiées !

La Procédure

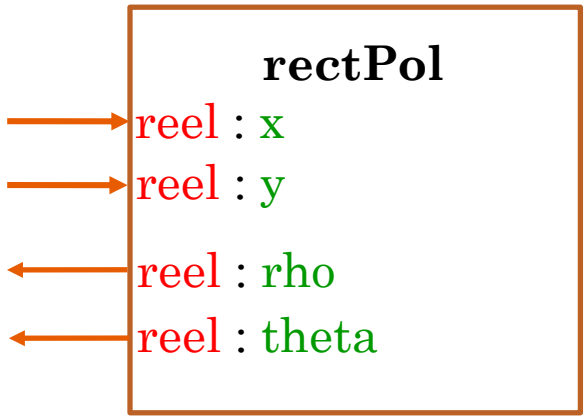


- Une procédure peut posséder
  - 0 ou plusieurs paramètres d'entrées
  - plusieurs paramètres de sortie ou aucun (si le SP consiste seulement à un affichage)
- Le.s résultat.s sont stocké.s dans le.s paramètre.s de sortie
  - Si un paramètre d'entrée doit être modifié dans la procédure, on parle alors d'un **paramètre d'entrée/sortie** (entrée ET sortie à la fois)
- Structure de l'entête

  Procédure nomProc (liste des paramètres)
- Corps :
  - Déclaration des variables locales (visibles uniquement dans le SP)
  - Description des actions à réaliser

Déclaration d'une Procédure

- Procédure non typée
- Notation différente entre les Entrées et les Sorties



Les entrées **soulignées**

Les sorties **encadrées**

PROCEDURE nom_Procedure (liste des paramètres *formels* et de leur type)
| **DECLARATIONS_LOCALES**
| | % Les variables déclarées ici sont visibles
| | % UNIQUEMENT dans ce SP
| **ACTIONS**
| | %Description des actions à faire pour
| | % atteindre son objectif
FIN_PROC

PROCEDURE rectPol (reel: x, y , reel : Rho, Theta)
| **DECLARATIONS_LOCALES**
| | reel : resultat
| |
| **ACTIONS**
| | %description des actions
| |
FIN_PROC

APPEL OU INVOCATION D'UNE PROCÉDURE

Appel nom_Procedure (liste des paramètres *effectifs*)

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  PROCEDURE  rectPol (reel: x, y ,
                          reel : rho, theta  )
   | ....
   |  FIN_PROC
DEBUT_PGME PRINCIPAL
    |     DECLARATIONS
    |      | reel : rhoA, thetaA
    |      | reel : a, b rhoB, thetaB
    |     ACTIONS
    |      | lire_clavier a, b
    |      |Appel rectPol( a, b, rhoA, thetaA)
    |      | Appel rectPol ( b ,a, rhoB, thetaB)
    |      |
FIN_PGME
```

Paramètres FORMELS

Paramètres EFFECTIFS : même ordre et correspondance des types

- Correspondance des type et de l'ordre entre les paramètres effectifs et les paramètres formels
- Les résultats sont stockés dans les paramètres de sortie

La Procédure : Synthèse

SOUS-PROGRAMMES

| % **Déclaration des SP**

DEBUT PGME PRINCIPAL

| **DECLARATIONS**

| **ACTIONS**

| % Appel ou invocation des SP

FIN PGME

PROCEDURE nomProcedure(liste des paramètres *formels* et de leur type)

| **DECLARATIONS LOCALES**

| % Les variables déclarées ici sont visibles

| % UNIQUEMENT dans ce SP

| **ACTIONS**

| % Description des actions à faire pour atteindre

| % son objectif

FIN PROC

Appel nomProcedure (liste des paramètres *effectifs*)

Une Procédure en C++

- Déclaration de la procédure
  - Commence par le mot clé **void**
  - Paramètres formels d'entrée **passés par valeur**
  - Paramètres formels de sortie **passés par référence**

```
void rectPol ( float x, float y, float & rho, float & theta)
        {
        ....
        }
```

- Appel dans le Prog. Appelant

```
rectPol (a,b, rhoA, thetaA) ;
```

Pas de type lors de l'appel

Passage par Référence

```
//SOUS-PROGRAMMES
   |  // déclaration des SP
   |  void rectPol ( float x, float y, float &
rho, float & theta)
   |     { .... }
   |
//DEBUT  PGME PRINCIPAL
   |  int main(void)   {
   |        //DECLARATIONS
   |        |    float rhoA,thetaA ;
   |        |    float  a, b rhoB, thetaB ;
   |        //ACTIONS
   |        |    cin>> a >> b ;
   |        |    rectPol( a, b, rhoA, thetaA);
   |        |    rectPol ( b ,a, rhoB, thetaB) ;
   |    }
//FIN  PGME
```

Passage par référence

- Une référence est un alias à un espace mémoire d'une variable notée : type& var
- Une fois la référence déclarée alias d'une autre variable, toutes les opérations que l'on croit effectuer sur l'alias (i.e. la référence) sont en fait exécutées sur la variable d'origine

Exemple ci-contre

float& rho et **float& theta** sont chacun une référence, un alias, des paramètres effectifs utilisées lors de l'appel.

Lors de l'appel de **rectPol**, les variables **rhoA** et **thetaA** sont utilisées et modifiées.

NB. Les paramètres a et b sont passées par valeur, donc la valeur de a (respect. b) **est copiée** dans le paramètre formel x (respect. y).Les variables a et b ne sont pas modifiées !

Exemple 1

Ecrire un SP afficheMessage qui affiche à l'écran « bonne journée », puis écrire le Prog Princ qui l'appelle.

1. Faire la vue externe
2. Type de SP ? Pourquoi
3. Faire entête et corps du SP
4. Faire PP

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  PROCEDIRE afficheMessage ( )
   |    |  ecrire_ecran "Bonne journée"
   |    |
   |  FIN_PROC
DEBUT_PGME PRINCIPAL
    | DECLARATIONS
    |       |
    | ACTIONS
    |       |
    |       | Appel afficheMessage()
    |       |
FIN_PGME
```

EXEMPLE 2

Ecrire un SP aireRectangle qui calcule l'aire d'un rectangle, puis écrire le PP adéquat.

1. Faire la vue externe
2. Type de SP ? Pourquoi
3. Faire entête et corps du SP
4. Faire PP

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  FONCTION aireRectangle: reel (reel : l,L)
   |   |    reel : aire
   |   |    aire ←l*L
   |   |    retourne aire
   |  FIN_FCT
DEBUT_PGME PRINCIPAL
    | DECLARATIONS
    |      | reel : larg, long, aire1
    | ACTIONS
    |      | ecrire_ecran "donner larg et long"
    |      |aire1 ←aireRectangle( larg, long )
    |      | ecrire_ecran "aire = ", aire1
    |      |
FIN_PGME
```

EXEMPLE 3

Ecrire un SP lireDimensions qui lit au clavier 2 réels et les retourne au Prog. Princ., puis écrire le Prog Princ qui l'appelle.

1. Faire la vue externe
2. Type de SP ? Pourquoi
3. Faire entête et corps du SP
4. Faire PP

```
SOUS-PROGRAMMES
   |  % déclaration des SP
   |  PROCEDURE lireDimensions(reel : x,y )
   |    |  ecrire_ecran "donner 2 réels"
   |    |  lire_clavier x,y
   |  FIN_PROC
DEBUT_PGME PRINCIPAL
    | DECLARATIONS
    |       | reel : Ax, Ay
    | ACTIONS
    |       |
    |       | Appel lireDimensions(Ax,Ay)
    |       |
FIN_PGME
```

EXEMPLE 4

Faire un PP qui demande la valeur de 2 réels, calcule l'aire du rectangle à partir de ces deux valeurs, affiche le résultat puis « bonne journée » en utilisant les SP précédent.

1. Faire la vue externe
2. Type de SP ? Pourquoi
3. Faire entête et corps du SP
4. Faire PP

```
SOUS-PROGRAMMES
   |   % déclaration des SP
   |    %déjà fait
DEBUT  PGME PRINCIPAL
   |  DECLARATIONS
   |       |   reel : lo, la, aire
   |   ACTIONS
   |       |
   |       | Appel lireDimensions( lo, la)
   |       | aire ←aireRectangle(lo, la)
   |       | ecrire_ecran « l'aire vaut » , aire
   |       | Appel afficheMessage()
FIN  PGME
```