Sous-programmes

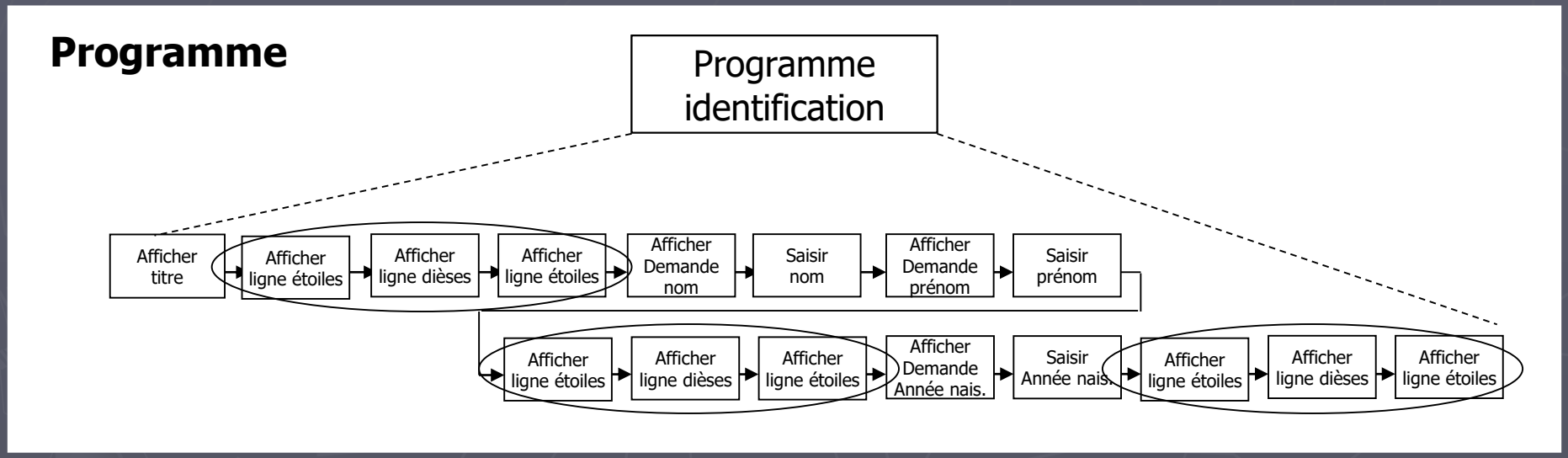
Introduction

- Sous-programme = traitement particulier appelé à s'exécuter à l'intérieur d'un autre programme
- Utilisation :
  - **quand un même traitement doit être réalisé plusieurs fois** dans un programme. On écrit un sous-programme pour ce traitement et on l'appelle à chaque endroit où l'on en a besoin
  - **pour organiser le code** , améliorer la conception et la lisibilité des gros programmes
- Il existe des sous-programmes prédéfinis pouvant être utilisés directement dans n'importe quel programme : **les librairies**

Sous programme

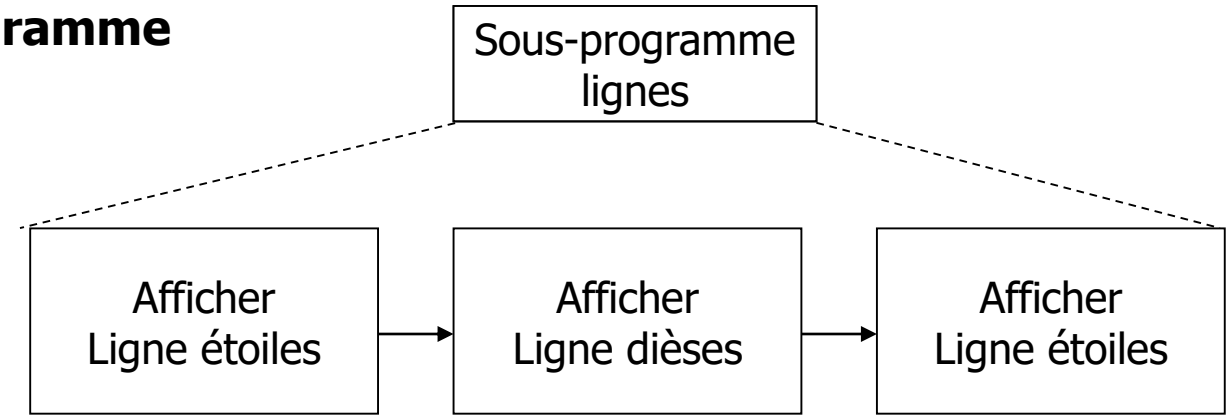
```
Programme d'identification
**********
##########
**********
Veuillez saisir votre nom
durand
Veuillez saisir votre prenom
alain
**********
##########
**********
veuillez saisir votre annee de naissance
1943
**********
##########
**********
```

Sous programme

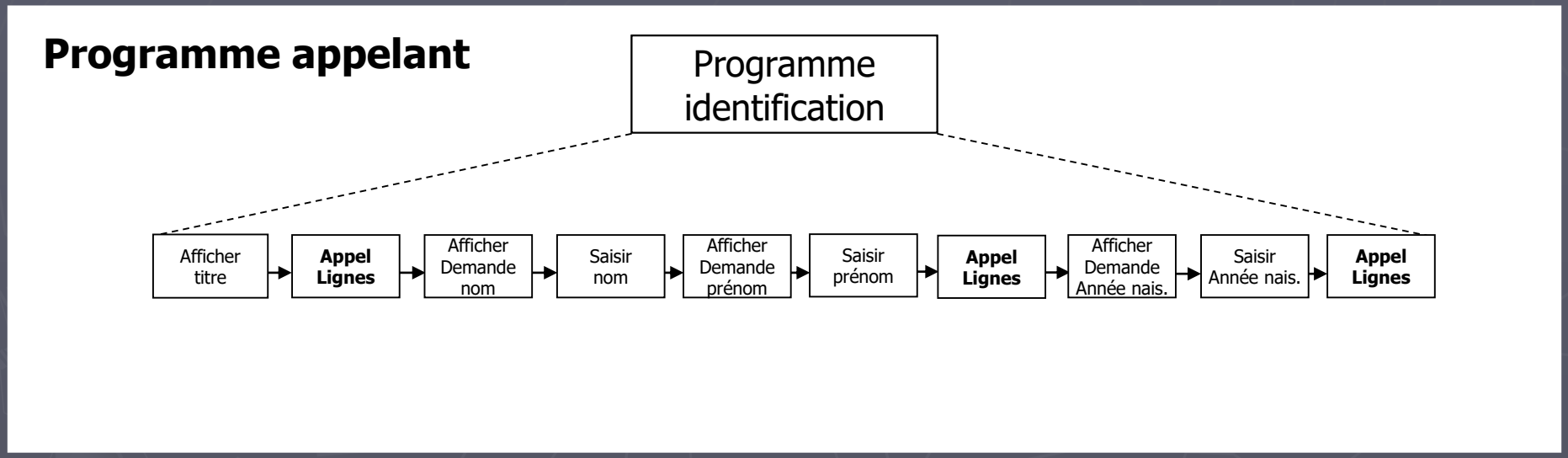


Sous programme

Sous-programme



Sous programme



Fonctions et procédures

- 2 sortes de sous-programmes :
  - les ***fonctions***
  - les ***procédures***
- L'appel d'une fonction est une expression, tandis que l'appel d'une procédure est une instruction :
  - une **fonction renvoie un résultat**
  - une **procédure ne renvoie rien**

Procédures

- Une procédure est un ensemble d'instructions regroupées sous un nom, qui réalise un traitement particulier dans un programme lorsqu'on l'appelle
- Comme un programme, une procédure possède
  - un nom
  - des variables
  - des instructions
  - un début
  - une fin

Syntaxe définition

- **DEFINITION** d'une procédure :

```
Procédure maProcedure( )
//déclaration des variables
Var var1 : entier
Début
/*
instructions 1
instructions 2
*/
FinProc
```

En-tête

Corps

Exemple

```
Procédure lignes( )
Var etoile, diese : chaine
Début
etoile <- «**********»
diese <- «##########»
Afficher etoile
Afficher diese
Afficher etoile
FinProc
```

Appel d'une procédure

- Pour déclencher l'exécution d'une procédure dans un programme, il suffit de l'appeler :
  - indiquer son nom suivi de parenthèses

```
PROGRAMME monProgr
VAR       varEntier1 : entier
          varChaine : chaîne
DEBUT
/*instuctions*/
maProcedure()
/*instuctions*/
FIN
```

Exemple

```
Programme id
Var   nom, prenom: chaine
       anneeNaissance : entier
Début
Afficher « Programme d'identification »
lignes()
Afficher « veuillez saisir votre nom »
Saisir nom
Afficher « veuillez saisir votre prenom  »
Saisir prenom
lignes()
Afficher « veuillez saisir votre annee de naissance »
Saisir anneeNaissance
lignes()
Fin
```

Exécution

- Appel d'une procédure
  1. arrêt momentané de l'exécution du programme appelant
  2. Exécution des instructions de la procédure
  3. Reprise de l'exécution du programme appelant là où il s'était arrêté (instruction suivant l'appel)
- Une procédure peut être appelée soit par un programme, soit par un autre sous-programme (qui lui même a été appelé)
- Les appels de sous-programmes peuvent s'imbriquer autant de fois que cela est utile

Exécution

```
Programme id
Var     nom, prenom: chaine
         anneeNaissance : entier
Début
Afficher « Programme d'identification »
ligneEtoile ()
Afficher « veuillez saisir votre nom »
Saisir nom
Afficher « veuillez saisir votre prenom »
Saisir prenom
ligneEtoile ()
Afficher « veuillez saisir votre annee de naissance »
Saisir anneeNaissance
ligneEtoile ()
Fin
```

```
Programme d'identification
**********
##########
**********
Veuillez saisir votre nom
durand
Veuillez saisir votre prenom
alain
**********
##########
**********
veuillez saisir votre annee de naissance
1943
**********
##########
**********
```

```
etoile <- «**********»
diese <- «##########»
Afficher etoile
Afficher diese
Afficher etoile
```

```
etoile <- «**********»
diese <- «##########»
Afficher etoile
Afficher diese
Afficher etoile
```

Variables locales

- Les variables déclarées dans une procédure ne sont pas utilisables dans le programme appelant
- Les variables déclarées dans le programme appelant ne sont pas utilisables dans les procédures
- Chaque programme et sous-programme a son propre espace de variables, inaccessible par les autres
- les variables sont dites **LOCALES**
- Il peut exister des variables globales, mais leur usage est déconseillé (et donc sanctionné)

Paramètre

- Comment sous-programmes et programmes vont pouvoir communiquer des données ?
  - Grâce aux paramètres
  - Grâce aux messages
  - Grace au réseau
  - …
- Un paramètre est une variable particulière qui sert à la communication entre programme appelant et sous-programme.

Paramètre

- Un paramètre est une variable particulière qui sert à la communication entre programme appelant et sous-programme.
- Elle a un nom et un type

Syntaxe définition

- **DEFINITION** d'une procédure avec paramètre :

```
Procédure maProcedure(donnée monParamètre : entier)
//déclaration des variables locales
Var var1 : entier
Début
/*
instructions avec monParamètre
instructions avec var1
*/
FinProc
```

Paramètre formel

Exemple

```
Procédure lignes(donnée nbColonnes : entier)
Var          cpt : entier
Début
Pour cpt de 1 à nbColonnes Faire
   Afficher '*'
FinPour
Pour cpt de 1 à nbColonnes Faire
   Afficher '#'
FinPour
Pour cpt de 1 à nbColonnes Faire
   Afficher '*'
FinPour
FinProc
```

Appel avec paramètre

- Pour appeler une procédure avec un paramètre, il faut mettre la valeur du paramètre entre parenthèse

```
PROGRAMME monProgr
VAR         varEntier1 : entier
            varChaine : chaîne
DEBUT
/*instuctions*/
maProcedure(varEntier1)
maProcedure(3)
/*instuctions*/
FIN
```

Exemple

```
****************************************
########################################
****************************************
Programme d'identification
******************
##################
******************
Veuillez saisir votre nom
durand
Veuillez saisir votre prenom
alain
******************
##################
******************
veuillez saisir votre annee de naissance
1943
****************************************
########################################
****************************************
```

Programme appelant

```
Programme id
Const   largeur <- 20 : entier
Var     nom, prenom: chaine
        anneeNaissance : entier
Début
lignes(largeur * 2, 255,0,0)
Afficher « Programme d'identification »
lignes(largeur, 255,255,255)
Afficher « veuillez saisir votre nom »
Saisir nom
Afficher « veuillez saisir votre prenom  »
Saisir prenom
lignes(largeur, 255,255,255)
Afficher « veuillez saisir votre annee de naissance »
Saisir anneeNaissance
lignes(largeur * 2, 255,0,0)
Fin
```

Paramètre tableau

```
Procédure affTab(donnée tab : tableau[1..10] de réels)
Var   i : entier
Début
Pour i de 1 jusqu'à 10 Faire
   Afficher tab[i]
FinPour
FinProc
```

Paramètre tableau

```
Programme tableau

Var   monTab : tableau[1..10] de réels
      i : entier
Début
Afficher « Saisir 10 réels »
Pour i de 1 jusqu'à 10 Faire
   Saisir monTab[i]
FinPour
Afficher « les valeurs saisies sont »
affTab(monTab)
Fin
```

Plusieurs paramètres

- Une procédure peut avoir plusieurs paramètres :

```
Procédure maProcedure(donnée Par1 : entier, par2 : réel)
```

- Le nombre et l'ordre des paramètres à l'appel doit être exactement le même

```
maProcedure(varE, varR)
```

Exemple : définition

```
Procédure lignes(donnée nbetoiles, nbDiese : entier)
Var          cpt : entier
Début
Pour cpt de 1 jusqu'à nbetoiles Faire
   Afficher '*'
FinPour
Pour cpt de 1 jusqu'à nbDiese Faire
   Afficher '#'
FinPour
Pour cpt de 1 jusqu'à nbetoiles Faire
   Afficher '*'
FinPour
FinProc
```

Exemple

```
**************************************
##################
**************************************
Programme d'identification
*******************
##################
*******************
Veuillez saisir votre nom
durand
Veuillez saisir votre prenom
alain
*******************
##################
*******************
veuillez saisir votre annee de naissance
1943
**************************************
##################
**************************************
```

Exemple : programme appelant

```
Programme id
Const   largeur <- 10 : entier
Var     nom, prenom: chaine
        naissance : entier
Début
lignes(largeur * 2, largeur)
Afficher « Programme d'identification »
lignes(largeur, largeur)
Afficher « veuillez saisir votre nom »
Saisir nom
Afficher « veuillez saisir votre prenom  »
Saisir prenom
lignes(largeur , largeur)
Afficher « veuillez saisir votre annee de naissance »
Saisir naissance
lignes(largeur * 2 , largeur)
Fin
```

Fonctionnement

- Lors de l'appel de la procédure, la valeur du paramètre effectif passée en argument est copiée dans le paramètre formel (qui est une variable locale)
- La procédure effectue alors le traitement avec la variable
- La procédure n'utilise pas directement la variable mise en paramètre effectif : elle utilise sa valeur, qu'elle a recopiée dans sa propre variable locale (le paramètre formel)

Pièges

- Il est primordial de bien distinguer :
  - les paramètres **formels** qui se trouvent dans l'en-tête d'une procédure lors de sa définition
  - et les paramètres **effectifs** (ou arguments) qui sont placés entre parenthèses lors de l'appel

Paramètres formels

- placés dans la **définition** d'une procédure
- servent à **décrire** le traitement à réaliser par la procédure indépendamment des valeurs traitées
- Ce sont des variables locales à la procédure
- ils sont déclarés dans l'entête de la procédure

Paramètres effectifs

- placés dans **l'appel** d'une procédure
- Lors de l'appel, leur valeur est recopiée dans les paramètres formels correspondants
- Un paramètre effectif en donnée peut être
  - soit une variable du programme appelant (nomVar)
  - soit une valeur littérale (3)
  - soit le résultat d'une expression (5 * x)

Les Fonctions

- Une fonction est un ensemble d'instructions regroupées sous un nom, qui réalise un traitement particulier dans un programme lorsqu'on l'appelle et retourne un résultat au programme appelant
- Comme un programme et une procédure, une fonction possède :
  - un nom
  - des variables
  - des instructions
  - un début
  - une fin

Les Fonctions

- Sous-programmes retournant un et un seul résultat au programme appelant
- Les fonctions sont appelées pour récupérer une valeur
  (alors que les procédures ne renvoient aucune valeur)

- L'appel des fonctions est différent de l'appel des procédures :
  - L'appel d'une fonction doit **obligatoirement** se trouver à l'intérieur d'une instruction qui utilise sa valeur
  - Le résultat d'une fonction doit obligatoirement être retourné au programme appelant par l'instruction Retourne

Syntaxe définition

- **DEFINITION** d'une fonction :

```
Fonction maFonction(/*paramètres*/) : type
//déclaration des variables
Var maVar : entier
Début
/*instructions*/
Retourne maVar
FinFonction
```

En-tête

Corps

Exemple

```
FONCTION calSomN (Donnée e : entier) : entier
VAR  res, i : entier

DEBUT
res <- 0
Si e > 0
alors
   Pour i de 1 à e Faire
      res <- res + i
   FinPour
Finsi
Retourne Res
FINFONCTION
```

Instruction de retour

Appel de fonction

```
PROGRAMME monProgr
VAR       varE, varR : entier
          varChaine : chaîne
DEBUT
/*instuctions*/
varR <- maFonction(varE)
varR <- maFonction(5)
varR <- maFonction(3 * varE)
/*instuctions*/
FIN
```

Exemple

```
Programme sommesDesN1erEntiers

Var     x, somme : entier
        rep : caractère

Début
Répéter
   Afficher « Entrez un entier > 0 »
   Saisir x
   somme <- calSomN (x)
   Afficher «la somme des », x, « 1er entiers est », somme
   Afficher «voulez-vous continuez ? (O/N)»
   Saisir rep
Jusqu'à rep = 'N'
Afficher « au revoir »
Fin
```