UNIVERSITÉ FRANÇOIS - RABELAIS TOURS

**UFR Sciences et Techniques**
**Licence S&T 1ère année**

# Informatique — UE 102

*Architecture des ordinateurs et Algorithmique de base*

Jean-Yves Antoine

http://www.info.univ-tours.fr/~antoine/

UNIVERSITÉ FRANÇOIS - RABELAIS TOURS

UFR Sciences et Techniques
Licence S&T 1ère année

# Informatique UE 102

## Chap V — Sous-programmes : fonctions et itérations

*Où l'on apprend à appliquer le vieil adage du « diviser pour régner » en décomposant un problème en sous-problèmes*

# SOUS-PROGRAMMES

## DEFINITION

Sous-partie d'un programme qui fait sens en elle-même

⇨exemple : recherche des nombres parfaits à N (vu en TD)

- **programme principal** : gère la saisie clavier de N, les appels aux sous-programme pour réaliser la recherche et l'affichage des résultats
- **sous-programme recherche** : gère la boucle de test de tous les entiers compris entre 1 et N
- **sous-programme verif** : gère la vérification qu'un nombre donné est premier

## UTILITE

- Simplification de l'analyse (analyse descendante en sous-problème)
- Modularité
- Réutilisabilité du code
- Permet une programmation récursive *(cf. cours semestre 2)*

# SOUS-PROGRAMMES

Deux grandes classes de sous-programmes (en Pascal comme dans la plupart des autres langages impératifs)

## FONCTIONS

- Retourne un résultat en fonction de paramètres passés à la fonction
- En Turbo Pascal, le résultat ne peut être qu'un type simple

⇨ **Exemple** : fonction prédéfinie dans Pascal `racine := sqrt(nb);`

## PROCEDURES

- Travaille par effet de bord : modifie directement une ou plusieurs des variables passées en paramètre (passage par **référence**)
- Ne retourne rien

⇨ **Exemple** : procédure prédéfinie dans Pascal `readln(nn);`

# SOUS-PROGRAMMES : SYNTAXE PASCAL

- Constitue un programme en soi : **en-tête** de déclaration du sous-programme et de ses paramètres
- **Corps** de sous-programme entre `begin` et `end`.
- Définition dans la zone des déclarations du programme principal

```
program prog_ppal;
const …
var …
```

**Déclarations sous programmes**

```
begin
(* corps avec appels *)
end.
```

**Entête ss programme**

```
begin

(* corps ss programme *)

end;
```

# SOUS-PROGRAMMES : SYNTAXE PASCAL

## EN-TÊTE FONCTION

```
function nom(par1: type ; …; parN: type N) : typef;

const …    (* declarations locales à la fonction *)
var …
```

## EN-TÊTE PROCEDURE

passage par valeur

passage par référence

```
procedure nom(par1: type ; …; var parN: type N);

const …   (* declarations locales à la procedure *)
var …
```

Il est possible d'imbriquer des sous-programmes : déclaration d'un « sous-sous-programme » dans un sous-programme

# VARIABLE LOCALE / VARIABLE GLOBALE

```
program tout_solde;
const NB_PROD = 100;
var prix : array[1 .. NBPROD] of real;
    i : integer;
function solde(avt: real;reduc: real): real;
   var var_inutile : real;
   begin
       var_inutile := avt * reduc;
       solde := var_inutile;
   end;
(* program principal *)
begin
   …
   for i:=1 to NBPROD do
       prix[i] := solde(prix[i],0.5);
end.
```

variables globales

variable locale

Affectation du résultat

# PORTEE DES VARIABLES

## VARIABLE LOCALE

- Une variable locale n'a d'existence que dans le corps du sous-programme auquel elle appartient
- Une variable locale est considérée comme … globale dans un sous-programme qu'elle englobe

## VARIABLE GLOBALE

- Une variable globale a une existence dans le programme où elle est déclarée et les sous-programmes que ce dernier englobe

## CONFLITS DE PORTEE ET REGLES DE (BON) GENIE LOGICIEL

- Si une variable globale et locale on le même nom, c'est l'interprétation au titre de la variable locale qui prime dans le sous-programme ⇨ **éviter de donner des noms identiques** !
- Éviter d'utiliser directement une variable globale dans un sous-programme : **passage par paramètre**

# CONFLITS DE PORTEE

## EXEMPLE 1

```
program tout_solde_foo;
var prix : real;
    i : integer;
function solde(prix: real;reduc: real): real;
   var var_inutile : real;
   begin
       solde := prix * reduc;
   end;
(* program principal *)
begin
       prix := solde(prix,0.5);
end.
```

# CONFLITS DE PORTEE

## EXEMPLE 2

```
program tout_solde_foo;
var prix : real;
    i : integer;
function solde(avt : real;reduc: real): real;
   var prix : real;
   begin
        prix := avt * reduc;
        solde := prix ;
   end;
(* program principal *)
begin
      prix := solde(prix,0.5);
end.
```

# CONFLITS DE PORTEE

## EXEMPLE 3

```
program tout_solde_foo;
var prix : real;
    i : integer;
function solde(avt : real;reduc: real): real;
   begin
        solde := prix * reduc;
   end;
(* program principal *)
begin
      prix := solde(prix,0.5);
end.
```