ECOLE SUPÉRIEURE DE LA STATISTIQUE
ET DE L'ANALYSE DE L'INFORMATION

CHAPITRE II : LES PROCÉDURES ET LES FONCTIONS

AÏCHA EL GOLLI
aicha.elgolli@essai.ucar.tn
Novembre-décembre 2022

PROCÉDURES ET FONCTIONS

Dès qu'on commence à écrire des programmes importants, il devient difficile d'avoir une vision globale sur son fonctionnement et de traquer les erreurs, Que faire ?

décomposer le problème en sous problèmes et trouver une solution à chacun puis regrouper le tout dans un seul algorithme.

En Algorithmique, chaque solution partielle donne lieu à un sous-algorithme (fonctions et les procédures) qui fera partie d'un algorithme complet pour pouvoir être exécuté.

En programmation, les fonctions et les procédures sont des sous-pgs qui sont appelés par le corps principal du code et permettent de décomposer un programme complexe en plusieurs parties plus simples et relativement indépendantes.

Ceci a de nombreux avantages :

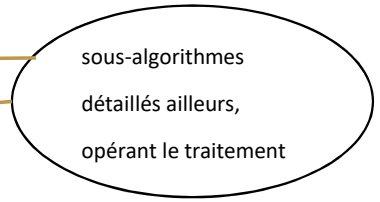
- ✓ Chaque fonction peut être **implémentée et testée séparément** ;
- ✓ Si le même traitement est nécessaire en plusieurs point du programme, on se contente d'**appeler** la fonction correspondante **plusieurs fois** au lieu de **dupliquer le code source** ;
- ✓ Une même fonction peut être utilisée dans **plusieurs programmes différents**, grâce au concept de **bibliothèque**
- ✓ avoir un **code clair** et **lisible**

IDENTIFIER LE RÔLE D'UN BLOC D'INSTRUCTIONS

```
Algorithme Puissances
variables
uneVal, puissance : réels ;
cpt, nbPuissances : entiers ;
début
afficher ("Donnez un nombre réel quelconque : ") ;
entrer (uneVal) ;
afficher ("Combien de puissances successives souhaitez-vous ? ") ;
entrer (nbPuissances) ;
{afficher 7 étoiles * et passer à la ligne}
pour (cpt ← 1 à 7) faire
afficher ("*");
fpour
afficher ("\n");
{calcul des nbPuissances puissances successives de uneVal}
puissance← 1 ;
pour (cpt← 1 à nbPuissances) faire
puissance← puissance × uneVal ;
afficher ("La", cpt, "ième puissance de", uneVal, "est", puissance ) ;
fpour
fin
```

EN SIMPLIFIANT L'ÉCRITURE…

```
Algorithme Puissances
variables
uneVal, puissance : réel ;
nbPuissances : entier ;
début
afficher ("Donnez un nombre réel quelconque : ") ;
entrer (uneVal) ;
afficher ("Combien de puissances successives souhaitez-vous ? ") ;
entrer (nbPuissances) ;
{appel à la procédure etoiles}
etoiles (7);
{appel à la fonction calculPuissances}
puissance← calculPuissances (uneVal, nbPuisssances) ;
afficher ("La", cpt, "ième puissance de", uneVal, "est", puissance) ;
fin
```



SOUS-ALGORITHMES (PROCÉDURE, FONCTION)

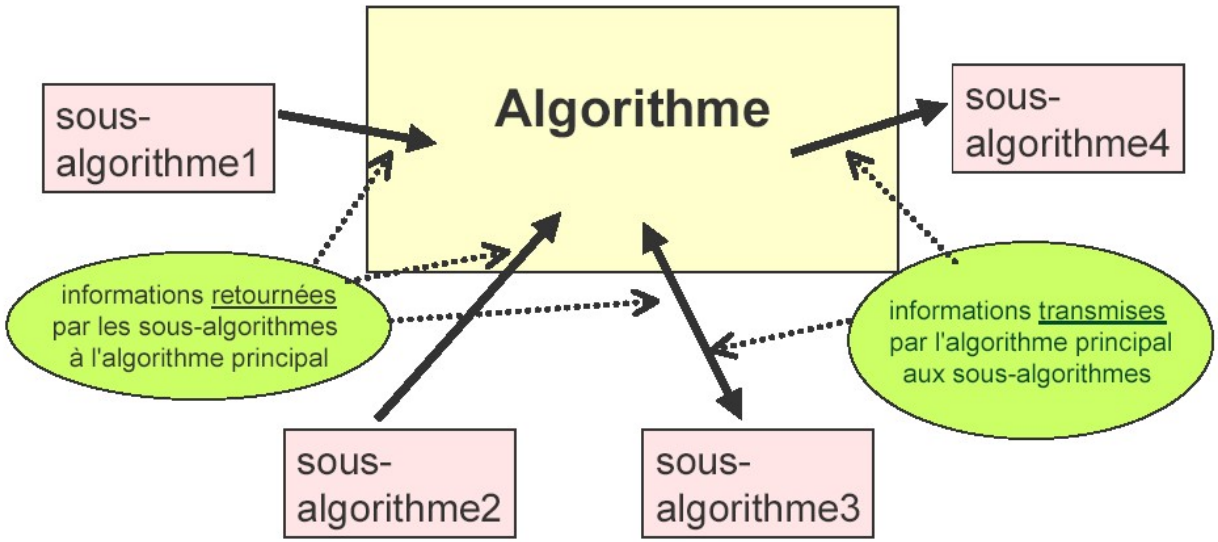
Sous-algorithmes

- Un algorithme appelle un sous-algorithme : cet algorithme passe "momentanément" le contrôle de l'exécution du traitement au sous-algorithme.
- Un sous-algorithme est conçu pour faire un traitement bien défini, bien délimité, si possible indépendamment du contexte particulier de l'algorithme appelant.

Remarque : un sous-algorithme peut en appeler un autre.

CIRCULATION DES INFORMATIONS

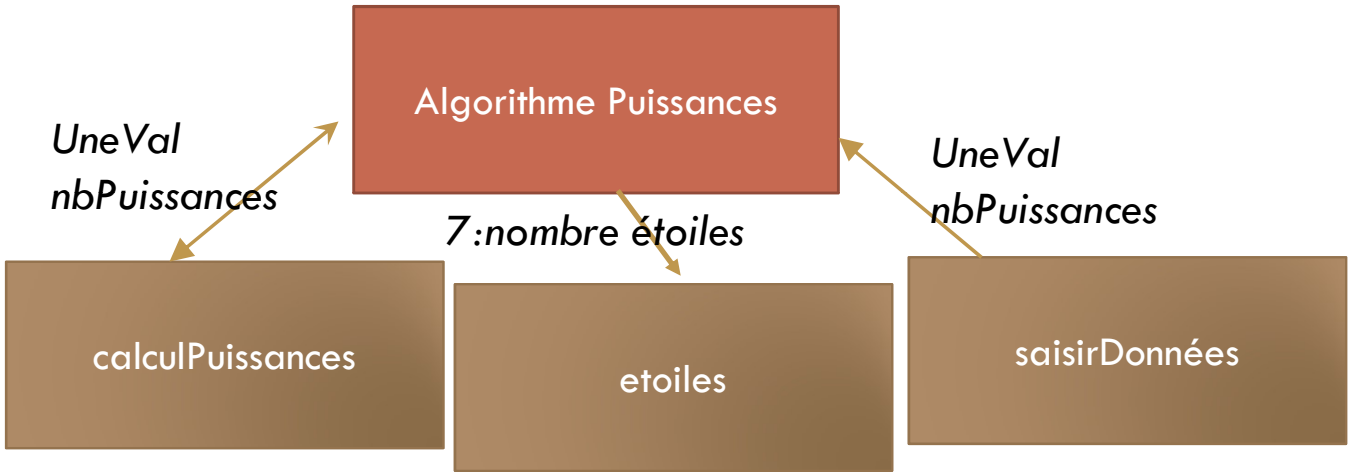
Comment les informations circulent...



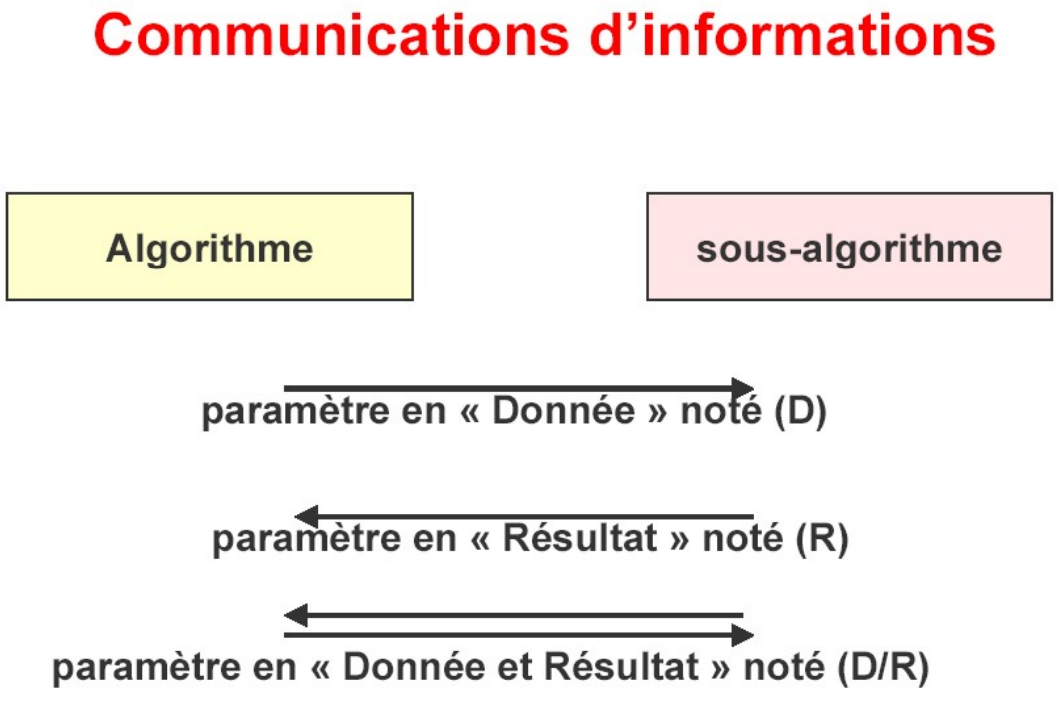
EN SIMPLIFIANT L'ÉCRITURE…

```
Algorithme Puissances
 variables
 uneVal, puissance : réel ;
 nbPuissances : entier ;
début
 saisirDonnées(uneVal, nbPuissances)
 etoiles (7);
 puissance← calculPuissances (uneVal, nbPuisssances) ;
 afficher ("La", cpt, "ième puissance de", uneVal, "est", puissance) ;
fin
```

CIRCULATION DES INFORMATIONS



TYPES DE PARAMÈTRES



passage de paramètre se fait par l'utilisation aussi par le mot-clé **Var**

PARAMÈTRES/ARGUMENTS


```
Procédure saisirDonnées(r val : reel, r nb : entier)
début
....
fin
```

paramètres de la procédure

```
Algorithme Puissances
variables
uneVal, puissance : réel ;
nbPuissances : entier ;
début
saisirDonnées(uneVal, nbPuissances)
etoiles (7);
puissance← calculPuissances (uneVal, nbPuisssances) ;
afficher ("La", cpt, "ième puissance de", uneVal, "est", puissance) ;
fin
```

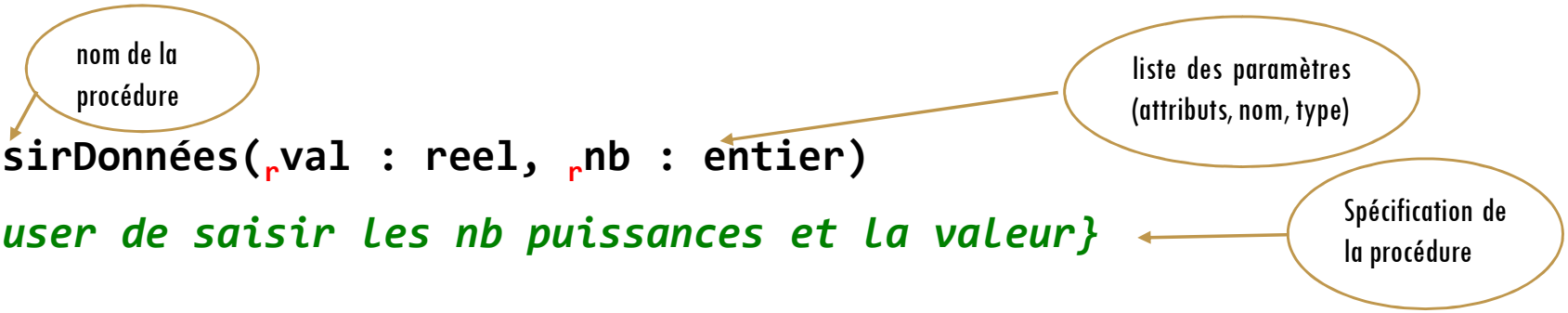
Arguments ou param effectifs de l'appel de la procédure

```
Procédure etoiles(n : entier)
début
....
fin
```

PARAMÈTRES ET ARGUMENTS

- Le nombre de paramètres doit correspondre au nombre d'arguments.
- Le type du $k^{ème}$ argument doit correspondre au type du $k^{ème}$ paramètre.
- Le nom du $k^{ème}$ argument a, de préférence, un nom différent de celui du $k^{ème}$ paramètre.
- Un paramètre défini en "Donnée" **doit** correspondre à un argument qui possède une valeur dans l'algorithme appelant au moment de l'appel.
- Un paramètre défini en "Résultat" doit recevoir une valeur dans le sous-algorithme.

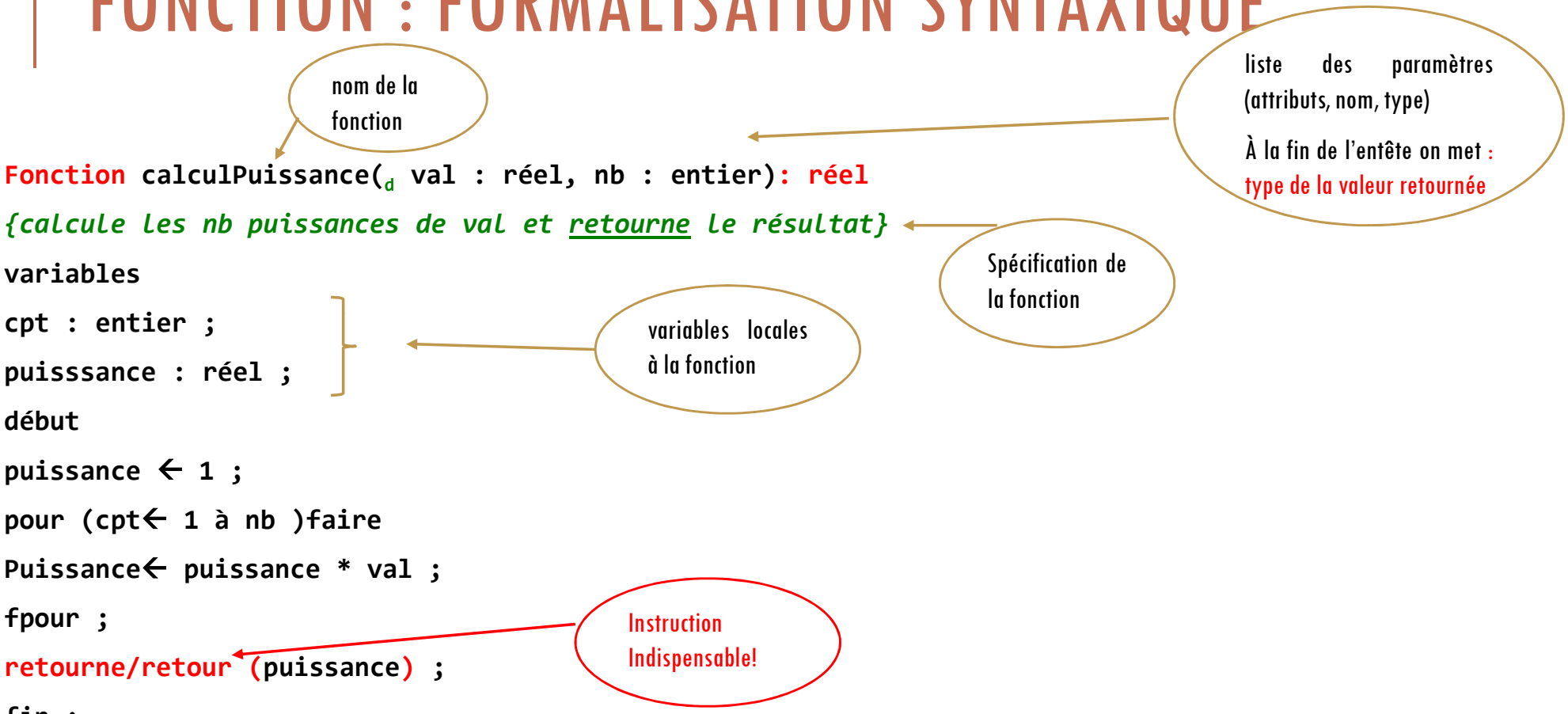
PROCÉDURE : FORMALISATION SYNTAXIQUE




```
Procédure saisirDonnées(r val : reel, r nb : entier)
 {permet à l'user de saisir les nb puissances et la valeur}
début
afficher ("Donnez un nombre réel quelconque : ") ;
entrer (val) ;
afficher ("Combien de puissances successives souhaitez-vous ? ") ;
entrer (nb) ;
fin
```

```
Procédure etoiles(d n: entier)
début
pour (cpt ← 1 à n) faire
afficher ("*");
fpour
afficher ("\n");
fin
```

FONCTION : FORMALISATION SYNTAXIQUE



COMPARAISON : FONCTIONS ET PROCÉDURES

- **Définition:**

```
Procédure surf1(long : réel, larg : réel, r aire :réel)
début
aire←  long × larg ;
fin
```

```
Fonction surf2(long :réel, larg : réel) : réel
variable aire : réel ;
début
aire←  long × larg ;
retourne (aire) ;
fin
```

Utilisation (appel):

```
Algorithme Exemple
variables surface, longueur, largeur : réel ;
début
longueur ←  10 ; largeur ←  25 ;
surf1(longueur, largeur, surface) ; {utiliser surf1
surface ←  surf2(longueur, largeur) ; ou surf2}
fin
```

CARACTÉRISTIQUES DES DEUX CLASSES DE SOUS-ALGORITHMES (PROCÉDURE/FONCTION)

- Une procédure, comme une fonction, est appelée dans un algorithme/sous-algo par son nom suivi de la liste des arguments correspondant à sa liste de paramètres.
- Lors de l'appel d'une fonction, la valeur retournée doit être exploitée dans une instruction (affectation, expression arithmétique ou logique, affichage, ...).
- Dans une fonction, la valeur retournée explicitement remplace le paramètre d'attribut résultat (R) auquel on veut donner une importance particulière.
- Dans une procédure comme dans une fonction, la liste des paramètres peut être vide.

RAPPEL ALGORITHMIQUE

Définition Procédure

Procédure Nom_de_la_procédure (pf1 : type 1 , pf2 : type 2 , …)
Les déclarations des variables et constantes locales

DÉBUT

Instructions de la procédure

FIN

Les paramètres peuvent être donnée (d) ou donnée/résultat (d/r)

Définition Fonction

Fonction Nom_de_la_fonction (pf1 : type 1 , pf2 : type 2 , …): **type**
Les déclarations des variables et constantes locales

DÉBUT

Instructions de la fonction

retourne Résultat

FIN

Les paramètres peuvent être donnée (d) ou donnée/résultat (d/r)

Appel:
Nom_de_la_fonction/procédure (<liste des paramètres effectifs>) ;

EXERCICE

```
Procédure ESSAI (i : entier )

DÉBUT
i ← 3 * i ;

Écrire ( "Le paramètre i =" , i ) ;
FIN
```

```
Procédure ESSAI (d/r i : entier )

DÉBUT
i ← 3 * i ;
 Écrire ( "Le paramètre i =" , i ) ;
FIN
```

ALGORITHME PASSAGE
VAR

x : entier ;
DÉBUT

x←3;

ESSAI (x) ; {*Appel de la procédure ESSAI*}

Ecrire ("Après appel x = " , x) ;
FIN

EXEMPLE

```
Algorithme calculNotes
{calcule la moyenne des notes saisies }
variables note, somme, moyenne : réel ;
nbnotes : entier ; ok : booléen ;
début
somme← 0 ; nbnotes←  0 ; {initialisations}
afficher ("Donnez la première note (note négative pour terminer)") ;
entrer(note) ;
tant que (note >=0) faire {traitement : saisies des notes et calcul du total}
ajouterNote(note, somme, nbnotes) ;
afficher ("Donnez la note suivante (note négative pour terminer) " ) ;
entrer(note) ;
ftq
ok←  calculMoyenne(somme,nbnotes,moyenne) ; {calcul moyenne}
si (ok) {affichage résultat}
alors afficher("La moyenne des ", nbnotes, "notes est", moyenne) ;
sinon afficher("Pas de notes, pas de moyenne!") ;
fin
```

```
Procédure ajouterNote(uneNote :réel , d/r laSomme : réel, d/r nb :entier)
{Ajoute uneNote à laSomme et incrémente nb}
début
LaSomme←laSomme+uneNote;
nb←nb+1;
fin
```

```
Fonction calculMoyenne(laSomme:réel, nb:entier, d/r
laMoyenne:réel):booléen
{Calcule la moyenne des nb notes dont le total est laSomme; retourne vrai
si calcul possible,faux sinon}
variable
Début
Si (nb<>0) alors
     laMoyenne←laSomme/nb;
     Retourne vrai;
Sinon
     laMoyenne←0,0;
     Retourne faux;
FinSI;
fin
```

Algorithme UnTest

variables a,b,c : **entiers**
début
a ← 1 ;
b ← 2 ;
afficher ("Entrée : (a, b) = (", a , ", ", b , ")") ;
test1(a,b,c) ;
afficher ("Sortie : (a, b, c) = (", a , ", ", b , ", ", c, ")") ;
fin

Procédure test1(x : entier, ***d/r*** y :entier, ***r*** z :entier)
début
x ← x + 1 ;
y ← y + 1 ;
z ← test2(x , y) ;
afficher ("fin test1 : (x, y, z) = (",x , ", ", y , ", ", z , ")") ;
fin

Fonction test2(v1 : entier, ***d/r*** v2 : entier) : entier
variable rés : **entier ;**
début
afficher ("début test2 : (v1, v2) = (", v1, ", ", v2, ")") ;
v1 ← v1 + 1 ;
v2 ←v2+ 1**;**
rés ← v1 + v2 ;
retour (rés) ; **fin**

DÉFINITION D'UNE FONCTION EN C

```
<TypeRés> NomFonct (<TypePar1> <NomPar1>, <TypePar2> <NomPar2>, ...)
 {
      <liste des variables locales>
       <instructions>
      return résultat ;
  }
```

Appel:
Nom_de_la_fonction (<liste des paramètres effectifs>) ;

DÉFINITION D'UNE PROCÉDURE EN C

```
Void NomProc (<TypePar1> <NomPar1>, <TypePar2> <NomPar2>, ...)   {
     <liste des variables locales>
      <instructions>
  }
```

Appel:
Nom_de_la_procédure (<liste des paramètres effectifs>) ;

EXEMPLE

```
int produit(int x, int y)
{ return (x*y) ;}
Void echange( int, int );
void main()
{       int n1, n2;
        printf("donnez un entier ") ;
        scanf("%d",&n1);
        printf("donnez un entier ") ;
        scanf("%d",&n2);
        printf("le produit de %d et %d vaut : %d\n", n1, n2, produit(n1, n2)) ;
        echange(n1,n2);
        printf("n1 vaut %d et n2 vaut %d \n", n1, n2) ;
}
void echange(int a, int b){
int tmp= a;
a=b;
b=tmp;
printf("a vaut %d et b vaut %d \n", a, b) ;
}
```

Il est demandé **d'écrire un algorithme** qui demande à l'utilisateur d'entrer une valeur entière positive appelée (Valeur) puis:

a. qui indique à l'utilisateur si Valeur est un nombre à 2 chiffres,
b. qui affiche la factorielle de Valeur,
c. et qui saisit des valeurs au nombre de Valeur puis affiche la plus grande valeur saisie.

Utilisez les procédures et fonctions

Déposez votre algorithme avec ses procédures et fonctions sur le classroom en version numérique dans un fichier texte et ce au maximum le mercredi à 23h59

VARIABLES GLOBALES

Le programme suivant ne se compile pas car on essaye d'accéder la variable a de la fonction main() depuis la fonction test().

```
#include <stdio.h>

void test ()

{  printf ("a = %d\n", a) ;}


void main ()

{ int a ;

  a = 5;

  test ();}
```

VARIABLES GLOBALES

L'exemple suivant en revanche est correct :

```
#include <stdio.h>
int a;
void test ()
{ printf ("a = %d\n", a) ;}

void main ()
{ a = 5 ;
  test () ;}
```

LES VARIABLES LOCALES

Les variables locales ne sont accessibles qu'à l'intérieur de la fonction dans laquelle elles sont déclarées. Elles n'ont aucun rapport avec des variables globales du même nom.

Exemple

```
#include <stdio.h>

int n ;

void fonct ( )

{  int p ;

  int n = 5;

  p = 4;

/* ici on fait référence à la variable locale n */

  printf ("n = %d\n", n) ;

}
```

```
void main ()
{  int p = 7;
  n = 3;
  printf ("p = %d\n", p) ;
  fonct();}
```

affichage:

p = 7

n = 5