Chapitre 5
Procédures, fonctions et modes de passage de paramètres

Objectifs pédagogiques du chapitre

Objectif général

A la fin du chapitre, l'étudiant est capable :

- d'écrire une procédure ou/et une fonction pour mieux rédiger un algorithme,
- d'analyser une procédure ou/et une fonction dans un algorithme.

Objectifs pédagogiques du chapitre

Objectifs spécifiques

A la fin du chapitre, l'étudiant est capable :

- de décrire des raffinements successifs
- de décrire correctement une procédure ou une fonction
- d'utiliser correctement une procédure ou une fonction
- d'utiliser correctement les principaux modes de passage des paramètres

Contenu

- Introduction aux raffinements
- Description d’une procédure
- Utilisation d’une procédure
- Description d’une fonction
- Utilisation d’une fonction
- Principaux modes de passage de paramètres

Introduction aux raffinements

- L'écriture d'algorithmes sur des dizaines voire des centaines de pages n'est pas aisée si l'on n'utilise pas l'**approche descendante** s'appuyant sur les raffinements successifs qui donnent la possibilité d'avoir une vision globale de l'algorithme.
- Chaque raffinement décrit de manière un peu plus détaillée l'enchaînement logique des opérations de la solution d'un sous problème (application de la politique du « **diviser pour régner** »).
- On arrête les raffinements dès que la description devient abordable en une ou deux pages.

Introduction aux raffinements

Exemple

Ecrire un algorithme permettant de saisir les deux notes d'algorithmique des 90 étudiants de TC1, de calculer la moyenne de chaque étudiant dans cette matière, de calculer et d'afficher les statistiques principales de cette classe en algorithmique à savoir la moyenne de la classe, la plus forte moyenne et la plus faible moyenne de cette classe. On suppose disposer de deux tableaux Nom et Prenom renfermant déjà les noms et prénoms des 90 étudiants concernés.

Introduction aux raffinements

1) Spécification

En supposant disposer de deux tableaux Nom et Prenom renfermant déjà les noms et prénoms des N (N=90) étudiants de TC1, l'algorithme permet de :

- saisir les deux notes d'algorithmique de ces étudiants ;
- calculer la moyenne de chaque étudiant dans cette matière ;
- calculer et d'afficher la plus forte moyenne, la plus faible moyenne et la moyenne de cette classe en algorithmique.

Introduction aux raffinements

2) Variables en entrée

Constante N ← 90 ; /* nombre d'étudiants de TC1 */

Nom est de type tableau [1..N] chaînes de caractères, correspondant aux noms des étudiants de TC1, ce tableau étant supposé initialisé ;

Prenom est de type tableau [1..N] chaînes de caractères, correspondant aux prénoms des étudiants de TC1, ce tableau étant aussi supposé initialisé

Introduction aux raffinements

3) Variables en sortie

Note1 est de type tableau [1..N] réels, correspondant aux premières notes des étudiants de TC1 en algorithmique ;

Note2 est de type tableau [1..N] réels, correspondant aux deuxièmes notes des étudiants de TC1 en algorithmique ;

TMoy est de type tableau [1..N] réels, correspondant aux moyennes des étudiants de TC1 en algorithmique ;

Introduction aux raffinements

3) Variables en sortie

MoyClasse est de type réel, correspondant à la moyenne de la classe de TC1 en algorithmique ;

MinMoyClasse est de type réel, correspondant à la plus faible moyenne des étudiants de TC1 en algorithmique ;

MaxMoyClasse est de type réel, correspondant à la plus forte moyenne des étudiants de TC1 en algorithmique

Introduction aux raffinements

4) Variables en entrée-sortie

Néant

5) Variables intermédiaires

Som est de type réel, correspondant à la somme des moyennes des étudiants de TC1 en algorithmique ;

i est de type entier naturel, correspondant à l'indice de parcours des tableaux

Introduction aux raffinements

7) Description de l'enchaînement logique

Début

/* Saisie des deux notes de chacun des étudiants */

SaisirNotesEtudiants ;

/* Les tableaux Note1 et Note2 sont censés actuellement initialisés avec les notes des étudiants en algorithmique */

/* Calcul de la moyenne de chacun des étudiants */

CalculerMoyenneDeChaqueEtudiant ;

/* Le tableau TMoy est censé actuellement initialisé avec les moyennes des étudiants en algorithmique */

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

/* Calcul des statistiques de la classe en algorithmique, à savoir la moyenne de la classe, la plus forte moyenne de la classe et la plus faible moyenne de la classe */

CalculerStatistiquesClasse ;

/* Les variables MoyClasse, MaxMoyClasse et MinMoyClasse sont censées contenir les statistiques en algorithmique de la classe de TC1*/

AfficherStatistiquesClasse

Fin

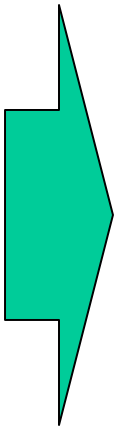
Introduction aux raffinements

SaisirNotesEtudiants
CalculerMoyenneDeChaqueEtudiant
CalculerStatistiquesClasse
AfficherStatistiquesClasse

=

Blocs à détailler par <u>Raffinements successifs</u>

Les raffinements successifs viennent après la fin du bloc principal de description de l'enchaînement logique selon le format



<u>Raffinement</u> de XXXX

Liste des instructions décrivant le traitement associé au bloc et pouvant donner lieu à son tour à d'autres raffinements

<u>FinRaffinement</u> de XXXX

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

Raffinement de **SaisirNotesEtudiants**

/* On saisit les deux notes de chacun des étudiants après affichage de leurs noms et prénoms */

```
Pour i allant de 1 à N faire
    Ecrire ("Entrez la 1re note de ", Nom(i), " ", Prenom(i)) ;
    Lire (Note1(i)) ;
    Ecrire ("Entrez la 2e note de ", Nom(i), " ", Prenom(i)) ;
    Lire (Note2(i))
FinPour
```

FinRaffinement de **SaisirNotesEtudiants**

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

Raffinement de **CalculerMoyenneDeChaqueEtudiant**

/* Calcul de la moyenne de chacun des étudiants */

Pour i allant de 1 à N faire

TMoy(i) ← (Note1(i)+Note2(i))/2 /* division réelle */

FinPour

/* Le tableau TMoy est actuellement initialisé
avec les moyennes des étudiants en algorithmique */

FinRaffinement de **CalculerMoyenneDeChaqueEtudiant**

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

Raffinement de **CalculerStatistiquesClasse**

/* Initialisation de la moyenne de la classe, de la plus forte moyenne et de la plus faible moyenne de la classe avec les valeurs du 1er étudiant de la classe */

MinMoyClasse ← TMoy(1) ;
MaxMoyClasse ← TMoy(1) ;
Som ← TMoy(1) ;

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

```
/* Prise en compte des autres étudiants de la classe */
   Pour i allant de 2 à N faire
   |   Som ← Som+TMoy(i) ;
   |   Si TMoy(i) < MinMoyClasse alors
   |   |   MinMoyClasse ← TMoy(i) /* nouveau min */
   |   FinSi ;
   |   Si TMoy(i) > MaxMoyClasse alors
   |   |  MaxMoyClasse ← TMoy(i) /* nouveau max */
   |   FinSi
   FinPour
```

Introduction aux raffinements

7) Description de l'enchaînement logique (suite)

/* Les variables Som, MaxMoyClasse et MinMoyClasse sont bien initialisées */

MoyClasse ← Som/N /* division réelle */

/* Les variables MoyClasse, MaxMoyClasse et MinMoyClasse contiennent les statistiques de TC1 en algorithmique */

FinRaffinement de **CalculerStatistiquesClasse**

Introduction aux raffinements

7) Description de l'enchaînement logique (suite et fin)

Raffinement de **AfficherStatistiquesClasse**

/* Affichage des statistiques de la classe contenues dans les variables *MoyClasse*, *MaxMoyClasse* et *MinMoyClasse* */

Ecrire ("La moyenne de la classe est ", *MoyClasse)* ;
Ecrire ("La plus forte moyenne de la classe est ", M*axMoyClasse*) ;
Ecrire ("La plus faible moyenne de la classe est ", *MinMoyClasse)*

FinRaffinement de **AfficherStatistiquesClasse**

Introduction aux procédures et fonctions

Que dire des procédures ?

Introduction aux procédures

- Un raffinement correspond à une **procédure dans sa forme la plus simple** avec comme caractéristiques que les **variables** sur lesquelles porte le traitement doivent être **de portée globale**, de sorte que les modifications effectuées sur ces variables au sein de la procédure soient conservées et visibles en dehors de la procédure.
- Une **procédure** est semblable à une « boite » capable d'utiliser **des données en entrée** pour produire, parfois en utilisant **des données intermédiaires**, **des données en résultat**.
- L'emploi des procédures **augmente la lisibilité** et la **réutilisabilité** des algorithmes produits, en **évitant les séquences répétitives** et en favorisant l'analyse aisée des erreurs de logique.

Introduction aux procédures

- Une **procédure porte un nom** et doit être complètement décrite avant toute utilisation.
- Allure d'une **procédure**

Procédure nomProcédure

/* Commentaire sur ce que fait la procédure */

Liste descriptive des paramètres en entrée

Liste descriptive des paramètres en sortie

Liste descriptive des paramètres en entrée/sortie

Liste descriptive des variables locales

Description de l'enchaînement logique associé

FinProcédure nomProcédure

Introduction aux procédures

Liste descriptive des paramètres en entrée

Ce sont les variables censées renfermer des valeurs positionnées pour que lors de l'appel de la procédure, cette dernière les utilise en lecture seulement.

Elles sont décrites comme d'habitude avec des facettes de type et de commentaire.

Introduction aux procédures

Liste descriptive des paramètres en sortie

Ce sont les variables censées renfermer les valeurs que la procédure produira en résultat et qui devront être visibles en dehors de la procédure.

Elles sont décrites comme d'habitude avec des facettes de type et de commentaire.

Introduction aux procédures

Liste descriptive des paramètres en entrée/sortie

Ce sont les variables censées renfermer les valeurs qui, dans la procédure, seront exploitées aussi bien en lecture qu'en écriture/modification.

Leurs modifications devront être visibles en dehors de la procédure.

Elles sont décrites comme d'habitude avec des facettes de type et de commentaire.

Introduction aux procédures

Liste descriptive des variables locales

Ce sont des variables intermédiaires dont la portée se limite à la procédure. Elles sont initialisées dans la procédure et les modifications qu'elles vont connaître ne seront pas visibles en dehors de la procédure.

Elles sont décrites comme d'habitude avec des facettes de type et de commentaire.

Introduction aux procédures

- Les **paramètres** sont les **seuls moyens de communication** avec la procédure : ils constituent **l'interface** de communication de la procédure.

==>*Avant tout appel de procédure, les paramètres censés être utilisés par la procédure doivent être correctement positionnés.*

Introduction aux procédures

- Allure d'un **appel de procédure**

  nomProcedure (liste des paramètres en entrée,
  liste des paramètres en sortie,
  liste des paramètres en entrée-sortie)

- Lors d'un **appel de procédure** il y a substitution des **paramètres formels** (employés au moment de la description de la procédure) avec les **paramètres effectifs** ou **arguments** (paramètres employés au moment de l'appel de la procédure) selon l'ordre des paramètres employé lors de la description de la procédure.

Introduction aux procédures

Procédure **SaisirNotesEtudiants**

/* Cette procédure permet de saisir les deux notes de chacun des N étudiants après affichage de leurs noms et prénoms */

Paramètres en entrée :

N est de type entier, correspondant au nombre d'étudiants concernés ;

Nom est de type Tableau[1..N] chaînes de caractères, correspondant au tableau des noms d'étudiants ;

Prenom est de type Tableau[1..N] chaînes de caractères, correspondant au tableau des prénoms d'étudiants

Paramètres en sortie :

Note1 est de type Tableau[1..N] réels, correspondant au tableau des notes du 1er devoir en algorithmique ;

Note2 est de type Tableau[1..N] réels, correspondant au tableau des notes du 2e devoir en algorithmique

Introduction aux procédures

Paramètres en entrée-sortie : néant ;

// Cette procédure emploie, lors de son appel, 5 paramètres pouvant

// être : Nx, nomX, prenomX, noteA, noteB.

Variables locales :

i est de type entier, correspond à l'index de parcours des tableaux ;

```
Pour i allant de 1 à N faire
    Ecrire ("Entrez la 1re note de ", Nom(i), " ", Prenom(i)) ;
    Lire (Note1(i)) ;
    Ecrire ("Entrez la 2e note de ", Nom(i), " ", Prenom(i)) ;
    Lire (Note2(i))
FinPour
FinProcédure SaisirNotesEtudiants
```

Introduction aux procédures

Procédure **CalculerMoyenneDeChaqueEtudiant**

/* Elle réalise le calcul de la moyenne en algorithmique de chacun des étudiants */

Paramètres en entrée :

N est de type entier, correspondant au nombre d'étudiants concernés ;

Note1 est de type Tableau[1..N] réels, correspondant au tableau des notes du 1er devoir en algorithmique ;

Note2 est de type Tableau[1..N] réels, correspondant au tableau des notes du 2e devoir en algorithmique ;

Paramètres en sortie :

TMoy est de type Tableau[1..N] réels, correspondant au tableau des moyennes des étudiants en algorithmique ;

Introduction aux procédures

Paramètres en entrée-sortie : néant ;

// Cette procédure emploie 4 arguments lors de son appel.

Variables locales :

i est de type entier, correspondant à l'indice de parcours des tableaux ;

Pour i allant de 1 à N faire

| TMoy(i) ← (Note1(i)+Note2(i))/2

FinPour

FinProcédure **CalculerMoyenneDeChaqueEtudiant**

Introduction aux procédures

Procédure **CalculerStatistiquesClasse**

/* Elle calcule les statistiques de la classe en algorithmique */

Paramètres en entrée :

/ A préciser dans le cadre d'un TD */*

Paramètres en sortie :

/ A préciser dans le cadre d'un TD */*

Paramètres en entrée-sortie :

/ A préciser dans le cadre d'un TD */*

Variables locales :

/ A préciser dans le cadre d'un TD */*

Bloc logique

/ A préciser dans le cadre d'un TD */*

FinProcédure **CalculerStatistiquesClasse**

Introduction aux procédures

Procédure **AfficherStatistiquesClasse**

/* Affichage des statistiques de la classe en algorithmique */

Paramètres en entrée :

MoyClasse est de type réel, correspondant à la moyenne de la classe à afficher ;

MaxMoyClasse est de type réel, correspondant à la meilleure moyenne de la classe à afficher ;

MinMoyClasse est de type réel, correspondant à la plus faible moyenne de la classe à afficher ;

Paramètres en sortie : néant ;

Paramètres en entrée-sortie : néant ;

// Cette procédure emploie 3 arguments lors de son appel.

Variables locales : néant ;

Introduction aux procédures

Ecrire ("La moyenne de la classe est ", *MoyClasse)* ;
Ecrire ("La meilleure moyenne de la classe est ", M*axMoyClass*e) ;
Ecrire ("La plus faible moyenne de la classe est ", *MinMoyClasse)*

FinProcédure **AfficherStatistiquesClasse**

Introduction aux procédures

- Exemple d'utilisation des procédures

/* On suppose :

- la constante Nb initialisée ;
- les tableaux NomEt et PrenomEt bien initialisés ;
- les tableaux NoteA, NoteB et TabMoy bien déclarés ;
- les variables MoyCl, MaxMoy, MinMoy déclarées ;
- les descriptions correctes des procédures SaisirNotesEtudiants, CalculerMoyenneDeChaqueEtudiant, CalculerStatistiquesClasse et AfficherStatistiquesClasse disponibles.

*/

7) Description de l'enchaînement logique

Début

SaisirNotesEtudiants (Nb, NomEt, PrenomEt, NoteA, NoteB) ;
CalculerMoyenneDeChaqueEtudiant (Nb, NoteA, NoteB, TabMoy) ;
CalculerStatistiquesClasse (Nb, TabMoy, MoyCl, MaxMoy, MinMoy) ;
AfficherStatistiquesClasse (MoyCl, MaxMoy, MinMoy)

Fin.

Appel *des 4 procédures l'une après l'autre*

Introduction aux procédures

Exercice nº1

Ecrire une procédure qui calcule la somme des éléments d'un tableau de N réels et la met à disposition à travers l'un de ses paramètres.

Exercice nº3

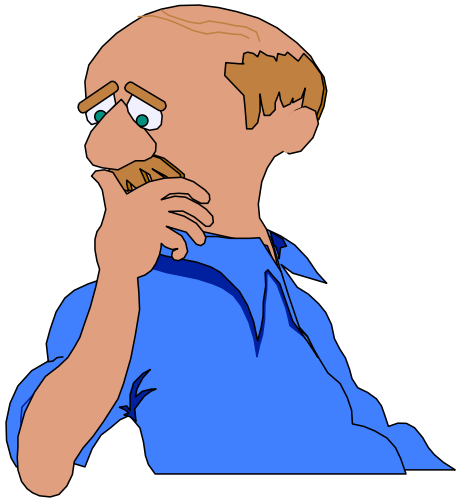
Ecrire une procédure qui calcule la distance entre deux points A(ax,ay) et B(bx,by) et la met à disposition à travers l'un de ses paramètres.

Exercice nº3

Ecrire une procédure qui élève un réel x à la puissance b, avec entier naturel supérieur à 0, le résultat étant mis à disposition à travers l'un de ses paramètres

Introduction aux fonctions

Que dire des fonctions ?



Introduction aux fonctions

- Une **fonction** est une **procédure** dont le **nom est capable de supporter un résultat.**

- Dans une fonction, le **nom de cette fonction fait partie implicitement** de la liste des **paramètres en sortie**.

- le **nom de la fonction est associé à un type de donnée** et peut figurer dans les expressions portant sur des objets (variables et constantes) de son type.

Introduction aux fonctions

- Allure d'une **fonction**

Fonction nomFonction est de type typeFonction

/* Commentaires sur ce que fait la fonction et sur

le type du résultat supporté par le nom de la fonction*/

Liste descriptive des paramètres en entrée

Liste descriptive des autres paramètres en sortie

Liste descriptive des paramètres en entrée/sortie

Liste descriptive des variables locales

Description de l'enchaînement logique associé

Retourner *expression*

FinFonction nomFonction

Introduction aux fonctions

- Exemple 1 de description de fonction

Fonction Moy2Val est de type réel

/* Elle calcule la moyenne de deux nombres réels qu'elle retourne comme résultat par l'intermédiaire du nom de la fonction*/

Paramètres en entrée :

Val1 est de type réel, correspondant à la 1re valeur à employer pour la moyenne ;

Val2 est de type réel, correspondant à la 2e valeur à employer pour la moyenne ;

Autres paramètres en sortie : néant ;

Paramètres en entrée-sortie : néant ;

// cette fonction emploie 2 arguments lors de son appel

Variables locales :

MoySimple est de type réel, correspondant à la moyenne calculée ;

MoySimple ← (Val1+Val2)/2 ;

Retourner *MoySimple*

FinFonction Moy2Val

Introduction aux fonctions

- Exemple 2 de description de fonction

Fonction Moy2Val est de type réel

/* Elle calcule la moyenne de deux nombres réels qu'elle retourne comme résultat par l'intermédiaire du nom de la fonction*/

Paramètres en entrée :

Val1 est de type réel, correspondant à la 1re valeur à employer pour calculer la moyenne ;

Val2 est de type réel, correspondant à la 2e valeur à employer pour calculer la moyenne ;

Autres paramètres en sortie : néant ;

Paramètres en entrée-sortie : néant ;

// Cette fonction emploie 2 arguments lors de son appel

Variables locales : néant ;

Retourner (Val1+Val2)/2

FinFonction Moy2Val

Introduction aux fonctions

- Une **fonction** est utilisée par invocation de son nom : on parle d'**appel de fonction**.

- Les **paramètres** et le **nom de la fonction** sont les **seuls moyens de communication** entre la fonction et son environnement d'appel : ils constituent **l'interface** de communication de la fonction.

==>*Avant tout appel de fonction, les paramètres censés être utilisés par elle doivent être correctement positionnés.*

Introduction aux fonctions

- Allure d'un **appel de fonction**

  resultat ← **nomFonction** (liste des paramètres en entrée,
  liste des autres paramètres en sortie et en entrée-sortie)

ou dans une expression algébrique combinant des opérateurs

nomFonction (liste des paramètres en entrée,
liste des autres paramètres en sortie et en entrée-sortie)

La valeur retournée par le nom de la fonction est soit affectée à **resultat,** soit employée lors de l'évaluation de l'expression dans laquelle elle se retrouve.

Introduction aux fonctions

- Lors d'un **appel de la fonction** il y a substitution des **paramètres formels** (employés au moment de la description de la fonction) avec les **paramètres effectifs** ou **arguments** (paramètres employés au moment de l'appel de la fonction) selon l'ordre des paramètres employé au moment de la description de la fonction.

- Exemple :

  moy ← **Moy2Val**(n1, n2) ; // n1 et n2 étant des variables
  // de type réel censées initialisées

Introduction aux fonctions

/* On suppose :
- la constante N initialisée ;
- les tableaux NoteX et NoteY bien initialisés ;
- le tableau TablMoy déclaré et devant être initialisé ;
- la variable i déclarée comme de type entier naturel ;
- la fonction **Moy2Val** déjà décrite.

*/

Pour i allant de 1 à N faire

    /* *appel de la fonction N fois avec la paire de notes de chaque étudiant* */

    TablMoy(i) ← **Moy2Val** (NoteX(i), NoteY(i));

FinPour

Introduction aux fonctions

- Remarques

- L'emploi de la fonction suppose le **respect de l'interface de communication** avec la fonction.

- Le **nom de la fonction** supporte un résultat, ce qui a permis de le placer **à droite dans une instruction** d'affectation ou dans une expression algébrique, **traitement interdit avec une procédure**.

- L'affectation, au sein de la fonction, du résultat obtenu au nom de la fonction est **obligatoire** et ne se réalise qu'à travers l'instruction « **Retourner expression** » toujours présente dans une fonction.

- Le **type de la fonction** doit être **compatible** avec le type de la variable censée recevoir le résultat retourné par le nom de la fonction.

Introduction aux fonctions

Exercice n°1

Ecrire une fonction calculant et retournant la somme des éléments d'un tableau de N réels.

Exercice n°2

Ecrire une fonction calculant et retournant la distance entre deux points A(ax,ay) et B(bx,by) du plan réel.

Exercice n°3

Ecrire une fonction qui élève un réel x à la puissance b correspondant à un entier naturel supérieur à 0, le résultat étant mis à disposition à travers le nom de la fonction.

Modes de passage des paramètres

- Définition

Le mode de passage d'un paramètre désigne la manière dont une fonction/procédure récupère les valeurs communiquées via son **interface de communication**.

Deux principaux modes de passage de paramètres existent :

- le mode de passage par valeur
- le mode de passage par adresse

- Cette précision est apportée en complétant la facette de type.

Ex. : Val1 est de type réel, passé par valeur, correspondant à la 1re valeur à employer pour calculer la moyenne ;

Modes de passage des paramètres

- Description du mode de passage par valeur

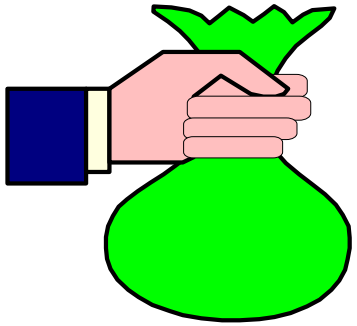
Passage par valeur de paramètre
ou
argument en entrée seulement

Au sein de la procédure ou de la fonction, c'est avec une **copie** de la variable désignée qu'on travaille.

Modes de passage des paramètres

- Contexte d'emploi du mode de passage par valeur

  Lorsqu'aucune modification de la valeur d'un paramètre dans le contexte d'une procédure ou d'une fonction ne doit être répercutée/visible en dehors de ce contexte. On parle alors de passage de paramètre par valeur.



Modes de passage des paramètres

Mode de passage de paramètre par valeur

Avantages

- La valeur de la donnée originale est préservée, même si par mégarde au sein de la procédure/fonction on venait à modifier le paramètre.
- Il convient pour passer des paramètres de type peu encombrant à protéger en écriture et les fonctions non récursives.

Inconvénients

- Lenteur due à la création de la variable locale puis la recopie de la valeur du paramètre avant le démarrage de la procédure/fonction.
- Plus grande consommation d'espace mémoire.

Modes de passage des paramètres

- Description du mode de passage par adresse

Passage par adresse de paramètre
ou
argument en entrée-sortie

Au sein de la procédure ou de la fonction,
c'est avec la variable **originale** qu'on travaille.

Modes de passage des paramètres

- Contexte d'emploi du mode de passage par adresse

Lorsque toute modification de la valeur d'un paramètre dans le contexte de la procédure ou de la fonction doit au contraire être répercutée/ visible en dehors de ce contexte, on parle alors de passage de paramètre par adresse ou en entrée-sortie.

Modes de passage des paramètres

Mode de passage de paramètre par adresse

Avantages

- Rapidité d'exécution.
- Economie d'espace mémoire.
- Plus adapté au passage des paramètres de type encombrant et les fonctions récursives.

Inconvénients

- Risque de modification accidentelle et irréparable des données.
- Difficultés à suivre la trace des variables durant l'exécution.

Modes de passage des paramètres

La **valeur modifiée** du paramètre **ne vous intéresse pas** à la fin de l'exécution de la procédure ou de la fonction ?

passage par valeur

Paramètre

=

copie de la variable

La **valeur modifiée** du paramètre **vous intéresse** à la fin de l'exécution de la procédure ou de la fonction ?

passage par adresse

Paramètre

=

variable originale

Modes de passage des paramètres

Exercice

Préciser les paramètres employés dans l'écriture de chacune des procédures et fonctions décrites précédemment.

FIN

QUESTIONS ?

Problème

Ecrire un algorithme permettant :

- de saisir la moyenne en algorithmique et celle en programmation des 5 étudiants de TC1 ;
- de calculer la moyenne en algorithmique et programmation de chacun de ces 5 étudiants, et
- d'afficher l'ensemble des 5 moyennes précédemment calculées.