

Algorithmique

6. Procédures et Fonctions

1

Plan du cours

- I. Introduction
- II. Variables
- III. Structures alternatives
- IV. Structures itératives
- V. Tableaux
- VI. Fonctions et procédures**
- VII. Fichiers

▶ 2

Algorithmique ESI 2023-2024

2

Sommaire

- I. Intérêt et motivation
- II. Sous-procédures
- III. Fonctions
- IV. Portée des variables

▶ 3

Algorithmique

ESI

2023-2024

3

Intérêt et motivation

4

Problématique

- ▶ Généralement, un algorithme comporte plusieurs parties, chacune effectuant un traitement donné
- ▶ Exemple : gestion des étudiants, gestion des clients, gestion de stock...
- Programme trop long et/ou trop complexe

▶ 5

Algorithmique

ESI

2023-2024

5

Problématique

- ▶ Plusieurs concepteurs sont amenés à collaborer sur un même algorithme, chacun sur une ou plusieurs parties
- ▶ Nécessité de réutiliser le code
- ▶ Exemple : algorithme de recherche dichotomique, algorithme de tri des tableaux ...

▶ 6

Algorithmique

ESI

2023-2024

6

Problématique

- ➔ L'idéal est de décomposer un algorithme en plusieurs « petits » algorithmes
 - ▶ Chacun traite une partie du problème
 - ▶ Au besoin, ils s'appellent entre eux
- ➔ Utilisation de sous-procédures et de fonctions

▶ 7

Algorithmique

ESI

2023-2024

7

Intérêt et motivation

- ▶ Pourquoi découper un seul algorithme en plusieurs « petits » algorithmes ?
 - ▶ Améliorer la lisibilité de l'algorithme et, par conséquent celle du programme qui sera implémenté
 - ▶ Faciliter la maintenance de l'algorithme / programme
 - ▶ Favoriser la réutilisation de l'algorithme / programme

▶ 8

Algorithmique

ESI

2023-2024

8

```

ALGORITHME exemple
VAR tab1[10], tab2[10], tab[10] : réel
DEBUT
  //remplir le tableau tab1
  POUR i ← 0 à 9
    Lire(tab1[i])
  i SUIVANT
  //remplir le tableau tab2
  POUR i ← 0 à 9
    Lire(tab2[i])
  i SUIVANT
  //remplir le tableau tab3
  POUR i ← 0 à 9
    Lire(tab3[i])
  i SUIVANT
  //trier le tableau tab1
  POUR i ← 0 à 9
    POUR j ← i + 1 à 10
      SI tab1[j] < tab1[i] ALORS
        x ← tab1[i]
        tab1[i] ← tab1[j]
        tab1[j] ← x
      FINSI
    j SUIVANT
  i SUIVANT
  //trier le tableau tab2
  POUR i ← 0 à 9
    POUR j ← i + 1 à 10
      SI tab2[j] < tab2[i] ALORS
        x ← tab2[i]
        tab2[i] ← tab2[j]
        tab2[j] ← x
      FINSI
    j SUIVANT
  i SUIVANT
  //trier le tableau tab3
  POUR i ← 0 à 9
    POUR j ← i + 1 à 10
      SI tab3[j] < tab3[i] ALORS
        x ← tab3[i]
        tab3[i] ← tab3[j]
        tab3[j] ← x
      FINSI
    j SUIVANT
  i SUIVANT
  //autres instructions du programme
  ...
FIN

```

9

Intérêt et motivation

```

ALGORITHME exemple
VAR tab1[10], tab2[10], tab[10] : réel
...
DEBUT
  //remplir les tableaux
  Remplir(tab1, 10)
  Remplir(tab2, 10)
  Remplir(tab3, 10)

  //trier les tableaux tab1
  Trier(tab1, 10)
  Trier(tab2, 10)
  Trier(tab3, 10)
FIN

```

10

Intérêt et motivation

- ▶ Le corps de l'algorithme est appelé procédure principale
- ▶ Les instructions qu'on regroupe pour effectuer un traitement particulier sont appelées **sous-procédures** ou **fonctions**
- ▶ On fait appel à ces sous-procédures ou fonctions à partir de la procédure principale

▶ II

Algorithmique

ESI

2023-2024

11

Sous-procédures

12

Définition

- ▶ Une **sous-procédure** est un ensemble d'instructions décrivant une action simple ou complexe
- ▶ Elle est définie par
 - ▶ Son nom
 - ▶ Ses arguments (ou paramètres)

▶ I3

Algorithmique

ESI

2023-2024

13

Syntaxe

- ▶ Syntaxe de la définition

```

PROCÉDURE nomProcédure(arg1:type, ..., argN:type)
VAR ...
DÉBUT
...
FINPROCÉDURE
  
```

▶ I4

Algorithmique

ESI

2023-2024

14

Syntaxe

► Syntaxe de l'appel

```

DEBUT
//instructions
...
nomProcédure(arg1, ..., argN)
//autres instructions
...
FIN

```

► I5

Algorithmique

ESI

2023-2024

15

Syntaxe

► Exemple

```

ALGORITHME exemple

VAR name : chaîne

PROCÉDURE Message(nom:chaîne)
DÉBUT
    Afficher("Bonjour " & nom)
FINPROCÉDURE

DEBUT
    Afficher("Saisir votre nom")
    Lire(name)
    Message(name)
FIN

```

► I6

Algorithmique

ESI

2023-2024

16

Syntaxe

► Exemple

ALGORITHME exemple

PROCÉDURE Message()

DÉBUT

Afficher("Bonjour!")

FINPROCÉDURE

DEBUT

Message()

FIN

► I7

Algorithmique

ESI

2023-2024

17

Arguments

- Les arguments d'une sous-procédure sont mis entre parenthèses lors de sa déclaration et des appels
- Ils servent à transmettre des informations de la procédure principale (ou le programme appelant) à la sous-procédure

► I8

Algorithmique

ESI

2023-2024

18

Arguments

- ▶ Une sous-procédure peut avoir 0 ou n arguments
 - ▶ Même sans arguments, on garde les parenthèses en les laissant vides
- ▶ Les arguments sont passés soit **par valeur** soit **par adresse**

▶ 19

Algorithmique

ESI

2023-2024

19

Passage des arguments

- ▶ Passage par valeur
 - ▶ Mode de passage par défaut
 - ▶ Syntaxe de la définition

```

PROCÉDURE nomProcédure(arg1:type, ..., argN:type)
VAR ...
DÉBUT
...
FINPROCÉDURE

```

▶ 20

Algorithmique

ESI

2023-2024

20

Passage des arguments

► Passage par valeur

► Mode de passage par défaut

► Syntaxe de la définition

► Syntaxe de l'appel

DEBUT

//instructions

...

nomProcédure(*var1*, ..., *varN*)

//autres instructions

...

FIN

► 21

Algorithmique

ESI

2023-2024

21

Passage des arguments

► Passage par valeur

► Lors de l'appel, la sous-procédure crée N nouvelles variables : *arg1*, ..., *argN*

► Chaque nouvelle variable va recevoir la **valeur** des variables *var1*, ..., *varN* (dans l'ordre de déclaration) → elle sera une copie du paramètre passé par valeur et sera détruite à la fin de l'exécution de la sous-procédure

► **Les variables *var1*, ..., *varN* ne sont pas modifiées**

► 22

Algorithmique

ESI

2023-2024

22

Passage des arguments

► Passage par valeur

ALGORITHME exemple

VAR n : entier

PROCÉDURE Tripler(*arg* : entier)

DÉBUT

arg \leftarrow arg*3

FINPROCÉDURE

DEBUT

n \leftarrow 2

Tripler(n)

Afficher(n)

FIN

► 23

Algorithmique

ESI

2023-2024

23

Passage des arguments

► Passage par valeur

► Avantage

- Les variables du programme appelant sont protégées contre toute modification de leur contenu

► Inconvénients

- Les variables utilisées ne peuvent être que des paramètres en entrée, jamais en sortie
- Utilisation de la mémoire

► 24

Algorithmique

ESI

2023-2024

24

Passage des arguments

- ▶ Passage par adresse
 - ▶ Syntaxe de la définition

```
PROCÉDURE nomProcédure(arg1:type*, ..., argN:type*)
VAR ...
DÉBUT
...
FINPROCÉDURE
```

▶ 25

Algorithmique

ESI

2023-2024

25

Passage des arguments

- ▶ Passage par adresse
 - ▶ Syntaxe de la définition
 - ▶ Syntaxe de l'appel

```
DEBUT
//instructions
...
nomProcédure(&var1, ..., &varN)
//autres instructions
...
FIN
```

▶ 26

Algorithmique

ESI

2023-2024

26

Passage des arguments

► Passage par adresse

- Lors de l'appel, la sous-procédure crée N nouvelles variables : $\text{arg1}, \dots, \text{argN}$
- Chaque nouvelle variable va recevoir l'**adresse** des variables $\text{var1}, \dots, \text{varN}$ (dans l'ordre de déclaration) → toute modification de $\text{arg1}, \dots, \text{argN}$ sera répercutée sur $\text{arg1}, \dots, \text{argN}$
- **Les variables $\text{var1}, \dots, \text{varN}$ peuvent être modifiées**

► 27

Algorithmique

ESI

2023-2024

27

Passage des arguments

► Passage par adresse

ALGORITHME *triplerAlgo*

VAR n : entier

PROCÉDURE *Tripler*(arg :entier*)

DÉBUT

$\text{arg} \leftarrow \text{arg} * 3$

FINPROCÉDURE

DEBUT

$n \leftarrow 2$

Tripler(& n)

Afficher(n)

FIN

► 28

Algorithmique

ESI

2023-2024

28

Passage des arguments

▶ Passage par adresse

▶ Avantage

- ▶ Grâce aux pointeurs, les variables du programme peuvent être modifiées pour répondre à un besoin particulier → les variables utilisées peuvent être des paramètres en entrée et en sortie

▶ Inconvénient

- ▶ Risque d'écraser par erreur les variables du programme

▶ 29 appelant

Algorithmique

ESI

2023-2024

29

Passage des arguments

▶ Remarques

- ▶ Un tableau est toujours passé par adresse, et son identificateur représente son adresse

▶ 30

Algorithmique

ESI

2023-2024

30

Passage des arguments

► Remarques

ALGORITHME *nomAlgorithme*

VAR notes[10] : réel

PROCÉDURE RemplirTableau(**tab[]**:réel, n: entier)

VAR i : entier

DÉBUT

POUR i \leftarrow 0 à n – 1

Afficher("Donner la valeur de l'élément N° ", i+1)

Lire(tab[i])

i **SUIVANT**

FINPROCÉDURE

DEBUT

RemplirTableau(**notes**, 10)

FIN

► 31

Algorithmique

ESI

2023-2024

31

Passage des arguments

► Remarques

- Un tableau est toujours passé par adresse, et son identificateur représente son adresse
- Une sous-procédure peut faire appel à une autre sous-procédure
- Une sous-procédure peut faire appel à elle-même → **récurtivité**

► 32

Algorithmique

ESI

2023-2024

32

Exercices

- Écrire une sous-procédure qui permet d'afficher les nombres de 0 à 100

► 33

Algorithmique

ESI

2023-2024

33

Exercices

```

ALGORITHME algoExercice

PROCÉDURE AfficherNombres()
VAR i : entier
DÉBUT
  POUR i ← 0 à 100
    Afficher(i)
  i SUIVANT
FINPROCÉDURE

DÉBUT
  AfficherNombres()
FIN

```

► 34

Algorithmique

ESI

2023-2024

34

Exercices

- ▶ Écrire une sous-procédure qui calcule et affiche la somme et le produit de trois nombres réels passés en paramètre

▶ 35

Algorithmique

ESI

2023-2024

35

Exercices

```

ALGORITHME algoExercice

VAR a, b, c: réel

PROCÉDURE SommeEtProduit(x, y, z : réel)
VAR somme, produit : réel
DÉBUT
    somme  $\leftarrow$  x + y + z
    produit  $\leftarrow$  x*y*z
    Afficher(somme, produit)
FINPROCÉDURE

DÉBUT
    Lire(a,b,c)
    SommeEtProduit(a,b,c)
FIN
  
```

▶ 36

Algorithmique

ESI

2023-2024

36

Exercices

- ▶ Modifier la sous-procédure précédente pour stocker la somme et le produit dans des variables accessibles à partir de la procédure principale

▶ 37

Algorithmique

ESI

2023-2024

37

Exercices

ALGORITHME algoExercice

VAR a, b, c, somme, produit: réel

PROCÉDURE SommeEtProduit(x, y, z:réel, s:réel*, p:réel*)

DÉBUT

s \leftarrow x + y + z

p \leftarrow x*y*z

FINPROCÉDURE

DÉBUT

Lire(a,b,c)

SommeEtProduit(a,b,c, &somme, &produit)

Afficher(somme,produit)

FIN

▶ 38

Algorithmique

ESI

2023-2024

38

Exercices

- ▶ Écrire une sous-procédure qui permet de permuter les valeurs de deux variables

▶ 39

Algorithmique

ESI

2023-2024

39

Exercices

ALGORITHME algoExercice

VAR a, b: réel

PROCÉDURE Permutation(x: réel*, y:réel*)

VAR z : réel

DÉBUT

z \leftarrow x

x \leftarrow y

y \leftarrow z

FINPROCÉDURE

DÉBUT

Lire(a,b)

Permutation(&a, &b)

FIN

▶ 40

40

Exercices

- ▶ Écrire une sous-procédure qui permet de remplacer un réel par sa valeur absolue

▶ 41

Algorithmique

ESI

2023-2024

41

Exercices

```

ALGORITHME algoExercice

VAR a: réel

PROCÉDURE ValeurAbsolue(x: réel*)
DÉBUT
  SI  $x < 0$  ALORS
     $x \leftarrow -x$ 
  FINSI
FINPROCÉDURE

DÉBUT
  Lire(a)
  ValeurAbsolue(&a)
FIN

```

▶ 42

Algorithmique

ESI

2023-2024

42

Fonctions

43

Définition

- ▶ Une **fonction** est une sous-procédure particulière qui **retourne un résultat** à la procédure principale (ou le programme appelant)
- ▶ Elle est définie par
 - ▶ Son nom
 - ▶ Ses arguments (ou paramètres d'entrée)
 - ▶ Son type de retour

▶ 44

Algorithmique ESI 2023-2024

44

Syntaxe

► Syntaxe de la définition

```
FONCTION nomFonction(arg1:type, ..., argN:type) : type
VAR ...
VAR resultat : type
DÉBUT
...
Retourner resultat
FINFONCTION
```

► 45

Algorithmique

ESI

2023-2024

45

Syntaxe

► Syntaxe de l'appel

```
DEBUT
//instructions
...
var ← nomFonction(arg1, ..., argN)
//autres instructions
...
FIN
```

► 46

Algorithmique

ESI

2023-2024

46

Arguments

- ▶ Une fonction sans arguments d'entrée s'écrit avec les parenthèses vides ()
- ▶ Les arguments sont utilisés de la même manière que pour les sous-procédures

▶ 47

Algorithmique

ESI

2023-2024

47

Fonctions prédéfinies

- ▶ Les langages de programmation offrent des fonctions prédéfinies qui servent à effectuer des traitements qui ne peuvent être effectués autrement ou qui sont trop complexes à écrire par le programmeur
- ▶ Elles prennent comme argument des chaînes de caractères, des nombres, des tableaux...

▶ 48

Algorithmique

ESI

2023-2024

48

Fonctions prédéfinies

► Exemples

- **Length**(*texte:chaîne*) : retourne le nombre de caractères d'une chaîne de caractères
- **Mid**(*texte:chaîne, car:caractère, n: entier*) : extrait une chaîne de caractère à partir de *texte*. Cette chaîne commence au caractère *car* et faisant *n* caractères de long
- **Asc**(*car:caractère*) : retourne le code ASCII d'un caractère
- **Car**(*code:entier*) : retourne le caractère correspondant à un code ASCII

► 49

Algorithmique

ESI

2023-2024

49

Fonctions prédéfinies

► Exemples

- **Entier**(*x:réel*) : retourne la partie entière d'un nombre
- **ConvNum**(*texte:chaîne*) : convertit un nombre qui est saisi et stocké comme une chaîne de caractères au type réel
- **ConvChaîne**(*x:réel*) : convertit un nombre qui est saisi et stocké comme nombre au type chaîne de caractères

► 50

Algorithmique

ESI

2023-2024

50

Exercices

- Écrire une fonction qui calcule la somme de trois nombres réels passés en paramètre

► 51

Algorithmique

ESI

2023-2024

51

Exercices

```

ALGORITHME algoExercice

VAR a, b, c, somme: réel

FONCTION Somme(x, y, z: Réel) : réel
VAR somme: réel
DÉBUT
    somme  $\leftarrow$  x + y + z
    Retourner somme
FINFONCTION

DÉBUT
    Lire(a,b,c)
    somme  $\leftarrow$  Somme(a,b,c)
FIN
  
```

► 52

Algorithmique

ESI

2023-2024

52

Exercices

```

ALGORITHME algoExercice

VAR a, b, c, somme: réel

FONCTION Somme(x, y, z:réel) : réel
DÉBUT
    Retourner x + y + z
FINFONCTION

DÉBUT
    Lire(a,b,c)
    somme ← Somme(a,b,c)
FIN
  
```

► 53

Algorithmique

ESI

2023-2024

53

Exercices

- Écrire une fonction qui permet de calculer la valeur absolue d'un réel

► 54

Algorithmique

ESI

2023-2024

54

Exercices

ALGORITHME algoExercice

VAR a: réel

FONCTION ValeurAbsolue(x: réel) : réel

DÉBUT

SI $x < 0$ **ALORS**

$x \leftarrow -x$

FINSI

Retourner x

FINFONCTION

DÉBUT

Lire(a)

$a \leftarrow \text{ValeurAbsolue}(a)$

FIN

► 55

Algorithmique

ESI

2023-2024

55

Exercices

- Quels sont les problèmes de cet algorithme ?

► 56

Algorithmique

ESI

2023-2024

56

Exercices

ALGORITHME algoExercice

VAR a, b : entier

VAR c : caractère

FONCTION ValeurAbsolue(x: réel) : réel

DÉBUT

SI $x < 0$ **ALORS**

$x \leftarrow -x$

FINSI

Retourner x

FINFONCTION

DÉBUT

Lire(a)

Lire(c)

$b \leftarrow \text{ValeurAbsolue}(10 * a - 20)$

$a \leftarrow \text{ValeurAbsolue}(c)$

$\text{ValeurAbsolue}(a) \leftarrow 0$

FIN

► 57

57

Exercices

- Écrire une fonction qui permet de vérifier si un tableau est trié avec un ordre croissant ou non trié

► 58

Algorithmique

ESI

2023-2024

58

```

FONCTION TableauTrie(tab[ ]: réel, taille: entier) : booléen
VAR i : entier
VAR trie : booléen
DÉBUT
    trie ← VRAI
    i ← 0
    TANTQUE trie ET i < taille-1 FAIRE
        SI tab[i] > tab[i+1] ALORS
            trie ← FAUX
        FINSI
        i ← i + 1
    FINTANTQUE
    Retourner trie
FINFONCTION

```

▶ 59

Algorithmique

ESI

2023-2024

59

Protée des variables

60

Définition

- ▶ La **portée d'une variable** est l'ensemble des sous-programmes (sous-procédures et fonctions) où elle est connue et peut donc être utilisée
- ▶ Une variable peut être
 - ▶ Locale
 - ▶ Globale

▶ 61

Algorithmique

ESI

2023-2024

61

Variable locale

- ▶ Une variable **locale** est une variable qui est déclarée au sein d'un sous-programme et qui, par conséquent, ne peut être utilisée que dans ce sous-programme

```

PROCÉDURE AfficherNombres()
VAR i : entier
DÉBUT
  POUR i ← 0 à 101
    Afficher(i)
  i SUIVANT
FINPROCÉDURE
  
```

▶ 62

Algorithmique

ESI

2023-2024

62

Variable globale

- ▶ Une variable **globale** est une variable qui est déclarée au sein de la procédure principale
- ▶ Toute sous-procédure ou fonction appelée par la procédure principale connaît cette variable et peut l'utiliser

▶ 63

Algorithmique

ESI

2023-2024

63

Variable globale

- ▶ Une sous-procédure ou fonction peut déclarer une variable globale qui, une fois créée, peut être utilisée par tous les autres sous-programmes de la procédure principale

```

PROCÉDURE AfficherNombres()
VAR GLOBALE i : entier
DÉBUT
  POUR i ← 0 à 101
    Afficher(i)
  i SUIVANT
FINPROCÉDURE
  
```

▶ 64

Algorithmique

ESI

2023-2024

64

Portée des variables

- ▶ Si un sous-programme déclare une variable locale qui a le même identificateur qu'une variable globale
- ▶ La variable locale est utilisée par le sous-programme où elle est définie
- ▶ La variable globale devient inaccessible par ce sous-programme

▶ 65

Algorithmique

ESI

2023-2024

65



Algorithmique

6. Procédures et Fonctions

66