LEÇON 0 : LES SOUS-ALGORITHMES

AYIKPA KACOUTCHY JEAN : Enseignant - Chercheur

Table des matières

Objectifs

Ce cours devra vous permettre de:

- *Décrire* une procédure et *Savoir* faire appelle à une procédure ;
- *F*aire appel à une fonction ;
- *Manipuler* les variables locales et globales ;
- *Savoir* utiliser les paramètres.

Introduction

Lorsque l'on progresse dans la conception d'un algorithme, ce dernier peut prendre une taille et une complexité croissante (lorsqu'il dépasse deux pages il devient difficile à comprendre et à gérer). De même des séquences d'instructions peuvent se répéter à plusieurs endroits. La solution consiste alors à découper l'algorithme en plusieurs parties plus petites. Ces parties sont appelées des sous-algorithmes.

Les sous-algorithmes sont écrit séparément du corps de l'algorithme principal sera appelé par celui-ci quand ceci sera nécessaire.

Il existe deux types de sous-algorithmes à savoir : *les procédures et les fonctions.*

Ossature de l'algorithme principal

Algorithme Nom_algo

< Déclaration des prototypes>

Var

<Déclaration des variables globales>

Début

<Corps de l'algorithme>

Fin

I- Les procédures

Définition : 1.1 Définition

Une procédure est une série d'instructions regroupées sous un nom, qui permet d'effectuer des actions par un simple appel de la procédure dans un algorithme ou dans un autre sous-algorithme.

Syntaxe : 1.2 Déclaration d'une procédure

Procédure nom_proc(liste de paramètres)

Var

Nom_variable : type

Début

Instruction(s)

Finproc

Après le nom de la procédure, il faut donner la liste des paramètres (s'il y en a) avec leurs types respectifs. Ces paramètres sont appelés paramètres formels.

Remarque : Remarque :

Les paramètres sont facultatifs, mais s'il n'y en a pas, les parenthèses doivent rester présentes.

Exemple : Exemple :

Énoncé :

Écrire une procédure qui affiche à l'écran une ligne de 10 arobase puis passe à la ligne suivante.

Solution :

Procédure arobase() *// nom de la procédure ici nous n'avons aucun paramètre*

Var

i : entier *// la variable qui nous permettra de faire la variation de la répétition*

Début

Pour i ←1 à 10 faire *//permet de répéter 10 fois*

Afficher "@" *//affichera @@@@@@@@@@*

FinPour

FinProc

1.3 L'appel d'une procédure

Pour déclencher l'exécution d'une procédure dans un algorithme ou sous-algorithme, il suffit de l'appeler. L'appel de procédure s'écrit en mettant le nom de la procédure, puis la liste des paramètres, séparés par des virgules.

A l'appel d'une procédure, l'algorithme interrompt son déroulement normal, exécute les instructions de la procédure, puis retourne à l'algorithme appelant et exécute l'instruction suivante.

Syntaxe : Syntaxe

Nom_proc(liste de paramètres)

Remarque

Les paramètres utilisés lors de l'appel d'une procédure sont appelés *paramètres effectifs.*

Ces paramètres donneront leurs valeurs aux *paramètres formels.*

Exemple : Exemple :

Pour l'appelle de la procédure de l'exemple ci-dessus il suffit d'appeler le nom de la procédure dans l'algorithme principal :

arobase ()

Exercice

II

[Solution n°1 p 23]

1- Une procédure permet :

- ◯ de regroupé une série d'instructions ramenant un seul résultat
- ◯ de regroupé une série d'instructions ramenant pouvant ramener plusieurs résultat
- ◯ de regroupé une série d'instructions regroupées sous un mon et ramenant un seul résultat

Exercice

[Solution n°2 p 23]

2- Quelle est la syntaxe correcte ?

- ◯ Procédure multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Renvoyer C
  Finproc

- ◯ Procédure multiplication(A,B,C : entier) : entier
  Var
  Début
  C ← A * B
  Renvoyer C
  Finproc

- ◯ Procédure multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Finproc

II- Les fonctions

IV

Définition : 2.1 Définition

La fonction est un sous-algorithme admettant des paramètres et retournant un seul résultat et un seul de type simple qui peut apparaître dans une expression, dans une comparaison, à la droite d'une affectation, etc.

Syntaxe : 2.2 Déclaration d'une procédure

Fonction nom_Fonct (liste de paramètres) : *type*

Var

Nom_variable : type

Début

Instruction(s)

Renvoyer Expression

Fin

La syntaxe de la déclaration d'une fonction est assez proche de celle d'une procédure à laquelle on ajoute un type qui représente le type de la valeur retournée par la fonction et une instruction *Renvoyer* Expression.

Cette dernière instruction renvoie à l'algorithme appelant le résultat de l'expression placée à la suite du mot clé *Renvoyer*.

Remarque : Remarque :

Les paramètres sont facultatifs, mais s'il n'y pas de paramètres, les parenthèses doivent rester présentes.

Exemple : Exemple :

Énoncé :

Définir une fonction qui renvoie le plus grand de deux nombres différents.

Solution :

Fonction Max(X: entier, Y: entier) : entier *//nom de la fonction avec en paramètre les deux variables contenant chacune une valeur*

Début

Si X > Y Alors *// vérifie laquelle des variables et la plus grande*

Renvoyer X *// renvoie le X dans ce cas et sort de la fonction*

Sinon

Renvoyer Y *// renvoie le Y dans ce cas et sort de la fonction*

FinSi

FinFonc

2.3 L'appel d'une fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivi des paramètres effectifs. C'est la même syntaxe qu'une procédure.

A la différence d'une procédure, la fonction retourne une valeur. L'appel d'une fonction pourra donc être utilisé dans une instruction (affichage, affectation, ...) qui utilise sa valeur.

Syntaxe : Syntaxe

VARIABLE ← *Nom_Fonc*(list de paramètres)

Exercice

V

[Solution n°3 p 23]

1- Une fonction permet :

- ◯ de regroupé une série d'instructions ramenant au minimum un seul résultat
- ◯ de regroupé une série d'instructions ramenant pouvant ramener plusieurs résultat
- ◯ de regroupé une série d'instructions regroupées sous un mon et ramenant un seul résultat

Exercice

VI

[Solution n°4 p 23]

2- Quelle est la syntaxe correcte ?

◯ Fonction multiplication(A,B,C : entier)
Var
Début
C ← A * B
Renvoyer C
Finfonc

◯ Fonction multiplication(A,B,C : entier) : entier
Var
Début
C ← A * B
Renvoyer C
Finfonc

◯ Fonction multiplication(A,B,C : entier)
Var
Début
C ← A * B
Finfonc

III- Les Variables locales - globales

VII

3.1- Les variables globales

Une variable déclarée dans la partie déclaration de l'algorithme principale est appelée *variable globale* .

Elle est accessible de n'importe où dans l'algorithme, même depuis les procédures et les fonctions.

Elle existe pendant toute la durée de vie du programme.

3.2- Les variables locales

Une variable déclarée à l'intérieur d'une procédure (ou une fonction) est dite *variable locale.*

Elle n'est accessible qu'à la procédure au sein de laquelle elle est définie, les autres procédures n'y ont pas accès.

La durée de vie d'une variable locale est limitée à la durée d'exécution de la procédure (la fonction).

Exercice

VIII

[Solution n°5 p 24]

Une variable est dite locale lorsqu'elle

- ◯ est accessible dans tout l'algorithme
- ◯ a une valeur fixe dans un sous algorithme
- ◯ n'est accessible qu'au sous algorithme dans laquelle elle a été créée.

VI- Les paramètres

IX

Définition : 4.1- Définition

Les *paramètres* sont les voies de transmission des données entre les sous-algorithmes.

Son rôle principal est de transmettre des informations entre le programme appelant et la fonction ou la procédure appelée.

4.2- Catégories des paramètres

Il existe deux catégories de paramètre à savoir les paramètres *formels* et *effectifs*

Définition : 4.2.1- Les paramètres Formels

Les *paramètres formels* sont des paramètres dont leurs valeurs ne sont pas connues lors de la création de la procédure ou de la fonction.

Définition : 4.2.2- Les paramètres Effectifs

Les *paramètres effectifs ou réels* sont des paramètres utilisés lors de l'appel de la procédure ou de la fonction.

4.3- Les Types de paramètres

Il existe trois types de paramètre à savoir les paramètres en *entrée*, les paramètres en *sortie* et les paramètres en *entrée / sortie.*

Définition : 4.3.1- Paramètre en entrée

On dit qu'un paramètre est en *entrée lorsqu'il reçoit une valeur au départ et qu'il ne peut pas retourner une autre valeur durant l'exécution du sous-algorithme.* On le note *{E}.*

Définition : 4.3.2- Paramètre en sortie

On dit qu'un paramètre est en *sortie lorsqu'il n'a aucune valeur au départ et qu'il va retourner une valeur durant l'exécution du sous-algorithme.* On le note *{S}.*

Définition : 4.3.3- Paramètre en entrée / sortie

On dit qu'un paramètre est en *entrée / sortie lorsqu'il peut recevoir une valeur au départ et retourner une autre valeur durant l'exécution du sous-algorithme. On le note {E/S}.*

4.4- Passage de paramètres

Les échanges d'informations entre sous-algorithme se font par l'intermédiaire de paramètres.

Il existe trois modes de passages de paramètres qui permettent des usages différents : *Passage par valeur, par référence (ou par adresse) et par résultat.*

Définition : 4.4.1- Passage par valeur

Dans ce type de passage, le paramètre formel reçoit uniquement une copie de la valeur du paramètre effectif. *La valeur du paramètre effectif ne sera jamais modifiée.*

Autrement dit à l'appel de la fonction , *les paramètres formels sont remplacés par les valeurs réelles. Toute modification des paramètres formels dans le sous-algorithme n'entraîne aucune modification des valeurs réelles (paramètres effectifs) à la sortie.*

Définition : 4.4.2- Passage par référence ou par adresse

Dans ce type de passage, la procédure ou la fonction utilise l'adresse du paramètre effectif. *Lorsqu'on utilise l'adresse du paramètre, on accède directement à son contenu.* Ainsi, à l'appel de la fonction ou de la procédure, les paramètres formels sont remplacés par les valeurs réelles. *Toute modification des paramètres formels dans le sous-algorithme modifie les valeurs réelles (paramètres effectifs) à la sortie.*

Définition : 4.4.3- Passage par résultat

Ce mode de passage est identique au mode de passage par adresse sauf qu'à l'appel de la fonction ou de la procédure, elle ne s'attend pas à recevoir de paramètres effectifs, ceux-ci sont générés par le sous-algorithme et remplace donc les paramètres formels.

Exercice

[Solution n°6 p 24]

Calcul(X, Y , Z)

De quel type de paramètres (X, Y , Z) s'agit-il ?

- ◯ Formels
- ◯ Effectifs

Exercice

XI

[Solution n°7 p 24]

Quel type de paramètre dois je utiliser lorsque mon sous algorithme doit recevoir une valeur de départ et ensuite ramené une autre valeur après exécution du sous algorithme ?

- ◯ Paramètre en entrée : {E}
- ◯ Paramètre en sortie : {S}
- ◯ Paramètre en entrée / sortie : {E/S}

V- Applications

XII

Exemple : Exercice 1 :

Énoncé :

En utilisant les sous-algorithmes, écrire un algorithme qui permet de faire la somme de deux nombres.

Résolution :

// Dans notre exercice nous allons diviser notre algorithme en trois parties

Tout d'abord il faut créer un sous-algorithme pour la récupération des données. Étant donné que nous devrions avoir plusieurs résultats nous allons utiliser une procédure

Procédure recupdonne({S} nbre1, nbre2 : entier) *//Entête de la procédure comportant le nom et les différents paramètres en sortie car nbre1et nbre2 n'ont aucune valeur de départ*

Var *// Ici nous n'avons pas de variable locale*

Début

Afficher "Donner le premier nombre" *// Envoi le message : Donner le premier nombre*

Saisir nbre1 *// Récupère la valeur saisie dans la variable nbre1*

Afficher "Donner le deuxième nombre" *// Envoi le message : Donner le deuxième nombre*

Saisir nbre2 *// Récupère la valeur saisie dans la variable nbre2*

Fin

//Ensuite nous allons utiliser un autre sous-algorithme qui permettra de calculer la somme des deux nombres, en occurrence nous allons utiliser une fonction car la somme de deux nombres retourne un résultat.

Fonction somme({E} nbre1, nbre2 : entier) : entier *//Entête de la fonction comportant le nom et les différents paramètre en entrée car nbre1 et nbre2 récupèrent une valeur de départ*

Var

Som : entier *//variable locale à la fonction qui récupère le résultat de la Som*

Début

Som ← nbre1 + nbre2 *// fait le calcul de la somme des deux variables*

Renvoyer Som *// retourne le résultat obtenu*

Fin

// Enfin nous allons utiliser un sous-algorithme pour afficher le résultat, ici nous allons utiliser une procédure pour faire l'affichage

Procédure affichage({E}nbre1,nbre2,result : entier) *//Entête de la procédure comportant le nom et les différents paramètre en entrée car nbre1,nbre2 et result auront valeur de départ qu'on affichera*

Var

Début

Afficher nbre1," + ",nbre2," = ", result *// affichera l'opération suivie du résultat*

Fin

// L'Algorithme Principal

Algorithme Somme

//Prototype des différents sous-algorithme (les différents entêtes des sous-algorithmes)

Procédure recupdonne({S} nbre1, nbre2 : entier)

Fonction somme({E} nbre1, nbre2 : entier) : entier

Procédure affichage({E} nbre1, nbre2, result : entier)

Var

X,Y,Z : entier *// déclaration des variables globales puisqu'il s'agit de faire la somme il nous faut trois variable, deux variable pour récupérer les opérantes et une autre pour le résultat*

Début

recupdonne(X,Y) *// appel la procédure pour récupérer les données du calcul, ici il s'agit de l'appelant qui fait appelle à la procédure appelée*

//X va récupérer la valeur de nbre1 dans la procédure appelée

//Y va récupérer la valeur de nbre2 dans la procédure appelée

//Les paramètres effectifs reçoivent leur valeur selon la position des paramètres formels

Z ← somme(X,Y) *// appel de la fonction pour faire l'opération somme avec les variables X et Y qui vont donner leur valeur selon leurs positions au variables formels nbre1 et nbre2. Une fois les variables récupérées la fonction somme fera le calcul et ramènera le résultat à la variable Z*

Affichage(X,Y,Z) *//appel la procédure affichage qui va donner les valeurs aux variable nbre1,nbre2 et result en respectant la position des variables effectifs*

Fin

Définition : Exercice 2 :

Énoncé :

En utilisant les sous-algorithmes, écrire un algorithme qui permet de résoudre une équation du seconde degré dans R.

Résolution :

// Dans notre exercice, pour résoudre cette équation il nous faut d'abord :

1. *Avoir les différentes valeurs des coefficients de l'équation (Ax^2 +Bx +C = 0) à savoir A,B, et C*
2. *Calculer le discriminant de l'équation*
3. *Calculer les valeurs de X1 et X2 en fonction du résultat du discriminant*
4. *Donner la ou les solution(s) de l'équation s'il y en a .*

//Pour résoudre ce problème, il nous faut tout d'abord récupérer les différentes valeurs mentionnées du point (1), ce qui donnera lieu de créer une procédure qui récupère les coefficients

Procédure recupcoef({S} A, B, C : Réel) *//Entête de la procédure comportant le nom et les différents paramètres en sortie car A,B et C n'ont aucune valeur de départ*

Var *// Ici nous n'avons pas de variable locale*

Début

Afficher "Donner la valeur du coefficient A" *// Envoi le message : Donner la valeur du coefficient A*

Saisir A *// Récupère la valeur saisie dans la variable A*

Afficher "Donner la valeur du coefficient B" *// Envoi le message : Donner la valeur du coefficient B*

Saisir B *// Récupère la valeur saisie dans la variable B*

Afficher "Donner la valeur du coefficient C" *// Envoi le message : Donner la valeur du coefficient C*

Saisir C *// Récupère la valeur saisie dans la variable C*

Fin

// Ensuite en regardant le point (2) il faut calculer le discriminant, étant donné que le calcul du discriminant donne un seul résultat nous allons utiliser une fonction.

Fonction calculdiscri({E} A, B, C : Réel) : Réel*//Entête de la fonction comportant le nom et les différents paramètre en entrée car car A,B et C récupèrent une valeur de départ*

Var

disc : Réel*//variable locale à la fonction qui récupère le résultat de la disc*

Début

disc ← (B * B - 4*A*C *) // Calcul du discriminant*

Renvoyer disc *// retourne le résultat obtenu*

Fin

// Concernant le points (3) nous allons calculer les différents résultats de X1 et X2 mais les résultats sont conditionnés par la valeurs du discriminant. Nous avons plusieurs résultats de ce fait nous allons utiliser une procédure

Procédure calculsolution({E} A, B, C, Discri : Réel,{S}X1, X2 : Réel) *//Entête de la procédure comportant le nom et les différents paramètres en entrée car A,B,C et Discri récupère une valeur de départ tandis que X1 et X2 sont en sortie car elles n'ont aucune valeur de départ*

Var *// Ici nous n'avons pas de variable locale*

Début

Si Discri > 0 Alors *// vérifie pour voir si le discriminant est supérieur à zéro si oui alors il calcule les solutions X1 et X2.*

X1 ←(-B - sqrt(Discri))/ 2*A *// calcule la solution X1*

X2 ←(-B + sqrt(Discri))/ 2*A *// calcule la solution X2*

Sinon

SI Discri = 0 Alors *// vérifie pour voir si le discriminant est égal à zéro si oui alors il calcule une solution double X1 = X2*

X1 ←(-B)/ 2*A *// calcule X1*

X2 ←X1 *// affecte la valeur X1 à X2*

Finsi

Finsi

// le cas où le discriminant est inférieur à zéro ne peut pas être énuméré ici car aucun calcule n'est fait donc il sera énuméré lors de la procédure d'affichage

Fin

//Enfin pour le point (3) nous allons afficher les différentes solutions, dans ce cas on utilisera une procédure

Procédure affichage({E} Discri, X1, X2 : Réel) *//Entête de la procédure comportant le nom et les différents paramètres en entrée car X1 et X2 Discri récupère une valeur de départ*

Var *// Ici nous n'avons pas de variable locale*

Début

Si Discri >0 Alors *// Vérifie si le discriminant est supérieur à zéro donc il aura deux solutions à afficher X1 et X2*

Afficher " X1 = ",X1 *// Affiche le résultat de X1*

Afficher " X2 = ",X2 *// Affiche le résultat de X2*

Sinon

Si Discri = 0 Alors *// Vérifie si le discriminant est égal à zéro donc il aura une solution double X1 = X2*

Afficher "X1 = X2 = ",X1 *// Affiche le résultat de la solution double*

Sinon *// Vérifie si le discriminant est inférieur à zéro donc il n'y a aucune solution dans R*

Afficher " Pas de solution dans R " *// Affiche Pas de solution dans R*

Finsi

Finsi

Fin

// L'Algorithme Principal

Algorithme Equation

//Prototype des différents sous-algorithme (les différents entêtes des sous-algorithmes)

Procédure recupcoef({S} A, B, C : Réel)

Fonction calculdiscri({E} A, B, C : Réel) : Réel

Procédure calculsolution({E} A, B, C, Discri : Réel,{S} X1, X2 : Réel)

Procédure affichage({E} Discri, X1, X2 : Réel)

Var

CoefA, CoefB, CoefC, Dis, S1,S2 : entier *// déclaration des variables globales puisqu'il s'agit de résoudre une équation du second dans R*

Début

recupcoef(CoefA,CoefB,CoefC) *// appel la procédure pour récupérer les coefficients de l'équation, ici il s'agit de l'appelant qui fait appelle à la procédure appelée*

//CoefA va récupérer la valeur de A dans la procédure appelée

//CoefB va récupérer la valeur de B dans la procédure appelée

//CoefC va récupérer la valeur de C dans la procédure appelée

//Les paramètres effectifs reçoivent leur valeur selon la position des paramètres formels

Dis ← calculdiscri(CoefA,CoefB,CoefC) *// appel de la fonction pour faire l'opération du calcul du discriminant avec les variables CoefA, CoefB et CoefC qui vont donner leurs valeurs selon leurs positions au variables formels A,B et C. Une fois les variables récupérées la fonction calculdiscri fera le calcul et ramènera le résultat à la variable Dis*

calculsolution(CoefA,CoefB,CoefC,Dis,S1,S2) *//appel de la procédure pour calculer les solutions de l'équation. Les variables CoefA, CoefB et CoefC qui donneront leurs valeurs selon leurs positions aux variables formels A,B et C. les variables S1 et S2 n'auront pas de valeur de départ mais par la suite lorsque la procédure sera exécutées S1 aura la valeur de X1 et S2 la valeur de X2*

affichage(Dis,S1, S2) *//appel la procédure affichage se chargera d'afficher le résultat. Les variables Dis,S1 et S2 donneront leurs valeurs selon leurs positions aux variables formels Discri, X1 et X2.*

Fin

Solutions des exercices

> Solution n°1

Exercice p. 7

- ○ de regroupé une série d'instructions ramenant un seul résultat
- ◉ de regroupé une série d'instructions ramenant pouvant ramener plusieurs résultat
- ○ de regroupé une série d'instructions regroupées sous un mon et ramenant un seul résultat

> Solution n°2

Exercice p. 8

- ○ Procédure multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Renvoyer C
  Finproc
- ○ Procédure multiplication(A,B,C : entier) : entier
  Var
  Début
  C ← A * B
  Renvoyer C
  Finproc
- ◉ Procédure multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Finproc

> Solution n°3

Exercice p. 11

- ○ de regroupé une série d'instructions ramenant au minimum un seul résultat
- ○ de regroupé une série d'instructions ramenant pouvant ramener plusieurs résultat
- ◉ de regroupé une série d'instructions regroupées sous un mon et ramenant un seul résultat

Exercice p. 12

> Solution n°4

- ○ Fonction multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Renvoyer C
  Finfonc
- ◉ Fonction multiplication(A,B,C : entier) : entier
  Var
  Début
  C ← A * B
  Renvoyer C
  Finfonc
- ○ Fonction multiplication(A,B,C : entier)
  Var
  Début
  C ← A * B
  Finfonc

> Solution n°5

Exercice p. 14

- ○ est accessible dans tout l'algorithme
- ○ a une valeur fixe dans un sous algorithme
- ◉ n'est accessible qu'au sous algorithme dans laquelle elle a été créée.

> Solution n°6

Exercice p. 17

- ○ Formels
- ◉ Effectifs

> Solution n°7

Exercice p. 18

- ○ Paramètre en entrée : {E}
- ○ Paramètre en sortie : {S}
- ◉ Paramètre en entrée / sortie : {E/S}