Fonctions et procédures

I2 - Algorithmique et Programmation structurée

Julien Saunier, Alexandre Pauchet, Pierrick Tranouez

Rappels

Analyse descendante

Principe

Décomposer le code en unités de traitement génériques

Décomposition de code

- Éviter la redondance
- Alléger le code / niveaux d'imbriquations
- Améliorer la lisibilité
- Éviter les erreurs

Il existe deux types de sous-programmes :

- les fonctions
- les procédures

Fonction

Exemple

fonction minimum2 (a,b : **Entier**) : **Entier**
    Déclaration res : **Entier**
debut
    si a $\geq$ b **alors**
        res $\leftarrow$ b
    sinon
        res $\leftarrow$ a
    finsi
    retourner res
fin

Les fonctions 1 / 4

- Les fonctions sont des sous-programmes admettant des paramètres d'entrée et retournant **un ou plusieurs** résultats (comme les fonctions mathématiques y=f(x,y,...))
  - les paramètres d'entrée sont en nombre fixe (≥ 0)
  - une fonction possède un ou plusieurs type de valeur retournée
  - la valeur de retour est spécifiée par l'instruction **retourner**
- Généralement le nom d'une fonction est soit un nom (par exemple *minimum*), soit une question (par exemple *estVide*)

Les fonctions 2 / 4

On déclare une fonction de la façon suivante :

fonction *nom de la fonction* **(***paramètre(s) d'entrée de la fonction***) :** *types des valeurs retournées*

Déclaration *variable(s) locale(s)*

debut

instructions de la fonction avec au moins une fois l'instruction

retourner *valeur(s)*

fin

Les fonctions 3 / 4

fonction differenceEntreDates **(**mois1, mois2 : **Naturel** ; an1, an2 : **Entier) : Entier**

Déclaration difference : **Entier**

debut

difference ← an2 - an1

difference ← difference * 12

difference ← difference + (mois2 - mois1)

retourner difference

fin

Les fonctions 4 / 4

Signature d'une fonction

La signature d'une fonction contient

- Le mot-clé fonction
- Le nom de la fonction
- Les paramètres formels d'entrée
- Le(s) type(s) de retour

fonction differenceEntreDates (mois1, mois2 : Naturel ; an1, an2 : Entier) : Entier

Exercice

Ecrire une fonction prenant en entrée un tableau de réels et le nombre d'éléments qu'il contient et renvoyant les valeurs minimum et maximum contenues.

Appel de fonction

Appel

L'exécution d'une fonction A se fait par un appel dans un sous-programme B.

- l'appel se fait par une instruction en utilisant le nom de la fonction A ;
- chaque paramètre d'entrée de A se voit affecté une valeur lors de l'appel ;
- à la fin de l'exécution de A, celle-ci reprend à l'instruction suivant l'appel dans le sous-programme B ;
- la ou les valeurs retournées peuvent être utilisé dans B pour une affectation, un calcul, ...

Exercice

Ecrire un programme principal permettant la saisie d'éléments dans un tableau de réels, puis affichant les valeurs minimales et maximales.

Exemple complet 1 / 5

Validité d'une date

Écrire

- une fonction qui renvoie le nombre de jours d'un mois donné (en représentant un mois par un naturel) ;
- une fonction qui détermine si une date, exprimée par un jour et un mois (par des naturels), est valide ;
- un programme principal qui demande la saisie d'une date et vérifie qu'elle est valide.

On ne tiendra pas compte des années bissextiles dans ce programme.

Exemple complet 2 / 5

```
fonction nombreJours (unMois : Naturel) : Naturel
    Déclaration  res : Naturel
debut
    si (unMois = 2) alors
        res ← 28
    sinon
        si (unMois = 4) ou (unMois = 6) ou (unMois = 9) ou (unMois = 11)
        alors
            res ← 30
        sinon
            res ← 31
        finsi
    finsi
    retourner res
fin
```

Exemple complet 3 / 5

```
fonction estUneDateValide (unJour, unMois : Naturel) : Booleen
debut
    si (unMois > 0) et (unMois ≤ 12) alors
        retourner (unJour > 0) et (unJour ≤ nombreJours(unMois))
    finsi
    retourner Faux
fin
```

Exemple complet 4 / 5

```
Déclaration: jour1, mois1 : Naturel
debut
    lire(jour1, mois1)
    si estUneDateValide(jour1,mois1) alors
        ecrire('La date est valide')
    sinon
        ecrire('La date n'est pas valide')
    finsi
fin
```

Question

Comment modifier la fonction nombreJours pour tenir compte des années bissextiles ? (ne donnez que la signature)

Exemple complet 5 / 5

2 possibilités

- `nombreJours(unMois : Naturel; uneAnnee : Entier) : Entier`
- `nombreJours(unMois : Naturel; bissextile : Booleen) : Entier`

Paramètres

Paramètres formels Variables donnant le type mais non la valeur d'exécution (e.g. leJour, leMois : Integer)

Paramètres effectifs Valeurs ou variables concrètes données lors de l'exécution (e.g. jour1, mois 1 qui possèdent une valeur suite aux readln)

Les fonctions en Pascal 1 / 2

Déclaration et définition

déclaration C'est donner la signature d'une fonction pour une définition ultérieure

définition C'est donner le corps de la fonction

- Souvent on déclare et définit en même temps
- Si on veut déclarer puis définir (cas de sous-programmes qui s'appellent mutuellement), il faut faire suivre la déclaration du mot clé forward

Attention : une fonction en Pascal ne peux retourner qu'une seule valeur.

Les fonctions en Pascal 2 / 2

Syntaxe

```
function nom_fonction(variable1 : type1; ..) : type_retour;
  declarations_locales
begin
  ...
  nom_fonction:=....
end
```

sommeDesDiviseurs

```
function estUneDateValide(unJour, unMois : Integer) : Boolean;
begin
   if (unMois >0) and (unMois <= 12) then
      estUneDateValide := (unJour > 0) and
           (unJour <= nombreJours(unMois))
   else estUneDateValide := false;
end;
```

procédures

Les Procédures 1 / 1

- On déclare une procédure de la façon suivante :
  procédure *nom de la procédure* **(E** *paramètre(s) en entrée* **;S** *paramètre(s) en sortie* **;E/S** *paramètre(s) en entrée/sortie***)**

  Déclaration *variable(s) locale(s)*

  debut

  instructions de la procédure

  fin
- On appelle une procédure comme une fonction, en indiquant son nom suivi des paramètres effectifs entre parenthèses

Paramètres d'entrée/sortie 1 / 3

Paramètres d'entrée :

- valeurs nécessaires à l'exécution d'un algorithme, mais non modifiées (ou modifiables)

Paramètres de sortie :

- servent à stocker les valeurs produites

Paramètres d'entrée/sortie :

- Valeur initiale nécessaire au fonctionnement de l'algorithme
- et potentiellement modifiée par l'algorithme

Paramètres d'entrée/sortie 2 / 3

```
procédure ajoutValeur (E r : Reel ;E/S tab :  Tableau[1..MAX] de
Reel ; taille : Entier)
debut
    si taille < MAX alors
        taille ← taille+1
        tab[taille] ← r
    finsi
fin
```

Signature d'une procédure

Définition

La **signature** d'une procédure la désigne de manière unique dans un programme (on ne peut pas avoir 2 procédures de même signatures) et se compose :

- de son nom
- de ses paramètres formels (entrée, sortie, entrée/sortie)

Exemples

- `procedure ajoutValeur (E r : Reel;E/S tab: Tableau[1..MAX] de Reel ; taille : Entier)`

Appel de procédure

Appel

L'exécution d'une procédure A se fait par un appel dans un sous-programme B.

- l'appel se fait par une instruction en utilisant le nom de la procédure ; chaque paramètre d'entrée de A se voit affecté une valeur lors de l'appel ;
- chaque paramètre de sortie et d'entrée/sortie de A est associé à une variable lors de l'appel ;
- à la fin de l'exécution de A, celle-ci reprend à l'instruction suivant l'appel dans le sous-programme B ;
- les variables correspondant aux paramètres de sortie et d'entrée/sortie prennent la valeur affectée par la procédure A, ...
- les variables correspondant aux paramètres d'entrée ne peuvent pas être modifiées

Paramètres d'entrée/sortie 3 / 3

Utiliser la procedure ajoutValeur pour initialiser les valeurs d'un tableau

Exemple 1 / 5

Une étape du jeu du 421

Écrire

- une procédure qui affecte une valeur initiale aléatoire entre 1 et 6 à 3 nombres entiers (simulant un jet de 3 dés)
- une procédure qui propose à l'utilisateur de relancer un dé et indique s'il l'a fait
- une procédure qui prend comme paramètres 3 valeurs de dés et demande à l'utilisateur quels dés il souhaite relancer ; une nouvelle valeur aléatoire est affectée aux dés relancés et une valeur booléenne fournie par la procédure indique si aucun dé n'est relancé
- un programme principal qui effectue un premier jet de dés et propose à l'utilisateur de relancer certains dés au maximum 2 fois

Exemple 2 / 5

procédure jetInitial **(S** de1, de2, de3 : **Naturel)**
debut
    de1 $\leftarrow$ random(6) + 1
    de2 $\leftarrow$ random(6) + 1
    de3 $\leftarrow$ random(6) + 1
fin

Exemple 3 / 5

```
procédure relancerUnDe (E/S de : Naturel ; stable : Booleen)
    Déclaration  relance : Caractere
debut
    ecrire('Voulez-vous relancer le dé (o/n) ?')
    lire(relance)
    si relance = 'o' alors
        de ← random(6) + 1
        stable ← Faux
    finsi
fin
```

Exemple 4 / 5

```
procédure relancer (E/S de1, de2, de3 : Naturel ; S stable : Booleen)
debut
    stable ← Vrai
    ecrire('Premier de')
    relancerUnDe(de1, stable)
    ecrire('Deuxieme de')
    relancerUnDe(de2, stable)
    ecrire('Troisieme de')
    relancerUnDe(de3, stable)
fin
```

Exemple 5 / 5

```
Déclaration: d1, d2, d3, nb :Naturel ; stop : Booleen
debut
    jetInitial(d1, d2, d3)
    nb ← 0
    stop ← Faux
    repeter
        ecrire('de 1 = ', d1, ' ; de 2 = ', d2, ' ; de 3 = ', d3)
        relancer(d1, d2, d3, stop)
        si non stop alors
            nb ← nb + 1
        finsi
    jusqu'a ce que (nb = 2) ou (stop = Vrai)
    ecrire('Jet final : de 1 = ', d1, ' ; de 2 = ', d2, ' ; de 3 = ', d3)
fin
```

Les procédures en Pascal

Déclaration et définition

déclaration C'est donner la signature d'une procédure pour une définition ultérieure

définition C'est donner le corps de la procédure

On déclare souvent les procédures par leur définition. On peut aussi déclarer la procédure avant sa définition avec le mot clé forward

Paramètres

- d'entrée : uniquement avec un identifiant et un type
- de sortie : on utilise le mot-clé `var` avant l'identifiant suivi du type
- d'entrée/sortie : idem que les paramètres de sortie

Définition de procédures

Syntaxe

```
procedure nom_procedure([var]variable1 : type1; ..);
declarations_locales
begin
...
end
```

echanger deux entiers

```
procedure swap(var a,b : Integer);
var temp :  Integer;
begin
    temp:=a;
    a:=b;
    b:=temp
end;
```

Quelques procédures standards en Pascal

Procédures d'entrée et sortie standard

WRITE et WRITELN Cette procédure permet d'afficher à l'écran des valeurs de type Boolean, Integer, Real, Char et String. Si il y a plusieurs valeurs, elle sont séparées par des ,

READLN Cette fonction permet à l'utilisateur de saisir des valeurs à l'aide du clavier. Les types acceptés sont Integer, Real, Char et String

Fonctions et procédures : Portée des variables

Visibilité

- La **portée** d'une variable correspond aux emplacements du programme où elle est visible, c-à-d utilisable.
- Les variables déclarées localement à une procédure ont pour seule portée cette procédure
- Une déclaration locale se fait
  - dans les paramètres d'une procédure
  - dans le champ `Déclaration` après la signature de la procédure
  - dans le champ `Déclaration` avant le début du programme principal

Fonction ou procédure ? 1 / 2

- Une fonction
  - ne modifie jamais de valeurs du sous-programme l'appelant
  - effectue un calcul et retourne une ou plusieurs valeurs
  - est assimilable à une valeur
- Une procédure
  - permet de modifier l'état du programme (variables, affichage...)
  - peut ne pas avoir de sortie
  - n'est pas assimilable à une valeur

Fonction ou procédure ? 2 / 2

- Une fonction est utilisée pour
  - un calcul déterministe
  - améliorer la lisibilité du programme (pas de modification des paramètres d'entrée, utilisation des valeurs de retour explicite dans le sous-programme appelant
- Une procédure est utilisée pour
  - modifier l'état du programme (variables effectives, affichage...)
  - un sous-programme non déterministe, principalement les interactions avec l'utilisateur