Algorithmique fonctionnelle - Fonctions

2 octobre 2009

Fonctions

Relation binaire

Le **produit cartésien** de 2 ensembles A et B est l'ensemble $A \times B$ ayant pour éléments les couples (a, b) avec $a \in A$ et $b \in B$.

Une **relation binaire** définie sur A et B est une partie du produit cartésien $A \times B$ notée aRb.

Fonctions

Fonction

- Etant donné 2 ensembles, l'ensemble de départ E et l'ensemble d'arrivée F, une fonction f de E vers F est la relation binaire associant à chaque élément x de l'ensemble de départ un élément $f(x)$ de l'ensemble d'arrivée
- on appelle alors f, une application de E dans F
- E est appelé **domaine**, F, **codomaine**
- $E \longrightarrow F$ est la **signature** de la fonction f

On note, par exemple :

$$f : \mathbb{R} \longrightarrow \mathbb{R}$$
$$x \longrightarrow x * x$$

Fonctions

En informatique

Une fonction est une suite d'instructions qui peut prendre ensemble de paramètres en entrée et qui produit (retourne) un résultat à la fonction appelante

Notation

Fonction nomDeLaFonction(liste de paramètres) : **type retourné**
Variables
  variables locales
Début
  instructions
  Retourner(resultat)
Fin

Fonctions

Notation

Signature de la fonction :
calculCarre : Entier $\longrightarrow$ Entier

Fonction calculeCarre(x : Entier) : Entier
Variables
  res : Entier
Début
  res $\leftarrow$ x*x
  Retourner(res)
Fin

Fonctions

Conditions

On notera, en en-tête, les conditions :

- pour les variables en entrée : **préconditions**
- pour l'élément retourné : **postconditions**

Fonctions

Notation

calculCarre : Entier $\longrightarrow$ Entier
precondition : x est un Entier
postcondition : valeur au carré de x

Fonction calculeCarre(x : Entier) : Entier
Variables
    res : Entier
Début
  res $\leftarrow$ x*x
  Retourner(res)
Fin

Fonctions

Fonctions pré-existantes

- Numérique
  - ent : Réel $\longrightarrow$ Entier
  - sqrt : Entier $\longrightarrow$ Réel
  - sin : Réel $\longrightarrow$ Réel
  - cos : Réel $\longrightarrow$ Réel
  - alea : Entier $\longrightarrow$ Entier
  - ...

Fonctions

Fonctions pré-existantes

- Texte
  - longueur : Châîne $\longrightarrow$ Entier
  - extrait : Châîne $\times$ entier $\times$ Entier $\longrightarrow$ chaine
  - ascii : Châîne $\longrightarrow$ Entier
  - caractere : Entier $\longrightarrow$ Châîne

Fonctions

Fonctions

Une fonction **retourne** un résultat, il est donc nécessaire de récupérer ce résultat dans une variable :

$$res \leftarrow calculCarre(12)$$

Fonctions

Fonction principale

On appelle **fonction principale**, la 1re fonction appelée, celle qui appellera les autres.

Notation

Programme NomDuProgramme
Variables
  variables locales
Début
  instructions
Fin

Fonctions

Exemple

Programme TestCarre
Variables
  res : Entier
Début
  res ← calculCarre(12)
  Ecrire(res)
Fin

Fonctions

Définitions

- le déroulement d'un programme en langage fonctionnel consiste en l'application d'une/plusieurs fonctions à ses/leurs arguments
- le programme appelle une fonction qui appelle une fonction qui appelle...

Variables locales et globales

Définitions

- **locale** : déclarée au sein d'une fonction visible uniquement par celle-ci
- **globale** : visible dans tout le programme, par l'ensemble des fonctions

Variables locales et globales

Exemple 1 : Variable locale

Fonction maFonction(y : Entier) : Entier
Variables
  tmp : Entier
Début
  tmp ← y*3
  Retourner(tmp)
Fin
Programme Test
Variables
  res, x : Entier
Début
  x ← 7
  res ← maFonction(x)
  Ecrire(res) //affiche quoi ?
Fin

Variables locales et globales

Exemple 2 : Variable locale (passée par valeur !)

Fonction maFonction(y : Entier) : Entier
Variables
  tmp : Entier
Début
  tmp ← y+2
  y ← 8
  Retourner(tmp)
Fin

Programme Test
Variables
  res, x : Entier
Début
  x ← 7
  res ← maFonction(x)
  Ecrire(res) //affiche quoi ?
  Ecrire(x) // 7 ou 8 ?
Fin

Variables locales et globales

Exemple 3 : Variable globale

Variables global : Entier

Fonction maFonction(y : Entier) : Entier
Variables
  tmp : Entier
Début
  tmp ← y+2
  global ← 8
  Retourner(tmp)
Fin

Programme Test
Variables
  res, x : Entier
Début
  global ← 7
  res ← maFonction(global)
  Ecrire(res) //affiche quoi ?
  Ecrire(global) // 7 ou 8 ?
Fin

Algorithme de qualité

Règles à respecter

1. Objectifs
   - Architecture claire
   - Modules concis et spécifiques
   - Modules testables et modifiables de manière indépendante
2. Mise en oeuvre
   - Limiter un maximum les variables (notament les globales)
   - Adapter ses modules aux unités fonctionnelles de votre application
   - Découper son algorithme pour le rendre léger, lisible et performant

Avantages et inconvénients

- Avantages
  - simple, intuitif
  - proche d'une description mathématique (récurrence)
  - facile à analyser
  - bien adapté à certaines structures de données
- Inconvenients
  - gourmand en ressources (utilisation d'une pile de stockage des informations)

Exercices

- Écrivez une fonction qui réalise la somme de 3 entiers passés en paramètre
- Écrivez un programme qui initialise 3 entiers, appelle la somme et affiche le résultat
- Écrivez une fonction qui réalise la somme de 6 entiers passés en paramètre
- Écrivez une fonction qui calcule le min de 2 entiers passés en paramètre
- Écrivez une fonction qui réalise le min de 3 entiers passés en paramètre