Fonctions

Définitions

Fonction

- Une fonction est un **algorithme indépendant**
- L'appel avec ou sans paramètres de la fonction déclenche l'exécution de son bloc d'instructions
- Une fonction se termine **en retournant ou non une valeur**

Procédure

- Une procédure est une fonction qui **ne retourne aucune valeur**

Structure d'une fonction

Fonction nomFonction(liste des paramètres) : typeRetourné

Début

Bloc d'instructions

Fin

Pour faciliter la lecture des algorithmes, on retiendra les principes suivant en pseudo langage :

- Le nom d'une fonction commence par une minuscule
- Le nom d'une fonction ne comporte pas d'espace
- Si le nom d'une fonction est composé de plusieurs mots, on fera commencer chaque mot par une majuscule (à l'exception bien sûr du 1er mot qui reste en minuscule) et on évitera les traits d'union dont l'utilisation est réservée au nom des algorithmes principaux. Exemple : calculDeRemise

Exécution d'une fonction

- Le programme principal s'interrompt pour appeler la fonction
- La fonction exécute son bloc d'instructions
- Elle s'arrête dès qu'une instruction *Retourne* est exécutée
- Sinon elle s'arrête à la fin de son bloc d'instructions
- Le programme appelant reprend alors son exécution

Pourquoi écrire des fonctions ?

- Le code de l'algorithme principal est **plus simple** et **plus court** s'il fait appel à des fonctions
- Une fonction peut être **appelée plusieurs fois**
- **Une seule modification** dans une fonction est **automatiquement répercutée** pour tous les algorithmes appelant cette fonction
- Une fonction peut être conçue comme un **outil mis à disposition d'autres développeurs**

Fonction sans valeur retournée

On précise que la fonction retourne « vide ».

Exemple : fonction d'affichage du mot « Bonjour »

```
Fonction afficheBonjour() : vide
Début
    Écrire (« Bonjour ») ;
Fin
```

La fonction est utilisée par un algorithme tel que celui-ci :

```
Algorithme utiliser-fonction
Début
    afficheBonjour() ;
Fin
```

Fonction avec valeur retournée

Une fonction peut retourner une valeur au programme appelant

Cette valeur est unique et son retour signifie l'arrêt de la fonction

Exemple :

Fonction lireNote() : entier

Variables : note : entier ;

Début

    Écrire (« Merci de saisir une note ») ;

    Lire (note) ;

    Retourne (note) ;

Fin

Les paramètres

Un paramètre est une variable **locale** à une fonction

Au début de la fonction il possède la valeur passée par le programme appelant

Exemple de passage de paramètres :

```
Fonction maxDeDeuxValeurs(p1 : entier, p2 : entier) : entier
Variables : résultat : entier
Début
    Si p1 < p 2 alors
        résultat ← p2 ;
    Sinon
        résultat ← p1 ;
    Fin si
    Retourne (résultat) ;
Fin
```

Les paramètres

Cette fonction est appelée par l'algorithme suivant :

```
Algorithme utiliser-fonction-maxDeDeuxValeurs
Variables : valeur1, valeur2, maxi: entiers ;
Début
    Écrire (« Saisir une 1ère valeur ») ;
    Lire (valeur1) ;
    Écrire (« Saisir une 2ème valeur ») ;
    Lire (valeur2) ;
    maxi ←maxDeDeuxValeurs(valeur1, valeur2) ;
    Écrire (« La plus grande valeur est : », maxi) ;
Fin
```

Les paramètres

- On **dissocie** complètement ce que l'on appelle les **environnements de données**, à savoir les variables utilisées dans l'algorithme principal et celles qui le sont dans une fonction
- Elles n'ont **aucune existence commune**
- Celles utilisées dans la fonction n'existent qu'entre le début et la fin d'exécution de la fonction
- En mémoire cela correspond à des **adressages indépendants**
- Il n'existe aucun moyen pour l'algorithme principal d'avoir accès aux variables de la fonction, ni à la fonction d'avoir accès aux variables de l'algorithme principal

Les paramètres

Les seuls échanges se font :

- De l'algorithme à la fonction en passant des valeurs grâce aux paramètres
- De la fonction vers l'algorithme en retournant une seule et unique valeur

Il est donc possible que 2 variables, l'une utilisée dans l'algorithme principal, l'autre dans une fonction, **portent le même nom et ne soient pas du même type** puisqu'elles sont utilisées de manière différente **dans des environnements de données différents.**

Les paramètres

- Au cœur d'une fonction on va distinguer **paramètres** et **variables**
- Les paramètres sont des valeurs passées par l'algorithme principal et connues dès le début de la fonction
- Les paramètres sont déclarés dans la définition de la fonction, **il n'est donc pas nécessaire de les redéfinir dans la section des variables**
- Il n'est pas obligatoire de les nommer comme les variables utilisées lors de l'appel de la fonction
- Les variables d'une fonction sont **locales à la fonction**, donc **inconnues du programme principal**
- Les variables d'une fonction sont définies dans la section « Variables » de la fonction
- Les variables du programme principal sont inconnues de la fonction

Technique d'écriture d'une fonction

On met en place ce que l'on appelle la **signature** de la fonction, à savoir :

- Le nom de la fonction
- Les paramètres de la fonction : nom, type et ordre des paramètres
- Le type de valeur retournée

Cette signature est essentielle car elle permet le **polymorphisme paramétrique** : deux fonctions peuvent avoir le même nom et des paramètres différents (nombre, type)

Le polymorphisme paramétrique **garantit automatiquement l'exécution de la bonne fonction** associée au bon nombre de paramètres et à leurs types

Les programmes identifient une fonction par sa signature et pas uniquement par son nom