

Introduction à l'informatique

Chapitre 5 - Fonctions

R. Groult, F. Levé

UFR des Sciences
Université de Picardie Jules Verne, Amiens

mardi 8 novembre 2011

Utilisation de fonctions prédéfinies

- La plupart du temps, les fonctions sont regroupées par *thématique*. Ces regroupements sont souvent appelés « *bibliothèques* » (*library* en anglais).
- Utiliser une fonction prédéfinie nécessite généralement de *préciser dans quelle bibliothèque* elle se trouve.
 - Exemple : en java, pour utiliser l'instruction `lire`, on doit importer `java.util.Scanner` qui contient les fonctions `next()`, `nextInt()`,...
 - Parfois les bibliothèques courantes sont prises en compte par défaut.
- La façon d'utiliser les fonctions des bibliothèques diffère selon les langages et les fonctions.
 - Exemple : en java, pour calculer la racine carrée de 5, on doit écrire : `Math.sqrt(5)`
- Il faut toujours *se reporter au manuel* du langage qu'on utilise pour connaître l'usage des fonctions prédéfinies (nombre et type des paramètres, valeur retournée et son type)
 - En java, consulter la documentation de l'API, disponible en ligne : <http://download.oracle.com/javase/1.5.0/docs/api/>

Écrire vos propres fonctions : pourquoi ?

- Problème complexe à résoudre : le *découper en sous-parties* et *écrire des fonctions* résolvant ces différentes sous-parties, *utilisées par un algorithme principal*.
 - Le code est plus lisible, les erreurs plus faciles à débusquer.
- Des fonctions peuvent *être utilisées facilement* plusieurs fois à différents moments de l'exécution, ou par des programmes différents, avec différents paramètres.
 - Le code est facilement réutilisable.
- Si on doit modifier un code utilisé à plusieurs endroits, il faut modifier beaucoup de code. Si on a écrit une fonction pour ce code, il suffit de *faire une seule modification*.
 - Le code est facile à maintenir et à modifier.

Sur des programmes de taille importante, on utilise des *fonctions réparties dans plusieurs fichiers sources* : ces programmes sont dits *modulaires*.

Deux sortes de fonctions

Syntaxe de fonction « avec retour »

fonction avec retour type nomFonction (paramètres)

déclaration des variables;

début

instructions de la fonction ;

retourne valeur ;

fin

Syntaxe de fonction « sans retour »

fonction sans retour nomFonction (paramètres)

déclaration des variables ;

début

instructions de la fonction ;

fin

Écriture d'une fonction : entête

Dans l'*entête* de la fonction, il faut préciser :

- Si la fonction est « *avec retour type* » ou bien « *sans retour* »
 - S'il s'agit d'une fonction « avec retour », préciser le type du résultat retourné par la fonction, et **ne pas oublier de retourner ce résultat avant la fin de la fonction.**
- le **nom** de la fonction
 - Ce que l'on veut en respectant les mêmes règles que pour les noms de variables.
- la **liste des paramètres** de la fonction.
 - On peut trouver zéro, un ou bien plusieurs paramètres entre parenthèses. S'il y en a plusieurs, ils sont séparés par des virgules.
 - Chaque nom de paramètre est précédé de son type.

Écriture d'une fonction : corps de la fonction

La partie après l'entête est le **corps de la fonction** : enchaînement des instructions nécessaires à la production du résultat.

- Déclaration de variables locales si nécessaire
 - obligatoire avant utilisation
- Portée des variables :
 - Une variable déclarée dans une fonction n'est *visible* que dans cette fonction.
 - On peut utiliser des variables ayant le même nom dans des fonctions différentes.

Appel d'une fonction

« nomFonction(liste de variables, d'expressions ou de valeurs) »

- La valeur de chaque paramètre de l'appel est *passée aux paramètres* de la fonction, dans le même ordre :
 - le **nombre de paramètres** de l'appel de la fonction est égal au nombre de paramètres de la déclaration de la fonction.
 - les **types des paramètres** de l'appel de la fonction sont les mêmes que les paramètres de la déclaration de la fonction.
- Les *noms des variables* utilisées lors de l'appel à la fonction peuvent être différents des *noms des paramètres* utilisés lors de la déclaration de cette fonction.
- Les paramètres de l'appel peuvent être des expressions.
- Si la fonction est « avec retour », récupérer/utiliser la valeur retournée.

fonction avec retour réel**solutionAXPlusB** (réel a, réel b)

réel x ;

début

 $x \leftarrow -b/a$;

retourne x ;

fin

réel val1, val2, sol ;

début

écrire "Donnez a et b" ;

lire val1, val2 ;

sol \leftarrow **solutionAXPlusB**(val1, val2) ;

écrire "x =" + sol ;

fin

instruction	val1	val2	sol	aff.
réel val1, ...	?	?	?	
écrire "..."				...
lire val1, val2 ;	5	10		
sol \leftarrow solu..				

solutionAXPlusB(5, 10)			
instruction	paramètres		variables
	a	b	x
fonction avec ...	5	10	
réel x ;			?
$x \leftarrow -b/a$;			-2
retourne x ;	retour -2		

sol \leftarrow solu..			-2	
écrire "x=...";				x=-2

Autre exemple

fonction sans retour afficheEntiers (entier n)

```
/* affichage des entiers de 1 à n */
```

```
entier i ;
```

```
début
```

```
    pour (i allant de 1 à n pas 1) faire
```

```
        écrire i ;
```

```
    finpour
```

```
fin
```

```
entier n ;
```

```
début
```

```
    écrire "Donnez un entier" ;
```

```
    lire n ;
```

```
    afficheEntiers(n);
```

```
fin
```

Démarche d'écriture (conseils)

1. Commencer par l'**en-tête de la fonction**

- déterminer les **données du problème** à résoudre et leur type (paramètres de la fonction)
- déterminer le **type du retour**
- préciser le **commentaire** en expliquant bien le rôle de la fonction (le problème résolu) et le rôle des paramètres (données)
- écrire les **préconditions** :
 - Cas général : domaine des valeurs pouvant être prises par les variables pour que le problème ait du sens
 - Cas particulier : restriction du problème par l'énoncé

2. Écrire le **corps de la fonction**

- suite d'instructions résolvant le problème

Les règles de présentation vues précédemment doivent également être respectées.

```
import java.util.Scanner;
public class EssaiEquation1erDegre
{
    public static double solutionAXPlusB(double a, double b)
    /* calcul de la solution de l'equation ax+b=0
       precondition : a !=0;
    */
    {
        double x;
        x = -b/a;
        return x ;
    }
    public static void main(String[ ] args)
    {
        double val1, val2, sol;
        System.out.println("Donner un réel");
        val1=(new Scanner(System.in)).nextDouble();
        System.out.println("Donner un autre réel");
        val2=(new Scanner(System.in)).nextDouble();
        sol=solutionAXPlusB(val1, val2);
        System.out.println("x=" + sol);
    }
}
```