Les sous-programmes

http://alexandre-mesle.com

22 août 2020

Lorsqu'un programme fait plusieurs milliers de lignes, comment :

- Eviter de faire des pâtés ?
- Trouver les erreurs rapidement ?
- Faire des modifications facilement ?

Lorsqu'un programme fait plusieurs milliers de lignes, comment :

- Eviter de faire des pâtés ?
- Trouver les erreurs rapidement ?
- Faire des modifications facilement ?

Lorsqu'un programme fait plusieurs milliers de lignes, comment :

- Eviter de faire des pâtés ?
- Trouver les erreurs rapidement ?
- Faire des modifications facilement ?

Définition

Nous allons utiliser des **sous-programes** pour subdiviser un programme. Il en existe deux types :

- Les **procédures**
- Les fonctions

Définition

Nous allons utiliser des **sous-programes** pour subdiviser un programme. Il en existe deux types :

- Les **procédures**
- Les **fonctions**

Définition

Une **procédure** est un ensemble d'**instructions** portant un **nom**

Définition

Procédure *nomprocedure()*
| corps de la procédure
Fin procédure

Exemple

Procédure *afficheBonjour()*
| **Afficher** *"Bonjour !"*
Fin procédure

Définition

On **exécute** une procédure en utilisant son nom.

On dit aussi que l'on **appelle** la procédure, ou encore qu'on l'**invoque**.

Exemple

```
afficheBonjour()
```

Définition

On **exécute** une procédure en utilisant son nom.

On dit aussi que l'on **appelle** la procédure, ou encore qu'on l'**invoque**.

Exemple

```
afficheBonjour()
```

Définition

On **exécute** une procédure en utilisant son nom.

On dit aussi que l'on **appelle** la procédure, ou encore qu'on l'**invoque**.

Exemple

```
afficheBonjour()
```

Exemple

Sans procédure

Algorithme : Affiche bonjour

Début

    Afficher *"Bonjour !"*

Fin

Exemple

et avec une procédure

Algorithme : Affiche bonjour

Procédure *afficheBonjour()*
| **Afficher** *"Bonjour !"*
Fin procédure
Début
| *afficheBonjour*()
Fin

Exemple

Algorithme : Affiche bonjour

Procédure *afficheBonjour()*
  Afficher *"Bonjour !"*
Fin procédure
Procédure *afficheAuRevoir()*
  Afficher *"Au revoir !"*
Fin procédure
Procédure *afficheBonjourEtAuRevoir()*
  afficheBonjour()
  afficheAuRevoir()
Fin procédure
Début
  afficheBonjourEtAuRevoir()
Fin

Définition

Une **variable locale** est une variable déclarée dans une procédure.

Exemple

Procédure *afficheUnADix()*
    entier : *i*
    Pour *i* $\longleftarrow$ 1 *à* 10
        Afficher *i*
    Fin pour
Fin procédure

Attention : une procédure ne peut pas utiliser les variables locales d'une autre procédure. Par exemple, **ceci est mal** :

Exemple

Procédure *afficheUn()*
    entier : *i*
    j ⟵ 1
    Afficher *j*
Fin procédure
Procédure *afficheDeux()*
    entier : *j*
    i ⟵ 2
    Afficher *i*
Fin procédure

Définition

Un **identificateur** est un nom choisi par le programmeur.

Exemple

- Les noms des variables sont des identificateurs parce qu'ils choisis par le programmeur pour représenter des variables
- Les noms des procédures sont aussi des identificateurs parce qu'ils sont choisis par le programmeur.

Définition

Un **identificateur** est un nom choisi par le programmeur.

Exemple

- Les noms des variables sont des identificateurs parce qu'ils choisis par le programmeur pour représenter des variables
- Les noms des procédures sont aussi des identificateurs parce qu'ils sont choisis par le programmeur.

Définition

Un **identificateur** est un nom choisi par le programmeur.

Exemple

- Les noms des variables sont des identificateurs parce qu'ils choisis par le programmeur pour représenter des variables
- Les noms des procédures sont aussi des identificateurs parce qu'ils sont choisis par le programmeur.

Définition

Un **mot-clé** (ou **mot réservé**) est un mot imposé par le langage de programmation.

Exemple

Les mots *Si*, *fin*, *alors* *procedure*... sont des mots-clés.

Vous vous douterez que les mots réservés le sont parce qu'il est impossible de les choisir comme nom de variable.

Définition

Un **mot-clé** (ou **mot réservé**) est un mot imposé par le langage de programmation.

Exemple

Les mots *Si*, *fin*, *alors* *procedure*... sont des mots-clés.

Vous vous douterez que les mots réservés le sont parce qu'il est impossible de les choisir comme nom de variable.

Définition

Un **mot-clé** (ou **mot réservé**) est un mot imposé par le langage de programmation.

Exemple

Les mots *Si*, *fin*, *alors* *procedure*... sont des mots-clés.

Vous vous douterez que les mots réservés le sont parce qu'il est impossible de les choisir comme nom de variable.

- Une procédure ne peut pas utiliser les variables locales d'une autre procédure
- Mais il existe un moyen de contourner cette restriction
- Les procédures se transmettent des informations avec des passages de paramètre.

- Une procédure ne peut pas utiliser les variables locales d'une autre procédure
- Mais il existe un moyen de contourner cette restriction
- Les procédures se transmettent des informations avec des passages de paramètre.

- Une procédure ne peut pas utiliser les variables locales d'une autre procédure
- Mais il existe un moyen de contourner cette restriction
- Les procédures se transmettent des informations avec des **passages de paramètre**.

- Une procédure ne peut pas utiliser les variables locales d'une autre procédure
- Mais il existe un moyen de contourner cette restriction
- Les procédures se transmettent des informations avec des **passages de paramètre**.

Définition

Il y a **passage de paramètre** lorsque la procédure appelante transmet une information à la procédure appelée.

Exemple

Procédure *afficheValeur(entier : x)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Début
| *afficheValeur*(4)
Fin

la valeur 4 est **recopiée** dans la variable x puis est affichée par la procédure *afficheValeur*. Cet extrait affiche donc la valeur 4.

Exemple

Procédure *afficheValeur(entier : x)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Début
| *afficheValeur*(4)
Fin

la valeur 4 est **recopiée** dans la variable *x* puis est affichée par la procédure *afficheValeur*. Cet extrait affiche donc la valeur 4.

On aurait tout à fait pu utiliser une variable à la place du 4 :

Exemple

Procédure *afficheValeur(entier : x)*
  Afficher *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Procédure *afficheQuatre()*
  entier : *titi*
  titi $\longleftarrow$ 4
  afficheValeur(titi)
Fin procédure

Attention : *titi* et *x* sont deux variables différentes !

On aurait tout à fait pu utiliser une variable à la place du 4 :

Exemple

Procédure *afficheValeur(entier : x)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Procédure *afficheQuatre()*
| entier : *titi*
| *titi* $\longleftarrow$ 4
| *afficheValeur*(*titi*)
Fin procédure

Attention : *titi* et *x* sont deux variables différentes !

On aurait tout à fait pu utiliser une variable à la place du 4 :

Exemple

Procédure *afficheValeur(entier : x)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Procédure *afficheQuatre()*
| entier : *titi*
| *titi* $\longleftarrow$ 4
| *afficheValeur*(*titi*)
Fin procédure

Attention : *titi* et *x* sont deux variables différentes !

On aurait tout à fait pu utiliser une variable à la place du 4 :

Exemple

Procédure *afficheValeur(entier : x)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*, *x*
Fin procédure
Procédure *afficheQuatre()*
| entier : *titi*
| *titi* $\longleftarrow$ 4
| *afficheValeur*(*titi*)
Fin procédure

Attention : *titi* et *x* sont deux variables différentes !

Pour éviter toute confusion entre *titi* et x, nous utiliserons la terminologie suivante :

Définition

- Les paramètres déclarés dans l'entête de la procédure sont les **paramètres formels.**
- Les paramètres utilisés lors de l'appel de la procédure sont les **paramètres effectifs.**

Pour éviter toute confusion entre *titi* et x, nous utiliserons la terminologie suivante :

Définition

- Les paramètres déclarés dans l'entête de la procédure sont les **paramètres formels.**
- Les paramètres utilisés lors de l'appel de la procédure sont les paramètres effectifs.

Pour éviter toute confusion entre *titi* et x, nous utiliserons la terminologie suivante :

Définition

- Les paramètres déclarés dans l'entête de la procédure sont les **paramètres formels.**
- Les paramètres utilisés lors de l'appel de la procédure sont les **paramètres effectifs.**

Exemple

formel est le paramètre formel et *effectif* le paramètre effectif.

Procédure *afficheValeur(entier : formel)*
| **Afficher** *"La valeur de l'entier passé en paramètre est "*,
| *formel*
Fin procédure
Procédure *afficheQuatre()*
| entier : *effectif*
| *effectif* $\longleftarrow$ 4
| *afficheValeur(effectif)*
Fin procédure

Exemple

Il est possible de passer plusieurs valeurs en paramètre :

Procédure *afficheSomme(entier : x, y)*
  Afficher $x + y$
Fin procédure

Exemple

Cette procédure peut s'appeller de la façon suivante :

afficheSomme(A, B)
Lors de cet appel, A est recopié dans x et B est recopié dans y.

Exemple

Il est possible de passer plusieurs valeurs en paramètre :

Procédure *afficheSomme(entier : x, y)*
  Afficher $x + y$
Fin procédure

Exemple

Cette procédure peut s'appeller de la façon suivante :

afficheSomme(A, B)
Lors de cet appel, A est recopié dans x et B est recopié dans y.

Exemple

Il est possible de passer plusieurs valeurs en paramètre :

Procédure *afficheSomme(entier : x, y)*
  Afficher $x + y$
Fin procédure

Exemple

Cette procédure peut s'appeller de la façon suivante :

afficheSomme(A, B)
Lors de cet appel, A est recopié dans x et B est recopié dans y.

Exemple

Il est possible de passer plusieurs valeurs en paramètre :

Procédure *afficheSomme(entier : x, y)*
  Afficher $x + y$
Fin procédure

Exemple

Cette procédure peut s'appeller de la façon suivante :

afficheSomme(A, B)

Lors de cet appel, A est recopié dans x et B est recopié dans y.

Exemple

Il est possible de passer plusieurs valeurs en paramètre :

Procédure *afficheSomme(entier : x, y)*
  Afficher $x + y$
Fin procédure

Exemple

Cette procédure peut s'appeller de la façon suivante :

afficheSomme(A, B)
Lors de cet appel, A est recopié dans x et B est recopié dans y.

Exemple

Tentons d'échanger les valeurs de deux variables dans une procédure :

Procédure *echange(entier : x, y)*
    entier : *temp*
    temp $\longleftarrow$ *x*
    x $\longleftarrow$ *y*
    y $\longleftarrow$ *temp*
Fin procédure

Exemple

Tentons d'échanger les valeurs de deux variables dans une procédure :

Procédure *echange(entier : x, y)*

entier : *temp*
temp $\longleftarrow$ *x*
x $\longleftarrow$ *y*
y $\longleftarrow$ *temp*

Fin procédure

Exemple

On peut appeler cette procédure de la façon suivante :

echange(A, B)

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Exemple

On peut appeler cette procédure de la façon suivante :

$echange(A, B)$

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Exemple

On peut appeler cette procédure de la façon suivante :

$echange(A, B)$

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Exemple

On peut appeler cette procédure de la façon suivante :

echange(A, B)

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Exemple

On peut appeler cette procédure de la façon suivante :

$echange(A, B)$

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Exemple

On peut appeler cette procédure de la façon suivante :

$echange(A, B)$

L'exécution de ce sous-programme laisse les valeurs de A et B inchangées :

- x et y sont des copies de A et B
- Les valeurs des **copies** (x et y) sont echangées
- Mais les **originaux** (A et B) ne sont pas affectés par cette modification.

Définition

Deux identificateurs de variables i et j sont des **alias** s'ils représentent la même variable.

Autrement dit, les valeurs de i et j sont liées, dans le sens ou modifier l'une revient aussi à modifier l'autre.

Définition

Deux identificateurs de variables i et j sont des **alias** s'ils représentent la même variable.

Autrement dit, les valeurs de i et j sont liées, dans le sens ou modifier l'une revient aussi à modifier l'autre.

Définition

Il y a passage de paramètre par :

- **valeur** lorsque le paramètre effectif est recopié dans le paramètre formel.
- **référence** lorsque le paramètre effectif est un alias du paramètre formel.

Lors d'un **passage de paramètres par référence**, on transmet non pas les valeurs des variables, mais les variables elles-mêmes.

Définition

Il y a passage de paramètre par :

- **valeur** lorsque le paramètre effectif est recopié dans le paramètre formel.
- **référence** lorsque le paramètre effectif est un alias du paramètre formel.

Lors d'un **passage de paramètres par référence**, on transmet non pas les valeurs des variables, mais les variables elles-mêmes.

Définition

Il y a passage de paramètre par :

- **valeur** lorsque le paramètre effectif est recopié dans le paramètre formel.
- **référence** lorsque le paramètre effectif est un alias du paramètre formel.

Lors d'un **passage de paramètres par référence**, on transmet non pas les valeurs des variables, mais les variables elles-mêmes.

Définition

Il y a passage de paramètre par :

- **valeur** lorsque le paramètre effectif est recopié dans le paramètre formel.
- **référence** lorsque le paramètre effectif est un alias du paramètre formel.

Lors d'un **passage de paramètres par référence**, on transmet non pas les valeurs des variables, mais les variables elles-mêmes.

Définition

Il y a passage de paramètre par :

- **valeur** lorsque le paramètre effectif est recopié dans le paramètre formel.
- **référence** lorsque le paramètre effectif est un alias du paramètre formel.

Lors d'un **passage de paramètres par référence**, on transmet non pas les valeurs des variables, mais les variables elles-mêmes.

Exemple

Passage de paramètres par valeur :

```
Procédure echange(entier : x, y)
    entier : temp
    temp ⟵ x
    x ⟵ y
    y ⟵ temp
Fin procédure
Début
    entier : a, b
    a ⟵ 1
    b ⟵ 2
    echange(a, b)
    Afficher a, b ;
Fin
```

Exemple

Passage de paramètres par valeur :

Procédure *echange(entier : x, y)*
    entier : *temp*
    $temp \longleftarrow x$
    $x \longleftarrow y$
    $y \longleftarrow temp$
Fin procédure
Début
    entier : a, b
    $a \longleftarrow 1$
    $b \longleftarrow 2$
    $echange(a, b)$
    Afficher a, b ;
Fin

Exemple

Passage de paramètres par référence.

Procédure *echange(entier : x e/s, y e/s)*
    entier : *temp*
    temp $\longleftarrow$ *x*
    x $\longleftarrow$ *y*
    y $\longleftarrow$ *temp*
Fin procédure
Début
    entier : *a*, *b*
    a $\longleftarrow$ 1
    b $\longleftarrow$ 2
    echange(*a*, *b*)
    Afficher *a*, *b* ;
Fin

- *a* et *x* sont des alias
- *b* et *y* sont des alias

Exemple

Passage de paramètres par référence.

Procédure *echange(entier : x e/s, y e/s)*
- entier : *temp*
- $temp \longleftarrow x$
- $x \longleftarrow y$
- $y \longleftarrow temp$

Fin procédure
Début
- entier : a, b
- $a \longleftarrow 1$
- $b \longleftarrow 2$
- $echange(a, b)$
- **Afficher** a, b ;

Fin

- a et x sont des alias
- b et y sont des alias

Exemple

Passage de paramètres par référence.

Procédure *echange(entier : x e/s, y e/s)*
    entier : *temp*
    $temp \longleftarrow x$
    $x \longleftarrow y$
    $y \longleftarrow temp$
Fin procédure
Début
    entier : a, b
    $a \longleftarrow 1$
    $b \longleftarrow 2$
    $echange(a, b)$
    Afficher a, b ;
Fin

- a et x sont des alias
- b et y sont des alias

Exemple

Passage de paramètres par référence.

Procédure *echange(entier : x e/s, y e/s)*
    entier : *temp*
    $temp \longleftarrow x$
    $x \longleftarrow y$
    $y \longleftarrow temp$
Fin procédure
Début
    entier : a, b
    $a \longleftarrow 1$
    $b \longleftarrow 2$
    $echange(a, b)$
    Afficher a, b ;
Fin

- a et x sont des alias
- b et y sont des alias

Définition

- Par défaut, un paramètre est passé par valeur
- Un paramètre est passé par référence s'il porte l'indication e/s

Définition

- Par défaut, un paramètre est passé par valeur
- Un paramètre est passé par référence s'il porte l'indication **e/s**

On résumera les choses ainsi :

- La transmission de valeur depuis la procédure appelante vers la procédure appelée se fait avec des passages de paramètres.
- La transmission de valeur depuis la procédure appelée vers la procédure appelante se fait avec des passages de paramètres par référence.

On résumera les choses ainsi :

- La transmission de valeur depuis la procédure appelante vers la procédure appelée se fait avec des passages de paramètres.
- La transmission de valeur depuis la procédure appelée vers la procédure appelante se fait avec des passages de paramètres par référence.

On résumera les choses ainsi :

- La transmission de valeur depuis la procédure appelante vers la procédure appelée se fait avec des passages de paramètres.
- La transmission de valeur depuis la procédure appelée vers la procédure appelante se fait avec des passages de paramètres par référence.

- Lorsqu'une procédure modifie les valeurs de variables passées par référence, il se produit ce que l'on appelle **effet de bord**
- Plus un programme contient d'effets de bords, plus il est difficile de trouver les erreurs, le maintenir et le faire évoluer
- N'utilisez donc les passages par référence que lorsque vous ne pouvez pas faire autrement.

- Lorsqu'une procédure modifie les valeurs de variables passées par référence, il se produit ce que l'on appelle **effet de bord**
- Plus un programme contient d'effets de bords, plus il est difficile de trouver les erreurs, le maintenir et le faire évoluer
- N'utilisez donc les passages par référence que lorsque vous ne pouvez pas faire autrement.

- Lorsqu'une procédure modifie les valeurs de variables passées par référence, il se produit ce que l'on appelle **effet de bord**
- Plus un programme contient d'effets de bords, plus il est difficile de trouver les erreurs, le maintenir et le faire évoluer
- N'utilisez donc les passages par référence que lorsque vous ne pouvez pas faire autrement.

Il existe un autre mécanisme permettant au sous-programme appelé de transmettre une information au sous-programme appelant : celui des **valeurs de retour**.

Définition

Une **fonction** est un sous-programme fait pour transmettre une unique valeur au sous-programme appelant. On dit que cette valeur est **retournée**.

Il existe un autre mécanisme permettant au sous-programme appelé de transmettre une information au sous-programme appelant : celui des **valeurs de retour**.

Définition

Une **fonction** est un sous-programme fait pour transmettre une unique valeur au sous-programme appelant. On dit que cette valeur est **retournée**.

Il existe un autre mécanisme permettant au sous-programme appelé de transmettre une information au sous-programme appelant : celui des **valeurs de retour**.

Définition

Une **fonction** est un sous-programme fait pour transmettre une unique valeur au sous-programme appelant. On dit que cette valeur est **retournée**.

Il existe un autre mécanisme permettant au sous-programme appelé de transmettre une information au sous-programme appelant : celui des **valeurs de retour**.

Définition

Une **fonction** est un sous-programme fait pour transmettre une unique valeur au sous-programme appelant. On dit que cette valeur est **retournée**.

Exemple

Fonction *Un*() : *entier*
 | **Retourner** 1
Fin fonction

La fonction *Un*() transmet 1 au sous-programme appelant.

Exemple

Fonction *Un*() : *entier*
| **Retourner** 1
Fin fonction

La fonction *Un*() transmet 1 au sous-programme appelant.

Définition

Une fonction se déclare

Fonction *nom*(*parametres*) : *typeDeRetour*

...

Fin fonction

Définition

On retourne une valeur avec l'instruction

Retourner *valeur*

Définition

Une fonction se déclare

Fonction *nom*(*parametres*) : *typeDeRetour*

...

Fin fonction

Définition

On retourne une valeur avec l'instruction

Retourner *valeur*

Définition

Une fonction se déclare

Fonction *nom*(*parametres*) : *typeDeRetour*

...

Fin fonction

Définition

On retourne une valeur avec l'instruction

Retourner *valeur*

Définition

Une fonction se déclare

Fonction *nom*(*parametres*) : *typeDeRetour*

...

Fin fonction

Définition

On retourne une valeur avec l'instruction

Retourner *valeur*

Exemple

La fonction *carre* prend en paramètre une valeur x et retourne la valeur x^2.

Fonction *carre*(*entier* : x) : *entier*
| **Retourner** $x * x$
Fin fonction

Exemple

La fonction *carre* prend en paramètre une valeur x et retourne la valeur x^2.

Fonction *carre*(*entier* : x) : *entier*
| **Retourner** $x * x$
Fin fonction

On peut appeler une fonction de plusieurs façons :

Exemple

$y \longleftarrow carre(x)$

Exemple

Afficher x, $"*"$, x, $"="$, $carre(x)$

On peut appeler une fonction de plusieurs façons :

Exemple

$y \longleftarrow carre(x)$

Exemple

Afficher x, $"*"$, x, $"="$, $carre(x)$

On peut appeler une fonction de plusieurs façons :

Exemple

$y \longleftarrow carre(x)$

Exemple

Afficher x, $''*''$, x, $''=''$, $carre(x)$

Exemple

Afficher x, $"\hat{}4 = "$, $carre(carre(x))$

On remarque que l'utilisation des fonctions est davantage souple que celle des procédures :

- on peut imbriquer des appels de fonctions dans des expressions
- ce que l'on ne peut pas faire avec des procédures, qui elles sont des instructions.

Exemple

Afficher x, $"\hat{}4 = "$, $carre(carre(x))$

On remarque que l'utilisation des fonctions est davantage souple que celle des procédures :

- on peut imbriquer des appels de fonctions dans des expressions
- ce que l'on ne peut pas faire avec des procédures, qui elles sont des instructions.

Exemple

Afficher x, $''\hat{}4 = ''$, $carre(carre(x))$

On remarque que l'utilisation des fonctions est davantage souple que celle des procédures :

- on peut imbriquer des appels de fonctions dans des expressions
- ce que l'on ne peut pas faire avec des procédures, qui elles sont des instructions.

Exemple

Afficher x, $"\hat{}4 = "$, $carre(carre(x))$

On remarque que l'utilisation des fonctions est davantage souple que celle des procédures :

- on peut imbriquer des appels de fonctions dans des expressions
- ce que l'on ne peut pas faire avec des procédures, qui elles sont des instructions.

La notion de fonction en tant que sous-programme est à mettre en parallèle avec celle de fonction mathématique.

Exemple

Fonction $f(entier : x) : entier$
| **Retourner** $3 * x + 1$
Fin fonction

La fonction $f(x) = 3x + 1$ est une correspondance entre une valeur quelconque x et celle que l'on obtient en multipliant x par 3 et en lui ajoutant 1.

La notion de fonction en tant que sous-programme est à mettre en parallèle avec celle de fonction mathématique.

Exemple

Fonction $f(entier : x) : entier$
| **Retourner** $3 * x + 1$
Fin fonction

La fonction $f(x) = 3x + 1$ est une correspondance entre une valeur quelconque x et celle que l'on obtient en multipliant x par 3 et en lui ajoutant 1.

La notion de fonction en tant que sous-programme est à mettre en parallèle avec celle de fonction mathématique.

Exemple

Fonction $f(entier : x) : entier$
| **Retourner** $3 * x + 1$
Fin fonction

La fonction $f(x) = 3x + 1$ est une correspondance entre une valeur quelconque x et celle que l'on obtient en multipliant x par 3 et en lui ajoutant 1.

On l'appelle :

Exemple

Afficher *"L'image de* 4 *par f est "*, $f(4)$, *"."*

On peut considérer que l'**antécédent** est x et que la valeur retournée est l'**image** de la fonction.

On l'appelle :

Exemple

Afficher *"L'image de* 4 *par f est "*, $f(4)$, *"."*

On peut considérer que l'**antécédent** est x et que la valeur retournée est l'**image** de la fonction.

On l'appelle :

Exemple

Afficher *"L'image de* 4 *par f est "*, $f(4)$, *"."*

On peut considérer que l'**antécédent** est x et que la valeur retournée est l'**image** de la fonction.

- La seule chose qui nous intéresse avec une fonction est la valeur qu'elle retourne
- donc on évitera les effets de bords
- et l'on préfèrera des passages de paramètres par valeur dans les fonctions.

- La seule chose qui nous intéresse avec une fonction est la valeur qu'elle retourne
- donc on **évitera les effets de bords**
- et l'on préfèrera des passages de paramètres par valeur dans les fonctions.

- La seule chose qui nous intéresse avec une fonction est la valeur qu'elle retourne
- donc on **évitera les effets de bords**
- et l'on préfèrera des passages de paramètres par valeur dans les fonctions.

- La seule chose qui nous intéresse avec une fonction est la valeur qu'elle retourne
- donc on **évitera les effets de bords**
- et l'on préfèrera des passages de paramètres par valeur dans les fonctions.