Algorithmique et programmation – Exemples et vocabulaire

Table des matières

1 Exemples et vocabulaire

1.1 Affectation, calcul, entrée, sortie

1.1.1 Extrait du programme 2009

Instructions élémentaires (affectation, calcul, entrée, sortie).

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables :

– d'écrire une formule permettant un calcul ;
– d'écrire un programme calculant et donnant la valeur d'une fonction ; ainsi que les instructions d'entrées et sorties nécessaires au traitement.

1.1.2 Exemple 1 – Périmètre d'un rectangle

L'algorithme ci-dessous calcule le périmètre d'un rectangle quand on lui donne la largeur a et la longueur b de ce rectangle.

Entrées	Lire a
	Lire b
Traitement	p prend la valeur $2 \times (a + b)$
Sortie	Afficher "Le périmètre est :"
	Afficher p

1.1.3 Vocabulaire

Entrées

L'instruction « Lire a » signifie que l'algorithme enregistre le nombre qu'on lui donne dans une mémoire nommée « a ». C'est l'utilisateur qui va donner ce nombre. On dit que c'est une entrée.

Les différents verbes possibles sont :

– en français : lire, saisir, entrer ;
– en anglais : input, prompt.

Traitement – Affectation

Le traitement ici est un calcul et un stockage, l'algorithme
– calcule le périmètre en effectuant $2 \times (a + b)$;
– stocke le résultat dans une mémoire nommée « p ».
Ce stockage s'appelle une affectation.

On peut exprimer ou représenter cette affectation de différentes manières :

p prend la valeur $2 \times (a + b)$ ou p reçoit $2 \times (a + b)$ ou $p \leftarrow 2 \times (a + b)$

ou encore :

$2 \times (a + b)$ est affecté à p ou stocker la valeur de $2 \times (a + b)$ dans p ou $2 \times (a + b) \rightarrow p$

Sorties

L'instruction « Afficher "Le périmètre est :" » fait afficher un message.

L'instruction « Afficher p » fait afficher la valeur de la mémoire p, c'est à dire le résultat.

En anglais, l'instruction « afficher » se traduit par « display » ou « print ».

1.1.4 Algorithme de l'exemple 1 avec les calculatrices

TI 82, TI 83, TI 84

```
PROGRAM:PERIM
:Prompt A
:Prompt B
:2*(A+B)→P
:Disp "P ="
:Disp P
```

TI 89, TI 92

```
:perim()
:Prgm
:Local a,b
:Prompt a
:Prompt b
:2*(a+b)→p
:Disp "p ="
:Disp p
:EndPrgm
```

CASIO

```
======PERIM======
"A="?→A↵
"B="?→B↵
2×(A+B)→P
"P="↵
P◢
```

Remarque : pour les calculatrices TI 89 et TI 92, l'instruction `Local a,b` permet d'éviter des conflits avec d'autres programmes qui utilisent aussi les lettres a et b.

1.1.5 Algorithme de l'exemple 1 avec AlgoBox

```
1   VARIABLES
2     a EST_DU_TYPE NOMBRE
3     b EST_DU_TYPE NOMBRE
4     p EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     //Entrées
7     LIRE a
8     LIRE b
9     //Traitement
10    p PREND_LA_VALEUR 2*(a+b)
11    //Sortie
12    AFFICHER "Le périmètre est : "
13    AFFICHER p
14  FIN_ALGORITHME
```

Les lignes précédées du signe // sont des commentaires.

1.1.6 Algorithme de l'exemple 1 avec Python 3 et Scilab

Python 3

```
# Entrees
a=input("Largeur ? ")
b=input("Longueur ? ")
# Conversion en nombres
a=float(a)
b=float(b)
# Traitement
p=2*(a+b)
# Sortie
print("Le perimetre est : ",p)
```

Scilab

```
// Entrees
a=input("Largeur ? ");
b=input("Longueur ? ");
// Traitement
p=2*(a+b);
// Sortie
afficher("Le perimetre est : "+string(p))
```

Les lignes précédées du signe `#` en Python 3 et du signe `//` dans le langage de Scilab sont des commentaires.

En Python 3, l'instruction `float` sert à transformer le nombre saisi en un nombre décimal.

1.2 Instruction conditionnelle (si, alors, sinon)

1.2.1 Extrait du programme 2009

Boucle et itérateur, instruction conditionnelle

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables de programmer une instruction conditionnelle.

1.2.2 Exemple 2

Un magasin offre une remise de 30 € et si le montant total des achats est supérieur ou égal à 150 € la remise est de 50 €.

L'algorithme ci-dessous affiche le prix à payer quand on lui donne le montant total des achats.

Entrée	Lire m
Traitement	Si $m < 150$
	alors p prend la valeur $m - 30$
	sinon p prend la valeur $m - 50$
Sortie	Afficher "Le prix à payer est :"
	Afficher p

1.2.3 Vocabulaire

L'instruction ci-dessous est une instruction conditionnelle

Si $m < 150$
alors p prend la valeur $m - 30$
sinon p prend la valeur $m - 50$.

En effet
– l'instruction « p prend la valeur $m - 30$ » est exécutée à condition que $m < 150$;
– l'instruction « p prend la valeur $m - 50$ » est exécutée à condition que $m \geqslant 150$.

1.2.4 Algorithme de l'exemple 2 avec les calculatrices

TI 82, TI 83, TI 84

```
PROGRAM:REMISE
:Prompt M
:If M<150
:Then
:M-30→P
:Else
:M-50→P
:End
:Disp "P=",P
```

TI 89, TI 92

```
:remise()
:Prgm
:Local m
:Prompt m
:If m<150
Then
:m-30→p
:Else
:m-50→p
:EndIf
:Disp "p=",p
:EndPrgm
```

CASIO

```
======REMISE======
"M="?→M↵
If M<150↵
Then ↵
M-30→P
Else ↵
M-50→P
IfEnd
"P=":P◢
```

1.2.5 Algorithme de l'exemple 2 avec AlgoBox

```
1   VARIABLES
2     m EST_DU_TYPE NOMBRE
3     p EST_DU_TYPE NOMBRE
4   DEBUT_ALGORITHME
5     LIRE m
6     SI (m<150) ALORS
7       DEBUT_SI
8       p PREND_LA_VALEUR m-30
9       FIN_SI
10      SINON
11        DEBUT_SINON
12        p PREND_LA_VALEUR m-50
13        FIN_SINON
14    AFFICHER "Prix à payer : "
15    AFFICHER p
```

1.2.6 Algorithme de l'exemple 2 avec Python 3 et Scilab

Python 3

```
# Entree
m=input("Montant total des achats ? ")
# On transforme m en nombre decimal
m=float(m)
# Traitement
if m<150:
    p=m-30
else:
    p=m-50
# Sortie
print("Prix à payer : ",p)
```

Scilab

```
// Entree
m=input("Montant total des achats ? ");
// Traitement
if m<150 then
    p=m-30
else
    p=m-50
end
// Sortie
afficher("Prix a payer : "+string(p))
```

1.3 Calcul itératif de nombre d'itérations donné (boucle Pour)

1.3.1 Extrait du programme 2009

Boucle et itérateur, instruction conditionnelle

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables de programmer un calcul itératif, le nombre d'itérations étant donné.

1.3.2 Exemple 3

L'algorithme ci-dessous calcule des puissances successives de 7 : 7^8, 7^9, 7^{10}.

Pour des valeurs de n allant de 8 à 10, de 1 en 1
    a prend la valeur 7^n
    Afficher n et a
    Fin de la boucle "pour"

Cela veut dire que :

- Le nombre n prend d'abord la valeur 8, a prend la valeur 7^8, et on affiche a ;
- puis le nombre n prend la valeur 9, a prend la valeur 7^9, et on affiche a ;
- puis le nombre n prend la valeur 10, a prend la valeur 7^{10}, et on affiche a.

1.3.3 Vocabulaire

On appelle cela un **calcul itératif**.

Quand n prend la valeur 8, puis 9, puis 10, on dit que n est **itéré** c'est à dire que n est chaque fois augmenté de 1.

Une étape de calcul est appelée une **itération**, par exemple, dans l'algorithme précédent, on dit qu'il y a trois itérations.

Dès qu'une étape est terminée, n est augmenté de 1, et on recommence, et ainsi de suite, on appelle donc cela une **boucle**, et comme on écrit cette boucle sous la forme *Pour des valeurs de ...allant de ...à ...*, on appelle cela une **boucle Pour**.

1.3.4 Algorithme de l'exemple 3 avec les calculatrices

TI 82, TI 83, TI 84

```
PROGRAM:PUISSANC
:For(N,8,10,1)
:7^N→A
:Disp N,A
:End
```

TI 89, TI 92

```
:puissanc()
:Prgm :Local
n,a
:For n,8,10,1
:7^n→a
:Disp n,a
:EndFor
:EndPrgm
```

CASIO

```
======PUISSANC======
For 8→N To 10↵
7^n→A↵
N◢
A◢
Next↵
```

Remarque pour les calculatrices CASIO : le symbole ◢ fait s'afficher ce qui précède et provoque une pause, il faut donc appuyer sur EXE pour que l'exécution du programme se poursuive.

1.3.5 Algorithme de l'exemple 3 avec AlgoBox

```
1   VARIABLES
2     n EST_DU_TYPE NOMBRE
3     a EST_DU_TYPE NOMBRE
4   DEBUT_ALGORITHME
5     POUR n ALLANT_DE 8 A 10
6       DEBUT_POUR
7       a PREND_LA_VALEUR pow(7,n)
8       AFFICHER a
9       FIN_POUR
10  FIN_ALGORITHME
```

1.3.6 Algorithme de l'exemple 3 avec Python 3 et Scilab

Python 3

```
for n in range(8,11):
    a=7**n
    print(n,a)
```

Scilab

```
for n=8:10
a=7^n;
afficher(string(n)+string(a))
end
```

Remarque : l'instruction `for n in range(8,11):` veut bien dire « Pour des valeurs de n allant de 8 à 10, de 1 en 1 », en effet `range(8,11)` est la liste {8, 9, 10}.

1.4 Calcul itératif avec une fin de boucle conditionnelle (boucle Tant que)

1.4.1 Extrait du programme 2009

Boucle et itérateur, instruction conditionnelle

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables de programmer un calcul itératif, avec une fin de boucle conditionnelle.

1.4.2 Exemple 4

Nous allons résoudre par essais successifs l'équation $x^3 + x^2 = 100$.

La fonction f est définie par $f(x) = x^3 + x^2$ sur $[0\ ;\ 10]$.

Cette fonction est croissante sur $[0\ ;\ 10]$ et on peut calculer que : $f(4) = 80$ et $f(5) = 150$

Par conséquent on peut trouver une solution approximative de l'équation $x^3 + x^2 = 100$ sur $[4\ ;\ 5]$.

Nous allons donc procéder de la manière suivante :
- Calculer $f(4)$
- puis calculer $f(4,01)\,;\ f(4,02)\,;\ f(4,03)$ et ainsi de suite, et de faire continuer l'algorithme **tant que** l'image de chaque nombre est inférieure à 100.

Initialisation
x prend la valeur 4
y prend la valeur $4^2 + 4^3$
Traitement
Tant que $y < 100$
  x prend la valeur $x + 0,01$
  y prend la valeur $x^2 + x^3$
Fin de la boucle "tant que"
Sorties
Afficher $x - 0,01$ et x

1.4.3 Vocabulaire

Il s'agit encore d'un **calcul itératif** (voir plus haut le paragraphe 1.3.3), mais cette fois-ci on ne connaît pas le nombre d'itérations à effectuer. En revanche, on sait que l'on poursuit les étapes **tant que** $y < 100$. On dit que cette condition est une **fin de boucle conditionnelle** : si cette condition ($y < 100$) n'est pas respectée, on « sort » de la boucle et on passe aux instructions suivante (Afficher $x - 0,01$ et x).

On appelle donc cela un **calcul itératif, avec une fin de boucle conditionnelle** ou plus simplement une **boucle Tant que**.

1.4.4 Algorithme de l'exemple 4 avec les calculatrices

TI 82, TI 83, TI 84

```
PROGRAM:EQUATION
:4→X
:4^2+4^3→Y
:While Y<100
:X+0.01→X
:X^2+X^3→Y
:End
:Disp X-0.01,X
```

TI 89, TI 92

```
:equation()
:Prgm
:Local x,y
:4→x
:4^2+4^3→y
:While y<100
:x+0.01→x
:x^2+x^3→y
:EndWhile
:Disp x-0.01,x
:EndPrgm
```

CASIO

```
======EQUATION======
4→X↵
4^2+4^3→Y↵
While Y<100↵
X+0.01→X↵
X^2+X^3→Y↵
WhileEnd↵
X-0.01◢
X◢
```

1.4.5 Algorithme de l'exemple 4 avec AlgoBox

```
1   VARIABLES
2     x EST_DU_TYPE NOMBRE
3     y EST_DU_TYPE NOMBRE
4     a EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     x PREND_LA_VALEUR 4
7     y PREND_LA_VALEUR pow(4,3)+pow(4,2)
8     TANT_QUE (y<100) FAIRE
9       DEBUT_TANT_QUE
10      x PREND_LA_VALEUR x+0.01
11      y PREND_LA_VALEUR pow(x,3)+pow(x,2)
12      FIN_TANT_QUE
13    a PREND_LA_VALEUR x-0.01
14    AFFICHER a
15    AFFICHER x
16  FIN_ALGORITHME
```

1.4.6 Algorithme de l'exemple 4 avec Python 3 et Scilab

Python 3

```
# Initialisation
x=4
y=4**3+4**2
# Traitement
while y<100:
    x=x+0.01
    y=x**3+x**2
# Sorties
print(x-0.01,x)
```

Scilab

```
// Initialisation
x=4;
y=4^3+4^2;
// Traitement
while y<100
    x=x+0.01
    y=x^3+x^2
end
// Sorties
a=x-0.01;
afficher("a = "+string(a)+" et x = "+string(x))
```

Index