

Introduction à l'algorithmique - première partie -

Vasile-Marian SCUTURICI

Objectif

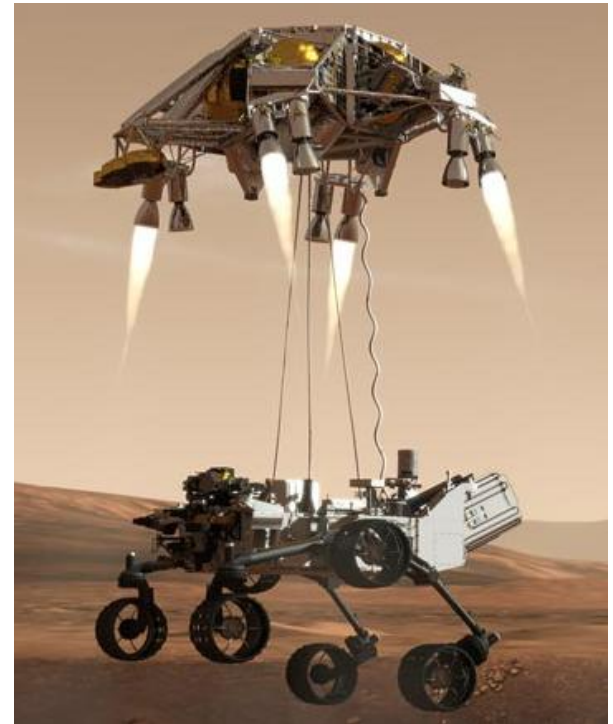
- Apprendre les bonnes bases pour programmer un ordinateur
 - Concevoir un algorithme
 - Evaluer un algorithme
 - Maîtriser quelques algorithmes utiles
- Structures de données -> 2^{ème} partie (Eric Guerin)
- Algorithmique avancée -> 3^{ème} partie (Christine Solnon)

Bibliographie

- **Introduction to algorithms**, Cormen, Leiserson, Rivest, Stein, MIT press, Third edition, 2009.
- **Algorithms and Data Structures**, Robert Sedgewick, Kevin Wayne, Princeton, 2007
- **Algorithms**, Dasgupta, Papadimitriou, and Vazirani, McGraw-Hill, 2006.
- **Competitive Programming**, Steven Halim, Felix Halim
- ...
- Support de cours (Moodle)
 - <http://moodle2.insa-lyon.fr/>

Motivation 1/2

- Algorithme = méthode pour résoudre un problème
- Applications :
 - Internet : routage, Web search
 - Robotique spatiale : Curiosity
 - Graphique : jeux vidéo
 - Transports : trajet optimal
 - Sécurité : https
 - Automobile : airbag
 - IA
 - ...



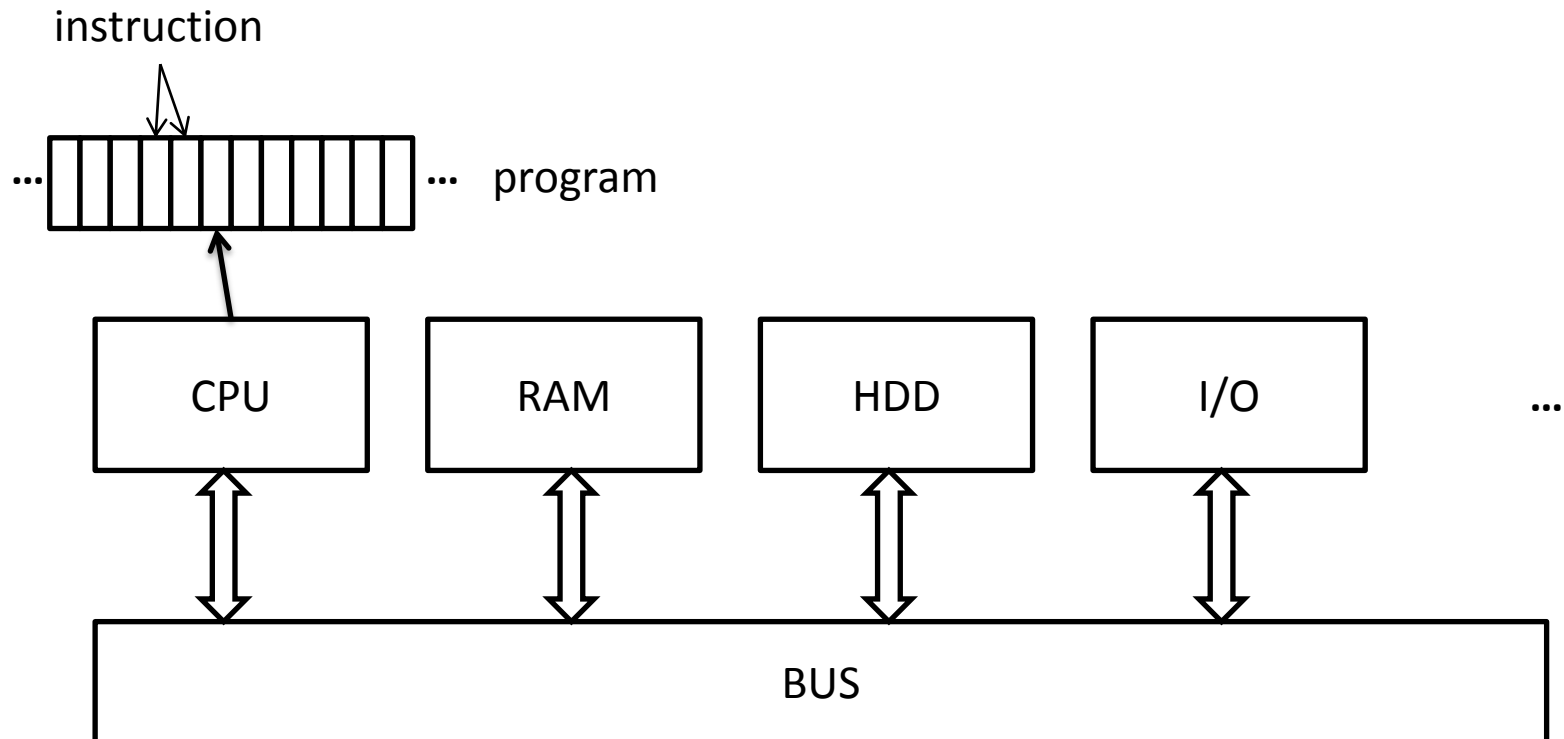
Motivation 2/2



Court historique

- Algorithme
 - L'origine du mot : Al-Khwarizmi, mathématicien perse du IX^{ème} siècle
 - Premiers algorithmes connus : les anciens grecs (Euclid, Eratosthène, Archimède)
 - Formalisation en 1936-1937 par Alan Turing : la machine Turing

Description (très) schématique d'un ordinateur



Programme

- Un ordinateur exécute des programmes, décrites via des instructions très simple
- Programme : permet de résoudre un problème
- Programme = algorithmes + structures de données (N. Wirth)

De problème à programme

- Plusieurs niveaux de description (langages) permettant de passer d'un problème décrite en langage naturel vers un code directement exécutable par une machine

Tours de Hanoï

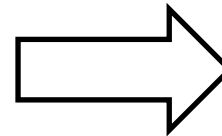
Le problème des tours de Hanoï est un jeu de réflexion imaginé par le mathématicien français Édouard Lucas, et consistant à déplacer des disques de diamètres différents d'une tour de « départ » à une tour d'« arrivée » en passant par une tour « intermédiaire », et ceci en un minimum de coups, tout en respectant les règles suivantes :

- on ne peut déplacer plus d'un disque à la fois,
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

On suppose que cette dernière règle est également respectée dans la configuration de départ.



Un modèle d'une Tour d'Hanoï (avec 8 disques)



```
10111001
11010010
00000100
10001001
00001110
00000000
00000000
00000000
10111001
11100001
00010000
10001001
00001110
00000010
```

Description d'un problème

- Problème initiale : langage naturel
 - description informelle, ambiguë et incomplète
- Spécification formelle : ce que doit faire le programme
 - non ambiguë, complète et correcte
 - Spécifie :
 - les paramètres en entrée,
 - les paramètres en sortie,
 - les pré-conditions,
 - la relation entre les paramètres en entrée et les paramètres en sortie,
 - les contraintes à respecter pour la résolution du problème (notamment les contraintes de ressources).

Algorithme

- spécification formelle -> élaboration d'un algorithme
- Algorithme
 - spécifie le “comment”, c'est à dire l'enchaînement d'opérations élémentaires à effectuer pour résoudre le problème
 - *Impérative ("comment") != déclarative ("quoi")*
- Algorithme
 - suite finie et non-ambiguë d'opérations ou d'instructions permettant de résoudre un problème




Description d'un problème - exemple

- Aller de Meyzieu à Cluj-Napoca
 - Départ : Meyzieu
 - Arrivée : Cluj-Napoca
 - Contraintes :
 - Voiture
 - Arrêt à Wien
 - Passer par la Suisse

« QUOI »

Algorithme - exemple


← from Meyzieu, France
to Cluj-Napoca, Romania


18 h 39 min (1,794 km)   

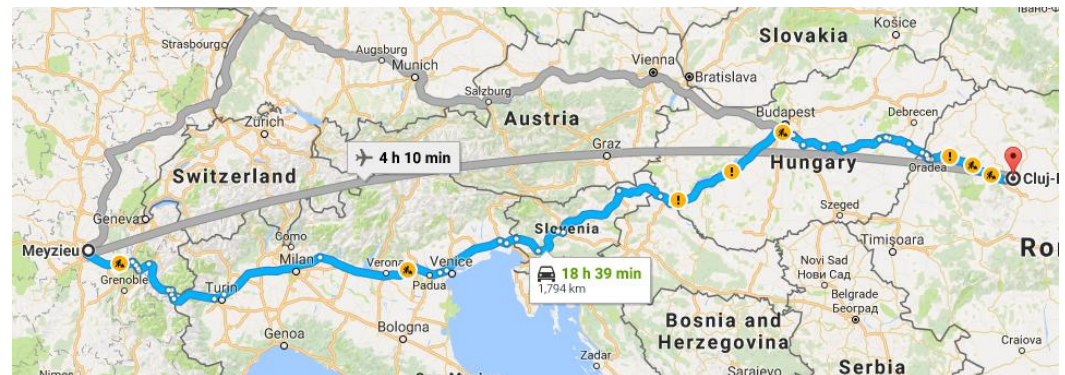
via A4
17 h 11 min without traffic
⚠ This route has tolls.
⚠ This route crosses through multiple countries.

Meyzieu
France

- Get on N346 in Décines-Charpieu from Rue de la République
5 min (2.1 km)
- ✓ Take A43, A32/E70, A4, E55, ... and M7 to Route 405/E60 in Újhartyán, Magyarország. Take exit 44 from E60/M5
13 h 8 min (1,413 km)

 Merge onto N346
7.0 km

 Use the right 2 lanes to merge onto A43 toward Grenoble/Chambéry/Aéroport Saint-Exupéry
⚠ Partial toll road



« COMMENT »

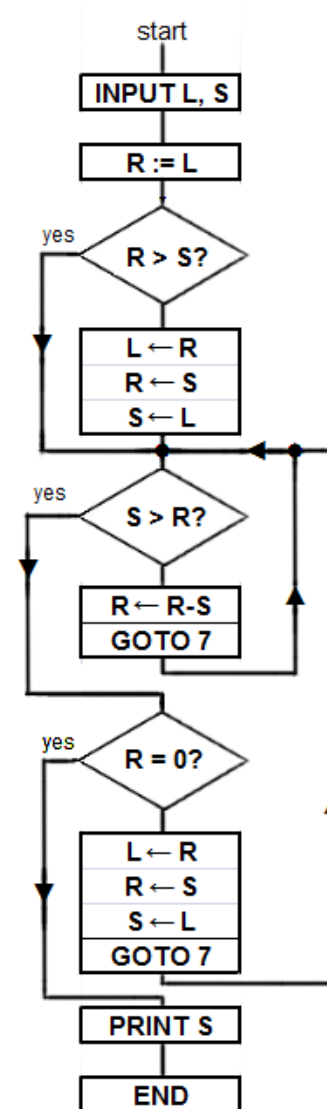
Algorithme - évaluation

- Utilisation de ressources
 - Temps processeur
 - Mémoire
- Terminaison : pour chaque entrée il se termine
- Correctitude : pour chaque entrée il produit la bonne sortie

Description d'un algorithme

- langage naturel
- graphiquement
- pseudo-code
- ...

```
MAX-HEAPIFY(A, i)
1  l ← LEFT(i)
2  r ← RIGHT(i)
3  if l ≤ heap-size[A] and A[l] > A[i]
4    then largest ← l
5  else largest ← i
6  if r ≤ heap-size[A] and A[r] > A[largest]
7    then largest ← r
8  if largest ≠ i
9    then exchange A[i] ↔ A[largest]
10     MAX-HEAPIFY(A, largest)
```



Pseudo-code

Partie statique - QUOI

Procédure : moyenne($a, b, resultat$)

Entrée : entier a
entier b

Sortie : réel $resultat$, correspondant à $(a + b)/2$

1 **début**

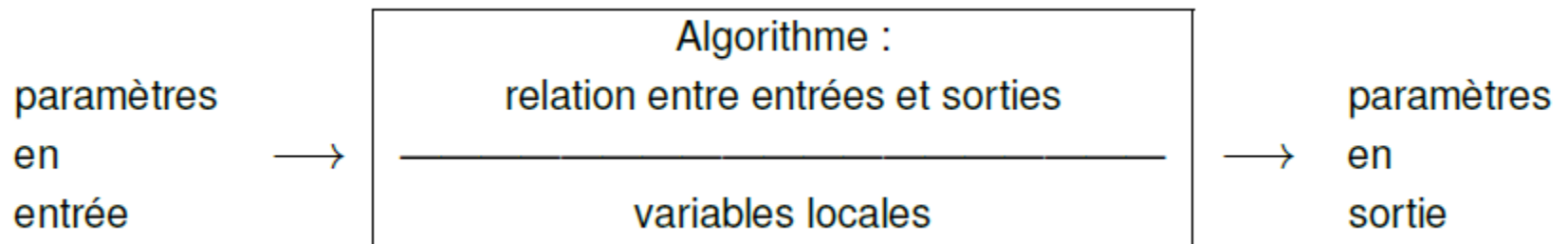
2 $resultat \leftarrow (a + b);$

3 $resultat \leftarrow resultat/2;$

Partie dynamique - COMMENT

Variables

- Nom, type, valeur



Variables - nom

- Nom/identificateur : représentation symbolique de l'adresse où est stockée la valeur de la variable en mémoire
- Donner un nom suggestif !
- Exemples :
 - Mot, Resultat, Perimetre, SommeTotale ...
 - ~~– i23, s12~~

Variables - type

- Permet d'interpréter les octets correspondant à la valeur de la variable
- entier
- réel
- car
- booléen (logique)
- tableau
- ...

Algorithme – partie statique

Procédure *nom-de-la-procédure*(*<noms-des-paramètres>*)

Entrée : pour chaque paramètre en entrée, préciser son type et son nom

Sortie : pour chaque paramètre en sortie, préciser son type et son nom

Précondition : Conditions sur les paramètres en entrée

Postcondition : Relation entre les paramètres en entrée et ceux en sortie

Déclaration : pour chaque variable locale, préciser son type et son nom

const : pour chaque constante, préciser son nom et sa valeur

début

Suite d'opérations élémentaires permettant de calculer les paramètres en sortie
en fonction des paramètres en entrée

(partie dynamique de l'algorithme)

Algorithme – partie statique - exemple

Procédure *racines*(a, b, c, r_1, r_2)

Entrée : réel a
réel b
réel c

Sortie : réel r_1
réel r_2

Précondition : $b^2 - 4ac \geq 0$ et $a \neq 0$

Postcondition : $ar_1^2 + br_1 + c = ar_2^2 + br_2 + c = 0$ ou autrement dit,
 r_1 et r_2 sont les deux solutions de
l'équation $ax^2 + bx + c = 0$

Déclaration : réel δ

début

Suite d'opérations élémentaires
permettant de calculer r_1 et r_2 à partir de a, b et c .

Algorithme – partie statique - exemple

- Forme abrégée

Procédure *racines*(a, b, c, r_1, r_2)

┌ **Entrée** : réel a, b, c

└ **Sortie** : réel r_1, r_2

Pseudo-code : instructions

- Affectation
- Expressions
- Instructions conditionnelles
- Boucles
- Entrée/sorties

Affectation

`nom-var ← expr`

- La variable *nom-var* recevra la valeur de l'expression *expr*
- Expression
 - Valeur explicite $a \leftarrow 25$
 - Valeur d'une variable $a \leftarrow b$
 - Résultat d'une opération entre d'autres expressions $r_1 \leftarrow r_1/3.0$

Expressions

entier e_1, e_2 ,
réel r_1 ,
logique b_1, b_2

- Operations arithmétiques

$$e_1 \leftarrow (10 \text{ mod } 3) + (5 \text{ div } 2)$$

$$e_2 \leftarrow e_1 * 2$$

$$r_1 \leftarrow 4.5 * 2.0$$

- Operations de comparaison

- Operations logiques

$$b_1 \leftarrow (r_1 > 4.2) \text{ ou } (e_1 = e_2)$$

$$b_2 \leftarrow \text{non}(b_1) \text{ et } r_1 \leq 4.2$$

Enchaînement d'instructions

- Séquentiel
- Alternatif
- Répétitif

Enchaînement séquentiel

Procédure *cercle*(R, D, P, S)

Entrée : réel R

Sortie : réel D, P, S

Précondition : $R \geq 0$

Postcondition : D, P et S contiennent respectivement le diamètre, le périmètre et la surface d'un cercle de rayon R

Déclaration : **const** : $\pi = 3.14$

début

$D \leftarrow 2 * R$

$P \leftarrow D * \pi$

$S \leftarrow R * R * \pi$

R	3
D	?
P	?
S	?

Mémoire ...

Enchaînement séquentiel

Procédure *cercle*(R, D, P, S)

Entrée : réel R

Sortie : réel D, P, S

Précondition : $R \geq 0$

Postcondition : D, P et S contiennent respectivement le diamètre,
le périmètre et la surface d'un cercle de rayon R

Déclaration : **const** : $\pi = 3.14$

début

$D \leftarrow 2 * R$

$P \leftarrow D * \pi$

$S \leftarrow R * R * \pi$

R	3
D	6
P	?
S	?

Enchaînement séquentiel

Procédure *cercle*(R, D, P, S)

Entrée : réel R

Sortie : réel D, P, S

Précondition : $R \geq 0$

Postcondition : D, P et S contiennent respectivement le diamètre, le périmètre et la surface d'un cercle de rayon R

Déclaration : **const** : $\pi = 3.14$

début

$D \leftarrow 2 * R$

$P \leftarrow D * \pi$

$S \leftarrow R * R * \pi$

R	3
D	6
P	18.84...
S	?

Enchaînement séquentiel

Procédure *cercle*(R, D, P, S)

Entrée : réel R

Sortie : réel D, P, S

Précondition : $R \geq 0$

Postcondition : D, P et S contiennent respectivement le diamètre, le périmètre et la surface d'un cercle de rayon R

Déclaration : **const** : $\pi = 3.14$

début

$D \leftarrow 2 * R$

$P \leftarrow D * \pi$

$S \leftarrow R * R * \pi$

R	3
D	6
P	18.84...
S	28.26...

Enchaînement séquentiel

Procédure *cercle*(R, D, P, S)

Entrée : réel R

Sortie : réel D, P, S

Précondition : $R \geq 0$

Postcondition : D, P et S contiennent respectivement le diamètre,
le périmètre et la surface d'un cercle de rayon R

Déclaration : **const** : $\pi = 3.14$

début

$D \leftarrow 2 * R$

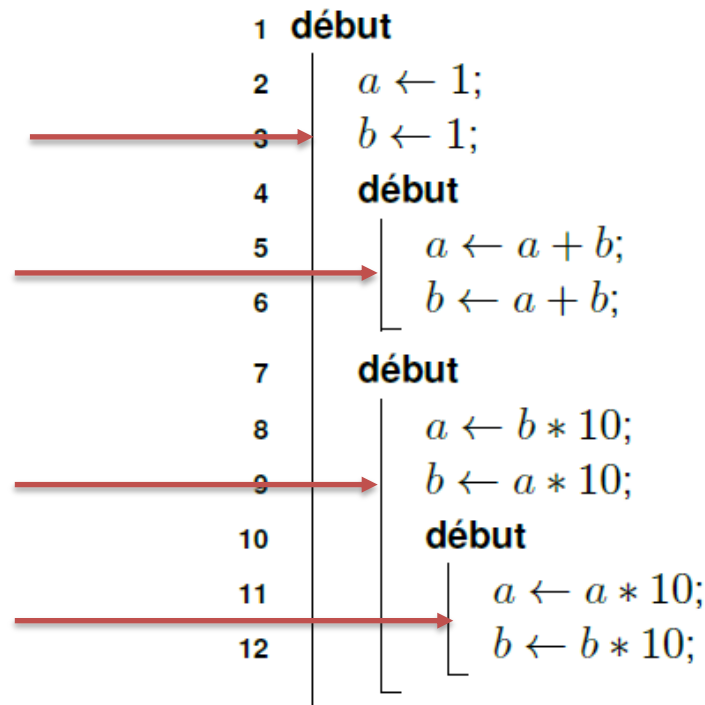
$P \leftarrow D * \pi$

$S \leftarrow R * R * \pi$

Coût : 3

Instructions : organisation en blocs

- Bloc = instructions avec le même niveau d'indentation
- Un bloc est considéré comme une instruction



Instructions : organisation en blocs

```
1  début
2  |   $a \leftarrow 1;$ 
3  |   $b \leftarrow 1;$ 
4  |  début
5  |  |   $a \leftarrow a + b;$ 
6  |  |   $b \leftarrow a + b;$ 
7  |  début
8  |  |   $a \leftarrow b * 10;$ 
9  |  |   $b \leftarrow a * 10;$ 
10 |  |  début
11 |  |  |   $a \leftarrow a * 10;$ 
12 |  |  |   $b \leftarrow b * 10;$ 
```

Coût : 8

Enchaînement alternatif

- L'instruction *si*

```
1 début
2   | si condition alors
3   |   | bloc du alors
4   | sinon
5   |   | bloc du sinon
```

```
1 début
2   | si condition alors
3   |   | bloc du alors
```

```
1 si condition1 alors
2   | suite-1
3 sinon si condition2 alors
4   | suite-2
5 sinon si condition3 alors
6   | suite-3
7 sinon si condition4 alors
8   | suite-4
9 sinon
10  | suite-5
```

Enchaînement alternatif - exemple

Procédure *min(a, b, resultat)*

Entrée : entier *a*
entier *b*

Sortie : entier *resultat*

début

si $a < b$ **alors**

resultat $\leftarrow a$

sinon

resultat $\leftarrow b$

<a=10, b=12>

a	10
b	12
resultat	?

Enchaînement alternatif - exemple

Procédure $\text{min}(a, b, \text{resultat})$

Entrée : entier a
entier b

Sortie : entier resultat

début

si $a < b$ **alors**

$\text{resultat} \leftarrow a$

sinon

$\text{resultat} \leftarrow b$

$\langle a=10, b=12 \rangle$

a	10
b	12
resultat	?

Enchaînement alternatif - exemple

Procédure $\text{min}(a, b, \text{resultat})$

Entrée : entier a
entier b

Sortie : entier resultat

début

si $a < b$ **alors**

$\text{resultat} \leftarrow a$

sinon

$\text{resultat} \leftarrow b$

$\langle a=10, b=12 \rangle$

a	10
b	12
resultat	10

Enchaînement alternatif - exemple

```
Procédure min(a, b, resultat)  
  Entrée      : entier a  
               entier b  
  Sortie      : entier resultat  
  début  
    si  $a < b$  alors  
      |  $resultat \leftarrow a$   
    sinon  
      |  $resultat \leftarrow b$ 
```

Coût : 2

1 **Procédure** *nb-solutions*(*a*, *b*, *c*, *nbSol*)

Entrée : réel *a*, *b*, *c*

Sortie : entier *nbSol*

Postcondition : *nbSol* = nombre de solutions sur les réels de l'équation $ax^2 + bx + c = 0$

Déclaration : réel *delta*

2 **début**

3 **si** *a* = 0 **alors**

4 **si** *b* = 0 **alors**

5 **si** *c* = 0 **alors**

6 | *nbSol* ← ∞

7 **sinon**

8 | *nbSol* ← 0

9 **sinon**

10 | *nbSol* ← 1

11 **sinon**

12 | *delta* ← *b* * *b* - 4 * *a* * *c*

13 **si** *delta* < 0 **alors**

14 | *nbSol* ← 0

15 **sinon si** *delta* = 0 **alors**

16 | *nbSol* ← 1

17 **sinon**

18 | *nbSol* ← 2

Coût : ?

1 **Procédure** *nb-solutions*(*a*, *b*, *c*, *nbSol*)

Entrée : réel *a*, *b*, *c*

Sortie : entier *nbSol*

Postcondition : *nbSol* = nombre de solutions sur les réels de l'équation $ax^2 + bx + c = 0$

Déclaration : réel *delta*

2 **début**

3 **si** *a* = 0 **alors**

4 **si** *b* = 0 **alors**

5 **si** *c* = 0 **alors**

6 | *nbSol* ← ∞

7 **sinon**

8 | *nbSol* ← 0

9 **sinon**

10 | *nbSol* ← 1

11 **sinon**

12 | *delta* ← *b* * *b* - 4 * *a* * *c*

13 **si** *delta* < 0 **alors**

14 | *nbSol* ← 0

15 **sinon si** *delta* = 0 **alors**

16 | *nbSol* ← 1

17 **sinon**

18 | *nbSol* ← 2

Coût : 3 ou 4 ou 5 → 5

Recherche du plus petit de 3 nombres

```
1 Procédure plus-petit(a, b, c, pp)
2   ...
3   début
4       si  $a < b$  alors
5           |  $pp \leftarrow a$ 
6       sinon
7           |  $pp \leftarrow b$ 
           //  $pp$  = plus petit nombre entre  $a$  et  $b$ 
8       si  $c < pp$  alors
9           |  $pp \leftarrow c$ 
```

Coût : ?

Enchaînement répétitif

tant que (*condition*) **faire**

└ bloc de la boucle

pour (*init; cond; passage*) **faire**

└ bloc de la boucle

répéter

└ bloc de la boucle

jusqu'à (*condition d'arrêt*);

L'instruction *tant que*

tant que (*condition*) **faire**

└ bloc de la boucle

- Répétition d'un bloc d'instruction un nombre de fois

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	?
cpt	?

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	1
cpt	?

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	1
cpt	0

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	1
cpt	0

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	2
cpt	0

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	2
cpt	1

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	2
cpt	1

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	4
cpt	1

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	4
cpt	2

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	4
cpt	2

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	8
cpt	2

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	8
cpt	3

tant que – exemple (x^n)

1 Procédure *puissance*(n, x, p)

 Entrée : entier n

 réel x

 Sortie : réel p

 Précondition : $n \geq 0$

 Postcondition : $p = x^n$

 Déclaration : entier cpt

2 début

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

n	3
x	2
p	8
cpt	3

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

$\langle n = 3, x = 2 \rangle$

Coût : ?

tant que – exemple (x^n)

1 **Procédure** *puissance*(n, x, p)

Entrée : entier n

$\langle n = 3, x = 2 \rangle$

 réel x

Sortie : réel p

Précondition : $n \geq 0$

Postcondition : $p = x^n$

Déclaration : entier cpt

2 **début**

3 $p \leftarrow 1$

4 $cpt \leftarrow 0$

5 **tant que** $cpt < n$ **faire**

 // invariant : $p = x^{cpt}$

6 $p \leftarrow p * x$

7 $cpt \leftarrow cpt + 1$

Coût : $1+1+3*3+1 = 12$

Coût : $1+1+3*n+1 = 3*(n+1)$

tant que - exemple

début

```
|  $p \leftarrow 1$   
|  $cpt \leftarrow 0$   
| tant que  $cpt < n$  faire  
|   // invariant :  $p = x^{cpt}$   
|    $p \leftarrow p * x$   
|    $cpt \leftarrow cpt + 1$ 
```

début

```
|  $p \leftarrow 1$   
|  $cpt \leftarrow 1$   
| tant que  $cpt \leq n$  faire  
|   // invariant :  $p = x^{cpt-1}$   
|    $p \leftarrow p * x$   
|    $cpt \leftarrow cpt + 1$ 
```

début

```
|  $p \leftarrow 1$   
|  $cpt \leftarrow n$   
| tant que  $cpt > 0$  faire  
|   // invariant :  $p = x^{n-cpt}$   
|    $p \leftarrow p * x$   
|    $cpt \leftarrow cpt - 1$ 
```

L'instruction *pour*

début

$p \leftarrow 1$
 $cpt \leftarrow 0$

initialisation

tant que $cpt < n$ **faire**

condition

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

traitement

$cpt \leftarrow cpt + 1$

passage

L'instruction *pour*

```
1 pour (init; cond; passage) faire  
2   | traitement
```

```
1 init  
2 tant que cond faire  
3   | traitement  
4   | passage
```

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0 ; cpt < n ; cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	?
cpt	?

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0 ; cpt < n ; cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	1
cpt	?

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0$; $cpt < n$; $cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	1
cpt	0

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0$; $cpt < n$; $cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	1
cpt	0

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0; cpt < n; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	2
cpt	0

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0; cpt < n; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	2
cpt	1

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0$; $cpt < n$; $cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	2
cpt	1

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0 ; cpt < n ; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	4
cpt	1

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0; cpt < n; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	4
cpt	2

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0$; $cpt < n$; $cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	4
cpt	2

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0 ; cpt < n ; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	8
cpt	2

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0; cpt < n; cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	8
cpt	3

L'instruction *pour* - exemple

$\langle n = 3, x = 2 \rangle$

Procédure *puissance*(n, x, p)

...

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0$; $cpt < n$; $cpt \leftarrow cpt + 1$) **faire**

// invariant : $p = x^{cpt}$

$p \leftarrow p * x$

n	3
x	2
p	8
cpt	3

Coût : ?

L'instruction *pour* - exemple

début

$p \leftarrow 1$

pour ($cpt \leftarrow 0 ; cpt < n ; cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt}$

$p \leftarrow p * x$

début

$p \leftarrow 1$

pour ($cpt \leftarrow 1 ; cpt \leq n ; cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt-1}$

$p \leftarrow p * x$

début

$p \leftarrow 1$

pour ($cpt \leftarrow 1 ; cpt \leq n ; cpt \leftarrow cpt + 1$) **faire**

 // invariant : $p = x^{cpt-1}$

$p \leftarrow p * x$

L'instruction *répéter*

- 1 **répéter**
- 2 | bloc de la boucle
- 3 **jusqu'à** (*condition d'arrêt*)

- 1 bloc de la boucle
- 2 **tant que** *condition d'arrêt* **faire**
- 3 | bloc de la boucle

L'instruction *répéter* - exemple

```
1 Procédure puissance( $n, x, p$ )  
2   ...  
3   début  
4        $p \leftarrow 1$   
5        $cpt \leftarrow 0$   
6       répéter  
7           si ( $cpt > 0$ ) alors  
8                $p \leftarrow p * x$   
9                $cpt \leftarrow cpt + 1$   
10      jusqu'à ( $cpt \geq n$ )
```

Instructions I/O

- saisir(x)
- afficher(x)

Rappel

- Programme = algorithmes + structures de données
- Algorithme :
 - Pseudo-code
 - Partie statique
 - Variables
 - Partie dynamique
 - Instructions
 - Coût \rightarrow min !

Problèmes – 1/2

- A. Calculer la somme de n nombres réels
- B. Calculer la moyenne de n nombres réels
- C. Même problème que B, mais la moyenne pour les nombres positifs (≥ 0)

Problèmes – 2/2

- <https://uva.onlinejudge.org/>
- <http://uhunt.felix-halim.net/>
- <http://france-ioi.fr>
- <http://codeforces.com/>
- TopCoder, Google CodeJam, Codingame, Prologin, Project Euler ...

Problème (UVA) : 1585 - Score

There is an objective test result such as ``OOXXOXXOOO". An `O' means a correct answer of a problem and an `X' means a wrong answer. The score of each problem of this test is calculated by itself and its just previous consecutive `O's only when the answer is correct. For example, the score of the 10th problem is 3 that is obtained by itself and its two previous consecutive `O's.

Therefore, the score of ``OOXXOXXOOO" is 10 which is calculated by ``1+2+0+0+1+0+0+1+2+3".

You are to write a program calculating the scores of test results.

?