

Initiation à la Programmation

<http://www.fil.univ-lille1.fr/portail>

Cédric Lhoussaine

Ext. M3, bureau 217

Tél: 03 28 77 85 70

Mel: Cedric.Lhoussaine@lifl.fr

Responsable du cours:

Éric Wegrzynowski

Eric.Wegrzynowski@lifl.fr

Évaluation

- ✓ TD : note sur 20 de contrôle continu
- ✓ TP : note sur 20 (contrôle en fin de semestre)
- ✓ EX : note sur 20 pour l'examen en fin de semestre

$$\text{Note finale} = (\text{TP} + 3\sup(\text{EX}, (2\text{EX} + \text{TD})/3))/4$$

Qu'est-ce que l'informatique ?

- ✓ **Une technologie**

Un ordinateur est une machine universelle capable de traiter automatiquement des informations

- ✓ **Un outil**

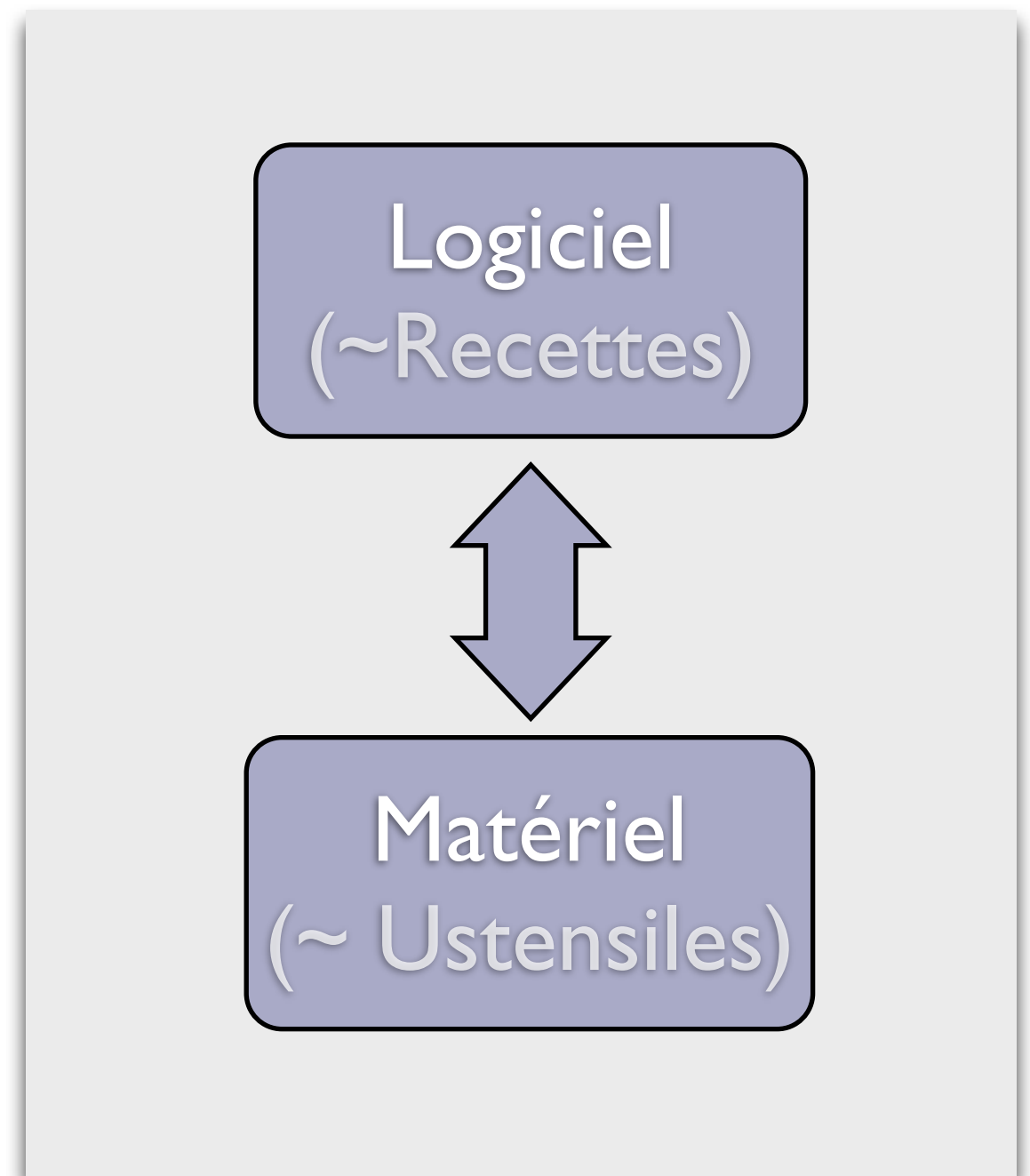
Pour la gestion, les communications, l'enseignement, les loisirs, ...

- ✓ **Une discipline scientifique**

Algorithmique, Théorie des langages, Programmation, Systèmes et Architecture, Bases de données, ...

Systemes informatiques

On distingue dans les
systemes informatiques
deux éléments constitutifs
distincts



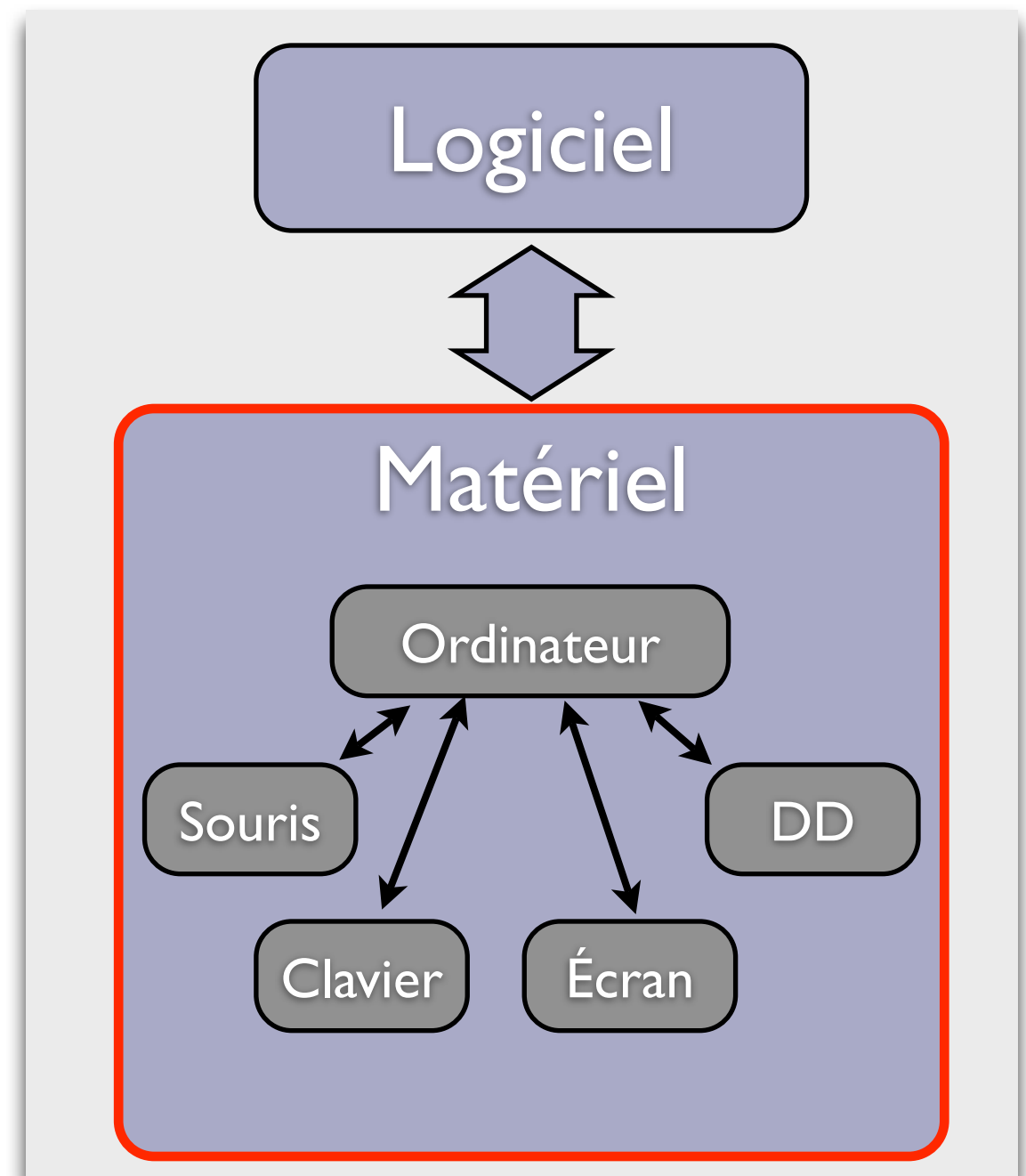
Systèmes informatiques

Stockage des données*
persistantes

Mémoires de masse

- ▶ DD (Disque Dur)
- ▶ Disquette, etc.

(*) *textes, images, etc.*



Systemes informatiques

Stockage des données

Mémoire centrale

- ▶ RAM (*Random Access Memory*)
- ▶ ROM (*Read Only Memory*)

Traitement des données

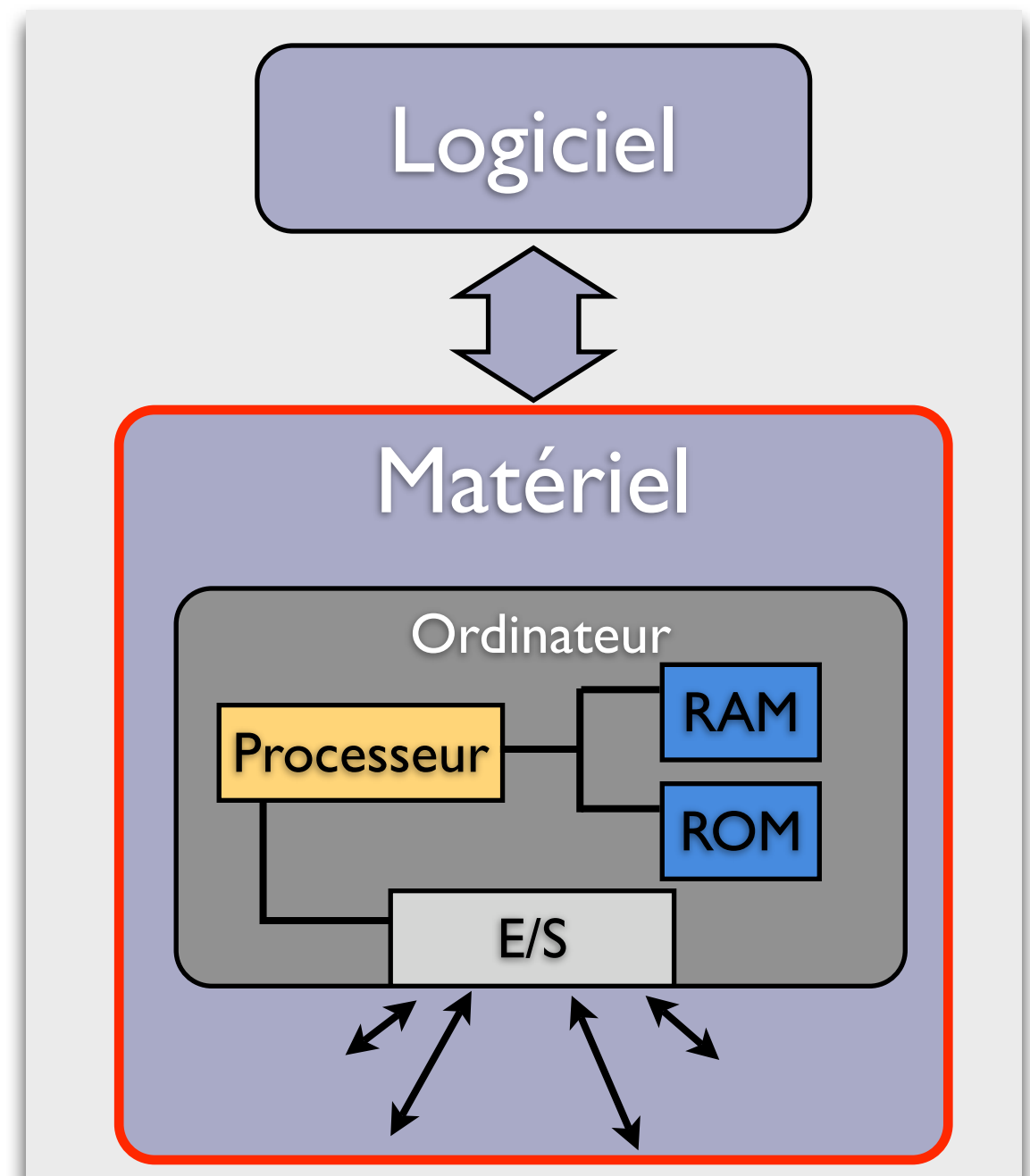
Processeur

Communications

Unités d'Entrées/Sorties

(Modèle de Von Neumann ~1940)

(Modèle théorique: Machine de Turing ~1936)

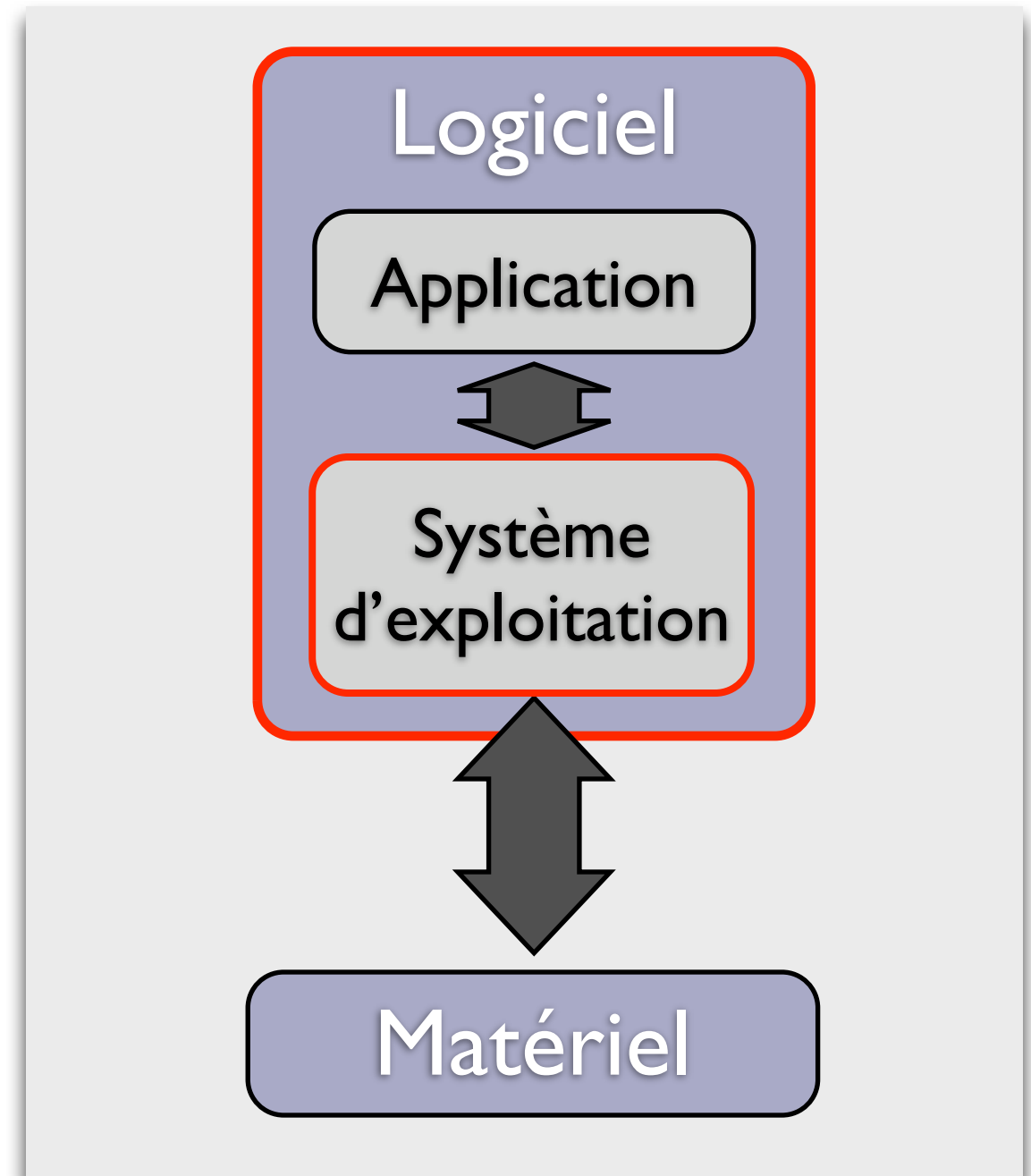


Systemes informatiques

Assure la liaison entre
ressources matérielles,
utilisateur(s) et applications

- ▶ Gestion du processeur
- ▶ Gestion de la mémoire vive
- ▶ Gestion des E/S
- ▶ Gestion des applications
- ▶ Gestion des droits
- ▶ Gestion des fichiers
- ▶ Gestion des services réseaux

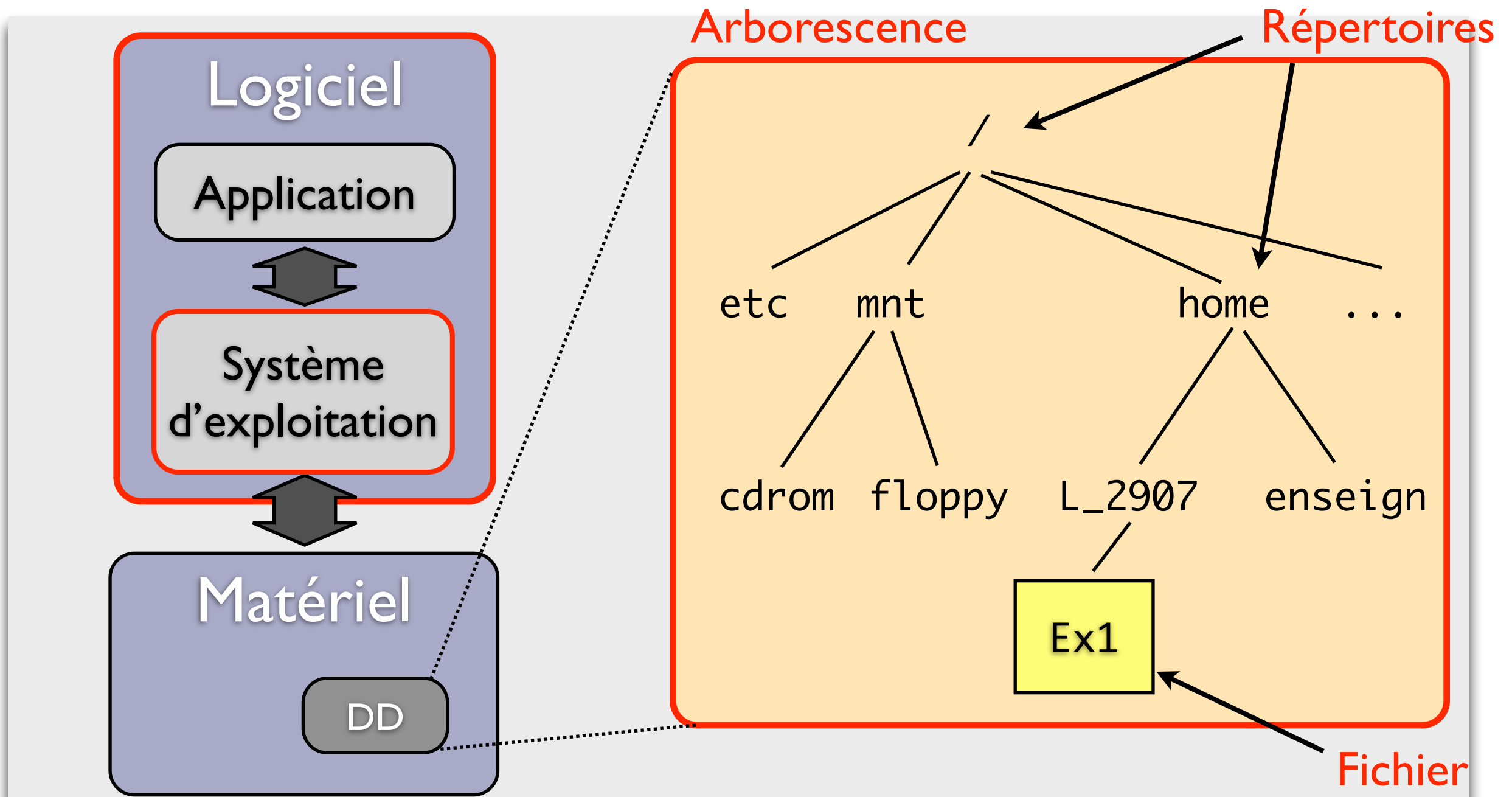
*(Windows XX, Mac OS, Amiga OS, Solaris,
GNU/Linux, FreeBSD, Palm OS,
Symbian OS, ...)*



Systèmes informatiques

Gestion des fichiers

Nom absolu du fichier Ex1: /home/dupont/Ex1

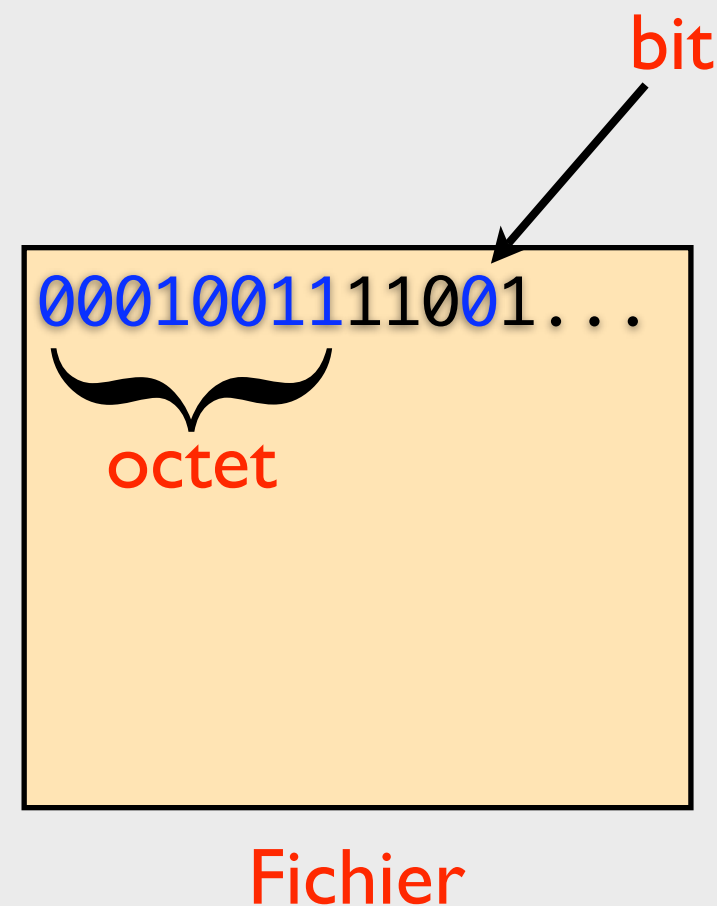
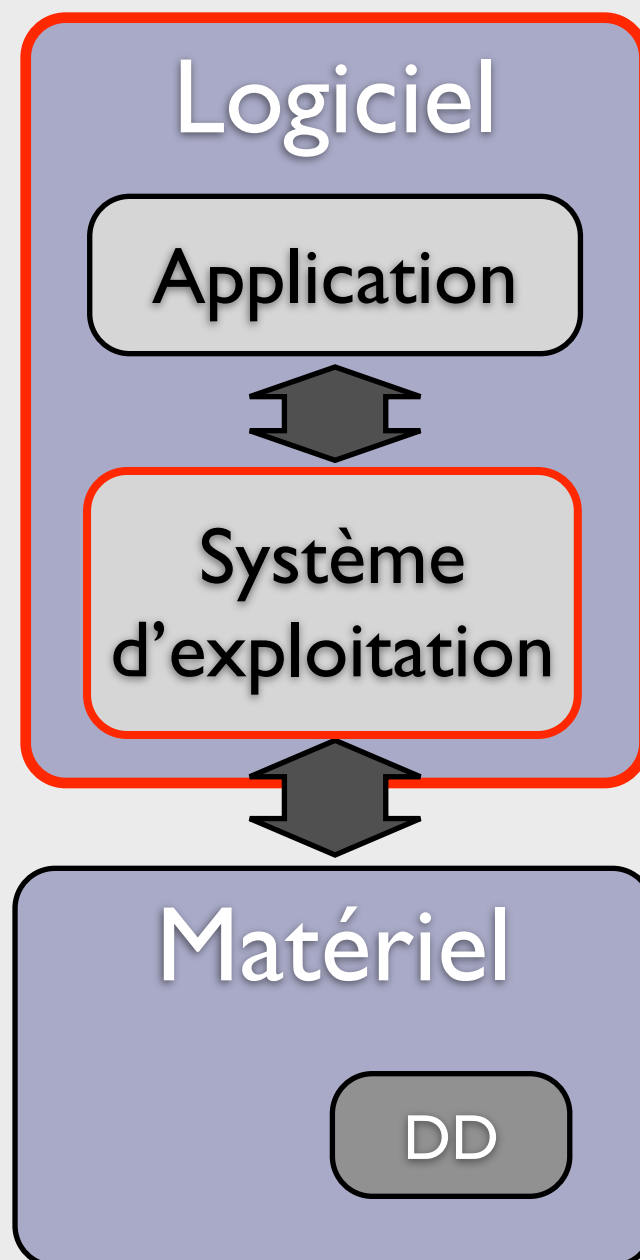


Systèmes informatiques

Fichiers

1 octet = 8 bits ($2^8 = 256$ octets possibles)

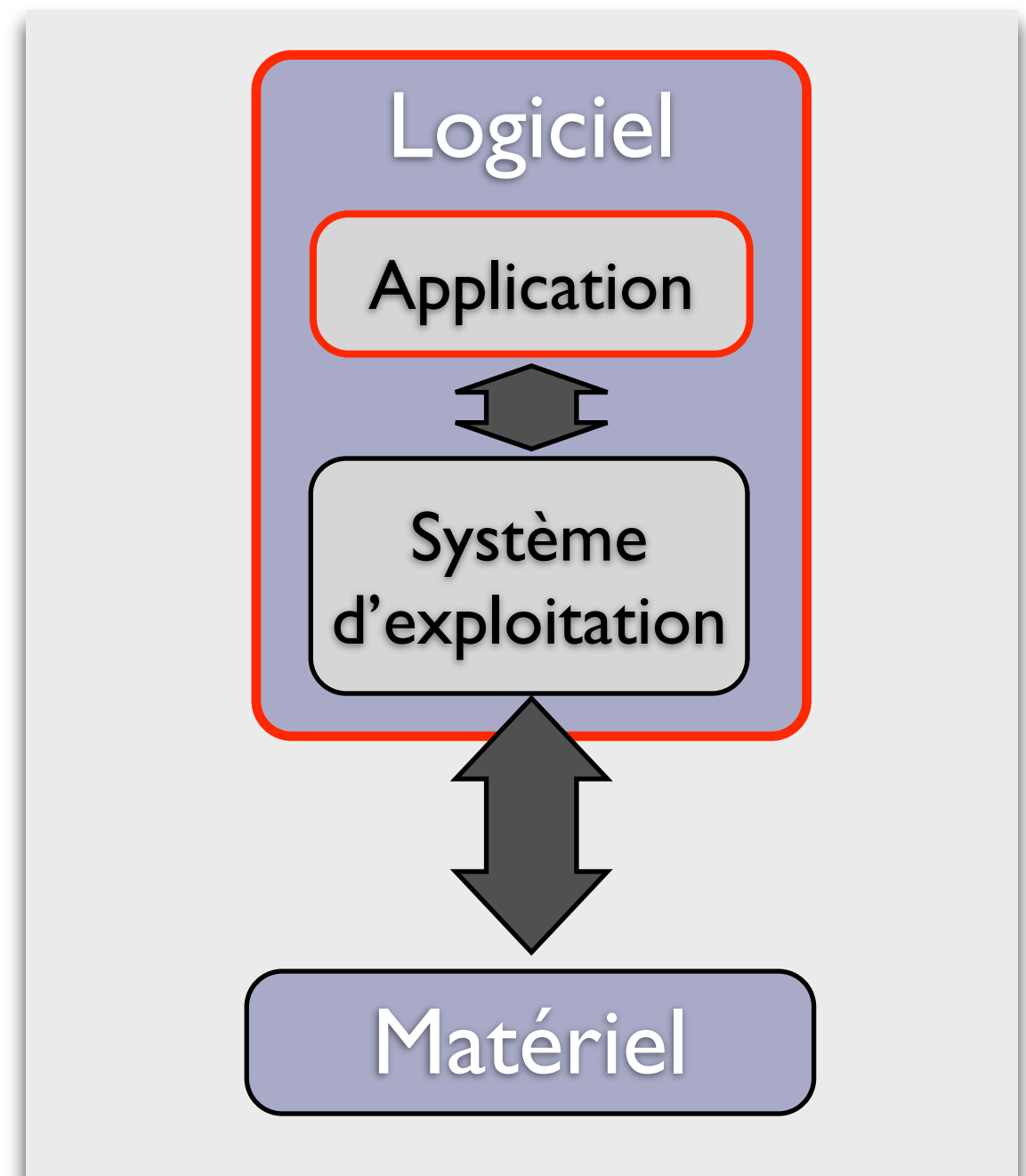
1 Ko = 1024 octets 1 Mo = 1024 Ko



Systèmes informatiques

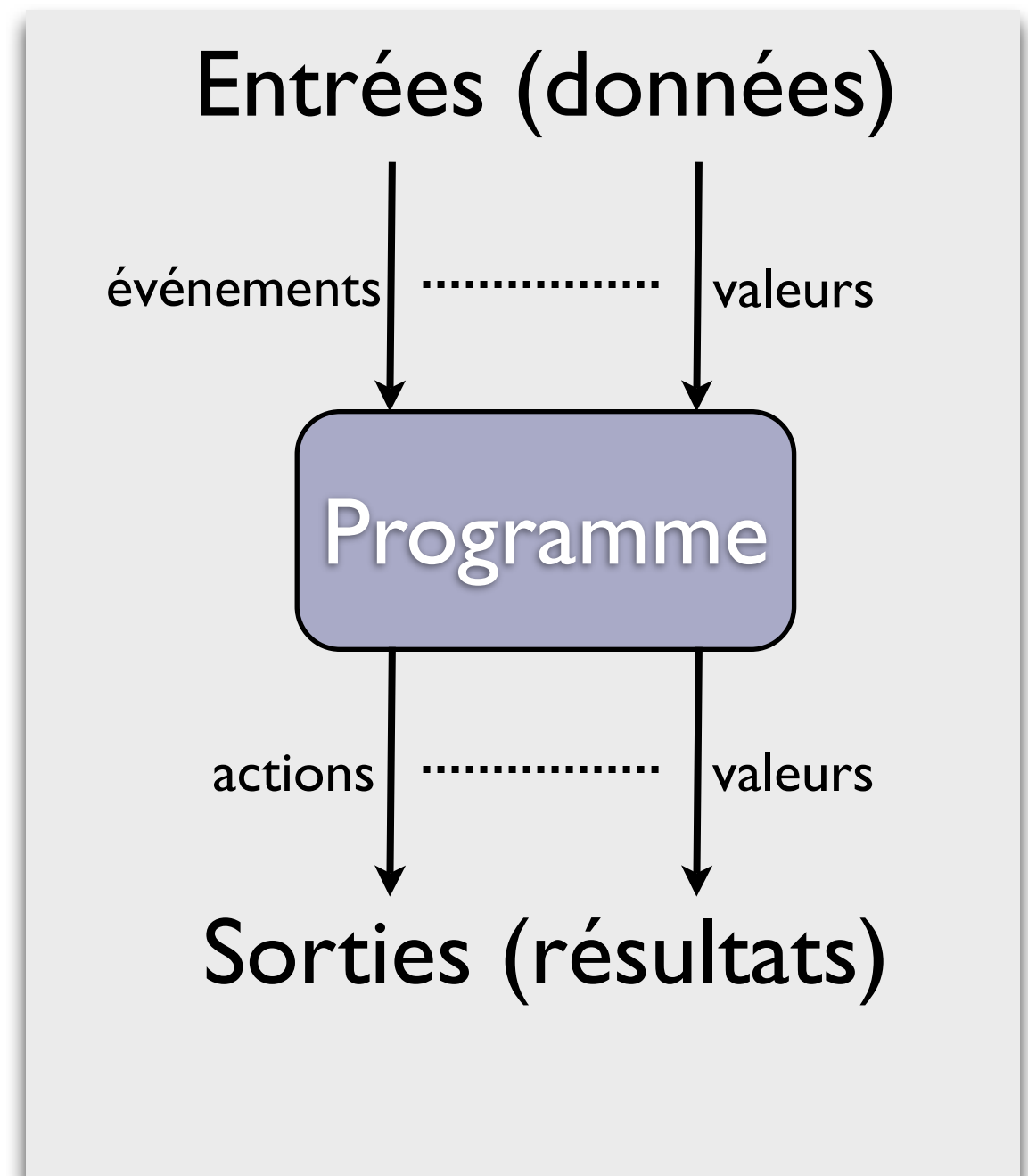
Programmes spécialisés à destination des utilisateurs

- ▶ Traitement de texte
- ▶ Éditeur de texte
- ▶ Tableur
- ▶ Manipulation/Traitement d'images
- ▶ Lecteurs audio/vidéo
- ▶ Courrier électronique
- ▶ Système de fenêtrage
- ▶ Jeux
- ▶ ...



Applications

- ✓ Application = (ensemble de) **programme**(s) élaborés pour réaliser une tâche
- ✓ Programme = séquence d'**instructions** qu'un ordinateur peut interpréter et exécuter



Applications

Un **langage** (de programmation) spécifie l'ensemble des **instructions** disponibles.

```
program HelloWorld;  
uses SysUtils;  
  
begin  
    write('Hello World!');  
    readln;  
end.
```



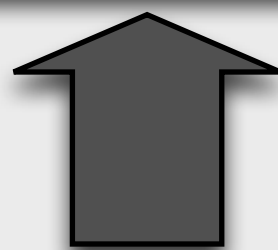
Fonctionnel

Orienté Objet

Impératif

Déclaratif

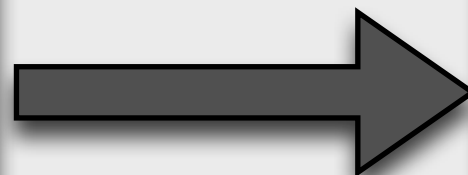
Aspects ...



Haut niveau

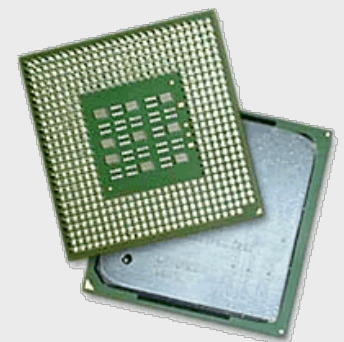
Bas niveau

```
pushl    %ebp  
movl     %esp, %ebp  
subl     $8, %esp  
andl     $-16, %esp  
movl     $0, %eax  
subl     %eax, %esp .....
```



Exécutable

```
100011101010111001110  
101011101110011101010  
100001011101010111110  
011101010111...
```



Applications

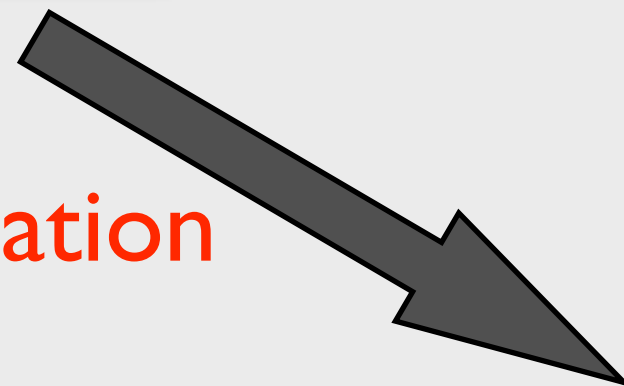
Les programmes sont écrits dans des langages de hauts niveaux et **compilés** dans des programmes directement **exécutables** par l'ordinateur.

Source

```
program HelloWorld;  
uses SysUtils;  
  
begin  
    write('Hello World!');  
    readln;  
end.
```

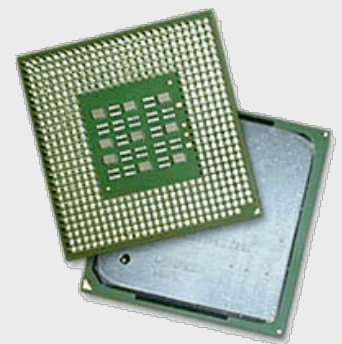


Compilation



Exécutable

```
100011101010111001110  
101011101110011101010  
100001011101010111110  
011101010111...
```



Objectifs

Initiation à la programmation *impérative*

- ✓ Analyse d'un problème
 - ▶ Identification des données et du résultat à atteindre
 - ▶ Représentation des données
- ✓ Résolution du problème
 - ▶ Recherche d'un algorithme de résolution
 - ▶ Implantation de l'algorithme

Vers le premier programme

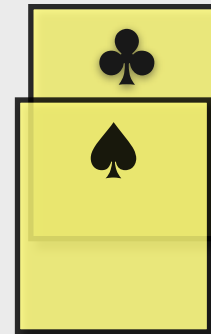
Programmation d'un robot manipulateur de cartes

- ✓ **Unité** cartes: ensemble des procédures et fonctions de manipulation de cartes (unités math, etc.)
- ✓ Il faut spécifier explicitement l'unité qu'on désire utiliser dans un programme Pascal.
- ✓ L'unité cartes contient
 - ▶ La définition (*codage*) des cartes
 - ▶ Les fonctions et procédures de manipulation (créer un tas, regarder la couleur d'une carte, ...)

Vers le premier programme

Situation initiale

1



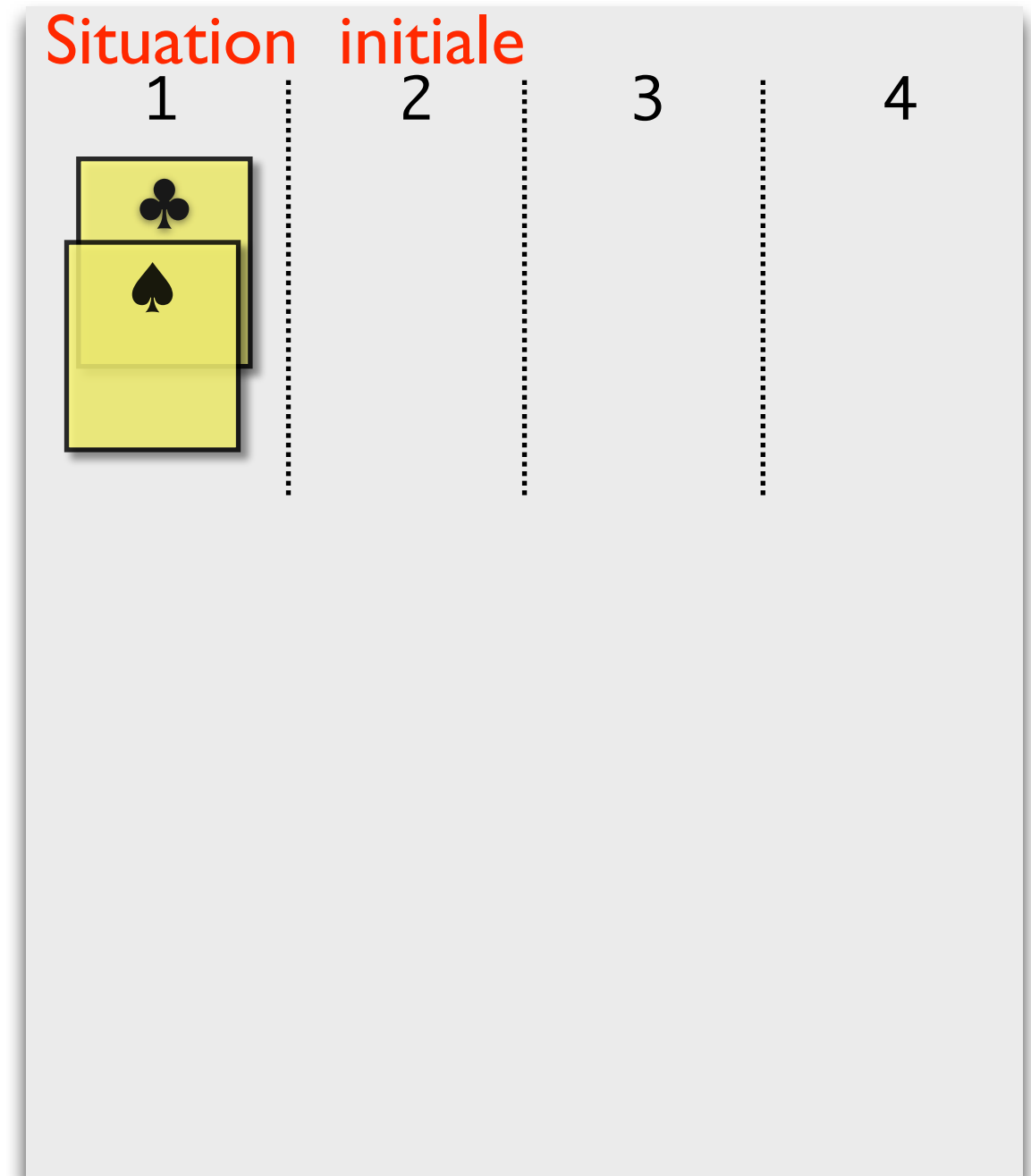
2

3

4

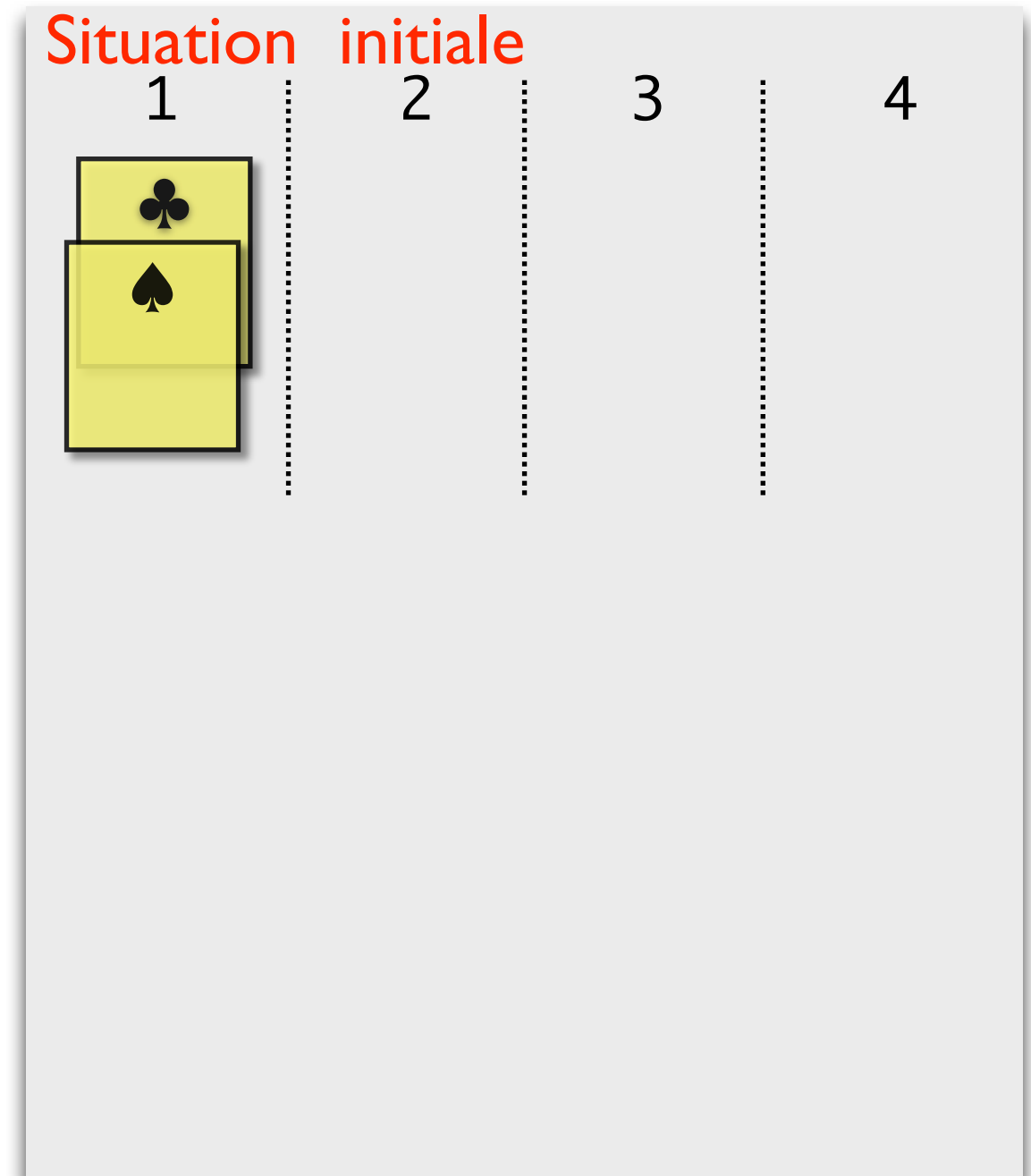
Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.



Vers le premier programme

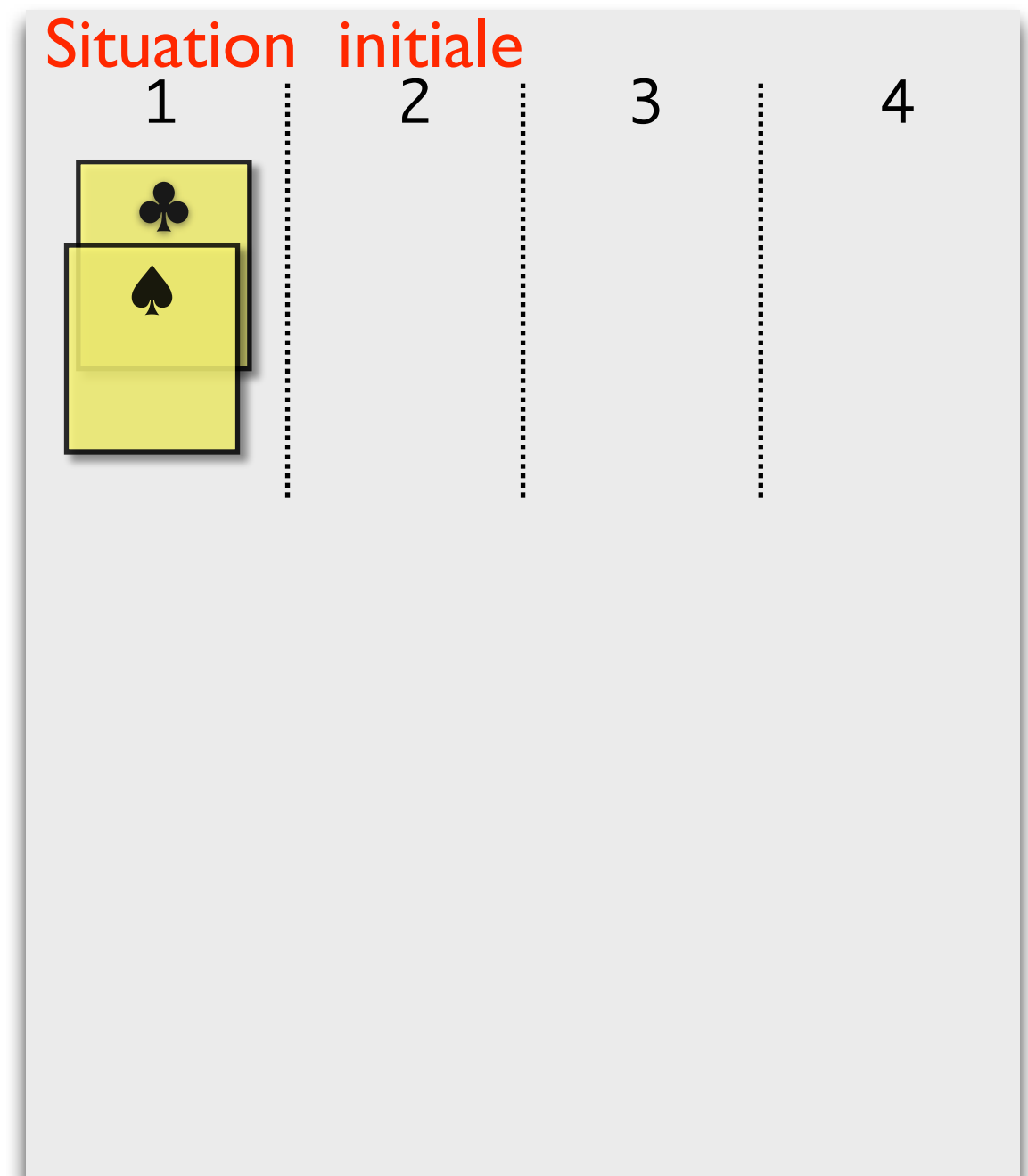
Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.



Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

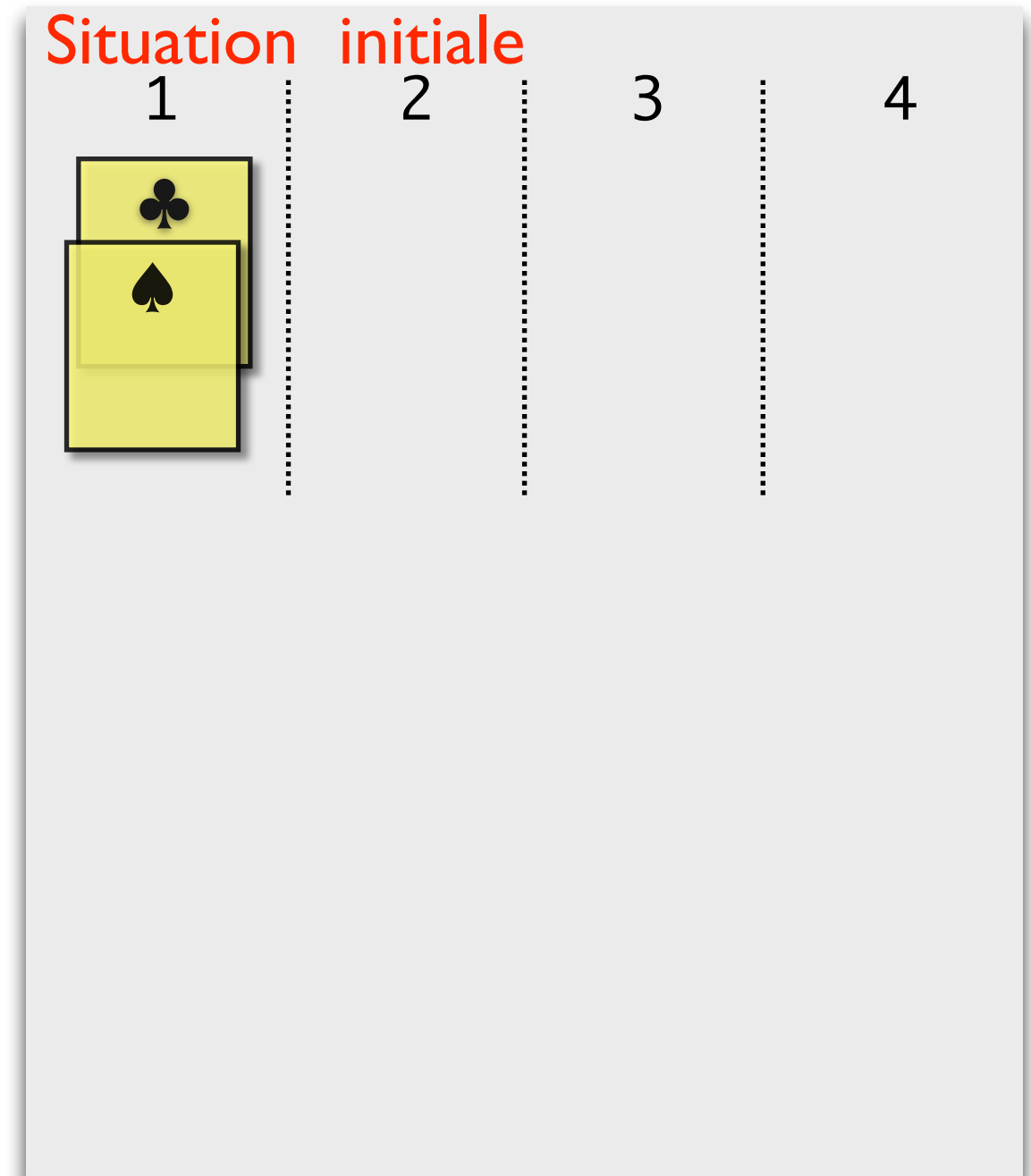


Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

1. Analyse du problème

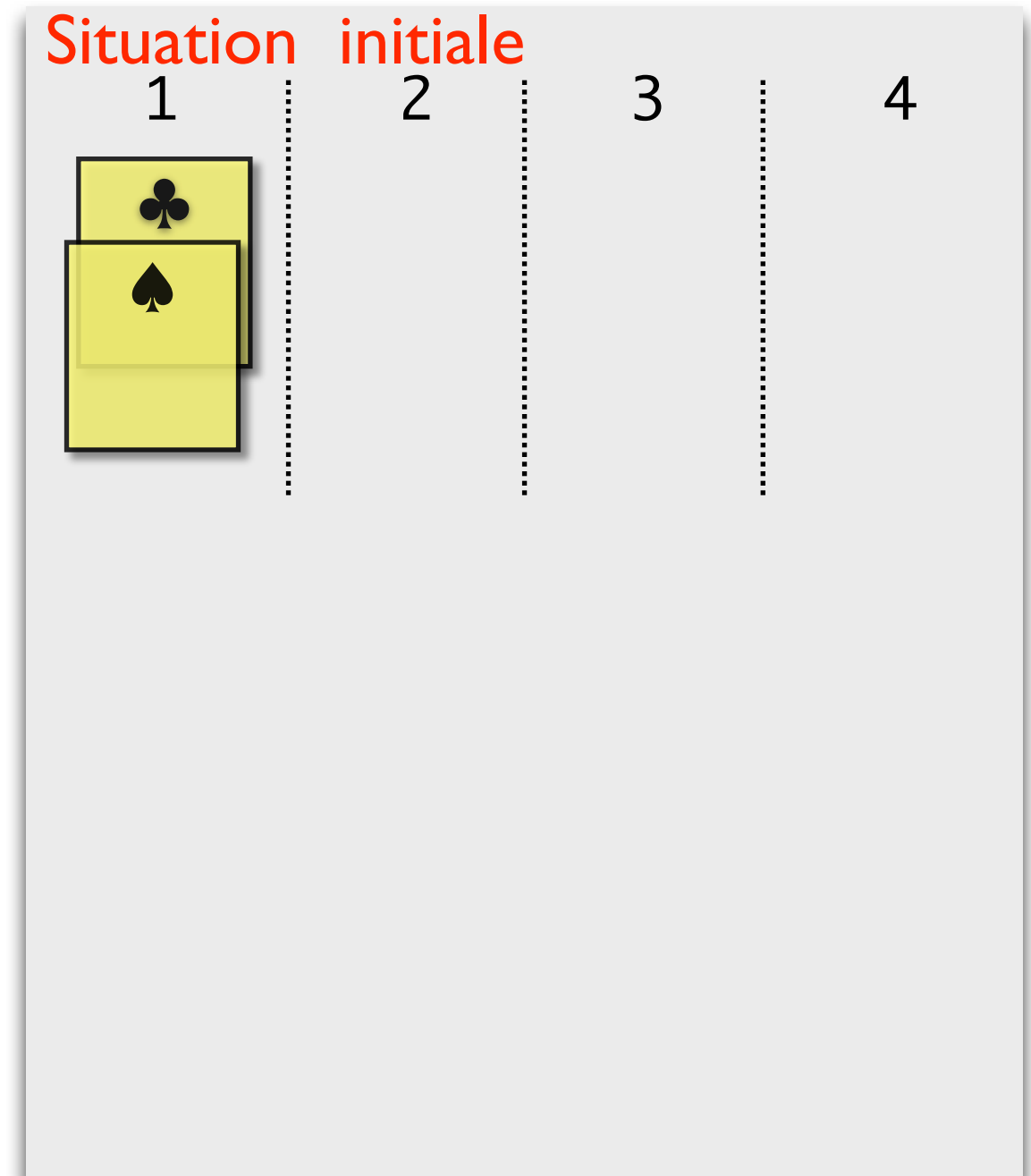


Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

1. Analyse du problème
2. Codage (en Pascal)

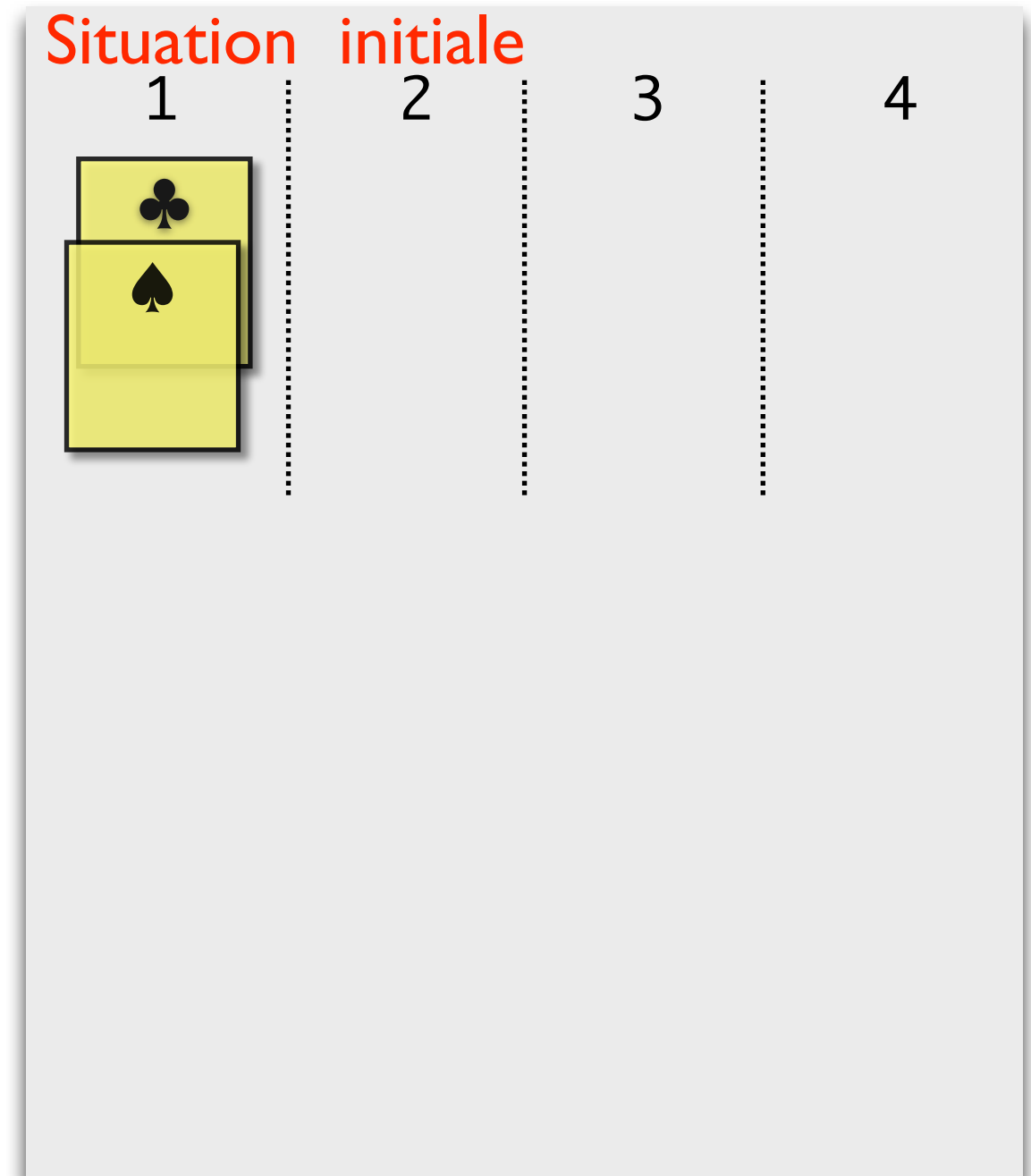


Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

1. Analyse du problème
2. Codage (en Pascal)
3. Tests

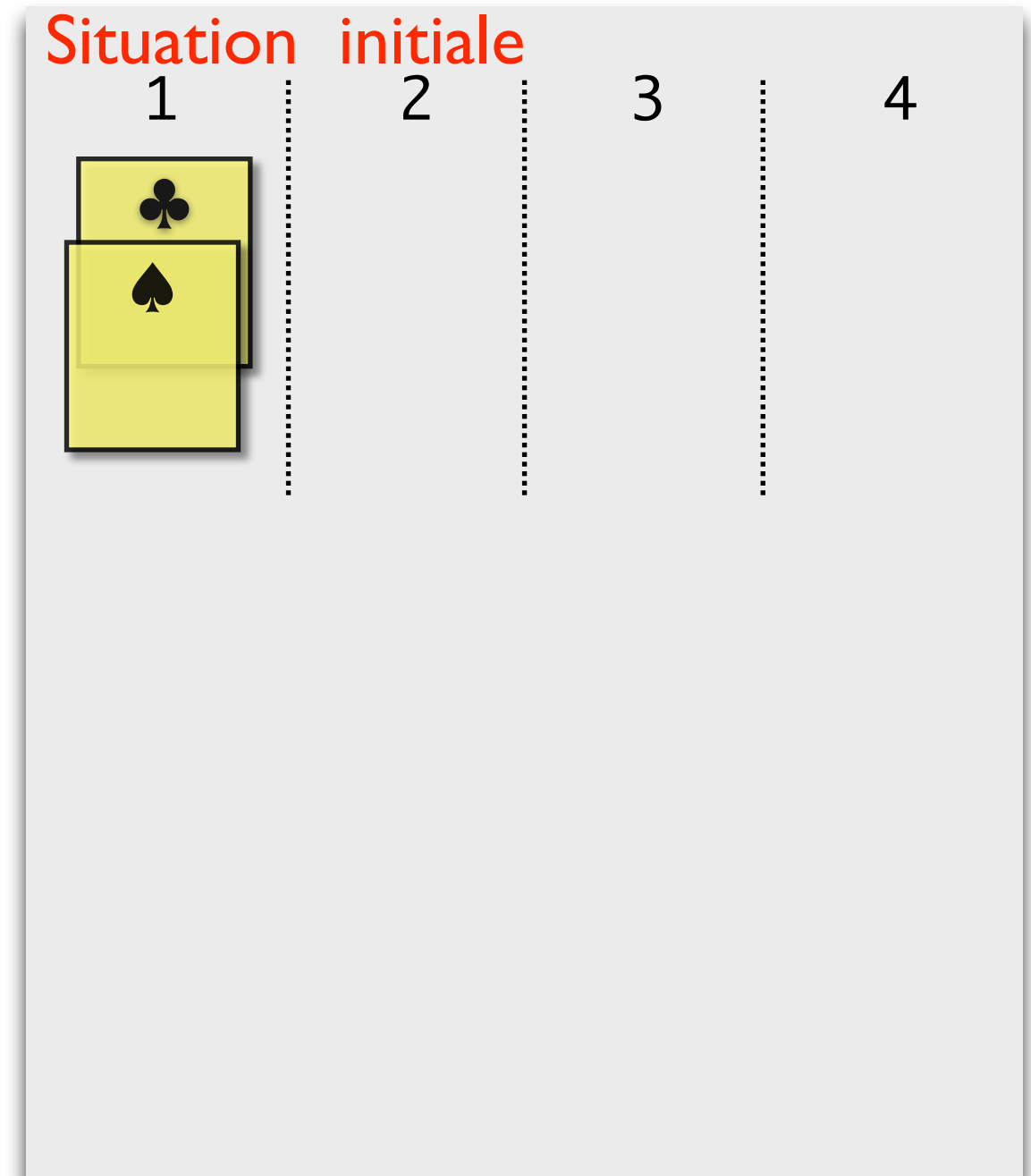


Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

1. Analyse du problème
2. Codage (en Pascal)
3. Tests



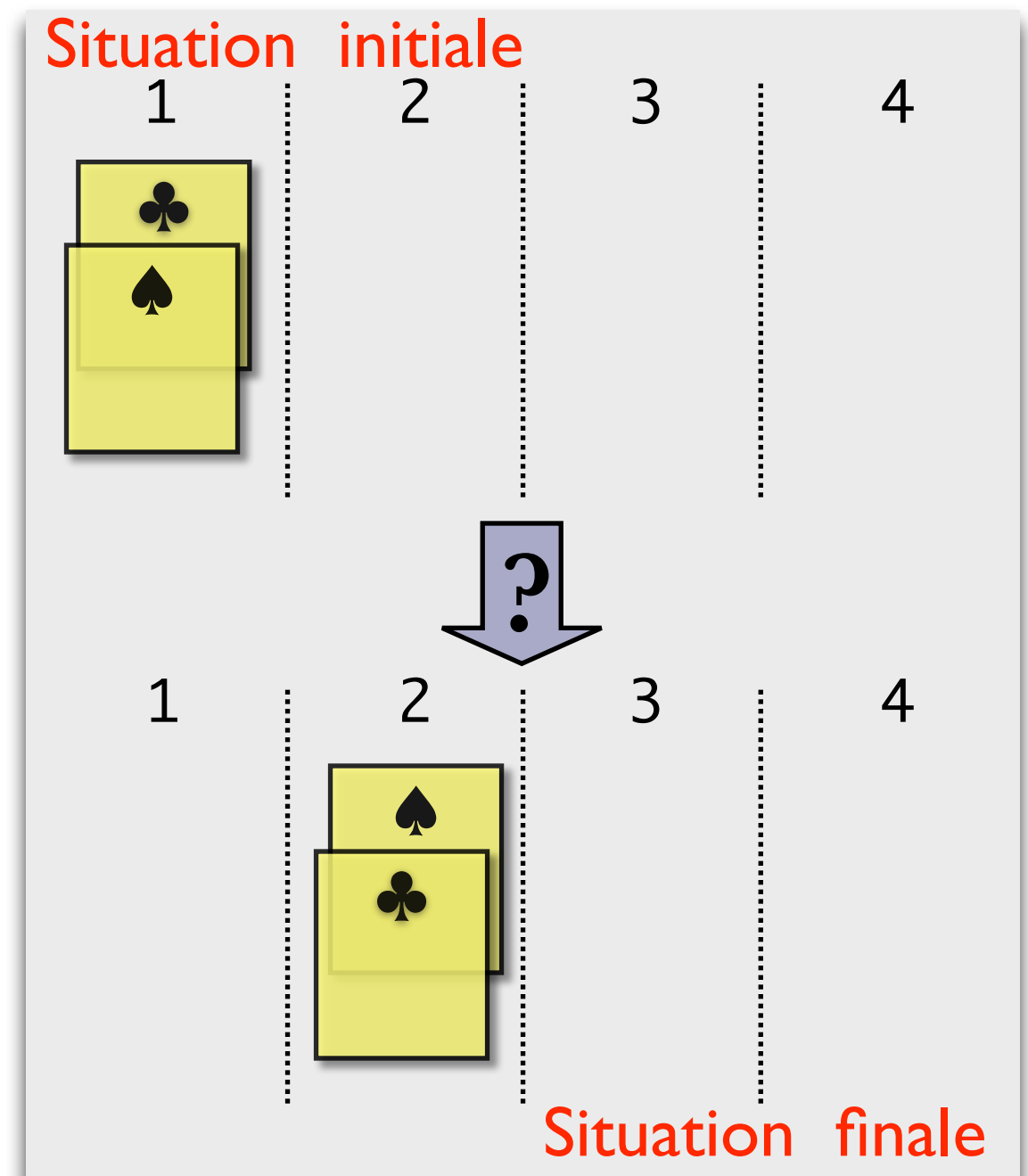
Vers le premier programme

Écrire un programme c'est décrire la suite des actions à réaliser pour passer d'une situation (ou configuration) initiale à une situation finale.

Plusieurs étapes:

1. Analyse du problème
2. Codage (en Pascal)
3. Tests

Exercice: Décrivez les actions à réaliser pour passer de la situation initiale à la situation finale ci-contre



Vers le premier programme

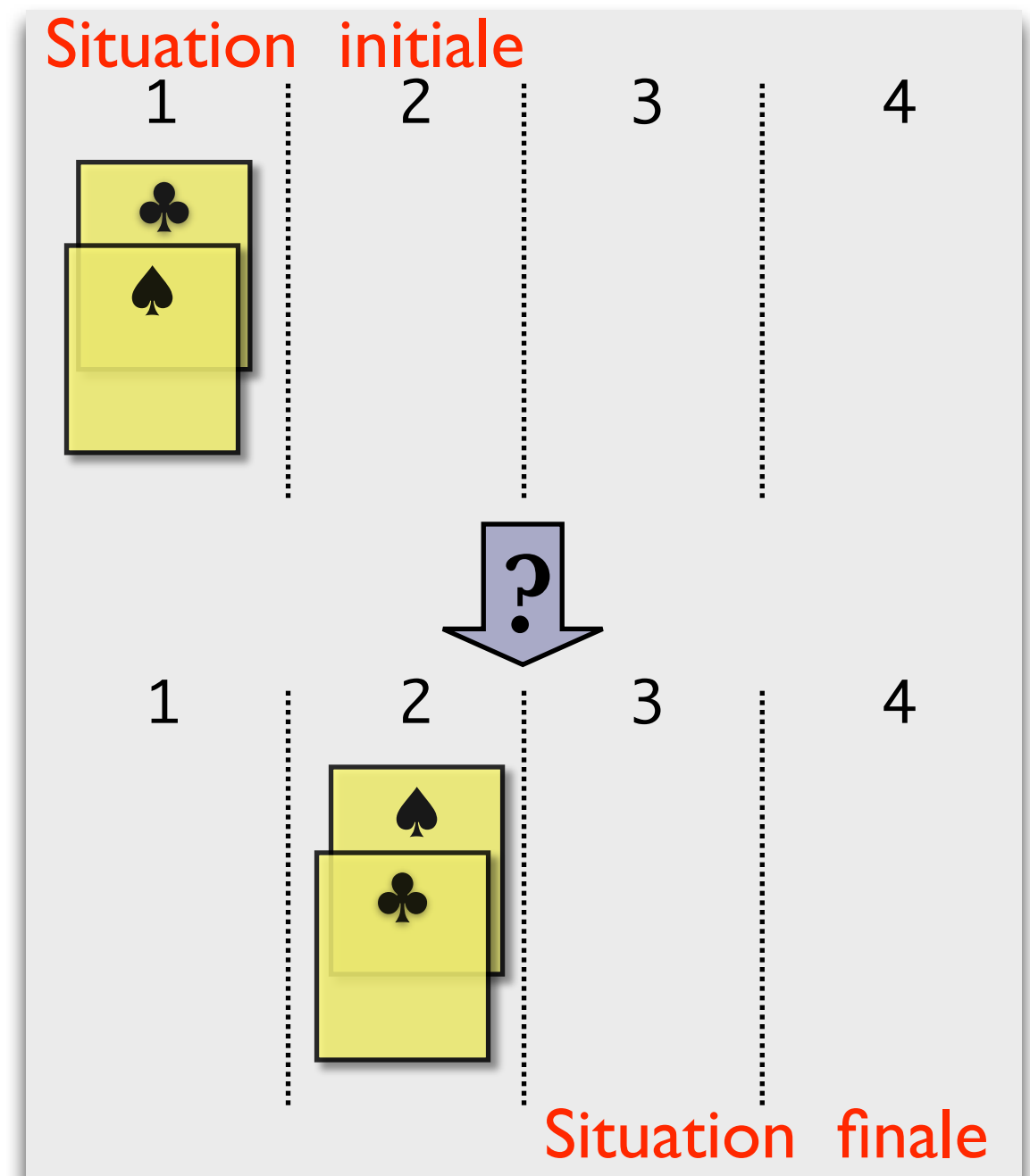
Analyse

✓ *Situation initiale*

Tas 1: 1 ♠ sur 1 ♣
autres tas vides

✓ *Situation finale*

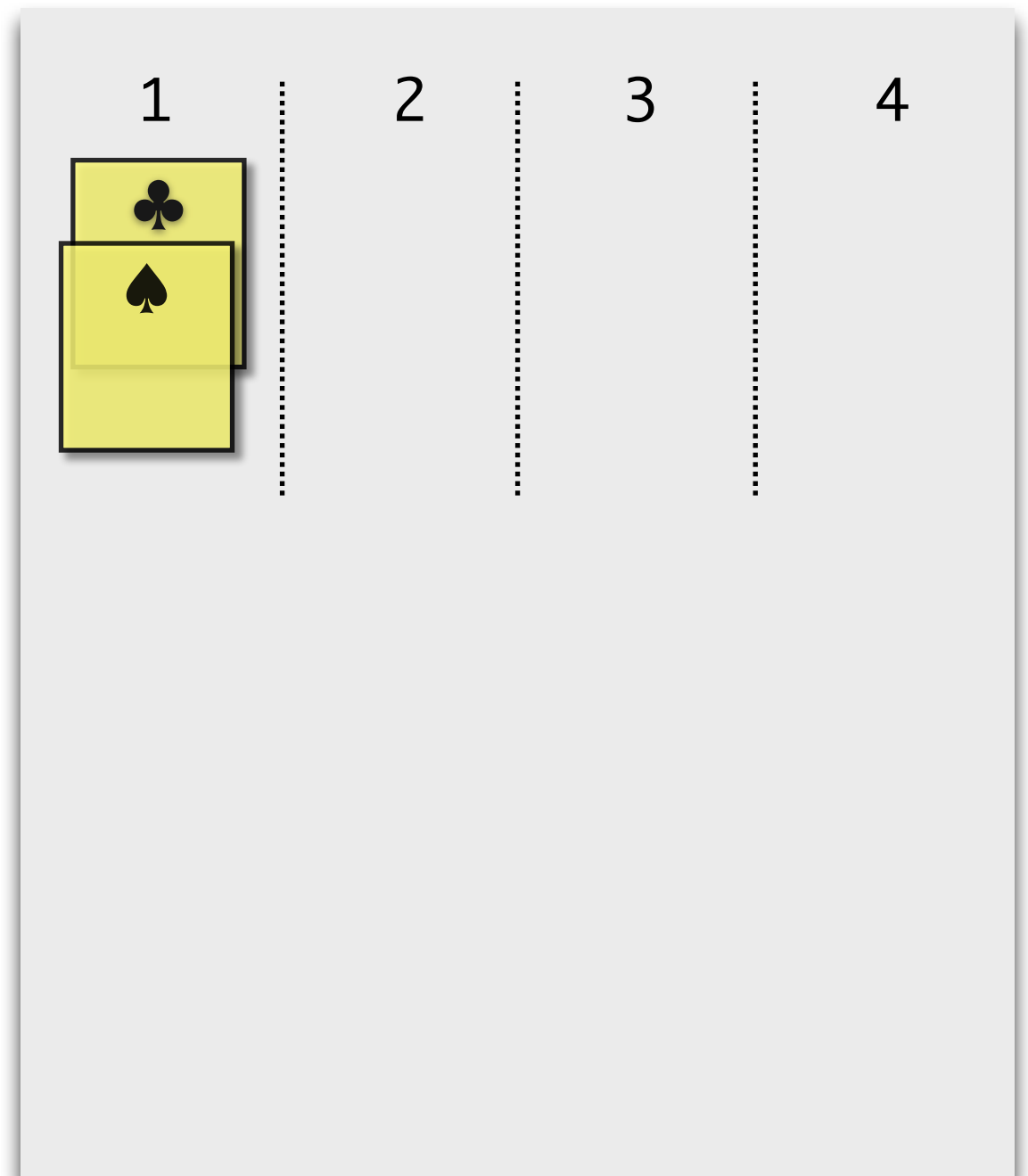
Tas 2: 1 ♣ sur 1 ♠
autres tas vides



Vers le premier programme

Analyse

✓ *Algorithme*

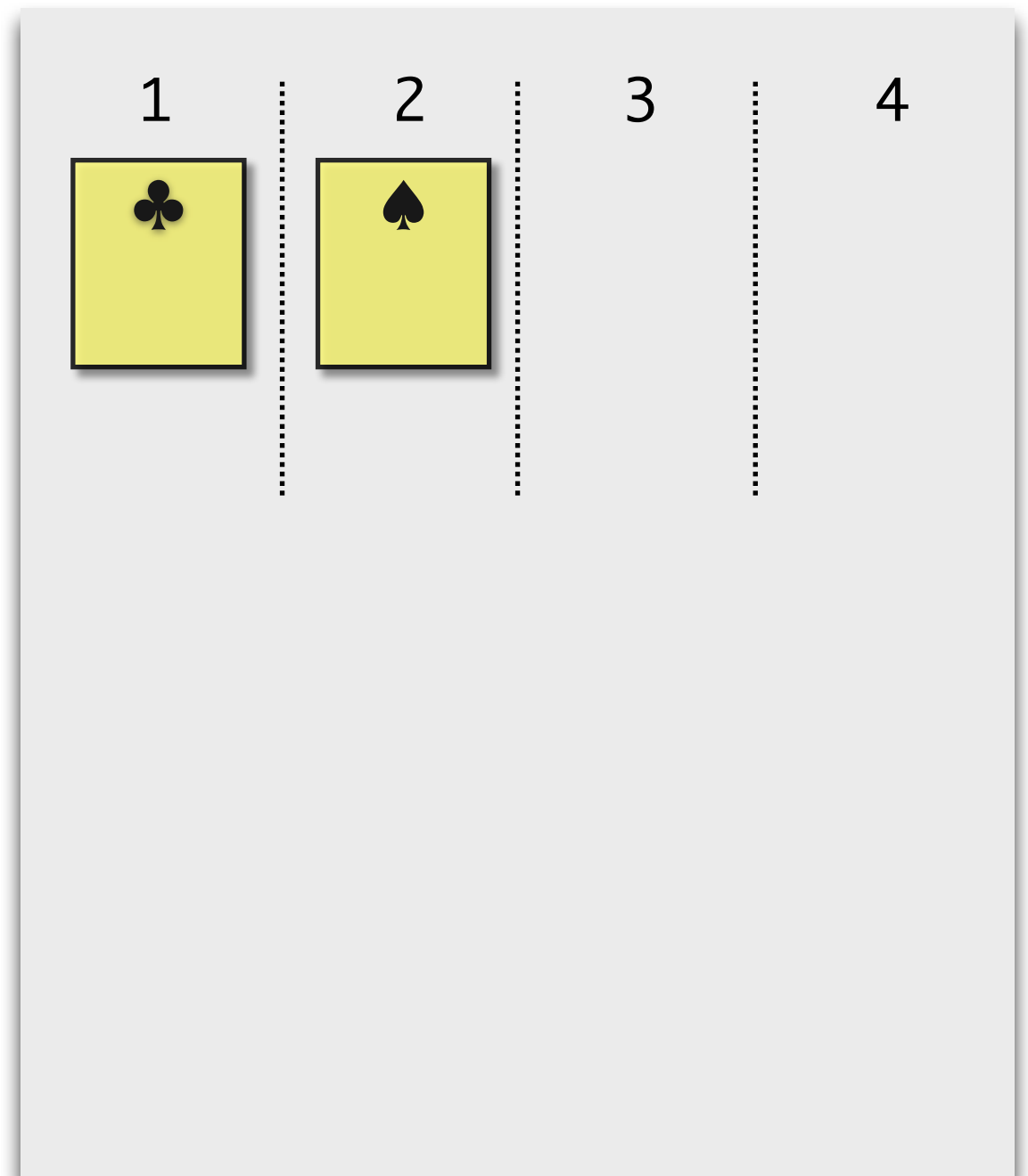


Vers le premier programme

Analyse

✓ *Algorithme*

1. déplacer la carte au
sommet du tas 1 sur le tas 2

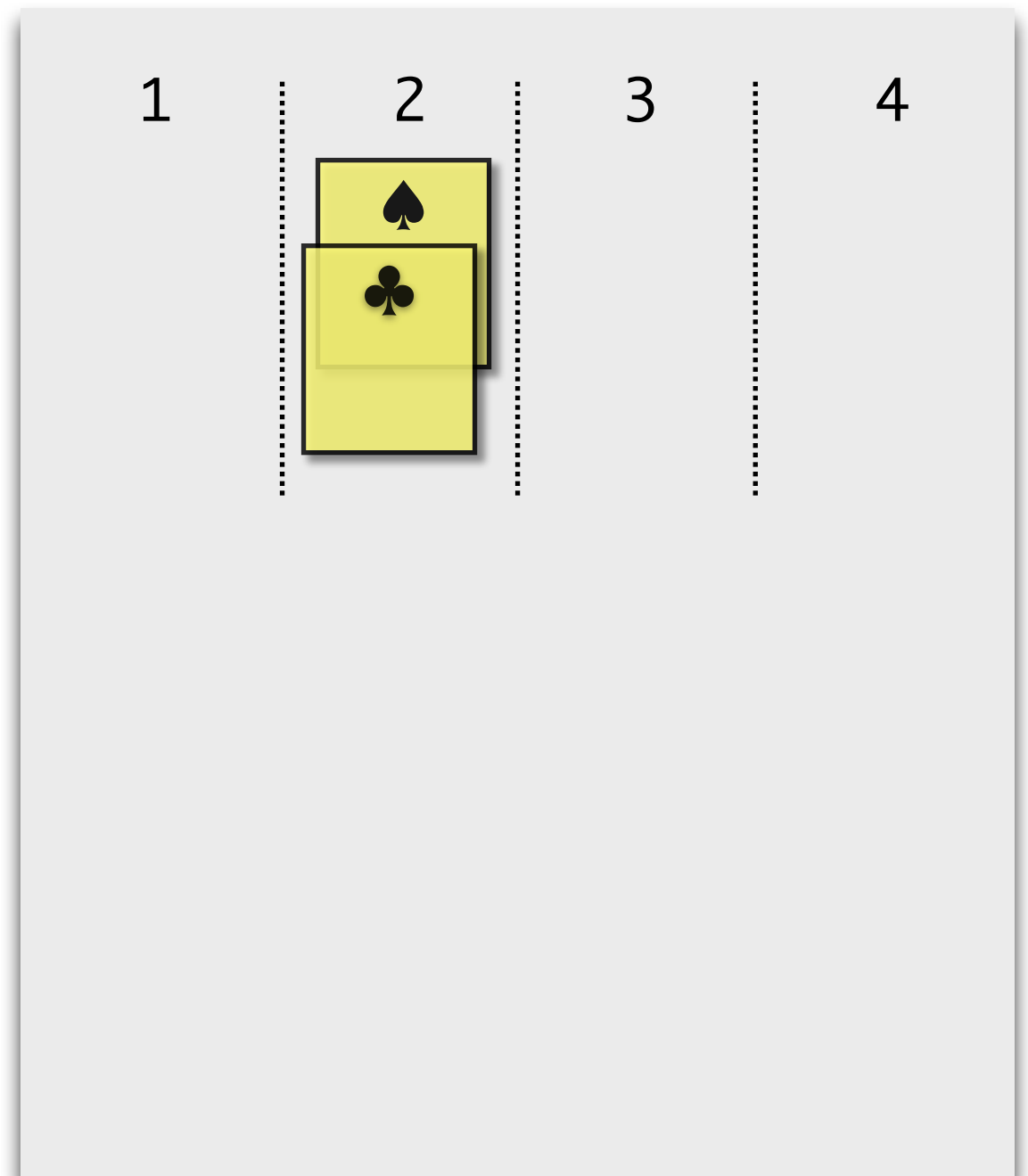


Vers le premier programme

Analyse

✓ *Algorithme*

1. déplacer la carte au sommet du tas 1 sur le tas 2
2. déplacer la carte au sommet du tas 1 sur le tas 2



Vers le premier programme

Codage

Fichier texte : exo1.pas

```
// Auteur  
// Date : 28/09/2005  
// Objet : exo1 manipulation de cartes  
// Etat initial : ...  
// Etat final : ...
```

```
program nom du programme;
```

```
uses
```

```
    unités séparées par des virgules;
```

```
begin
```

```
    instructions
```

```
end.
```

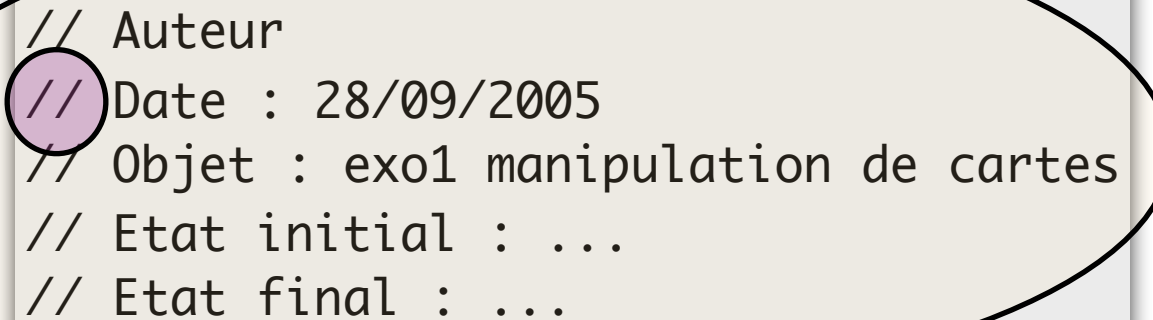
Vers le premier programme

Codage

Commentaire

Entête

Information utile pour la
documentation du
programme



```
// Auteur  
// Date : 28/09/2005  
// Objet : exo1 manipulation de cartes  
// Etat initial : ...  
// Etat final : ...
```

```
program nom du programme;
```

```
uses
```

```
unités séparées par des virgules;
```

```
begin
```

```
instructions
```

```
end.
```

Vers le premier programme

Codage

Exemple: programmes pour les instructions de manipulation des cartes.

Ensemble des programmes annexes à utiliser

```
// Auteur  
// Date : 28/09/2005  
// Objet : exo1 manipulation de cartes  
// Etat initial : ...  
// Etat final : ...
```

```
program nom du program;
```

```
uses
```

unités séparées par des virgules;

```
begin
```

instructions

```
end.
```

Vers le premier programme

Codage

Regroupement d'instructions encadrées par **begin** et **end**.

Bloc d'instructions

```
// Auteur  
// Date : 28/09/2005  
// Objet : exo1 manipulation de cartes  
// Etat initial : ...  
// Etat final : ...
```

```
program nom du programme;
```

```
uses  
    unités séparées par des virgules;
```

```
begin  
    instructions  
end.
```


Vers le premier programme

Représentation d'ensembles de tas

Chaîne de description de tas:

- ✓ T (= ♣), P (= ♥), K (= ♦), C (= ♠)
 - ✓ Si A et B sont des chaînes de descriptions de tas alors
 - ▶ AB (A surmonté de B)
 - ▶ A+B (A ou B)
 - ▶ [A] (A répété un nombre de fois positif *ou nul*)
- sont des chaînes de description de tas

Vers le premier programme

Exemples de chaînes de description

$$\text{TTP} = \{ \begin{array}{c} \text{♣} \\ \text{♣} \\ \text{♠} \end{array} \} \quad \text{TT+P} = \{ \begin{array}{c} \text{♣} \\ \text{♣} \end{array}, \begin{array}{c} \text{♠} \end{array} \}$$

$$[\text{TP}] = \{ \emptyset, \begin{array}{c} \text{♣} \\ \text{♠} \end{array}, \begin{array}{c} \text{♣} \\ \text{♠} \\ \text{♣} \end{array}, \begin{array}{c} \text{♣} \\ \text{♠} \\ \text{♣} \\ \text{♠} \end{array}, \dots \}$$

Vers le premier programme

Exercices:

Trouvez les ensembles de tas correspondant aux chaînes de description suivantes:

$(T+C)P$, $T+CP$, $[K+P]$, $[K] + [P]$, $C[P]$,
 $[T+P+C+K]$, $[[K][P]]$

Vers le premier programme

Exercices:

Parmi les tas suivants, lesquels appartiennent à $[P+C][K]$:
 \emptyset , PK, PCCK, PKKP, KKK, C, CCC

Même question pour $[[P+C][K]]...$

Vers le premier programme

Pour l'initialisation des tas on utilise l'instruction de l'unité cartes:

```
InitTas(n, s);
```

où n est un numéro de tas et s une chaîne de description (' ' représente le tas vide).

Exemples:

```
InitTas(1, 'TT');
```

initialise le premier tas avec deux ♣.

```
InitTas(2, 'T+P');
```

initialise le deuxième tas avec *aléatoirement* un ♣ ou un ♠

Vers le premier programme

Une unique instruction de l'unité cartes pour **modifier la configuration** des tas:

```
DeplacerSommet(n, p);
```

où n est le numéro du tas duquel on prend une carte et p celui du tas sur lequel on la pose.

Exemples:

```
DeplacerSommet(2, 4);
```

déplace la carte au sommet du tas 2 sur le sommet du tas 4.

Vers le premier programme

Une unique instruction de l'unité cartes pour **modifier la configuration** des tas:

```
DeplacerSommet(n, p);
```

où n est le numéro du tas duquel on prend une carte et p celui du tas sur lequel on la pose.

Attention:

```
DeplacerSommet(2,4);
```

Toujours s'assurer que le tas duquel on déplace une carte n'est pas vide. S'il est vide l'instruction `DeplacerSommet` déclenche l'**exception** `Tas_Vide`.

Vers le premier programme

Exercice:

*À l'aide des instructions InitTas
et DeplacerSommet écrire un
programme Pascal implantant
l'algorithme précédent.*

```
// Auteur  
// Date : 28/09/2005  
// Objet : exo1 manipulation de cartes  
// Etat initial : ...  
// Etat final : ...
```

```
program <nom du program>;
```

```
uses
```

```
    <unités séparées par des virgules>;
```

```
begin
```

```
    <instructions d'initialisation des tas>
```

```
    <instructions de déplacement>
```

```
end.
```


Vers le premier programme

Solution

```
// Auteur Cédric Lhoussaine
// Date : 28/09/2005
// Objet : exo1 manipulation de cartes
// Etat initial :
// Tas1='TT' Tas2='' Tas3='' Tas4=''
// Etat final :
// Tas1='' Tas2='TT' Tas3='' Tas4=''

program Exo1;

uses cartes;

begin
    // Initialisation des tas
    InitTas(1, 'TP');
    InitTas(2, '');
    InitTas(3, '');
    InitTas(4, '');

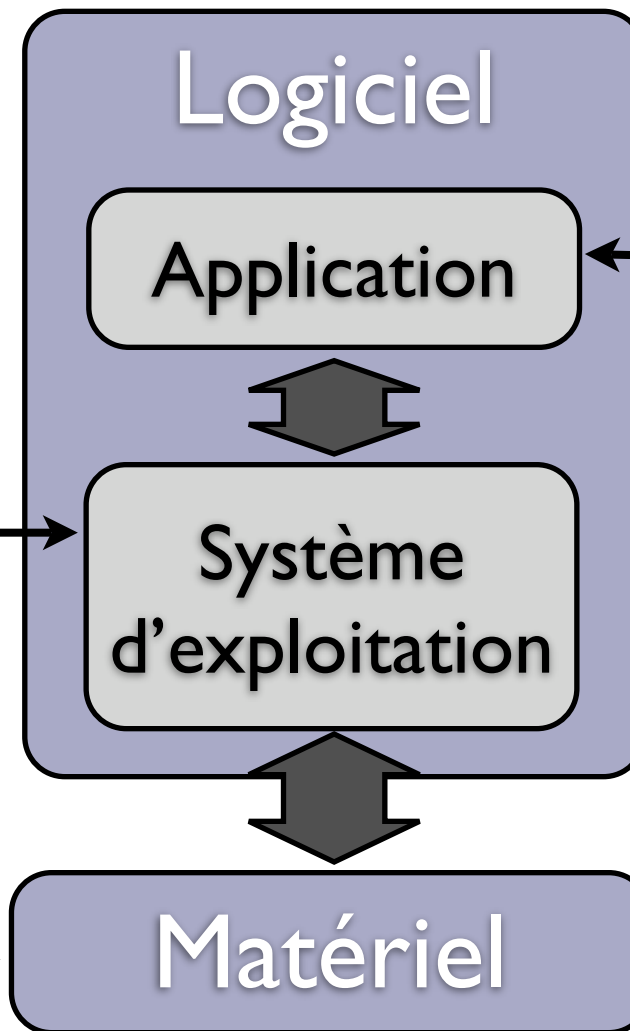
    // Déplacement des cartes
    DeplacerSommet(1,2);
    DeplacerSommet(1,2);
end.
```

Résumé

..... Systèmes informatiques

Interface appli/matériel
Gestion des fichiers
(Répertoires, fichiers, taille)

Application, Programme
Langage de programmation
(niveau, paradigme)
Compilation
(source, exécutable)



Ordinateur
Périphérique
Unités de stockage
(mémoire centrale/
mémoire de masse)

Résumé

Robot manipulateur de cartes

- ✓ Écrire un programme: *Analyse* (Algorithme), *Codage* (Pascal), *Tests* (en TP!)
- ✓ *Structure d'un programme* (entête, blocs, instructions, ...)
- ✓ *Unité cartes* (InitTas, DeplacerSommet)
- ✓ Représentation des tas (*chaînes de description*)
- ✓ Le premier programme Pascal