Initiation à la Programmation

http://www.fil.univ-lille1.fr/Licence123

Portée de constantes et variables, Boucles pour

Portée de variables et constantes

La portée d'une variable (ou constante) est la partie du programme où cette variable peut être utilisée.

La portée peut être

- ✓ globale: la variable peut être utilisée dans tout le programme,
- ✓ locale: la variable ne peut être utilisée que dans la procédure ou la fonction où celle-ci est déclarée.

On parle de variables globales et de variable locales.

Portée de variables et constantes

Déclaration de la variable globale `compteur`

Portée globale

```
// Entête habituelle...

program exemple;

var
   compteur : CARDINAL;

procedure incr;
begin
   compteur := compteur + 1;
end;

begin
   compteur := 0;
   // compteur = 0

   incr;
   // compteur = 1

   writeln(compteur);
end.
```

Portée de variables et constantes

Déclaration de la variable globale `compteur`

Déclaration de la variable locale `aux` et de la constante locale `val`

Portée locale de `val` et `aux`

Portée globale pour `compteur`

```
// Entête habituelle...

program exemple;

var
   compteur : CARDINAL;

procedure incr(const val:CARDINAL);
var
   aux : CARDINAL;
begin
   aux := 2 * val;
   compteur := aux;
end;

begin
   compteur := 0;
   // compteur = 0

   incr(1);
   // compteur = 2

   writeln(compteur);
end.
```

Portée de variables et constantes

Attention: une variable locale peut cacher une variable globale

```
// Entête habituelle...

program exemple;

var
   compteur : CARDINAL;

function incr(const val:CARDINAL):CARDINAL;
var
   compteur : CARDINAL;
begin
   compteur := 2 * val;
   incr := compteur;
end;

begin
   compteur := 0;
   // compteur = 0

   compteur := incr(1);
   // compteur = 2

   writeln(compteur);
end.
```

Portée de variables et constantes

Attention: une variable locale peut cacher une variable globale

Quelle est la valeur de `compteur` à l'issue du programme ?

```
// Entête habituelle...

program exemple;

var
   compteur : CARDINAL;

procedure incr(const val:CARDINAL);
var
   compteur : CARDINAL;
begin
   compteur := val + 1;
end;

begin
   compteur := 0;
   // compteur = 0

   incr(1);
   // compteur = ?

   writeln(compteur);
end.
```

Portée de variables et constantes

Déclaration de procédure:

```
procedure nom(const paramètre: type;...;const paramètre: type);
var
    variables locales;
const
    constantes locales;
begin
    Instructions;
end;
```

Portée de variables et constantes

Déclaration de fonction:

```
function nom(const paramètre: type;
                ...;
             const paramètre: type ) : type retour;
var
    variables locales
const
    constantes locales;
begin
    Instructions;
    nom := expression;
end;
```

Portée de variables et constantes

Toute variable n'apparaissant (et donc n'étant utile) que dans une fonction (ou procédure), doit être déclarée localement.

Boucle Pour

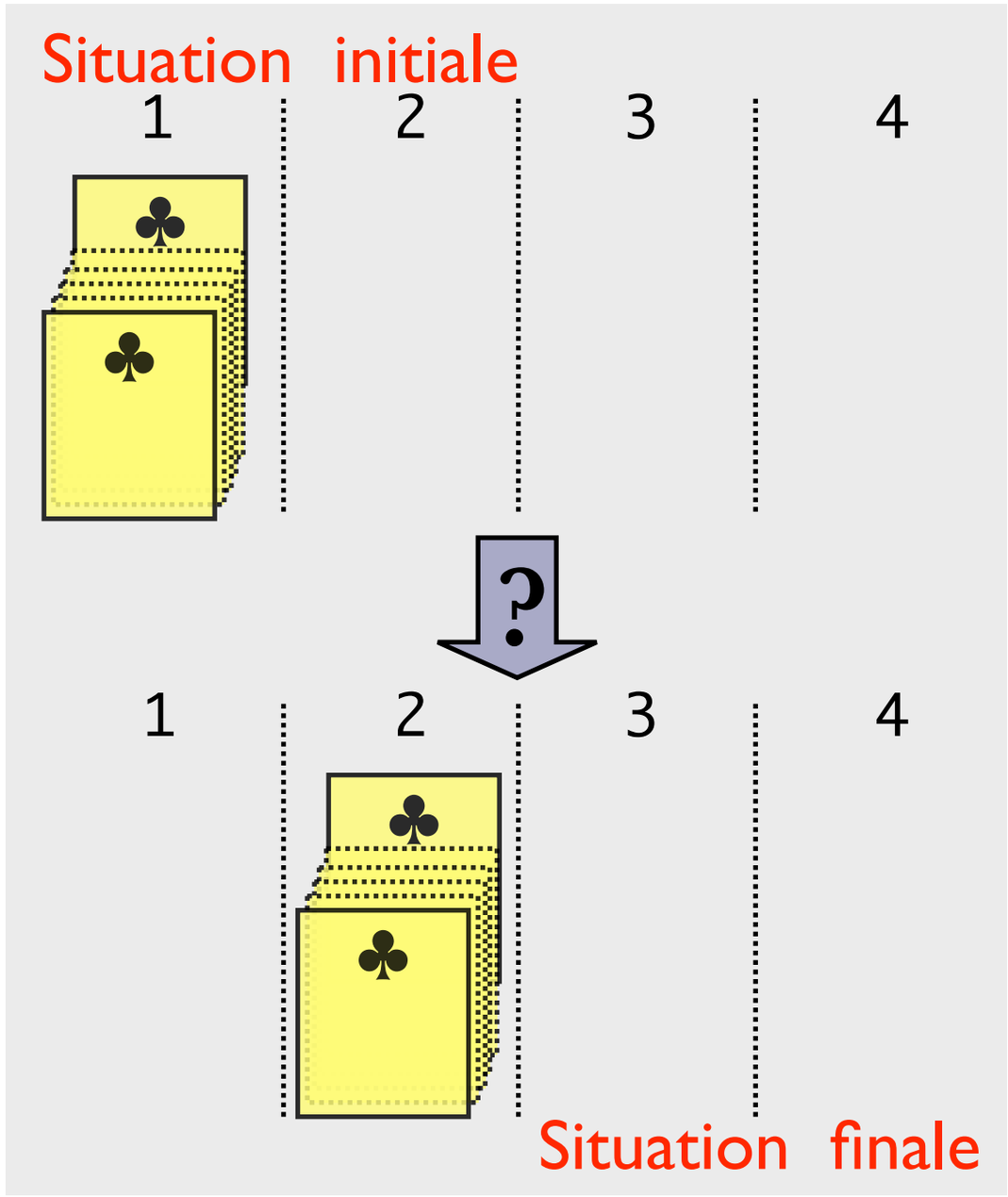
Rappel: on ne peut déplacer qu'une seule carte à la fois

SI: tas 1:T^{100}, autres tas vides
SF: tas 2:T^{100}, autres tas vides

Algorithme:

```
deplacerSommet(1,2);
deplacerSommet(1,2);
...
deplacerSommet(1,2);
```

} 100 x



Boucle Pour

Solution *Tant Que*

```
compteur := 1;
while (compteur <= 100) do
begin
    deplacerSommet(1,2);
    compteur := compteur + 1;
end;
```

Boucle Pour

Solution *Tant Que*

```
compteur := 1;
while (compteur <= 100) do
begin
    deplacerSommet(1,2);
    compteur := compteur + 1;
end;
```

Solution *Pour*

```
for compteur := 1 to 100 do
begin
    deplacerSommet(1,2);
end;
```

Boucle Pour

```
for variable := a to b do
begin
    Instructions;
end;
{ variable = b si b > a }
{ variable = a sinon }
```

variable parcours l'ensemble de valeurs comprises entre *a* et *b*.

Les *Instructions* sont donc exécutées *b*-*a*+1 fois.

Boucle Pour

Somme des 100 premiers entiers

Exercice: réécrire cet algorithme à l'aide d'une boucle Pour

```
S := 0;
i := 1;

{ S = 1 + ... + (i-1) }
while (i <= 100) do
begin
    S := S+i;
    { S = 1 + ... + i }
    i := i + 1;
    { S = 1 + ... + (i-1) }
end;
{ i = 101 }
{ S = 1 + ... + 100 }
```

Boucle Pour

Somme des 100 premiers entiers

```
S := 0;
i := 1;

{ S = 1 + ... + (i-1) }
while (i <= 100) do
begin
    S := S+i;
    { S = 1 + ... + i }
    i := i + 1;
    { S = 1 + ... + (i-1) }
end;
{ i = 101 }
{ S = 1 + ... + 100 }
```

```
S := 0;
for i := 1 to 100 do
begin
    { S = 1 + ... + (i-1) }
    S := S+i;
    { S = 1 + ... + i }
end;
{ i = 100 }
{ S = 1 + ... + 100 }
```

Boucle Pour

Boucle Pour à indice décroissant

```
for variable := a downto b
do
begin
    Instructions;
end;
{ variable = b si b < a }
{ variable = a sinon }
```

Boucle Pour

Boucle Pour à indice décroissant

```
for variable := a downto b do
begin
    Instructions;
end;
{ variable = b si b < a}
{ variable = a sinon }
```

(presque) Équivalent à :

```
variable := a;
while variable >= b do
begin
    Instructions;
    variable := variable - 1;
end;
{ variable = b - 1 si b < a }
{ variable = a sinon }
```

Bouble Pour

Piège: modification d'un indice de boucle Pour

```
S := 0;
for i := 1 to 100 do
begin
    S := S+i;
    i := i+1;
end;
```

Erreur de compilation:

```
Error: Illegal assignment to for-loop variable "i"
```

Bouble Pour

Piège: modifier *cachée* d'un indice de boucle Pour

```
S := 0;
for i := 1 to 100 do
begin
    S := S+i;
    p;
end;
```

avec:

```
procedure p;
begin
    i := i+1;
end;
```

Pas d'erreur de compilation mais le résultat est:

```
S = 1 + 3 + 5 + ... + 99
```

Conclusion: n'utiliser comme indice de boucle une variable qui ne joue aucun rôle ailleurs et définie le plus localement possible.