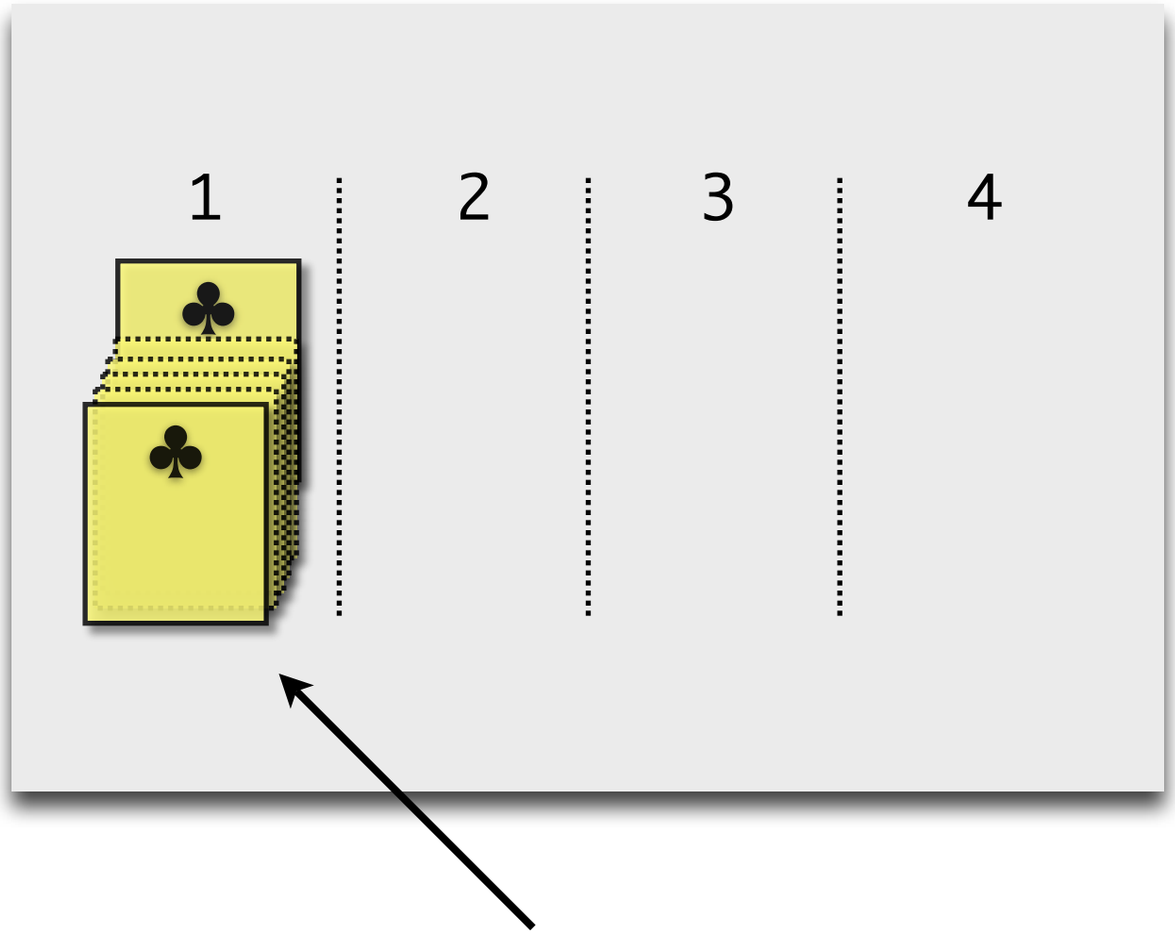
Initiation à la Programmation

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

Types et Variables

Motivation



Pour compter : variables et types données sont nécessaires

Types

Un type détermine le *sens* d'une donnée, d'un opérateur, d'une expression

L'expression

`PIQUE / CARREAU`

(*i.e.* diviser un pique par un carreau) n'a aucun sens! La nature (ou type) des objets `PIQUE` et `CARREAU` et le type de l'opérateur '`/`' déterminent la manière correcte de les utiliser.

Types

Les données manipulées par un programme
sont de différentes natures ou types

- ✓ Couleurs de cartes

  `PIQUE, TREFLE, COEUR, CARREAU` ∈ `COULEURS`

- ✓ Booléens

  `true, false` ∈ `BOOLEAN`

- ✓ Chaînes de caractères

  `'TP', 'T+P', '[P]', 'Bonjour',...` ∈ `STRING`

Types

Les données manipulées par un programme
sont de différentes natures ou types

- ✓ Ensemble de tas

  `1, 2, 3, 4` ∈ TASPOSSIBLES

- ✓ Entiers naturels

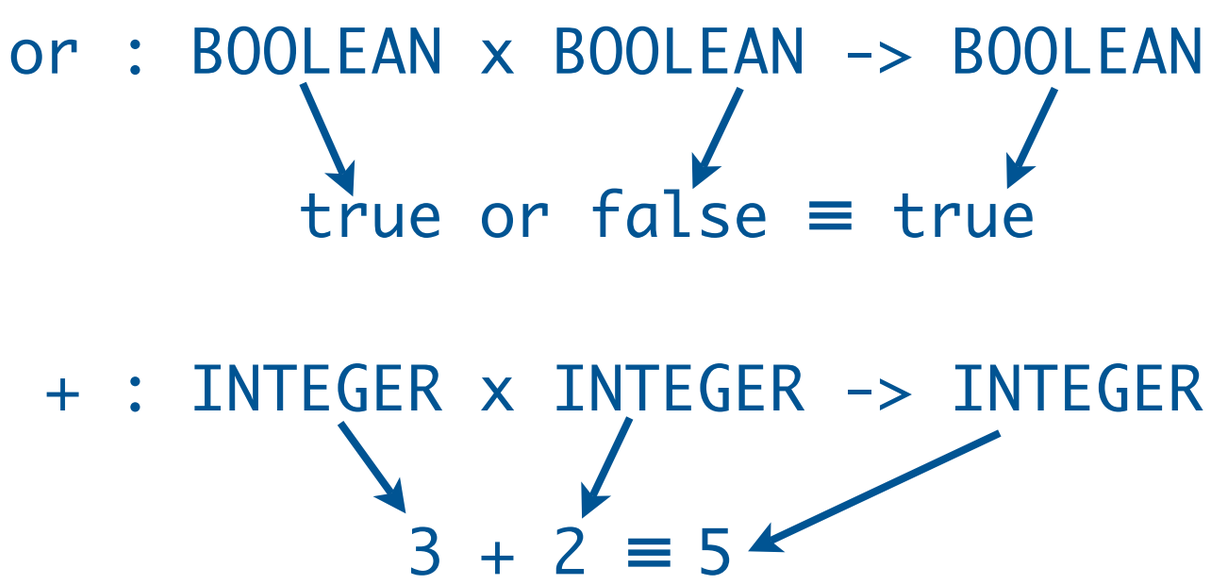
  `0, 1, 2, ....` ∈ CARDINAL

- ✓ Entiers relatifs

  `..., -2, -1, 0, 1, 2, ....` ∈ INTEGER

Types

Les opérateurs sont également typés.
Le type d'un opérateur détermine:
✓ les données auxquelles l'opérateur peut être appliqué
✓ le type du résultat de l'opérateur



Types

Exercice:
Donnez les types des opérateurs `and`, `not`, `-`, `<`, `TasNonVide`

Types

Certains opérateurs peuvent être polymorphes, i.e. ils admettent plusieurs types.

```
= : ∀T. (T x T -> BOOLEAN)
```

Donc = prend pour arguments deux objets de même type T (n'importe lequel) et a pour résultat un booléen.

Types

Les données et les opérateurs sur les données étant typés, les expressions sont donc également typées.

Le type d'une expression est le type du résultat de cette expression.

```
true or false : BOOLEAN
      car or : BOOLEAN x BOOLEAN -> BOOLEAN
```

Types

Une expression est bien typée (et donc valide) si les types des opérateurs sont respectés.

`((1 + 2) = 4) or TasNonVide(1) : BOOLEAN` est bien typé car:

```
1, 2 : INTEGER
+ : INTEGER x INTEGER -> INTEGER
```
} `1+2 : INTEGER`

```
4, 1+2 : INTEGER
= : INTEGER x INTEGER -> BOOLEAN
```
} `(1+2) = 4 : BOOLEAN`

```
1 : INTEGER
TasNonVide : INTEGER -> BOOLEAN
```
} `TasNonVide(1) : BOOLEAN`

```
(1+2)=4, TasNonVide(1) : BOOLEAN
or : BOOLEAN x BOOLEAN -> BOOLEAN
```
} OK

Types

Exercice:
Donnez le type des expressions suivantes et vérifiez si elles sont bien typées

```
-1 - 4

(true = (3 * 4)) or TasNonVide(1)

couleurSommet(1) and couleurSommet(2) = TREFLE
```

Variables

Servent à nommer des objets d'un type déterminé.

Caractérisées par un identificateur (nom) et un type.

Variables

Déclaration de variable

```
// Auteur
// Date : 11/10/2005
// Objet : exo1 manipulation de cartes
// Etat initial : ...
// Etat final : ...

program nom du program;

uses
    unités séparées par des virgules;

var
    identificateur : type;

procedure nom de procedure;
begin
    instructions;
end;

begin
    instructions d'initialisation des tas;
    instructions de déplacement;
end.
```

Variables

Exemple

```
// Auteur
// Date : 11/10/2005
// Objet : exo1 manipulation de cartes
// Etat initial : ...
// Etat final : ...

program exemple;

uses
   cartes;

var
   compteur : CARDINAL;

begin
    ...
end.
```

Variables

Une fois la variable déclarée on peut lui associer (affecter) un objet (i.e. une valeur)

Affectation:

identificateur := *expression*;

✓ La valeur de *expression* est associée à la variable *identificateur*
✓ L'*expression* doit être de même type que la variable *identificateur*

Variables

Attention: ne pas confondre l'affectation (:=) et la comparaison (=)

Affectation:

identificateur := *expression*;

Variables

Exemples

```
var compteur : CARDINAL;
```

✓ `{ compteur = ?? }`

```
   compteur := 2;
```

`{ compteur = 2 }`

✓ `{ compteur = 5 }`

```
   compteur := compteur * 2;
```

`{ compteur = 10 }`

✓ `compteur := true;`

Erreur: la variable `compteur` est déclaré `CARDINAL` mais est affecté à une variable de type `BOOLEAN`!

Afficher des données

Deux instructions pour afficher des données à l'écran:
`write` et `writeln`.
Prennent en argument une chaîne de caractères (`string`) ou une variable.

✓
```
write('nombre de cartes: ');
write(compteur);
```
affiche :
```
nombre de cartes: 2
```

✓
```
writeln('nombre de cartes');
writeln(compteur);
```
affiche :
```
nombre de cartes:
2
```

Afficher des données

Attention:

Pour afficher une apostrophe il faut la doubler:

```
write('L''UE InitProg est passionnante!');
```

Tous les types de données ne sont pas affichables:

```
var x:COULEUR;
...
x := PIQUE;
write(x);
Error: Can't read or write variable of this type
```

Compter les cartes

```
// Auteur EW
// Date : 11/10/2005
// Objet : compter cartes du tas 1

program compter;

uses
   cartes;

var
   compteur : CARDINAL;

begin
   // Initialisation des tas
   InitTas(1,'[T]');

   // Initialisation du compteur
   compteur := 0;
   // N : nb initial de cartes sur tas 1
   // n1 : nb actuel de cartes sur tas 1
   // on a : N = n1 + compteur
   // action : vider tas 1 en comptant
   // les cartes

   while tasNonVide(1) do
   begin
      // n1 + compteur = N
      DeplacerSommet(1,2);
      // n1 + compteur = N-1
      compteur := compteur + 1;
      // n1 + compteur = N
   end; { while }

   // n1 = 0, n1 + compteur = N
   // donc compteur = N

   writeln('nombre de cartes: ',
            compteur);
end.
```