L’information représentée par des termes – Enseignant

Ce chapitre se concentre sur la représentation de l’information. Dans une première étape de résolution de problème informatique, il est nécessaire de définir les informations pertinentes et celle qui ne le sont pas.

## Buts :

* Comprendre l’énumération
  + - Définition de termes
    - Abstraction des éléments réelles
    - Énumération finie (pour la taille d’une personne)
    - Énumération infinie (pour les couleurs)
* Comprendre la composition
  + - Énumération structurée et foncteurs
    - Surcharge d’un foncteur
    - Structure dynamique
    - Méthodologie pour construire une structure récursive
    - Structure de liste
    - Structure d’arbre binaire et leur représentation codée

## Définitions importantes

Termes

Foncteurs

Cas de base, cas récursif

Liste

Arbre (facultatif selon le niveau de la classe

## Description des activités

**Activité 1 :**

Description : Dans cette activité, il est demandé d’énumérer les ingrédients d’une pizza. Puis d’énumérer les manipulations des ingrédients.

But :Le but est d’illustrer la simplicité de l’énumération et de voir qu’une fois énumérés, on peut manipuler ces objets pour les transformer en pizza. C’est une illustration simple permet informellement d’observer des structures qui existent dans la vie courante et leur organisation.

Pratique : Nous conseillons que cette activité soit abordée seule ou en petit groupe, puis discutée avec l’enseignant.

**Activité 2 :**

Description : Énumération des branches du collège et du nombre d’heure.

But : Cette activité met en valeur la première étape de résolution d’un problème souvent négligée : la conception. En effet, c’est une dimension importante de savoir quelle est l’information que nous allons manipuler dans un problème.

Pratique : Nous conseillons que cette activité soit abordée en classe complète.

**Activité 3 :**

Description : Énumérer les couleurs

BUT : Cette activité a pour but de montrer que la discrétisation d’ensemble continu n’est pas toujours possible. En effet, il n’est pas toujours possible d’énumérer toutes les couleurs et que chaque élève a son propre type de représentation. Par exemple, certains utiliserons les couleurs de l’arc-en-ciel, mais comment représenter le brun, noir et blanc ?

Il est donc essentiel de transmettre que l’énumération dépend donc du problème que nous cherchons à résoudre.

Pratique : Nous conseillons de faire ce travail individuellement, puis de réunir la classe pour une discussion.

**Activité 4 :**

Matériel : Carte blanche.

Description : Écrire le nom des foncteurs et cas de base sur chaque carte et les emboiter pour composer des listes (par exemple).

BUT : Cette activité est un exercice d’informatique débranché permettant de comprendre la structure récursive.

Pratique : Nous conseillons de réaliser cette activité en petit groupe.

L’information représentée par des termes - élèves

L’informatique a pour but de représenter de multiples informations et de les traiter. Plusieurs approches sont possibles. Ce cours se concentre sur l’**énumération** et la **composition** à l’aide de termes.

**Activité 1 (en classe) :** Énumérer les ingrédients pour cuisiner une pizza ?

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………

Lorsque les ingrédients sont sélectionnés, il est possible de les manipuler et de les transformer pour qu’ils deviennent une pizza.

Comment constituer un cahier de recettes de pizza ou le menu de notre pizzeria ?

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………

……………………………………………………



# 1 Énumération de données

**Définition :** Les *Termes* sont une représentation abstraite d’objets (concrets ou abstraits). Par exemple, le terme « chat » en français désigne un certain type de mammifère.

Les langues naturelles, comme le français, répartissent les mots dans différentes catégories grammaticales : noms, verbes, adjectifs, déterminants, adverbes et prépositions. Ces dernières définissent certaines possibilités et contraintes pour la combinaison des mots entre eux. Nous pouvons donc définir les termes suivants représentant les différentes catégories : Nom, verbes, déterminant …

**L’énumération** est la première étape de la formalisation d’un problème. Pour réaliser cette étape, il faut réfléchir sur la nature des informations que l’on veut transmettre à l’ordinateur et les lister.

**Activité 2 :** Un collège (enseignant 10 branches) souhaite automatiser la construction de grilles horaires. Énumérer les branches dispensées dans le collège et le nombre d’heures :

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

Le collège contient 8 enseignants, énumérer les enseignants et les matières qu’ils enseignent :

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

…………………………………………………… ……………………………………………………

Pour faire les horaires de quoi d’autre avons-nous besoin ?



……………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………

…………………………………………………………………………………..………………………………………………………

Dans certains cas, il n’est pas possible d’énumérer des données.

Pourquoi ne pouvons-nous pas énumérer les nombres entiers ?

……………………………………………………………………

Comment énumérer la taille possible d’une personne ?

……………………………………………………………………

Les deux problèmes énoncés plus haut sont des problèmes de représentations des données bien connus en informatique. Pour résoudre un problème, un ordinateur a besoin de recevoir un nombre fini de valeurs. C’est pourquoi il est nécessaire de choisir des valeurs suffisamment représentatives permettant de coder les informations nécessaires avec une marge d’erreur acceptable.

Dans le cas des nombres entiers, nous choisirons donc un intervalle significatif correspondant au problème.

Pour représenter la taille, nous admettons donc qu’il peut y avoir une marge d’erreur qui est fixée en fonction du problème (par exemple deux chiffres après la virgule). Cette dernière technique est la **discrétisation**.

# 2 Énumération structurée de données

Pour construire des données complexes, les termes sont regroupés sous un « super-terme » que l’on appelle *foncteur*. Reprenons l’exemple de la grammaire française : les termes ( « verbes », « nom », « déterminent »,… ) sont assemblés (sans parler de l’ordre) pour former une structure plus complexe  qualifiée par le foncteur : phrase.

phrase sujet verbe complément

**Déf :** Un *foncteur* est un terme qui qualifie l’assemblage complexe de plusieurs termes. On notera donc la phrase ci-dessus par phrase(sujet, verbe, complément).

**Déf** : Le nombre de termes que regroupe un foncteur s’appelle l’*arité*.

Quelle est l’arité du foncteur « phrase » décrit plus haut ? ………………………………………………………

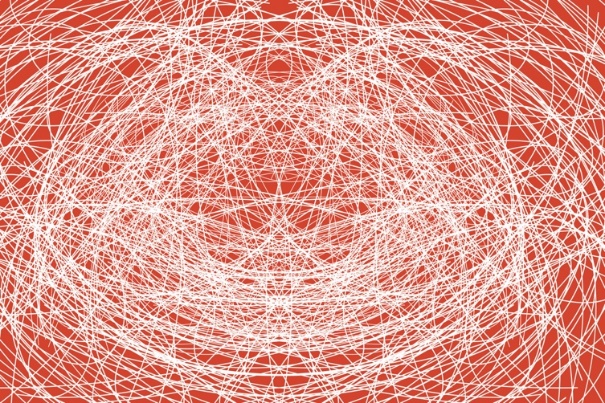
De plus, il est possible de manipuler plusieurs foncteurs de même nom, mais d’arité différente. Nous parlons alors de *surcharge* ou *overloading,* car un foncteur est « surchargé » avec plusieurs définitions. Ce qui crée une ambiguïté dans la définition du foncteur.

**Activité 3 :** Comment surcharger le foncteur « phrase »

Arité 1 : …………………………………………………………………

Arité 2 : …………………………………………………………………

Arité 3 : …………………………………………………………………



# 3 Structure de données dynamique

Il existe des données dont il n’est pas possible de définir la taille en avance. Par exemple, si nous voulons représenter une file d’attente s’étant formée devant un marchand de glace, il n’est pas possible de déterminer à l’avance la taille maximale. De plus, la taille va changer au cours du temps en fonction des arrivées et/ou des départs de personnes. Il est donc difficile, voire impossible de définir l’arité du foncteur représentant cette fin de la liste. Comment peut-on représenter ce type de structures ?

**Déf :** Une *structure dynamique* est une structure qui va évoluer au cours du temps. Comme la fin de la liste devant le marchand de glace.

## 3.1 Structure de données récursives

Une des manières de représenter des données de tailles variables est d’utiliser une *structure récursive*.

**Déf** : Pour définir une structure récursive, est définie par deux constructions :

* *Cas de base*: il est défini une situation initiale (souvent une simple constante).
* *Cas récursif*: il est défini la différence entre deux situations, c’est donc un   
  terme ayant une arité au moins égale à 1.

**Méthodologie** : Pour trouver une structure récursive, il est   
nécessaire d’effectuer les différentes étapes suivantes :

1. Identifier les différents états possibles
2. Définir un cas de base
3. Définir les transitions possibles vers un autre état
4. Définition d’un foncteur de transition

**Exemple** : Reprenons la fin de la liste devant notre marchand de glace.

1. Quels sont les états possibles : {0 client, 1 client, 2 clients…}
2. Quel est le cas de base ? Par exemple 0 client, ce qui correspond à l’état initial de notre modèle.
3. Quand observons-nous une transition par exemple pour passer de 1 à 2 clients (ou de 2 à 1 client)? Quand un client arrive (ou un client part).
4. Quel foncteur peut correspondre à cette transition ? Par exemple ClientAprès().

On construit donc une structure telle que ClientApres() qui représente l’encodage de la transition entre deux situations. L’arité de ce foncteur est de 1 et il sélectionne un objet de type « aucun » ou « clientAprès ».

On obtient donc pour une représentation de deux clients faisant la fin de la liste :

ClientAprès ClientAprès aucun



Ce type de constructions nous permet donc de représenter n’importe quels nombres de clients.

**ATTENTION : ClientAprès() ne représente pas directement les clients présents dans la file. Il n’est pas possible d’identifier quelqu’un avec ce type de structure. L’inclusion ClientAprès() peut à un moment représenté Alice qui arrive puis quelques secondes plus tard Bob qui part.**

**Remarque** : Comme nous pouvons le constater dans le schéma ci-dessus, l’arité ne change pas. Elle est toujours de 1. C’est ce que nous appelons une structure récursive.

Pour distinguer un terme récursif d’un terme qui ne l’est pas, nous utilisons une notation graphique différente. Les encadrés aux bords arrondis représentent des foncteurs non récursifs et les encadrés hexagonaux représentent les foncteurs récursifs.

Ce qui donne les constructions suivantes :

aucun et clientApres fin de la liste

Dans l’exemple ci-dessus, la dénomination « fin de la liste » ne représente ni une constante ni un terme composé. C’est un « trou » qu’il faut combler avec un terme construit avec les deux constructions possibles. Le « trou » peut donc être remplacé par n’importe quel terme.

Nous pouvons donc déduire de cette définition que nous finirons par utiliser la constante aucun, afin que la formule ne contienne plus de « trou ». C’est pourquoi nous nommons ce terme cas de base.

Il est donc très important de le définir sinon la structure est infinie.

**Activité :** Matériel cartes blanches.

- Écrire le nom d’un foncteur sur les cartes et dessiner un rectangle permettant de poser la carte suivante.

- Écrire le nom du cas de base sans emplacement pour poser la carte suivante.

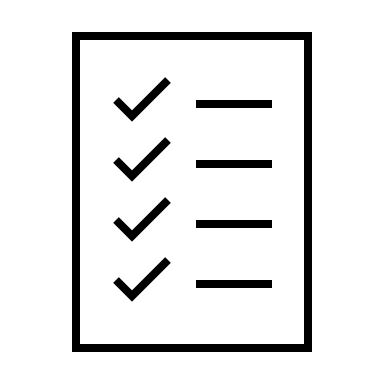
# 3.2 Listes

Comme nous l’avons vu précédemment la structure récursive nous permet de compter le nombre de clients en attente, il ne nous permet pas de les identifier. Comment inclure des informations supplémentaires pour que notre structure devienne une liste de personnes ?

Nous pouvons représenter une liste à l’aide de deux constructions :

vide et ajoute élément fin de la liste

Dans cette situation, le cas de base est la liste vide représentée par le terme vide.

Alors que le cas récursif est représenté par ajoute($element, $finDeLaListe) ajoute un élément contenant de l’information à la tête d’une autre liste (par exemple un prénom). De ce fait nous obtenons une construction comme suit :

Ajoute Leia ajoute Luke vide

Il est donc possible de construire n’importe quelle liste.

**Déf** : La *racine* est le premier élément de notre liste. C’est-à-dire la première personne dans la file d’attente. Il se compose d’une construction ajoute($element,$fin de la liste) de la constante Leia représentant une personne nommée « Leia » et une construction de construction identique.

Le deuxième terme est construit de la même manière en plaçant une personne prénommée Luck en deuxième position de la file. Le dernier terme est un terme vide qui représente la fin de liste.

## 3.3 Arbres binaires

Une structure de donnée importante en informatique est *l’arborescence* ou la structure d’*arbre*.

**Déf**: Un *arbre* représente les informations par des *nœuds* qui sont organisés de manière hiérarchique formées par des liens que l’on appelle des *branches* qui connectent deux nœuds entre eux. Nous avons donc une relation de *parent-enfant*, les nœuds plus généraux sont les parents et les plus spécifiques sont les enfants. Les nœuds terminaux sont des *feuilles* et le nœud le plus général est la racine.

**Remarque**: Lorsqu’on représente un arbre, nous le représentons à l’envers la racine orientée vers le haut.

Vêtement

Robes Pantalon

Robes Soleil Robes T-shirt Jeans Khakis

Dans l’exemple ci-dessus, la racine intitulée « vêtement » est le cas le plus général. Elle inclut n’importe quel type de vêtement. Elle est donc le parent de la catégorie « Robes » et « Pantalon » qui sont eux-mêmes parentes de catégorie plus spécialisée. Le « Jeans » est un nœud terminal donc une feuille.

**Remarque** : En informatique nous distinguons la structure arborescente de la structure de graphe, car :



* Le nœud d’un arbre ne peut avoir qu’un seul parent.
* Il ne peut pas y avoir de cycles. On ne peut donc pas être son propre descendant.
* Un arbre a une seule racine

**Activité :** Faire son propre arbre généalogique pour 2 générations.



Est-ce que ce type de structure correspond à un arbre informatique ? ………………………

Pourquoi ? …………………………………………………………………………………………………………………….

Un arbre peut être défini par les deux constructions :

Vide et nœud élément gauche élément droite

Il y a donc une construction vide qui représente l’arbre vide qui est notre cas de base. La seconde construction nœud($element, $gauche,$droite) représente le nœud parent d’un deux autres arbres. Cette construction est notre cas récursif. L’arbre illustré plus haut peut être représenté comme suit :

Nœud( vetements,

Nœud (robes,

Nœud(robes\_soleil, vide, vide),

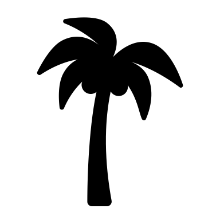
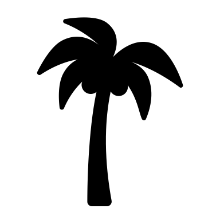
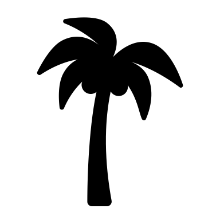
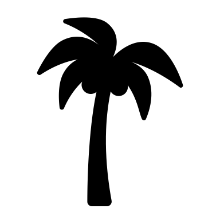
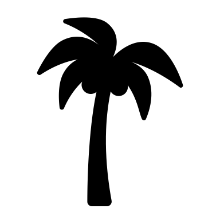
Nœud(robe\_t\_shirt, vide, vide)),

Noed(pantalons,

Nœud(jeans, vide,vide),

Nœud(khakis, vide,vide)))

Comme nous pouvons le remarquer chaque nœud prend un identifiant puis un sous-arbre. Dans le cas de nœud feuille ils prennent un arbre de type vide à gauche et à droite.



Exercices

## Énumération de données

**Exercice 1 :** Énumère l’ensemble des couleurs ?

**Exercice 2 :** Énumère l’ensemble des mammifères. Définir des termes pour chaque catégorie ?

**Exercice 3** : Comment représenter le poids d’une personne ?

**Exercice 4** : Quels sont les termes permettant de définir un mouvement d’un pion sur un quadrillage ?

**Exercice 5** : Quels sont les termes permettant de définir les différentes actions du jeu du moulin ?

**Exercice 6** : Quels sont les termes permettant de définir des races de chien différentes ?

**Exercice 7** : Énumère les fruits nécessaires pour faire une salade de fruits ?

## Énumération structurée de données

**Exercice 8**: Construire les nombres entiers à l’aide d’une structure de donnée

**Exercice 9:** Construire une suite de nombres pairs

**Exercice 10**: Construire la suite de Fibonacci par récurrence

**Exercice 11**: Construire les multiples de 3 par récurrence

**Exercice 12**: Énoncer et construire une structure décrivant les propriétés des triangles rectangles, longueur des côtés et angles.

**Exercice 13**: Construire une structure permettant de construire récursivement des polygones réguliers à n côtés à l’aide de triangles.

**Exercice 14:** Encodez à l’aide d’un terme la situation où quatre clients attendent dans la file.

## Liste

**Exercice 15**: Définir une liste contenant des canidés

**Exercice 16**: Définir une liste de mouvement que doit effectuer une personne au point A pour se rendre au point B.

**Exercice 17:** Définir une liste contenant trois personnes respectivement nommées Pierre, Ondine et Sacha.

**Exercice 18:** Définissez une liste non vide dont chaque élément représente un couple.

## Arbre

**Exercice 19**: Construire un arbre en ordonnant les termes {girafe, crocodile, éléphant, mammifère, reptile, lézard, triton, batracien, grenouille, invertébré, marsupiale, méduse, kangourou}.

La représentation obtenue est-elle un arbre ? Pourquoi ?

Peut-on créer une structure de donnée ?

**Exercice 20**: Les parties du corps humain de manière arborescente en prenant comme racine le « corps humain ».

**Exercice 21**: Les nombres binaires à 4 chiffres.

**Exercices 22**: « beau et mauvais » le temps possible des 4 prochains jours.