Formation ISN - A.4

Emmanuel Beffara, Edouard THIEL et Michel Van Caneghem

Université d'Aix-Marseille (AMU)

Janvier 2012

Les transparents de ce cours sont téléchargeables ici :
http://pageperso.lif.univ-mrs.fr/~edouard.thiel/ens/ISN/

Programme de la formation pour ISN

A. **Information numérique**
   - A.1 Codage numérique (3h)
   - A.2 Représentation numérique des données analogiques (3h)
   - A.3 Théorie de l'information (3h)
   - A.4 Information structurée 1 (3h)
   - A.5 Information structurée 2 (1,5h)
   - A.6 Contrôle de l'information (3h)
   - A.7 Informatique et société (3h)

B. **Langages et programmation**

C. **Algorithmique**

D. **Machines numériques**

E. **Projet**

A Information numérique
A.4 Information structurée

1. Notion de type
2. Structures de données de base
3. Structures de taille variable
4. Formats de fichier

1 - Notion de type

On a vu au cours A.1 :

- que l'on peut représenter des informations par des mots binaires (constitués d'un ou plusieurs mots machines, ou d'un nombre quelconque de bits).
- la nécessité de connaître la représentation et le codage d'un mot binaire pour l'utiliser.

Cette nécessité est vraie partout :

- pour les données dans la mémoire de l'ordinateur ;
- pour les variables employées dans les calculs ;
- pour les données enregistrées dans un fichier.

Types simples et composés

On appelle type la représentation et le codage d'un mot binaire.

On distingue les types simples et les types composés :

- Les types simples correspondent aux données qui peuvent être traitées directement par le processeur.
- Les types composés sont plus élaborés, et sont créés à partir des types simples.

Types simples

Les types simples dépendent du langage de programmation utilisé (C, Pascal, Java, Python, etc) et ont des représentations et codages qui peuvent varier selon la machine :

- Les booléens (faux, vrai) ;
- les entiers (de différentes tailles) ;
- les flottants (idem) ;
- les caractères ;
- les pointeurs (pour manipuler des adresses en mémoire).

Types simples — exemples

Les types simples dépendent du langage et leur taille peut dépendre de la machine :

Type simple	**C**		**Pascal**		**Java**		**Python**	
entier court	`short`	16			`short`	16		
entier	`int`	m	`integer`	16	`int`	32	`int`	m
entier long	`long`	32			`long`	64		
entier double	`long long`	64						
grand entier	(lib GMP)				`BigInteger`		`long`	
flottant simple	`float`	32	`real`	32	`float`	32		
flottant double	`double`	64			`double`	64	`float`	64
booléen	`int`	m	`boolean`		`boolean`	8	`bool`	
charactère	`char`	8	`char`	8	`char`	16	`str`	

(taille en bits ; 'm' = taille d'un mot machine)

Types composés

On distingue différents groupes de types composés :

- les types fournis par un langage (par exemple les chaînes de caractères) ;
- les types élaborés que l'on peut construire avec des primitives du langage (tableaux, enregistrements, etc) ;
- les types non directement fournis, mais qui sont fournis à côte du langage (graphes, arbres, etc) dans des bibliothèques.

Opérateurs

Les types simples et certains types composés sont associés à des opérateurs ; ils dépendent du langage :

Opérateurs	**Types**	**C**	**Pascal**	**Java**	**Python**
`+ - *`	ent., flottants	`+ - *`	`+ - *`	`+ - *`	`+ - *`
div réelle	flottants	`/`	`/`	`/`	`/`
div entière	entiers	`/ %`	`div mod`	`/ %`	`// %`
puissance	ent., flottants	`pow()`		`Math.pow()`	`**`
concat.	chaînes	`strcat()`	`+`	`+`	`+`
binaires	entiers	`~ & |`		`~ & |`	`~ & |`
logiques	booléens	`! && ||`	`not and or`	`! && ||`	`not and or`
égalités	simples	`== !=`	`= <>`	`== !=`	`== !=`
inégalités	simples	`< <=`	`< <=`	`< <=`	`< <=`

Il y a de nombreux autres opérateurs et langages !

2 - Structures de données de base

Une structure de donnée est une façon d'organiser des données afin de rendre possible leur mémorisation et leur traitement.

On associe un type à une structure de donnée.

On vient de voir les types simples. Les langages permettent de construire de nouveaux types (composés) à partir des types simples, et donc d'implémenter des structures de données.

Vocabulaire : on peut aussi appeler une structure de données un type abstrait, et le type associé un type concret.

Exemples de structures de données

- Les constantes ou les variables (entières, flottantes, booléennes, caractères, pointeur, etc).
- Les chaînes de caractères.
- Les tableaux à une ou plusieurs dimensions (vecteurs, matrices, etc) indicés par des entiers ; de taille fixe ou variable.
- Les tableaux associatifs, indicés par des clés (des chaînes de caractères).
- Les enregistrements (groupes de structures de données).
- Structures récursives (listes, arbres, graphes, ...).
- Bases de données.
- Format de fichier.

Il existe de nombreuses structures bien connues (parfois très complexes : description des n-faces d'un polytope dans $\mathbb{R}^n$, ...) ; on peut en inventer à l'infini.

Importance des structures de données

On choisit une structure de données pour modéliser un problème, en fonction de différents critères, parfois contradictoires :

- Minimisation du coût en mémoire.
- Efficacité en accès, insertion ou modification.
- Vitesse de calcul.
- Efficacité pour l'architecture machine.
- Contrôle des données.
- Disponibilités d'outils (langage, bibliothèque).

Types composés préexistants

Des types composés sont fournis par le langage avec des opérateurs, des fonctions ou des méthodes pour les manipuler :

- Les chaînes de caractères.
- Les tableaux à une dimension.
- Les enregistrements.

Les chaînes de caractères

Permet de mémoriser du texte : `"Bonjour les amis"`

- C'est une suite de caractère,
- accessibles par un indice entier.
- La chaîne a une longueur,
- qui peut être déduite d'un marqueur de fin (en général le caractère de code entier 0).

Les opérations courantes sont : obtention de longueur, la comparaison, la duplication, la concaténation (+), l'extraction ou la recherche d'une sous-chaîne.

Type préexistant :

C	Pascal	Java	Python
`char*`	`string`	`String`	`str`

Chaînes immuables

Dans certains langages, les chaînes sont immuables (Java String, Python str).

```
$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> s = "bonjour"
>>> s
'bonjour'
>>> type(s)
<type 'str'>
>>> s[1]
'o'
>>> s[1] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```

Les tableaux à une dimension

- C'est une suite de cases du même type (n'importe lequel),
- accessibles par un indice entier.
- Le tableau peut avoir une longueur max, fixe ou variable,
- et une longueur courante.

Exemple : un tableau d'entiers

1	1	2	3	5	8	13	21	34	?	?	?
0	1	2	3	4	5	6	7	8	9	10	11

Ne pas confondre : tableau de caractères et chaîne de caractères.

Types préexistants :

C	Pascal	Java	Python
`[]`	`array`	`[]`, `Array`	`list` (pas de tableau)

Les tableaux à une dimension — exemples

- Tableau de chaînes

"Bonjour"	"les"	"amis"	?	?	?
0	1	2	3	4	5

- Tableau de tableaux

	0	1	2	3
0	10	20	77	60
1	15	22	50	20
2	20	50	0	60
3	25	52	-5	80

→ Permet de simuler les tableaux en plusieurs dimensions (matrices, images, ...).

Les enregistrements

- C'est un groupe de structures de données (les champs),
- pouvant être de types différents,
- en nombre fixé,
- identifiées par un nom de champ ;
- les noms de champs sont immuables.

Exemple :

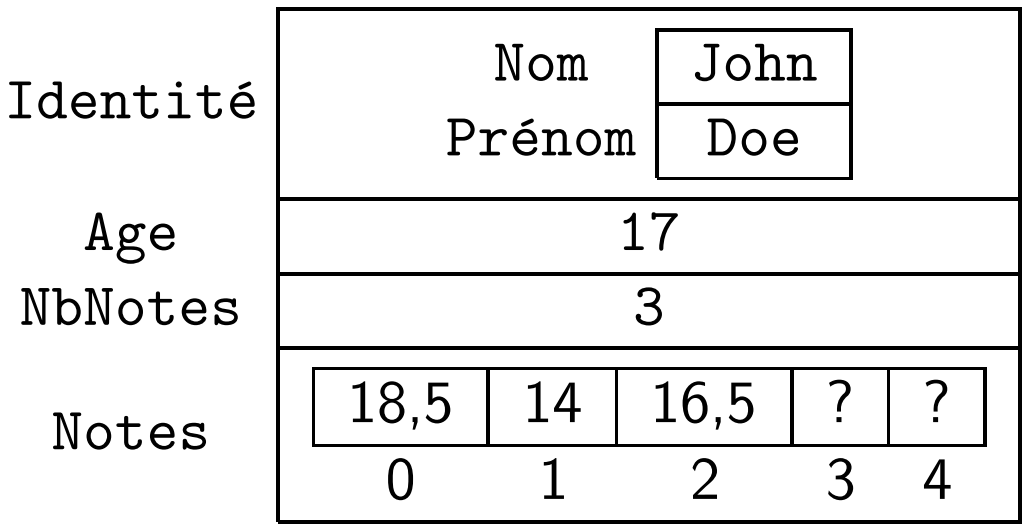
`Nom`	`John`
`Prénom`	`Doe`
`Age`	`17`

Type préexistant :

C	Pascal	Java	Python
`struct`	`record`	`class`	`dict` (pas d'enreg.)

Les enregistrements — exemple

- Les champs d'un enregistrement peuvent être des tableaux, des enregistrements, etc.



3 - Structures de taille variable

Nécessaires pour mémoriser des données dont le nombre ou la taille n'est pas connue à l'avance, ou varie au fil du temps.

- Listes, piles, files.
- Ensembles.
- Arbres.
- Graphes.
- Dictionnaires.

Listes, piles, files, ensembles

Structures de données variables simples :

- Liste : permet de regrouper des données ordonnées, accès libre.
- File : liste avec accès FIFO.
- Pile : liste avec accès LIFO.
- Ensemble : liste non ordonnée, accès libre.

On peut les implémenter avec des tableaux (de taille fixe suffisamment grande, ou élastiques) en C ou en Pascal ;

Java fournit des classes (`List`, ...) ;

Python fournit nativement les listes (`list`) et les ensembles (`set`).

Listes en Python

```
$ python
>>> v = ['Bonjour', 'les', 'amis', 123]
>>> type(v)
<type 'list'>
>>> v[1]
'les'
>>> v = v + ['encore', 'un']
>>> v
['Bonjour', 'les', 'amis', 123, 'encore', 'un']
>>> len(v)
6
```

Les listes remplacent les tableaux en Python.

Les graphes

Un graphe $G = (V, E)$ est constitué d'un ensemble de sommets V et d'un ensemble d'arêtes E entre des couples de sommets.

Un graphe peut être étiqueté, orienté, pondéré.

La théorie des graphes est un domaine de recherche informatique très vaste avec d'importantes applications.

- Euler (1735), problème des sept ponts de Königsberg.
- Coloration, théorème des 4 couleurs.
- Plus court chemin, algorithme de Dijkstra.
- Problème de flot maximum entre une source et un puit.
- etc

Représentation possible d'un graphe par une matrice d'adjacence.

Les arbres

Un arbre est un graphe acyclique orienté,

- dont un sommet est appelé la racine et n'a pas de parent,
- et tous les autres sommets ont un unique parent.

C'est une structure hiérarchique : chaque sommet possède 1 parent (sauf la racine), et 0, 1 ou plusieurs descendants (fils).

On distingue parmi les sommets de l'arbre :

- les nœuds (internes), qui ont des fils ;
- les feuilles (nœuds terminaux), qui n'ont pas de fils.

Les nœuds peuvent être étiquetés.

Exemple d'arbre : un système de fichier : les nœuds sont les dossiers, les feuilles sont les fichiers, les étiquettes sont les noms, la racine '/' est le premier dossier.

Les dictionnaires

Un dictionnaire (ou tableau associatif, ou map) associe un ensemble de clés à un ensemble de valeurs.

C'est une généralisation des tableaux, les indices entiers étant remplacés par des chaînes de caractères.

Les opérations standard sont :

- ajout ;
- modification ;
- suppression ;
- recherche.

Pour être efficaces, les structures de données utilisées utilisent des tables de hachage ou des arbres équilibrés.

Dictionnaires en Python

```
$ python
>>> d = { 'Nom' : 'John', 'Prénom' : 'Doe', 'Age' : 17 }
>>> d
{'Nom': 'John', 'Age': 17, 'Pr\xc3\xa9nom': 'Doe'}
>>> d['Age']
17
>>> d['Notes'] = [18.5, 14, 16.5]
>>> d
{'Nom': 'John', 'Age': 17, 'Pr\xc3\xa9nom': 'Doe',
'Notes': [18.5, 14, 16.5]}
>>> d['Notes'][1]
14
>>> del d['Notes']
>>> d
{'Nom': 'John', 'Age': 17, 'Pr\xc3\xa9nom': 'Doe'}
```

Les dictionnaires remplacent les enregistrements en Python.

4 - Formats de fichier

Un fichier est constitué d'une suite d'octets, mémorisés sur un support physique (disque dur, clé USB, SDcard, disque virtuel, etc).

Il est identifié par un nom, un chemin ; son nom peut avoir une extension. Par exemple

`/home/thiel/public_html/index.html`

Le format d'un fichier est l'organisation physique des données à l'intérieur de celui-ci ; c'est une sorte de structure de données.

Le format d'un fichier

- découle d'une norme, ou d'un standard de fait ;
- il peut être convert par un secret industriel, mais cela ne favorise pas l'interopérabilité $\longrightarrow$ rétro-ingéniérie.

Reconnaître un format

- En regardant sont extension : `.html` $\longrightarrow$ page HTML
- En demandant au système (Linux, MacOS) :

```
$ file index.html
index.html: HTML document text

$ file --mime-type index.html
index.html: text/html
```

Comment le sytème reconnait-il le format (Linux, MacOS) ?

- Il possède une base de données (`/usr/share/file/magic`)
- Il regarde le début du fichier, ou l'extension.

Un fichier excel ou ods (1)

Voici les notes d'évaluation des enseignants dans un tableur :

	A	B	C	D
1	Numéro	Nom	Prénom	Note
2	1	Beffara	Emmanuel	16,1
3	2	Lefèvre	Julien	17,2
4	3	Thiel	Edouard	18,3
5	4	Van Caneghem	Michel	19,4
6				17,75

On peut se demander comment représenter ces données dans un fichier.

Un fichier excel ou ods (2)

On peut décrire cette information comme une ligne de titres, puis des données de chaque colonnes (mis à part la ligne de titre) qui sont toutes de la même nature :

- La première colonne : un entier.
- La seconde colonne : une chaîne de caractères.
- La troisième colonne : une chaîne de caractères.
- La quatrième colonne : un nombre flottant (on utilise une virgule pour séparer la partie entière de la partie décimale).

Enfin une dernière ligne qui calcule la moyenne et qui contient une formule.

On peut remarquer que dans un tableur une cellule peut contenir n'importe quelle donnée.

Enfin il y a une certaine présentation : certaines cellules sont centrées, il peut y avoir des couleurs, tailles de polices, italique, etc.

Le format csv

CSV : Comma-separated values. Format de fichier texte pour coder des données tabulaires.

Voici ce que donne notre exemple (en utilisant le ';' comme séparateur) :

```
"Numéro";"Nom";"Prénom";"Note"
1;"Beffara";"Emmanuel";16,1
2;"Lefèvre";"Julien";17,2
3;"Thiel";"Edouard";18,3
4;"Van Caneghem";"Michel";19,4
;;;17,75
```

On peut remarquer que ce format (qui est très pratique) est assez pauvre :

- il n'y a que des nombres, et des chaînes de caractères (entre guillemets) ;
- les formules et la présentation ont complètement disparu !

Le format ods : Open Document Spreadsheet

```
<office:spreadsheet>
  <table:table table:name="Feuille1" table:style-name="ta1">
    <table:table-row table:style-name="ro1">
      <table:table-cell table:style-name="ce1" office:value-type="string">
        <text:p>Numéro</text:p>
      </table:table-cell>
      <table:table-cell table:style-name="ce1" office:value-type="string">
        <text:p>Nom</text:p>
      </table:table-cell>...
    </table:table-row>
    <table:table-row table:style-name="ro1">
      <table:table-cell table:style-name="ce1" office:value-type="float"
       office:value="1">
        <text:p>1</text:p>
      </table:table-cell>
      <table:table-cell table:style-name="ce3" office:value-type="string">
        <text:p>Beffara</text:p>
      </table:table-cell>...
      <table:table-cell table:style-name="ce3" office:value-type="float"
       office:value="16.1">
        <text:p>16,1</text:p>
      </table:table-cell>
    </table:table-row>...
  </table:table>
</office:spreadsheet>
```

Le format ods (2)

Il s'agit un meta-format xml (Extensible Markup Language) que l'on pourrait simplifier ainsi :

```
<spreadsheet>
    <table>
        <row>
            <cell>..</cell>
            <cell>..</cell>
        </row>
        <row>
            <cell>..</cell>
            <cell>..</cell>
            <cell>..</cell>
        </row>
        <row>
              ...
        </row>
    </table>
</spreadsheet>
```

Le format ods (3)

Ce format xml (plus général que html) sert a représenter de l'information. Il y a la notion d'attribut pour la mise en forme.

On peut dire que :

- un `spreadsheet` est une suite de tables ;
- une table est une suite de rangées ;
- une rangée est une suite de cellules.

Dans cette description on ne voit pas directement la notion de table avec un nombre de colonnes et de lignes fixées : ce n'est pas la représentation d'un tableau rectangulaire.

On sépare de manière claire l'information d'un côté et sa présentation de l'autre côté. Cette présentation est à son tour un objet que l'on peut également représenter en xml.

Le format ods (4)

Si on regarde la manière dont est présentée la première cellule qui correspond au numéro de l'enseignant, voici ce que l'on trouve :

```
<style:style style:name="\ro{ce1}" style:family="table-cell"
    style:parent-style-name="Default">

      <style:table-cell-properties style:text-align-source="fix"
          style:repeat-content="false" \ro{fo:border="0.26pt
          solid #0084d1"} />

      <style:paragraph-properties \ro{fo:text-align="center"}
          fo:margin-left="0cm" />

      <style:text-properties \ro{style:font-name="Trebuchet MS"
          fo:font-size="12pt"} style:font-size-asian="12pt"
          style:font-size-complex="12pt" />
</style:style>
```

Nous voyons donc l'universalité du format xml pour représenter toutes sortes d'informations : soit les données de notre table, soit une présentation.

Le format ods (5)

Si tout le monde se met d'accord sur le sens du précédent format, alors on peut travailler avec le même document sur différents systèmes, sur différentes machines et dans différentes langues. c'est l'interopérabilité.

- ods : fait partie d'une norme : OASIS Open Document Format for Office Applications (odf)
- xlsx : est une extension de nom de fichier pour tableur au format Office Open XML = norme ISO/IEC (IS 29500) créée par Microsoft. Même type de format xml que ods.

Le fait d'avoir 2 vraies normes permet relativement facilement de passer de l'une à l'autre : Excel permet de lire des documents ods et LibreOffice (OpenOffice) permet de lire des documents xlsx.

Il n'y a pas de raisons valables à utiliser une suite bureautique payante comme Office, par rapport à LibreOffice ; il vaut mieux utiliser une suite bureautique libre qu'une version piratée Office.

Les principaux formats de fichiers

Catégorie	Formats
Images	PNG, TIFF, JPEG, GIF, BMP
Dessin vectoriel	SVG, EPS
Son	OGG, FLAC, MP3, WAV, WMA, AAC
Vidéo	MPEG, OGM(DVD, DivX, XviD), AVI, Theora, FLV, MP4, MKV
Page	PDF, PostScript, HTML, XHTML
Document de bureautique	ODT, TXT, DOC(X), RTF, ODS, XLS(X), ...
Archives (fichier compressé)	7Z, TAR, GZIP, ZIP, LZW, RAR