63-11.2 : Fondements de la programmation

Fichiers (textes)

eddy.meylan@he-arc.ch

Les fichiers

- Un fichier est une collection ordonnée d'informations homogènes
- Leur nombre n'est à priori pas connu
- Ces données sont stockées ailleurs qu'en mémoire volatile (RAM)
  - Généralement sur un «disque», et dans un répertoire
- L'unité d'information est **l'enregistrement** (quelque soit son type)

Les fichiers

- De nos jours, les fichiers ne sont plus utilisés comme support de stockage.
  - Les bases de données existent depuis fort longtemps comme support éprouvé de stockage et manipulation de données persistantes
- Les fichiers textes sont encore utilisés, plutôt comme support d'échange de données (xml, csv *encore...*) ou de configuration/paramétrage (ini, cfg)
- La manipulation de fichiers depuis un langage de programmation est besogneux.
  - Chaque langage fourni via ses librairies des services facilitant la manipulation des fichiers et de leur contenu
- Dans le cadre de cette introduction, nous nous contenterons de manipuler des fichiers textes.
  - *Il en existe d'autres…*

Fichiers séquentiels, fichiers textes

- Un fichier séquentiel se caractérise par le fait que deux enregistrements contigus au niveau logique occupent deux emplacements consécutifs au niveau physique.
- L'accès séquentiel impose, pour lire ou écrire un enregistrement, d'avoir lu ou écrit les enregistrements précédents
- **Un fichier texte est un fichier séquentiel dont l'unité d'enregistrement est le caractère**

Fichier texte

- Pour vous, un fichier texte, c'est ca :

```
Petit Papa Noël
Quand tu descendras du ciel
Avec des jouets par milliers
N'oublie pas mon petit soulier
```

- Mais sur votre machine, c'est plutôt ca :

```
10111000100010101000111011010001010100101000101111101001010001010010010001001001010100 10
```


Fichiers textes

- Un fichier texte est constitué de caractères, organisés en lignes
  - Cette organisation est utile pour nous et/ou pour le programme manipulant un fichier texte
  - La machine elle ne stocke que des données binaires
- Un fichier texte ne contient pas le caractère en tant que tel, mais sa représentation numérique (Ascii, Unicode)
- La marque de fin de ligne est un caractère spécial, qui diffère selon les systèmes d'exploitation.
  - CR-LF pour DOS/Windows
  - LF pour Unix
- Ce sont les routines de lecture/écriture qui les traitent

Mise en forme d'un fichier texte

- Un texte comme nous nous le représentons…

```
Petit Papa Noel
Quand tu descendras du ciel
Avec des jouets par milliers
N'oublie pas mon petit soulier
```

  - Est mis en forme... Les sauts de lignes sont aussi des caractères

```
Petit Papa Noel<CR><LF>Quand tu descendras du ciel<CR><LF>Avec des jouets par milliers<CR><LF>N'oublie pas mon petit soulier
```

  - Chaque caractère, y.c les espaces et les saut de lignes, sont «encodés»

```
P  e  t  i  t     P  a  p  a     N  o  e  l  CR LF Q  u  a  n  t     …
80 101 116 105 116 32 80 97 112 97 32 78 111 101 108 13 10 81 117 97 110 100 32 …
```

  - Puis convertis en binaires…

Encodage

- Pour transformer
```
1011100010001010100011101101000101010010100010111110100101000101001001000100100101010010
```
en
```
Petit Papa Noël
Quand tu descendras du ciel
Avec des jouets par milliers
N'oublie pas mon petit soulier
```
il faut déterminer

  - Combien de bit représente un caractère ?
    - » Pour le transformer en un entier non signé
      - Généralement 8 (ou 16) bits
  - Quel caractère représente cet entier
    - » Utilisation d'une table d'encodage
      - A un nombre (entier) correspond un caractère
        - Convention commune

Table d'encodage : L'ancêtre !

- ASCII - *American Standard Code for Information Interchange*
- Première norme définissant en informatique un jeu de caractère standard
  - années 60
- Créé pour la langue anglaise
- Codé sur 8 bits, mais n'en utilise que 7 (!)
  - 2^7 -> 128 caractères

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII - Etendu

- Utilisation des 8 bits
  - 2^8 -> 256 caractères
- Décliné en plusieurs variantes qui utilise des jeux de caractères différents pour la plage 128-255 ☹
  - ISO-8859
  - ISO-8859-1
  - ISO-8859-2
  - ISO-8859-15
  - …
  - Window-1252

ASCII de base — Utilisation du 8ième bit

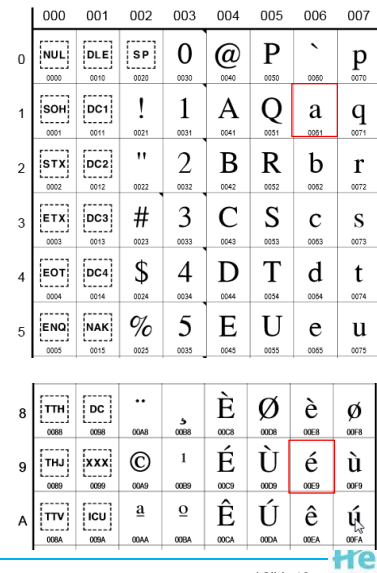
0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	Δ	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	"	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	"	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	'	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	'	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	Σ	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	Π	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	/	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	°
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	fi	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	fl	255	ˇ

Unicode

- Standard informatique développé par le Consortium Unicode
  - Le standard actuel
- Objectif : Définir un jeu de caractère universel
  - Prendre en compte chaque symbole/caractère de n'importe quel système d'écriture
  - Supprimer la profusion de pages de codes spécifiques
  - Augmenter l'interopérabilité et l'échange de données dans le monde
  - Simplifier le développement des logiciels

Unicode

- Attribue à chaque symbole/caractère
  - Un NOM
  - Un NOMBRE (points de code)
- Exemple avec 'a'
  - `LETTRE MINUSCULE LATINE A`
  - `U+0061`
- Exemple avec 'é'
  - `LETTRE MINUSCULE LATINE E ACCENT AIGU`
  - `U+00E9`
- *Unicode va bien plus loin, ce n'est pas qu'une nouvelle table de caractère (mais nous n'irons pas plus loin ici)*



Encodage actuel UTF-8

- Généralement, la table d'encodage utilisée actuellement est UTF-8
- Encode chaque point de code sur un nombre variable d'octets (de 1 à 4 octets)
  - 0 à 127 : encodage sur 1 seul octet
  - 128 à 2047 : encodage sur 2 octets
  - 2048 à 65536 : encodage sur 3 octets
  - …
  - *UTF-8 encapsule parfaitement la table ASCII !*

- 'é' : Unicode : point de code `U+00E9` (233 en décimal)
  - Donc est codé sur 2 octets, qui sont `C3 A9` en UTF-8
  - En ASCII étendu, le 'é' n'est codé que sur un seul octet, puisque ASCII (même étendu) code tout sur un octet seulement

Donc…

- Lorsque vous manipulez du texte, il faut TOUJOURS connaitre
  - Le jeu de caractère
  - L'encodage
    - Sinon, comment interpréter les octets ????

- Un classique

  Ã© au lieu de é

  Les caractères Ã et © sont selon Unicode aux points de code `C3` et `A9`…

Accès à un fichier texte

- Un fichier séquentiel texte ne supporte que 3 modes :
  - Lecture (consultation d'un fichier existant)
  - Ecriture (création d'un nouveau fichier)
  - Ajout (écriture à la fin du fichier existant)
- L'accès à un fichier se fait par un flux
  - une variable représentant le flux (sur le fichier)
  - Un flux est définit en entrée ou en sortie
  - C'est à l'ouverture du fichier que le lien physique se fait
- La syntaxe est passablement différente selon les langages, mais le principe reste (généralement) le même

Ouverture du fichier, liaison du flux au fichier

- Avant toute action, un fichier doit être ouvert
  - On lie un fichier physique à un flux et on définit le «sens» du flux

```
OPEN varFlux, "nom fichier", {lecture|écriture|Ajout}
```

  - `nom fichier` est le nom physique du fichier
    - avec le chemin complet ou relatif
  - L'ouverture en lecture d'un fichier inexistant génère une erreur
  - L'ouverture en écriture d'un fichier écrase le fichier s'il existe déjà

Fermeture du fichier

- Après traitement, un fichier doit être fermé

```
CLOSE varFlux
```

- Entraine la fermeture du fichier et la libération de la mémoire de la variable associée
  - Et éventuellement les verrous sur le fichier

Entrées/sorties

- La lecture et l'écriture dans un fichier texte se fait par des instructions d'entrées/sorties, en y spécifiant le flux
  - Lecture

```
Input (varFlux, maVar (E/S))
```

    - » Lit dans le flux `varFlux` et affect la variable `maVar`
  - Ecriture

```
Output (varFlux, maVar (E) )
```

    - » Ecrit le flux `varFlux` le contenu de la variable `maVar`
- Les lectures/écritures peuvent se faire
  - Par caractère
  - Par ligne

Fichiers séquentiels, notion de «curseur»

- Quelque soit le mode d'ouverture du fichier, il y a toujours un "curseur" qui pointe sur un endroit du fichier physique.
  - C'est la tête de lecture/écriture.
- A l'ouverture du fichier, elle se trouve naturellement au début du fichier, avant le premier enregistrement
  - Peut varier selon le langage
- Lorsque l'on lit un enregistrement, on lit l'enregistrement là où se trouve la tête de lecture.
  - La tête lit autant d'octets que nécessaire
  - Lorsque l'on vient de lire, la tête avance à l'enregistrement suivant

Fin de fichier

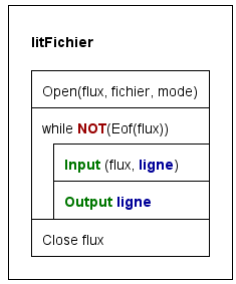
- La détection de la fin de fichier permet de savoir si la tête de lecture se trouve à la fin du fichier
  - Le curseur pointe «dans le vide»
- Cela se fait par la fonction `EOF`, qui retourne un booléen indiquant si la tête se trouve à la fin du fichier ou non.

```
Eof(varFlux):Booléen
```

  - *`EOF` : End Of File*

Algorithme classique

- Principe de lecture séquentielle de tous les enregistrements d'un fichier



  - Ici la lecture se fait ligne par ligne (avec une String) et les affiche à l'écran

Les fichiers Pascal

- Le langage Pascal met à disposition du programmeur des procédures et fonctions permettant de manipuler des fichiers
  - Doit connaitre le type de données d'un fichier pour pouvoir traiter ses enregistrements
  - Dans ce cours, nous nous intéressons uniquement aux fichiers texte,
- Pour les fichiers textes, les entrées sorties se font par les procédures standards
  - `Write, Writeln, Read, Readln`

Déclaration d'un flux

- Le flux est représenté par une variable
- A sa déclaration, on défini le type de fichier

```
VAR
   fLec  : Text;
...
```

  - `Text` est un type
    - » Il est équivalent à `FILE OF Char`

Ouverture du fichier (lecture)

- `Assign()` établit un lien entre la variable de flux et un nom de fichier physique.
  - Avec le chemin complet ou relatif
- `Reset()` ouvre le flux en lecture
  - Conserve les données existantes
  - Positionne la tête sur le 1er enregistrement.

```
Assign(fLec,CnomFichier);
Reset(fLec);
```

  - Si ce fichier n'existe pas, une erreur est levée

Pascal – Ouverture du fichier (écriture)

- Ouverture d'un fichier en écriture

```
Assign(fEcr,CnomFichier);
Rewrite(fEcr);
```

- Deux modes possibles (pour l'écriture)
  - **`Rewrite()`** ouvre le fichier :
    - Si le fichier n'existe pas, il est créé
    - Si le fichier existe déjà, il est écrasé (perte du contenu)
  - **`Append()`** ouvre le fichier :
    - Si le fichier n'existe pas une erreur est levée
    - Si le fichier existe déjà, le contenu est ajouté au fichier

Lecture dans un fichier

- Lire un caractère

```
Read(varFlux, unChar);
```

- Lire une ligne *(et le caractère de fin de ligne)*

```
Readln(varFlux, uneString);
```

- La détection de la marque de fin de ligne et de fin de fichier sont respectivement **`Eoln()`** et **`Eof()`**

Ecriture dans un fichier

- Ecrire un caractère

```
Write(varFlux, unChar);
```

- Ecrire une ligne

```
Writeln(varFlux, uneString);
```

  - Et sauter à la ligne…(*écrit le caractère de fin de ligne*)

- La lecture et l'écriture consistent simplement à lire ou écrire dans un flux, ce flux pouvant être le flux standard (la console) ou le flux fichier

- Caractère(s) de fin de ligne
  - Utile en cas de formatage de texte pour insérer manuellement une marque de fin de ligne
  - Unix LF (#10) Win CRLF(#13#10) MacOS (#13)

Fonction de contrôle et fermeture du fichier

- Détection de la fin de fichier

```
Eof(varFlux): Boolean
```

- Détection de la fin de ligne

```
Eoln(varFlux):Boolean
```

- Fermeture du flux sur le fichier

```
Close(varFlux);
```

Exemple de lecture/affichage d'un fichier en Pascal

- Lecture ligne par ligne

```
PROGRAM litfichier ;
VAR
  fLec    : Text ;
  ligneLue : String ;
BEGIN
  Assign (fLec, 'monfichier.txt') ;
  Reset(fLec);
  WHILE NOT Eof(fLec)DO
    BEGIN
      Readln(fLec,ligneLue) ;
      Writeln(ligneLue);
    END ;
  Close(fLec) ;
END.
```

Lit dans le fichier

Ecrit sur la sortie standard, l'écran

Exemple de copie d'un fichier en Pascal

- Caractère par caractère

```
PROGRAM copie ;
VAR
 fLec : Text ;
 fEcr : Text ;
 car  : Char ;
BEGIN
  Assign (fLec, 'fichiersource.txt') ;
  Reset(fLec);
  Assign (fEcr, 'fichierdestination.txt');
  Rewrite(fEcr);
  WHILE NOT Eof(fLec)DO
    BEGIN
      WHILE NOT Eoln(fLec)DO
        BEGIN
          Read(fLec,car) ;
          Write(fEcr,car);
        END ;
      Readln(fLec,car) ;
      Writeln(fEcr,car);
    END ;
 Close(fLec) ;
 Close(fEcr) ;
END.
```

Lecture d'un caractère

Ecriture d'un caractère

Lecture du saut de ligne

Ecriture du saut de ligne

Structure dans un fichier texte

- Un fichier texte est une suite de caractères !
- On veut (parfois) stocker une structure dans un fichier texte, on va donc «formater» la structure en une suite caractères
  - La manipulation d'un fichier «formaté» nécessite de connaitre sa structure (format)
- Le fichier formaté est généralement organisé en lignes :
  - Chaque ligne contient un enregistrement
  - Toutes les lignes ont le même format
- La transformation de ligne en enregistrement (et l'inverse) se fait avec des traitements de chaines (`String`)
  - Les nombres (Entier et/ou Réel) sont stockés en `String` et nécessitent des fonctions de conversion

Quelques organisations courantes

- Colonnes fixes
  - Chaque colonne à une longueur fixe
- Séparateur
  - Chaque colonne est séparé par un caractère connu *(ici l'espace)*
    - » Ne doit pas être contenu «dans» les colonnes
- CSV *(peut se lire avec Excel)*
  - Le séparateur est (généralement) le `;`
  - La première ligne contient le nom des colonnes
- Lignes
  - Chaque ligne est un champs
    - » Tous les enregistrements doivent avoir les même attributs

```
111Levert    Albert
222Hyacinthe Facteur
```

```
111 Levert Albert
222 Hyacinthe Facteur
```

```
numero;nom;prn
111;Levert;Albert
222;Hyacinthe;Facteur
```

```
111
Levert
Albert
222
Hyacinthe
Facteur
```

Traitement de chaines (String)

- Pour manipuler des fichiers textes, il est (très) souvent nécessaire de «traiter» des chaines de caractères
- `nomChaine[position]`
  - Extraction d'un caractère de la chaine à une position
    - » En Pascal, commence à 1 (!)
- `Length()`
  - Retourne la longueur de la chaine
- `Pos()`
  - Retourne la position d'une chaine dans une autre
    - » Très utile pour détecter les séparateurs (et leurs positions)
- `Copy()`
  - Retourne une sous chaine extraite d'une autre
    - » Très utile pour «découper» une ligne

Qq exemples

- `maChaine := 'Salut les lapins' ;`
  - `maChaine[2]` contient `'a'`
  - `Length(maChaine)` retourne `16`
  - `Pos('les',maChaine)` retourne `7`
  - `Pos('zut',maChaine)` retourne `0`
  - `Copy(maChaine,7,3)` retourne `'les'`

QQ commandes supplémentaires

- `Rename(f,'nomfichier');`
  - Renomme le fichier (doit être «assigné» et fermé)
- `Erase(f);`
  - Efface le fichier (doit être «assigné» et fermé)

- `Getdir`
  - Retourne le répertoire de travail courant
- `Chdir()`
  - Change de répertoire de travail
- `Mkdir()`
  - Crée un répertoire
- `Rmdir()`
  - Supprime un répertoire

Supplément gratuit pour les curieux