



## Devoir d'informatique

### Consignes

- Les scripts sont à déposer sur le site <http://envoi.lamartin.fr/> avant le 17 février 22h00.
- Concevoir un algorithme répondant à un problème précisément posé.
- Écrire des instructions conditionnelles avec alternatives, éventuellement imbriquées.
- Choisir un type de données en fonction d'un problème à résoudre.
- Traduire un algorithme dans un langage de programmation.
- Lire et écrire dans un fichier.

## 1 Présentation

### 1.1 Généralités



A l'origine, le MIDI (Musical Instrument Digital Interface) est une interface, une norme matérielle (câbles et prises) et logicielle (protocole d'échange de données) de communication entre instruments de musique électroniques. Avec un clavier Midi branché sur un ordinateur, il est possible d'enregistrer des accords, des rythmes, des mélodies, et même des orchestrations complètes grâce à des logiciels nommés arrangeurs ou séquenceurs.

Contrairement à un enregistrement classique à l'aide d'un microphone, ce ne sont pas des sons qui sont enregistrés, mais les notes jouées sur le clavier. La norme Midi attribue un numéro à chacune de ces notes. Le résultat de l'enregistrement est un fichier Midi composé uniquement de nombres : les numéros des notes et des indications sur l'interprétation ainsi que les numéros des sons à utiliser.



Les fichiers MIDI peuvent être lus sur PC via le lecteur **Windows Media Player**, sur Mac il est nécessaire d'installer l'application **VLC version 3.0.4**.

### 1.2 Éléments de structure à identifier dans ce DM

Il existe au moins deux plages dans un fichier MIDI :

- plage d'entête (commençant par "MThd")
- plages de données (commençant par "MTrk")

#### La plage de d'entête

La première plage de données renseigne sur la structure du fichier et la plus petite unité de temps considérée :

- 4 octets - nombre magique x4D546864 : "MThd"
- 4 octets - espace des spécifications du fichier : 0, 0, 0, 6 pour les 6 octets suivants
- 2 octets - type SMF (Standard Midi File) :
  - . 0, 0 : une seule plage de données où se mélangent les événements de plusieurs canaux;
  - . 0, 1 : plusieurs plages qui se suivent dans le fichier, mais jouées simultanément;
  - . 0, 2 : plusieurs plages, jouées l'une après l'autre (peu utilisé).
- 2 octets - nombre de plages de données ("MTrk"), de 1 à 65.535
- 2 octets - nombre de divisions de la noire (résolution temporelle du fichier)

#### La plage de données

Les plages sont composées de délais, d'événements musicaux, de contrôles et de métadonnées dont il sera question dans la section suivante.

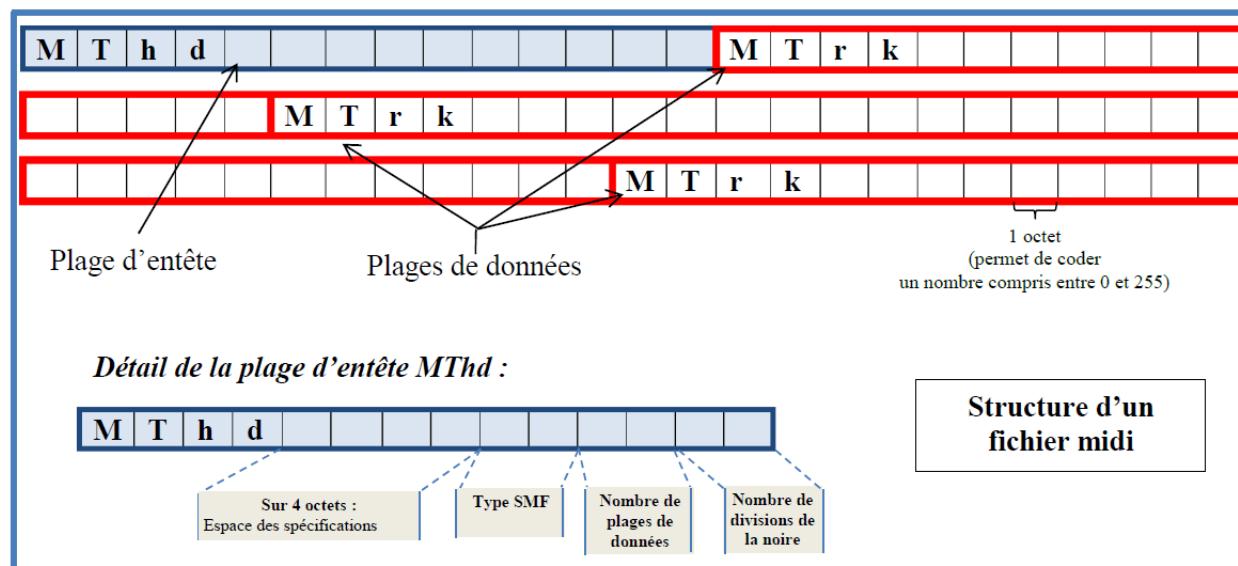


FIGURE 1 – Trame d'un fichier MIDI

Remarque : on utilise le code ASCII pour les lettres (par exemple 'a' est codé en décimal par 97). Pour passer du code ASCII au code décimal et inversement, se référer à l'annexe 1 en fin de document.

Pour répondre à votre curiosité, vous pouvez retrouver l'ensemble des informations sur cette norme en vous rendant sur le site : <https://www.jchr.be/linux/midi-format.htm>

## 2 Travail à effectuer

### 2.1 Lecture - Écriture d'un fichier MIDI et mise en forme des données

Contrairement aux fichiers habituellement utilisés, notamment dans le TP09, les fichiers midis sont des fichiers binaires. Nous allons lire ce fichier octet par octet, pour cela il suffit de **rajouter le caractère 'b'** au moment de l'ouverture du fichier.

```
f = open("jingle.mid", "rb") #ouverture en mode lecture, mode binaire\\
f = open("jingle2.mid", "wb") #ouverture en mode écriture, mode binaire
```

En dehors de l'ajout de ce caractère, vous utiliserez la procédure habituelle vue en cours pour écrire/lire un fichier.

On obtient alors des objets de type « bytes » de la forme `b'\x01\x02\x03'` qui contiennent les octets du fichier lu. On propose de convertir directement ces objets en listes d'entiers de la forme [1, 2, 3], que vous avez davantage l'habitude de manipuler. Pour ceci on utilise les fonctions suivantes :

```
listeEntier = list(ligneBytes) #conversion de bytes en liste d'entier\\
ligneBytes = bytes(listeEntier) #conversion d'une liste d'entier en bytes
```

#### Lecture du fichier

**Question 1** Écrire un programme qui lit le fichier jingle.mid et affiche dans le shell les lignes du fichier une à une.

Remarque : sur Pyzo, il faut changer le répertoire courant pour que lors de l'exécution le script aille chercher le fichier dans le dossier où se trouve les fichiers utiles. On utilise les instructions ci-dessous :

```
import os #importation de la bibliothèque os\\
os.chdir(r'D: \\$backslash$chemin') #changement de répertoire courant\\
#chemin est le chemin complet vers le dossier contenant le fichier midi
```

Une autre méthode, plus simple (toujours avec Pyzo), par un clic droit sur l'onglet de votre programme, sélectionner "Run File as script" pour la première exécution. Tous les fichiers utiles doivent être dans le même dossier.

## Mise en forme des données

**Question 2** Créez une liste `les_lignes_entier` qui contient les listes d'entiers correspondant à chaque ligne lue, en utilisant la fonction `list` présentée ci-dessus.

Les sauts de lignes n'ont rien de significatif ici (le caractère '\n' est un caractère comme un autre en MIDI : ne pas l'enlever) ; on souhaite donc regrouper l'ensemble des données en une seule liste, en mettant les lignes obtenues « bout à bout ».

**Question 3** Créez une liste `donnees` regroupant toutes les lignes. Vérifiez que cette liste contient bien 23783 éléments.

## Écriture d'un fichier MIDI

La suite du TP propose de modifier des éléments dans le fichier MIDI, puis d'écouter l'effet de la modification. Il faut pour cela être capable d'écrire un nouveau fichier MIDI à partir des données modifiées.

**Question 4** Écrire une fonction `écrire_midi` (`nom du fichier:str, liste:list`) qui écrit un fichier midi à partir d'une liste d'entiers. Cette fonction prend pour arguments le nom du fichier à générer (chaîne de caractères) et la liste de données à écrire (liste d'entiers). Cette fonction ne renvoie rien.

Aide : cette fonction reprend les étapes ci-dessus en sens inverse : conversion en bytes, puis écriture en binaire. Pas besoin de redécouper les lignes.

**Question 5** Tester cette fonction en générant le fichier `jingle2.mid` qui contient exactement les mêmes données que celles du fichier initial. Vérifier, en écoutant, que les sons n'ont pas été modifiés.

## 2.2 Étude de l'entête du fichier

A partir de la liste d'entiers `donnees` obtenue à la question 3 :

**Question 6** Afficher les 14 octets définissant la plage d'entête.

**Question 7** Répondre dans le script sous forme de commentaire :

- Quelle est la signification des quatre premiers termes (vous justifierez votre réponse par une ligne de code en utilisant le décodage ASCII voir annexe 1) ?
- Quel est type SMF utilisé ici ?
- Quel est le nombre de plages ?
- Quel est le nombre de divisions de la noire ?

## 2.3 Modification du tempo

**Question 8** Placer le nombre de divisions de la noire à 255. Écrire un nouveau fichier `jingle_tempo.mid`. Vérifier l'effet obtenu.

La modification pourra être conservée pour la suite du DM.

## 2.4 Identification des plages MTrk

Chaque plage de données commence par le mot `MTrk`.

**Question 9** Convertir le mot `b'MTrk` en entiers. Noter `L_MTRK` la liste des quatre entiers correspondants.

**Question 10** Créez une liste `les_indices` qui contient l'indice de début de chaque piste `MTrk`.

**Question 11** Valider le nombre de plages obtenues au regard de la question 7.

## 2.5 Modification d'un instrument

### Découverte des instruments utilisés dans le fichier `jingle.mid`

Les données de chaque piste « `MTrk` » contiennent des événements qui peuvent être par exemple : la production d'une note, le choix d'un instrument, la modification d'un timbre, la modification du volume ...  
La modification d'un instrument se code sur 2 octets :

- le premier octet « Cc » (en hexadécimal) indique un changement d'instrument sur le canal c.

- le second octet indique le code de l'instrument choisi (cf annexe 2).

**Exemple :** « C4 06 » : changement d'instrument sur le canal 4, code instrument : 06, correspondant au clavecin.

Sachant qu'un fichier MIDI contient au plus 16 canaux numérotés de 0 à 15, un changement d'instrument sera donc codé, en **hexadécimal**, de C0 à CF. En notation **décimale** un changement d'instrument est codé par 192 pour le canal 0; 193 pour le canal 1; 194 pour le canal 2; etc; 207 pour le canal 15.

**Note :** Un canal MIDI permet le transport des données pour commander un instrument équipé d'un port MIDI.

**Question 12** Rechercher, dans la liste données, tous les changements d'instrument : créer deux listes :

- une liste `les_instruments` contenant tous les codes instruments utilisés;
- une liste `les_canaux` qui contient le numéro du canal utilisé (de 0 à 15).

**Question 13** Indiquer, sous forme de commentaire, la liste des instruments utilisés, et le canal correspondant.

#### Modification d'un instrument

**Question 14** Remplacer l'instrument « violon » par un « jeu de cloche ». Écrire le résultat dans le fichier `jingle_cloche.mid`.

**Question 15** Écouter l'effet obtenu !

#### Facultatif : atténuation de la mélodie

**Question 16** En vous aidant de la norme MIDI (voir par exemple le site <https://www.jchr.be/linux/midi-format.htm>), modifier le « control change » associé au volume du canal « channel volume », pour diminuer fortement le volume de la mélodie (canal de l'ex violon). Écrire le fichier `jingle_acc.mid`.

## ANNEXE 1

Au sein d'un langage de programmation, les caractères sont représentés sous forme de nombre selon ce qu'on appelle le code ASCII.

On utilise deux fonctions prédéfinies en python : `ord` et `chr` qui permettent d'associer à chaque caractère un entier entre 0 et 255, et réciproquement. Voici un exemple :

```
>>> ord('a')
    97
>>> chr(97)
    'a'
```

Le tableau ci-dessous indique ces correspondances :

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	-	127	7F	177	

## ANNEXE 2

### Extrait de la norme General Midi : codage des instruments

Codage décimal	Pianos et claviers	Codage décimal	Cordes et timbales
0	Piano à queue	40	Violon
1	Piano de concert	41	Alto
2	Piano électrique	42	Violoncelle
3	Piano bastringue	43	Contrebasse
4	Piano électronique - 1	44	Cordes - effet trémolo
5	Piano électronique - 2	45	Pizzicato de cordes
6	Clavecin	46	Harpe
7	Clavicorde	47	Timbales
Codage décimal	Instruments chromatiques	Codage décimal	Ensembles et chœurs
8	Célesta	48	Ensemble de cordes - 1
9	Glockenspiel	49	Ensemble de cordes - 2
10	Boîte à musique	50	Cordes de synthèse - 1
11	Vibraphone	51	Cordes de synthèse - 2
12	Marimba	52	Chœurs - Aah
13	Xylophone	53	Chœurs - Oooh
14	Jeu de cloches	54	Chœurs de synthèse
15	Dulcimer	55	Tutti d'orchestre symphonique
Codage décimal	Orgues	Codage décimal	Cuivres
16	Orgue Hammond	56	Trompette
17	Orgue Hammond - effet percussion	57	Trombone
18	Orgue électronique	58	Tuba
19	Grand orgue	59	Trompette en sourdine
20	Harmonium	60	Cor d'harmonie
21	Accordéon	61	Pupitre des cuivres
22	Harmonica	62	Cuivres de synthèse - 1
23	Bandonéon	63	Cuivres de synthèse - 2
Codage décimal	Guitares	Codage décimal	Anches
24	Guitare acoustique nylon	64	Saxophone soprano
25	Guitare acoustique métal	65	Saxophone alto
26	Guitare électrique jazz	66	Saxophone ténor
27	Guitare électrique	67	Saxophone baryton
28	Guitare électrique - son amorti	68	Hautbois
29	Guitare électrique - son saturé	69	Cor anglais
30	Guitare électrique avec distorsion	70	Basson
31	Harmoniques de guitare	71	Clarinette
Codage décimal	Basses	Codage décimal	Vents
32	Basse acoustique	72	Piccolo
33	Basse électrique	73	Flûte traversière
34	Basse électrique avec médiator	74	Flûte à bec
35	Basse fretless	75	Flûte de pan
36	Slap basse - 1	76	Cruche (Jazz)
37	Slap basse - 2	77	Flûte shakuhachi
38	Basse de synthèse - 1	78	Sifflet
39	Basse de synthèse - 2	79	Ocarina