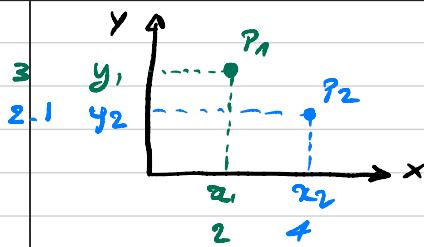


Sinfo2 M: - structures

11. 04. 2021



x₁, x₂, y₁ et y₂ sont des nœuds.

$$\Delta x = 2 \\ \Delta y = 0.9$$

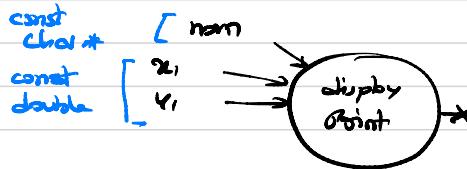
TD 2021/03/11:

- 1) créer les 4 variables x₁, y₁, x₂ & y₂
+ valeurs initiales
- 2) appeler une fonction "displayPoint"
qui affiche les coordonnées x et y
d'un point et son nom
 $P_1: (+2.0, +3.0)$
- 3) appeler une fonction qui calcule
et affiche la distance entre 2 points
"computeDistance"

Analysé → prototype
→ pseudo-code.

16' 55.

① display Point



`void displayPoint (const char * name, const double x, const double y);`

② compute Distance



`double computeDistance (const double x1, const double y1,
const double x2, const double y2);`

$$\text{Calcul} \rightarrow (\text{Pythagore}) \quad \text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Implémentation des 2 fonctions + test avec les valeurs en haut de la page.

17' 11

Point 2D

coordonnée x
coordonnée y
nom

double
double
char [...]

Structur en c → 1 nouveau type → choisir son nom
→ des données "cohérentes" entre elles
↳ "champs" → type et un nom.

Structur: sPoint2D

↳ x
↳ y
↳ name

← nom du nouveau type
type double
double
char [20]

```
typedef struct {
    double x;
    double y;
    char name [20];
} sPoint2D;
```

création d'un nouveau type

L'initialisation des valeurs
des champs s'effectue lors
de la création des variables.

Utilisation de la structur

sPoint2D p1 = { 2., 3., "P1" };
sPoint2D p2 = { 4., 2.1, "P2" };] création de 2 variables structurées

printf ("x = %f", p1.x);

p1.y

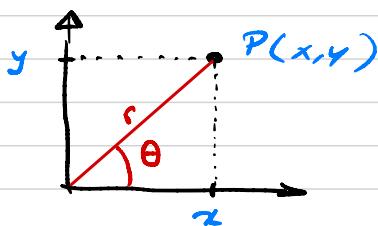
%s p1.name

TD 2021/03/11 b

1) créer la structure

2) adapter le TD 2021/03/11 aux le type structuré.

17/48



c2p: Cartesian to Polar
(x,y) \rightarrow (r,θ)

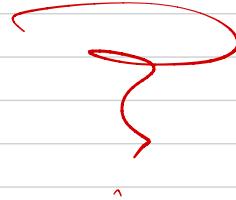
Créer une fonction telle que:



TD 2021/03/11c

- 1) créer la fonction c2p
- 2) ajouter un type struct pour le modèle polarique (r, θ)
- 3) afficher le (r, θ) pour les points P_1 et P_2 du TD 2021/03/11 b.

A terminer pour la session du 15. III. 2021 -

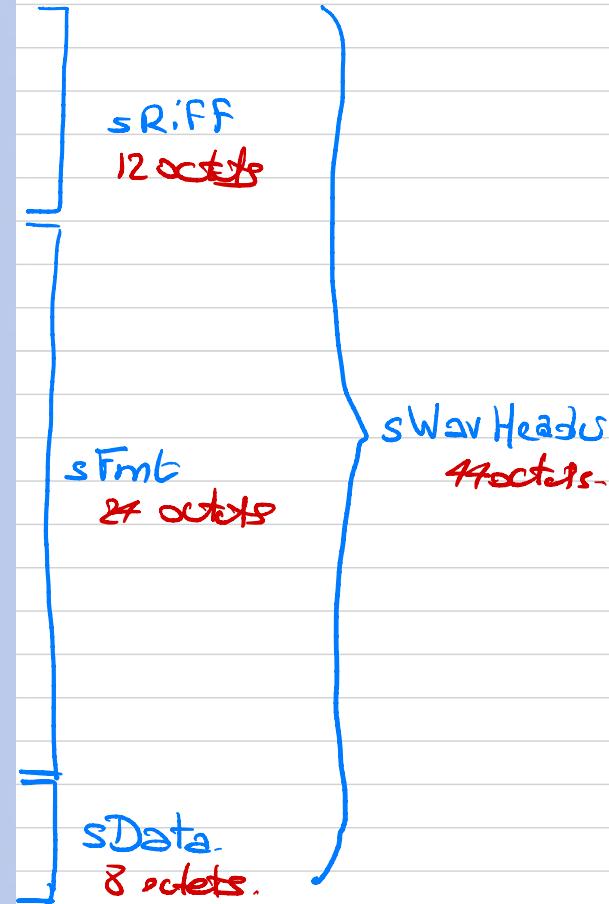


InfoM - Structure

15. 03. 2021

(!) que des valeurs entières

File offset (bytes)	field name	Field Size (bytes)
0	ChunkID	4
4	ChunkSize	4
8	Format	4
12	Subchunk1ID	4
16	Subchunk1 Size	4
20	AudioFormat	2
22	NumChannels	2
24	SampleRate	4
28	ByteRate	4
32	BlockAlign	2
34	BitsPerSample	2
36	Subchunk2ID	4
40	Subchunk2 Size	4
44	data	Subchunk2Size



- TD 20210315:
- ① création des 3 structures sRIFF, sFmt, sData.
 - ② affichage de leurs tailles en octets 12, 24, 8
 - ③ création de la structure sWavHeader
 - ④ affichage de sa taille en octet 44.

TD / TD 20210315 - data.zip

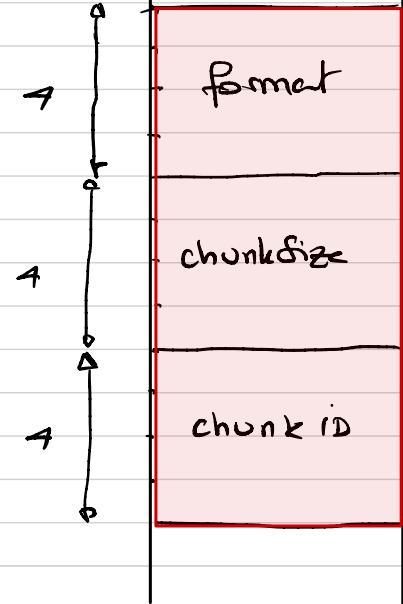
png (image de la structure)
92.wav
42s.wav

13h45

Padding en mémoire

```
typedef struct {
    uint32_t chunkId;
    uint32_t chunkSize;
    uint32_t format;
} sRiff;
```

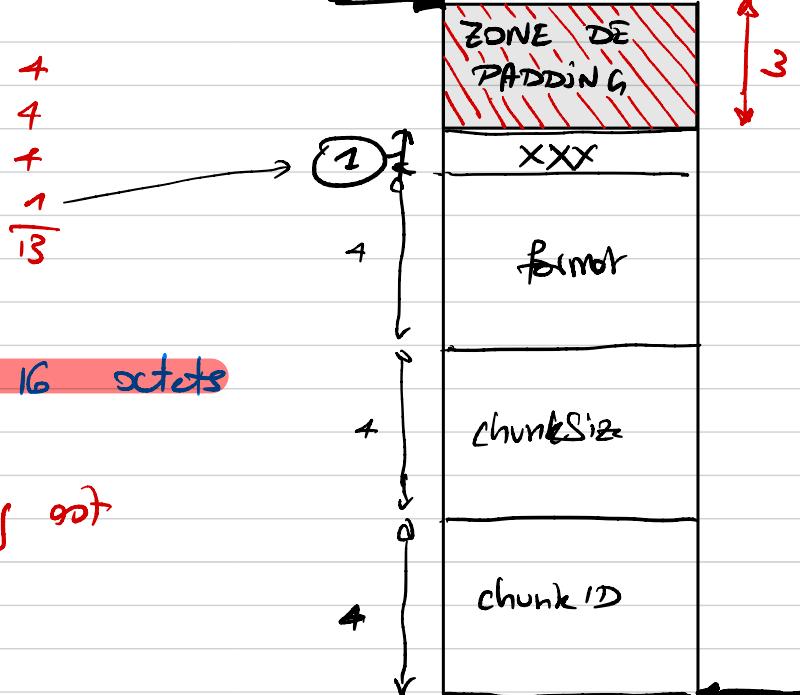
`sizeof(sRiff)` → 12 octets.



```
typedef struct {
    uint32_t chunkId;
    uint32_t chunkSize;
    uint32_t format;
    uint8_t xxx;
} sRiff;
```

n'ream
champs
10ctet

`sizeof(sRiff)` → 16 octets



(!) Toujours vérifier si le padding est
généré ou pas.

Pour supprimer le padding : `#pragma pack(1)`

`#pragma pack()`

```
typedef struct {
    uint32_t chunkId;
    uint32_t chunkSize;
    uint32_t format;
    uint8_t xxx;
} sRiff;
```

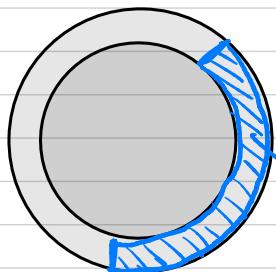
→ 13 octets

`#pragma pack($)`

Fichiers Binaires

15. Fév. 2021

ex: Fichier de type .WAV (son) → 42.wav (mono) 1 note de 42s.wav (stéréo) piano



42.wav

octets

42.wav : 23536
42s.wav : 47028

Fichier binaires



L'entête donne des informations sur les données -

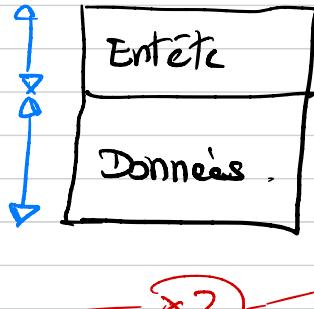
ex pour le .WAV : mono (fnt.numChannels = 1)
stéréo (= 2)

Fréquence d'échantillonage [Hz]
(fnt.sampleRate)

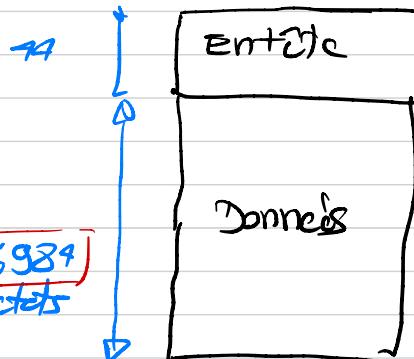
Mono
42.wav

sWavHeader 44

23492
octets



stéréo
42s.wav



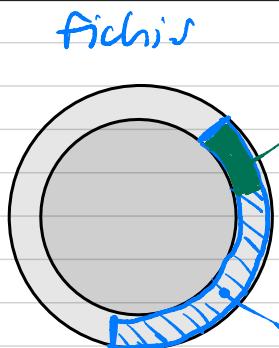
Lire le contenu du fichier pour afficher l'entête.

Fichier binaire. ⇒ ~~fscanf~~



fread

Utilisation fread.



42.wav

```
sWavHeader h;
FILE *f = NULL;
f = fopen("42.wav", "rb");
// test f != NULL
```

fread(&h, sizeof(sWavHeader), 1, f);

→ transfert 1x 44 octets depuis f vers h

20210315 (suite):

- lire l'entête du fichier → variable h
- afficher les éléments suivants (points jaunes)
- + PC123

```
typedef struct {
    uint32_t chunkId;
    uint32_t chunkSize;
    uint32_t format;
} sRiff;

typedef struct {
    uint32_t subChunk1Id;
    uint32_t subChunk1Size;
    uint16_t audioFormat;
    uint16_t numChannels;
    uint32_t sampleRate;
    uint32_t byteRate;
    uint16_t blockAlign;
    uint16_t bitsPerSample;
} sFmt;

typedef struct {
    uint32_t subChunk2Id;
    uint32_t subChunk2Size;
} sData;
```

Résumé

	42.wav	42s.wav
h.fmt.numChannels	1	2
sample Rate	22050	22050
bits Per Sample	16	16
subChunk2Size	83492	46984

|| finir la mise en point pour la
|| prochain fois (22.III.2021).