POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF1005A: Programmation procedurale

Révision final

Révision – Fonctions temporelles

- `variab = clock()` – retourne une matrice qui contient les éléments suivants :

  `temps = [année mois jour heure minute seconde]`

- `variab= cputime` - le temps(en secondes) d'utilisation du CPU par MATLAB depuis son démarrage
- `etime(temps1, temps2)` – calcule le temps écoulé en secondes entre deux matrices obtenues avec `clock()`.
- `tic`

  Instructions

  `toc`

  - fonctions qui calculent le temps écoulé entre les appels de `tic()` et `toc()`. `tic()` démarre le chronomètre et `toc()` affiche la valeur présente du chronomètre.

- `pause()` - sert à interrompre un programme jusqu'à ce que l'usager appuie sur une touche
- `pause(durée_en_secondes)` - le programme est arrêté durant le nombre de secondes spécifié en paramètre

Révision – Erreurs

Trois types d'erreur

- **de syntaxe**:

  i.e.

```
fro ou if a=b
```

- **de logique:**

  i.e.

```
a=10;
while a<20
   a=a-1;
end
```

- **de données:**

  i.e.

```
a=10;
   b=a(3)+10;
```

Révision – Erreurs

`error('ID_message', 'message', arg1, arg2, ...)`

- affiche un message d'erreur et arrête le programme

`[message, ID_message]=lasterr`

`lasterr` – retourne la dernière erreur

`warning ('message_d'avertissement',var1,var2,...)`

- affiche un message d'avertissement et n'arrête pas le programme

`lastwarn` – retourne le dernier avertissement

Révision – Fonctions

obligatoires

- `function [valeurs_de_retour] = nom_de_la_fonction (paramètres)`
  - `nargin`
  - `nargout`
- `function [] = ma_fonction(paramètres)`
- `function [valeurs de retour] = ma_fonction()`
- `function [varargout]=ma_fonction(varargin)`

  `X=varargin{k}(i)` (`length` pour le nombre de paramètres dans `varargin`).

Exemple initialisation `varargout: varargout={}`

Révision – Représentation interne de données

- La base 2
  Bit le plus significatif: bit ayant la pondération la plus élevée
  Bit le moins significatif: bit ayant la pondération la moins élevée (le dernier bit a droit)
- Conversion décimale-binaire
  - division par 2, si divisible on a 0 sinon 1 ou
  - puissances de 2
- Conversion binaire-octale (chiffre entre 0 et 7)
  - regrouper les bits par paquet de trois à partir du bit le moins significatif
  - calculer la valeur associée à chaque paquet
    1010100111 = 1 010 100 111 (1247) base 8
- Conversion binaire- hexadécimale (chiffre entre 0 et 9 plus A-F pour 10-15)
  - regrouper les bits par paquet de quatre à partir du bit le moins significatif
  - calculer la valeur associée à chaque paquet
    1010100111 = 10 1010 0111 (2A7) base 16

Révision – Représentation interne de données

- La base 2
  - Bit le plus significatif: bit ayant la pondération la plus élevée
  - Bit le moins significatif: bit ayant la pondération la moins élevée (le dernier bit a droit)
- Conversion décimale-binaire
  - division par 2, si divisible on a 0 sinon 1 ou
  - puissances de 2
- Conversion binaire-octale (chiffre entre 0 et 7)
  - Regrouper les bits par paquet de trois à partir du bit le moins significatif
  - Calculer la valeur associée à chaque paquet
    - 1010100111 = 1 010 100 111 (1247) base 8
- Conversion binaire- hexadécimale (chiffre entre 0 et 9 plus A-F pour 10-15)
  - Regrouper les bits par paquet de quatre à partir du bit le moins significatif
  - Calculer la valeur associée à chaque paquet
    - 1010100111 = 10 1010 0111 (2A7) base 16

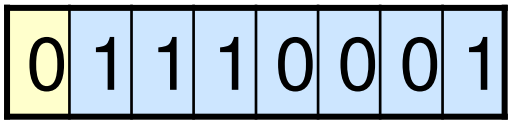
Révision – Représentation interne de données

Entiers signés : $[-2^{n-1} : (2^{n-1} -1)]$ (dans MATLAB `int8, int16, int32, int64`)

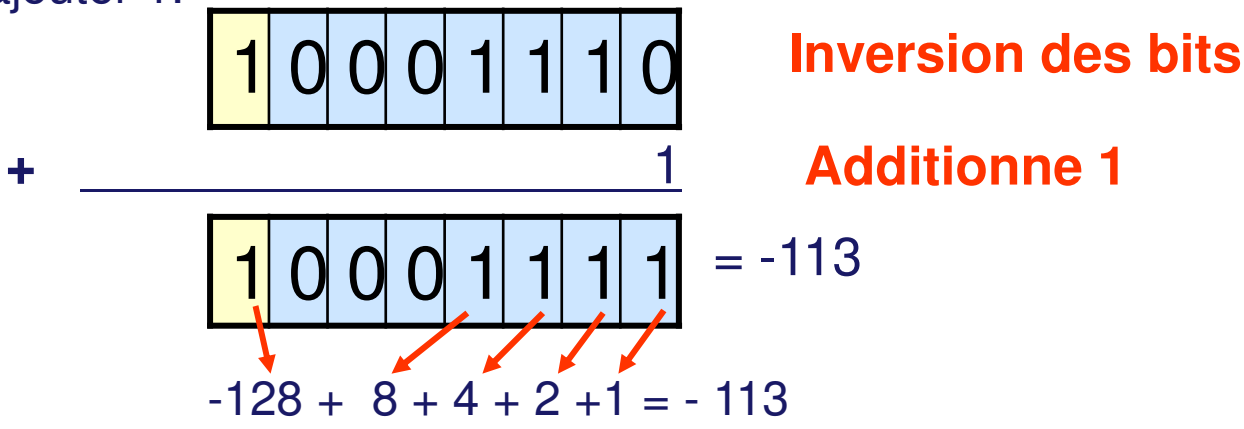
Le bit le plus significatif, le plus à gauche, précise le signe du nombre:

- 0 indique un nombre positif
- 1 indique un nombre négatif

i.e. représentation de 113 sur 8 bits:



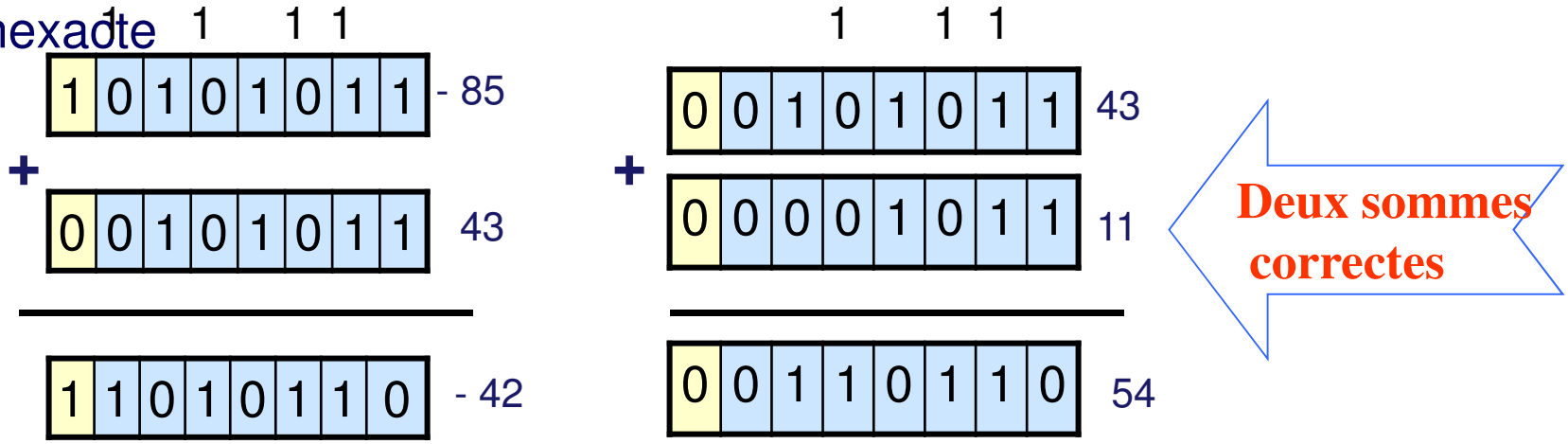
Pour obtenir la représentation du nombre négatif, il faut inverser tous les bits du nombre positif et y ajouter 1.



Révision – Représentation interne de données

Addition entiers signés, complément à 2

- Additionner bit à bit
- Ignorer la retenue
- Validité de la réponse
  - ✓ Si les deux opérandes sont de signe opposée, la réponse est exacte
  - ✓ Si les deux opérandes sont de même signe, mais que la réponse est de signe opposé, on dit qu'il y a *débordement* et la réponse est alors inexacte



Révision – Représentation interne de données

Nombres réels – notation scientifique

- un format de mémorisation standardisé. Il incorpore trois éléments: la mantisse, la base et l'exposant:

  $2{,}7 \times 10^5$

  2,7 est la mantisse

  10 est la base

  5 est l'exposant

- connaissant la base, il est possible de représenter le nombre à l'aide du triplet: (Signe, Exposant, Mantisse) (IEEE754)

	Nombre bit du signe	Nombre de bits de l'exposant	Nombre de bits de la mantisse
Simple précision 32 bits	1	8	23
Double précision 64 bits	1	11	52

Révision – Représentation interne de données

Nombres réels, règles norme IEEE 754:

- le bit de signe est 0 pour un nombre positif ou 1 pour un nombre négatif

- la mantisse est représentée par notation positionnelle. Le premier bit de la partie entière est sous-entendu puisqu'il est toujours 1.

- l'exposant est représenté par excès de $2^{n-1}-1$. Il s'agit de l'astuce utilisée pour combler l'absence du signe de l'exposant. En simple précision l'excès est de 127 tandis qu'en double précision l'excès est de 1023.

Révision – Représentation interne de données

Conversion de 132.147 en format simple précision

= 0 10000110 00001000010010110100010

Partie entière = 132		Partie fractionnnaire=147	
nbre	nbre bin	nbre	Partie ent. de (2×nb)
		0.147	0
132	0	0.294	0
66	0	0.588	1
33	1	0.176	0
16	0	0.352	0
8	0	0.704	1
4	0	0.408	0
2	0	0.816	1
1	1	0.632	1
0		0.264	0
		0.528	1
		0.056	0
		0.112	0
		0.224	0
		0.448	0
		0.896	1
		0.792	1

Pour arrondi.

Si 1, fait +1, sinon le dernier bit reste comme il est.

Dernier bit est arrondi à 1

132 . 147

10000100.0010010110100001 +1

10000100 = 1.0000100×2^7

1. 00001000010010110100010×2^7

Signe = 0

Exposant = 7 + 127 = 134

= 10000110

Mantisse = 00001000010010110100010

S	Exposant	Mantisse
0	1000 0110	0000 1000 0100 1011 0100 010

On a 23 bits pour la mantisse

Révision – Fichiers

Ouverture des fichiers:

```
ID_FIC = fopen('NOM_FIC',
  'TYPE_OUVERTURE')
```

vérification ouverture `ID_FIC ~=-1`

i.e.

```
fid=fopen('etudiants.txt','rt');
if fid == -1
  disp('Problème lors de l''ouverture');
else
  disp('Fichier ouvert');
end
```

Révision – Fichiers

Fermeture des fichiers:

`fclose(ID_FIC)`

vérification fermeture fichiers

`verification = fclose(ID_FIC)`

`verification :` variable qui indique si la fermeture s'est effectuée – retourne 0 si tout se passe bien ou -1 s'il y a eu des problèmes.

`ID_FIC :` Identificateur du fichier ouvert.

Révision – Fichiers

Pour savoir si on a lu un fichier au complet ou non, on doit vérifier que la fonction de lecture retourne une valeur erronée:

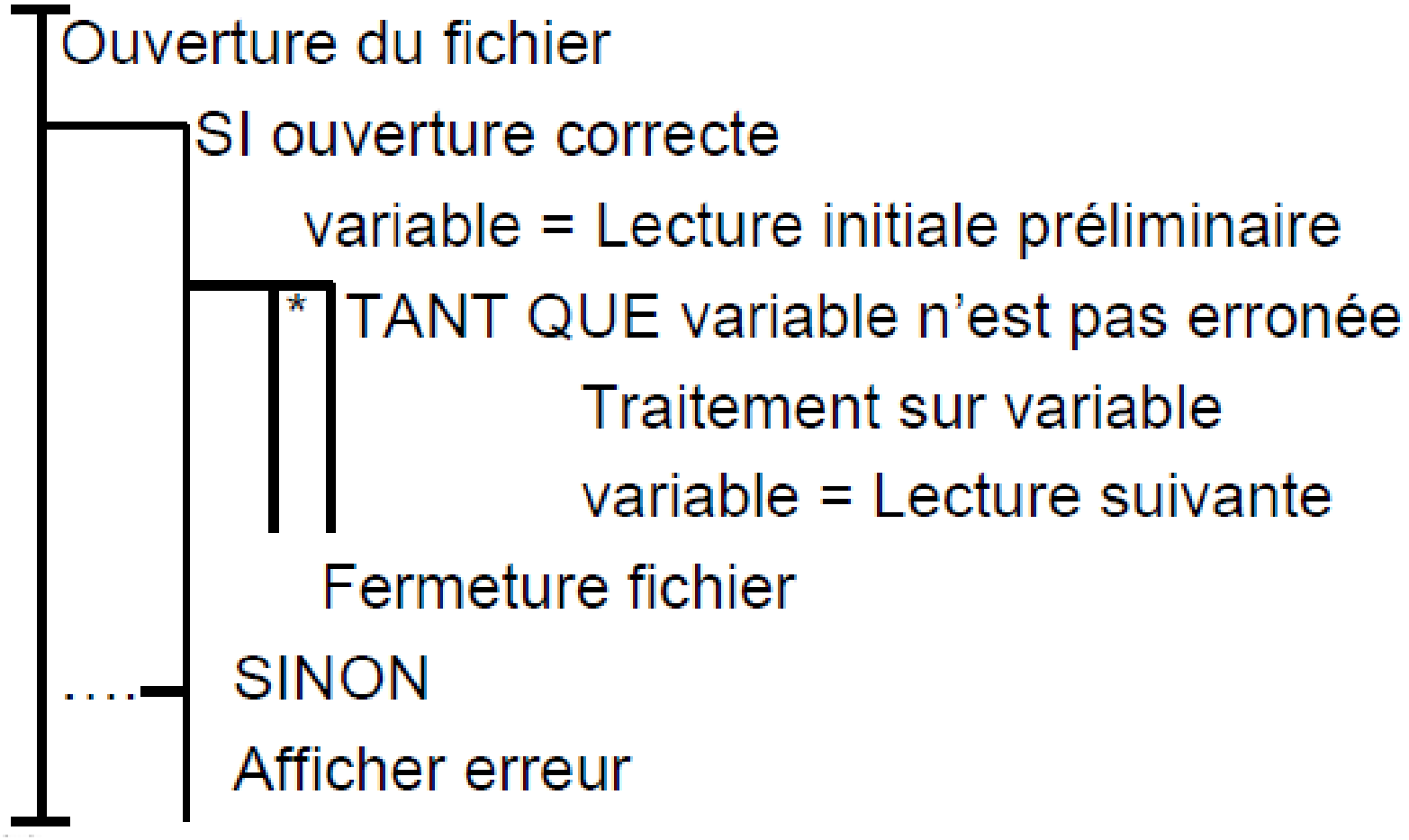
`fgetl()` donne `-1`, lorsqu'il y a une erreur de lecture (utilisation de `ischar()` *);*

`fscanf()` donne `[ ]` ou ' ' selon le format de lecture, lorsqu'il y a une erreur de lecture (utilisation de `isempty()`);

`fread()` donne `[ ],` lorsqu'il y a une erreur de lecture (utilisation de `isempty()` *);*

Une erreur de lecture inclut l'atteinte de la fin du fichier. Donc, il faut utiliser les valeurs de retour de ces fonctions pour détecter la fin d'un fichier dans un `while`.

Révision – Fichiers



Révision – Fichiers texte

Lecture d'un fichier texte: `fscanf()` ou `fgetl()`

- `data=fscanf(fid, format, taille)`
  - `data` = matrice contenant le résultat de la lecture
  - `fid` = le fichier qui a été ouvert avec fopen
  - `format` = chaine de caractères permettant de préciser des modes de conversion.
  - `taille` = argument qui spécifie combien d'éléments seront lus. Peut être un nombre, inf. ou une matrice m x n - [m,n].

    i. e. `nom=fscanf(ficID,'%s',1);`

- `ligne=fgetl(fid);` `%l`ecture d'une ligne complète (jusqu'à '\n').
  - `ligne` sera toujours une chaîne de caractères.
  - le fichier doit absolument contenir des `'\n'`.

Écriture dans un fichier texte:

- `variable=fprintf(fid, format, variable1,….)`
  - `fid` – fichier ouvert. Si `fid` est absent la sortie standard (l'écran) est utilisée
  - `'format'` est une chaine de caractère permettant de préciser le format de la variable i. e. `fprintf(ficID,'%s ',Equipe{i,1})`

Révision – Fichiers binaires

Lecture d'un fichier binaire

```
variable = fread(ID_FIC, taille, 'type')
```

`type` – contrôle le nombre de bits lus pour chaque élément et son interprétation en tant que caractère, entier ou réel.

i.e.
```
taille=fread(ficID,1,'int32');
nom=fread(ficID,taille,'char=>char')';
```

Écriture dans un fichier binaire:

```
compteur=fwrite(fid, data, 'type')
```

`compteur` - donne le nombre d' éléments écrits avec succès

`data` - la matrice a partir de laquelle les données sont écrites

`type` - donne l'interprétation et le nombre d'octets écrits pour chaque élément

i.e.
```
fwrite(ficID,length(Equipe{i,1}),'int32')
fwrite(ficID,Equipe{i,1},'char');
```

Révision – Fichiers binaires

Lecture d'un fichier binaire

```
variable = fread(ID_FIC, taille, 'type')
```

`type` – contrôle le nombre de bits lus pour chaque élément et son interprétation en tant que caractère, entier ou réel.

i.e.
```
taille=fread(ficID,1,'int32');
nom=fread(ficID,taille,'char=>char')';
```

Écriture dans un fichier binaire:

```
compteur=fwrite(fid, data, 'type')
```

`compteur` - donne le nombre d' éléments écrits avec succès

`data` - la matrice a partir de laquelle les données sont écrites

`type` - donne l'interprétation et le nombre d'octets écrits pour chaque élément

i.e.
```
fwrite(ficID,length(Equipe{i,1}),'int32')
fwrite(ficID,Equipe{i,1},'char');
```

Révision – Fichiers binaires

Lecture d'un fichier binaire

```
variable = fread(ID_FIC, taille, 'type')
```

`type` – contrôle le nombre de bits lus pour chaque élément et son interprétation en tant que caractère, entier ou réel.

i.e.
```
taille=fread(ficID,1,'int32');
nom=fread(ficID,taille,'char=>char')';
```

Écriture dans un fichier binaire:

```
compteur=fwrite(fid, data, 'type')
```

`compteur` – donne le nombre d' éléments écrits avec succès

`data` - la matrice à partir de laquelle les données sont écrites

`type` - donne l'interprétation et le nombre d'octets écrits pour chaque élément

i.e.
```
fwrite(ficID,length(Equipe{i,1}),'int32')
fwrite(ficID,Equipe{i,1},'char');
```

Révision – Fichiers binaires

- `fseek` – déplacement dans un fichier binaire
  - retourne `0` ou `-1`

  i.e. `status=fseek(fid, déplacement, origine)`

- `ftell` – donne le nombre d'octets entre le début du fichier et la position courante. La valeur `-1` indique une erreur.

- `feof - iseof=feof(fid)`

retourne `0` ou `1` suivant que la fin du fichier a été atteinte ou non.

Révision – Fichiers

- la fonction `movefile()` peut être utilisée pour changer le nom d'un fichier ou d'un répertoire.
  i.e. `movefile('nom_précédent','nouveau_nom')`
- la fonction `mkdir()` peut être utilisée pour créer un nouveau répertoire.
  i.e. `mkdir('nom_répertoire')`
- la fonction `delete()` peut être utilisée pour détruire un fichier.
  i.e. `delete('nom_fichier')`
- la fonction `rmdir()` peut être utilisée pour détruire un répertoire.
  i.e. `rmdir('nom_répertoire')`

Révision – Graphiques MATLAB

`figure (no_de_figure)`

`subplot(ligne,colonne,position)` – positionnement dans la fenêtre

`plot(X1,Y1,'format_ligne1',X2,Y2, 'format_ligne2',...)` – fonction de traçage

`set(nograph,'nom_propriété1', valeur_propriété1,...)`

- définition des propriétés spécifiques pour le graphique.

`axis([xmin xmax ymin ymax])` – format des axes

`xlabel('titre_axe_des_x') et ylabel('titre_axe_des_y')` – annotation des axes

`title('titre_du graphique_x')` – titre du graphique

`text(x,y,'commentaire',' format_commentaire')` – ajouter des commentaires

`legend('description1','description2',...,position)`

- ajouter une légende

Révision – Graphiques MATLAB

`hold on / off` – ajouter les prochains graphiques créés au graphique déjà existant

`colormap('Type')` – personnaliser la palette des couleurs

`print -format_fichier -options nom_fichier` – exporter le graphique

`loglog(X1,Y1,'format_ligne1',X2,Y2,'format_ligne2',..)` - échelle logarithmique

`semilogx(X1,Y1,'format_ligne1',X2,Y2,'format_ligne2',...)` ou `semilogy(X1,Y1,'format_ligne1',X2,Y2,'format_ligne2',...)` – échelle semi-logarithmique

`plotyy(X1,Y1,X2,Y2,'fonction1','fonction2')` – graphique avec deux axes Y

`bar(x,y,width,'style','couleur'),barh(x,y,width,'style','couleur')` graphique à barres