

Contexte et objectifs

Ce module s'adresse à des étudiants en 2^{ème} année de Sciences de la Terre à paris 6. C'est en général leur 1^{er} contact avec la programmation. Les compétences visées sont

- Maîtrise des structures simples de programmation
- Décomposition d'un problème en une succession d'étapes élémentaires.
- Élaboration de batteries de tests pour étudier le bon comportement de leur code.

Le langage retenu pour cet enseignement est Matlab (et son clone libre octave) parce qu'il est utilisé dans les laboratoires de Sciences de la Terre mais aussi parce qu'il est très facile d'obtenir une sortie graphique lors de traitement de données.

Contexte et objectifs

Par manque de support adapté mais aussi par manque de motivation, le travail personnel des étudiants était très limité. Ce projet avait pour but de permettre aux étudiants de s'entraîner à l'aide d'exercices corrigés (sans forcément donner une solution). Dans la mesure du possible ces exercices devaient utiliser des données aléatoires. WIMS était donc le support idéal pour ce projet.

NB : les travaux présentés seront publiés sous peu.

Les notions abordées

- Variables, affectations, expressions.
- Fonctions : séparation des variables, appel, arguments d'entrée et de sortie
- Structures simples : branchements (if), boucles (for, while)
- Tableaux 1D et 2D

Ces notions sont complétées en 3^{ème} année (fichiers, fonctions vectorielles, graphiques)

Des exercices rapides

Les notions abordées au début sont tout à fait propices à de petits exercices assez rapides mais permettant de vérifier la maîtrise des connaissances des fondamentales.

Exercice.

Parmi les noms ci dessous, cochez celui (ceux) qui peut (peuvent) être utilisé(s) comme nom de variable. Attention, certains noms de variables ne doivent pas être choisis même si ils sont syntaxiquement autorisés.

- grand
- var_1
- min
- a'
- trouvé
- ma moy
- ymin
- v3

Validité des noms de variables

Exercice.

Completez les trous:

```
>> a = 5;
>> b = 5;
>> 2 * b
    b = 
>> b - a
    ans = 
>> a = b + a;
>> 2 * a
    ans = 
>> b + 4
    ans = 
```

Évaluation ou affectation ?

Completez les trous:

```
>> 5 + 2 * 9 - 7
    ans = 
>> sqrt(16 - 7)
    ans = 
>> sqrt(16) - 7
    ans = 
>> 2 + 2 ^ 4 * 3
    ans = 
>>
```

Priorités

Exercices de "déroulements"

Ces exercices sont très utiles pour comprendre le comportement des branchements ("par où passe-t-on?") et des boucles ("Combien de fois passe-t-on ici?", "Quand est ce qu'on s'arrête?"). L'étudiant doit d'abord être à l'aise avec cette technique avant de pouvoir commencer à écrire ses propres programmes.

Exercice.

Grâce à la définition de la fonction `machin` (encadré de droite), complétez le(s) trou(s) dans la succession de lignes de commandes suivantes.

```
>> machin(-1)
ans = 
>> machin(2)
ans = 
>> machin(6)
ans = 
>> machin(7)
ans = 
>> machin(10)
ans = 
>>
```

```
function a = machin(x);
    if x <= 7
        a = 1;
        if x ~= 2
            a = 2;
        end
    else
        a = 3;
    end
```

Exercices spécifiques

- Les expressions booléennes
- Les tirages aléatoires (développement en cours)

Exercice.

Ecrire une expression booléenne qui correspond au tableau de valeurs suivant.

Attention vous devez appeler votre variable `a`.

<code>a</code> appartient à l'intervalle	<code>]-∞;3[</code>	<code>3</code>	<code>]3;4[</code>	<code>4</code>	<code>]4;12[</code>	<code>12</code>	<code>]12;∞[</code>
expression de référence	<code>true</code>	<code>true</code>	<code>false</code>	<code>false</code>	<code>true</code>	<code>true</code>	<code>false</code>

Entrez votre réponse :

Votre expression
(tentative 1/2)

Envoyer la réponse

Écriture de fonctions

Une fois les notions précédentes maîtrisées, les étudiants peuvent s'entraîner à écrire des fonctions respectant un cahier des charges donné dans l'énoncé. L'énoncé comporte aussi un exemple : appel de la fonction et valeurs renvoyées (les valeurs de l'exemple sont aléatoires). L'étudiant doit cette fois décomposer le problème en une succession d'étapes simples, c'est-à-dire construire un algorithme. La difficulté de cette construction est très variable d'un énoncé à l'autre. Ce type d'exercices, contrairement aux précédents, ne varie pas d'une fois sur l'autre.

Écriture de fonctions

Exercice.

Écrire une fonction, `compte`, qui compte le nombre de fois qu'apparaît, dans un tableau 1D (1^{er} argument d'entrée), une certaine valeur `v` (2^{ème} argument d'entrée).

Exemple:

```
>> compte([4,1,2,4,1,3,3,2,1,1,4],4)
ans = 3
```

Entrez votre réponse :

Votre fonction
(tentative 1/10)

```
function n=compte(T,v)
for i=1:length(T)
if T(i)==v
n=n+1;
end
end
```

Envoyer la réponse

Correction des réponses

Toutes les réponses (à l'exception des "exercices rapides") sont évaluées par l'intermédiaire d'appels à octave :

```
wims(exec octave commandes_octaves)
```

Dans le cas de l'écriture d'une fonction la correction se déroule en 2 étapes :

1 Test de structure

- Contient-elle le mot clé `function` ?
- A-t-elle le bon nom ?
- A-t-elle le bon nombre d'arguments (entrée et sortie) ?
- Est elle syntaxiquement correcte ?

- ### 2 Test de comportement : On utilise une batterie de tests prévue par l'auteur de l'exercice. On dispose aussi d'une solution fournie également par l'auteur. Pour chaque test, on compare les valeurs renvoyées par la fonction de l'étudiant avec les valeurs renvoyées par la fonction de l'enseignant. Les résultats sont résumés dans un tableau.

Test de votre fonction par des appels

	Avec la fonction de référence	Avec votre fonction
Test 1 Incorrect	>> compte([3,3,1,3,2,1,3,3,4,3,2],1) ans = 2	>> compte([3,3,1,3,2,1,3,3,4,3,2],1) ans = 1
Test 2 Correct	>> compte([1,2,1,1,1,3,4,1,1,3],2) ans = 1	>> compte([1,2,1,1,1,3,4,1,1,3],2) ans = 1
Test 3 Incorrect	>> compte([1,3,3,4,4,3,1,4,4,4,3,3,4,3],2) ans = 0	all outputs arguments have not been defined

Exemple d'énoncé

```
\text{enonce=
Écrire une fonction, <span id='code'> compte</span>, qui compte le nombre
de fois qu'apparaît, dans un tableau 1D ( $l$ er argument d'entrée),
une certaine valeur <span id='code'>v </span> ( $2^{\text{ème}}$  argument d'entrée.)}
\text{nfonc_etu=compte}
\text{nfonc_ens=compte}
\matrix{fonction_ens =
function n=\nfonc_ens(t,v)
  n=length(find(t==v));
}
\text{vex=randint(0..4)}
\integer{n1=randint(10..15)}
\integer{n2=randint(2..4)}
\text{Aex=slib(myrandwith 1,\n1,4,\n2,\vex)}
\text{Aex = slib(myoctavematrix \Aex)}
\matrix{argex=[\Aex],\vex}
\text{exemple_args=(\argex)}
\text{v1=randint(0..4)}
\integer{n1=randint(10..15)}
\integer{n2=randint(2..4)}
\text{A1=slib(myrandwith 1,\n1,4,\n2,\v1)}
\text{A1 = slib(myoctavematrix \A1)}
\text{v2=randint(0..4)}
\text{A2=slib(myrandwith 1,randint(10..15),4,1,\v2)}
\text{A2 = slib(myoctavematrix \A2)}
\text{v3=randint(0..4)}
\text{A3=slib(myrandwith 1,randint(10..15),4,0,\v3)}
\text{A3 = slib(myoctavematrix \A3)}
\text{tests=(([\A1],\v1) ([\A2],\v2) ([\A3],\v3)}
```