TP n°2 - Boucles while

Notions abordées

- boucle while
- variant et invariant de boucle
- à nouveau : fonctions, affichage et saisie, branchement conditionnel

Placez-vous sur la même machine que la semaine dernière et connectez-vous avec pour identifiant votre nom de famille en minuscules, et pour mot de passe celui que vous avez choisi la semaine dernière. Si vous n'aviez pas rempli la feuille pour indiquer votre place au dernier TP, appelez-moi une fois Debian lancé, mais avant de vous connecter.

En TP, vous devez faire le premier exercice de chaque partie.
Pour lundi prochain (avant 21h), vous me rendrez un autre exercice au choix.

Suites récurrentes

Exercice 1 Suites récurrentes linéaires

Soit $(a, b, c) \in \mathbb{Z}^3$. on considère la suite u définie par $u_0 = c$ et $\forall n \in \mathbb{N}$, $u_{n+1} = a\,u_n + b$.

Question 1 💻
Définissez une fonction `terme_u`, qui prend en argument les paramètres a, b et c ainsi qu'un entier naturel n, et qui renvoie la valeur de u_n. En commentaire vous indiquerez un invariant de boucle qui vous paraît pertinent. Proposez un jeu de tests pour cette fonction.

Question 2 ✍
Si $u0 > 0$, $a > 0$ et $b > 0$, que pouvez-vous dire de la suite u ? Est-elle croissante, décroissante, constante, convergente ou divergente ? Si elle en admet une, quelle est sa limite ?

Question 3 💻
Définissez une fonction `val_u_sup`, qui prend en argument les paramètres a, b et c ainsi qu'un entier M, et qui renvoie le rang du premier terme de la suite u qui dépasse la valeur M. Vous préciserez des hypothèses sur les arguments garantissant qu'un tel rang existe (des conditions suffisantes, je n'attends pas des conditions nécessaires). Proposez un jeu de tests pour cette fonction.

Soit $(a,b,c,d,e)\in\mathbb{Z}^5$. On note v la suite définie par $v_0=d$, $v_1=e$ et $\forall n\in\mathbb{N}$, $v_{n+2}=a\,v_{n+1}+b\,v_n+c$.

Question 4 💻
Définissez une fonction `terme_u`, qui prend en argument les paramètres a,b,c,d et e ainsi qu'un entier naturel n, et qui renvoie la valeur de v_n.
En commentaire vous indiquerez un invariant de boucle qui vous paraît pertinent. Proposez un jeu de tests pour cette fonction.

Question 5 💻
Définissez une fonction `terme_fibo`, qui prend en argument un entier naturel n, et qui renvoie la valeur du n-ième terme de la suite de Fibonacci. Proposez un jeu de tests pour cette fonction.
Cette fonction ne doit pas faire apparaître de boucle while.

Question 6 ✍ (bonus)
À quelle condition les termes de la suite v sont-ils tous non nuls ? Sous ces conditions, que pouvez-vous dire de la suite des ratios $(r_n)_{n\in\mathbb{N}}$ définie par $\forall n\in\mathbb{N}, r_n=\frac{v_{n+1}}{v_n}$? Converge-t-elle ? Si oui, quelle est sa limite ?

Question 7 💻 (bonus)
Définissez une fonction `rang_ratio_cvg`, qui prend en argument les paramètres a,b,c,d,e ainsi qu'un réel ε, et qui renvoie le rang à partir duquel deux termes de r sont distant d'au plus ε. Autrement dit il s'agit de retourner $\min\{n\in\mathbb{N}\,|\,|r_{n+1}-r_n|\leqslant\varepsilon\}$. Proposez un jeu de tests pour cette fonction.

Exercice 2 Suites de Syracuse (récurrentes non linéaires)

Pour $x\in\mathbb{N}^*$, on appelle suite de Syracuse de départ x, la suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$u_0=x \quad\text{et}\quad \forall n\in\mathbb{N},\ u_{n+1}=\begin{cases}\frac{u_n}{2} & \text{si } u_n \text{ est pair}\\ 3u_n+1 & \text{sinon}\end{cases}$$

Question 1
Définissez une fonction `prochain_syracuse` qui à partir d'un entier `a` calcule le terme qui le suivrait dans la suite de Syracuse. Proposez un jeu de tests pour cette fonction.

Question 2
Définissez une fonction `terme_syracuse` qui prend en argument un entier `x` et un entier `n` , et qui calcule le `n`-ième terme de la suite de Syracuse de départ `x`. En commentaire vous indiquerez un invariant de boucle qui vous paraît pertinent. Proposez un jeu de tests pour cette fonction.

Question 3
Définissez une fonction `duree_vol` qui prend en argument un entier `x` et qui retourne le rang du premier terme atteignant 1 dans la suite de Syracuse de départ x.

Question 4
Définissez une fonction `plus_long_vol` qui prend en argument un entier `n` et qui retourne le plus long vol pour les suites de Syracuse de point de départ dans l'intervalle d'entiers [1..`n`].

Un peu d'arithmétique

Le code de ces deux exercices peut être rassemblé dans un même fichier appelé **exo3.c**, ainsi que le code de la question suivante qui pourra vous être utile.

Question 0 💻
Définissez une fonction `racine_entiere` qui prend en argument un entier naturel `a` et qui retourne la partie entière de sa racine carrée, c'est-à-dire $\min\{n\in\mathbb{N}\,|\,n\leqslant\sqrt{a}\}$ ou encore $\min\{\sqrt{c}\,|\,c\in\mathbb{N}\ \texttt{et}\ c^2\leqslant a\}$.

Exercice 3 Test de primalité

Question 1 💻
Définissez une fonction `divise` qui prend en argument deux entiers relatifs `a` et `b` et qui retourne si `a` divise `b`. Proposez un jeu de tests pour cette fonction.

Question 2 💻
Définissez une fonction `est_permier` qui prend en argument un entier naturel `a` et qui retourne si `a` est un nombre premier. En commentaire vous indiquerez un invariant de boucle qui vous paraît pertinent. Proposez un jeu de tests pour cette fonction.

Question 3 ✍
Combien de comparaisons, respectivement de divisions (reste ou quotient), effectue la fonction `est_premier` que vous avez proposée (en fonction de la valeur de l'argument) ? Pouvez-vous réduire ces nombres ?
Soyez prêt à justifier votre réponse, et appelez-moi pour me proposer votre solution.

Exercice 4 Test de co-primalité

Question 1 💻
Définissez une fonction `co_divise` qui prend en argument trois entiers relatifs `a`, `b` et `c`, et qui retourne si a divise à la fois b et c. Votre fonction devra s'épargner de tester si a divise c dans le cas où a ne divise pas b. Vérifiez cela par quelques tests avec affichage supplémentaire. Après le test, vous laisserez ces instructions d'affichage en commentaire.

Question 2 💻
Définissez une fonction `premiers_entre_eux` qui prend en argument trois entiers relatifs `a` et `b` et qui retourne si ces entiers sont premiers entre eux, c'est-à-dire s'ils n'ont aucun diviseur commun autre que 1. En vous aidant de l'exercice précédent, vous essayerez de minimiser le nombre de comparaisons et de divisions faites par votre fonction. En commentaire vous indiquerez un invariant de boucle qui vous paraît pertinent. Proposez un jeu de tests pour cette fonction.

Question 3 ✍
Connaissez-vous un autre algorithme pour décider si deux entiers sont premiers entre eux ?

Pour s'amuser avec l'affichage

Exercice 5 Escaliers descendant vers la droite

On souhaite ici afficher un escalier à n marches. Chaque marche commence à gauche de l'écran et doit aller plus loin (*i.e.* plus à droite) que la marche précédente (*i.e.* celle du dessus), ainsi on obtient des escaliers descendant vers la droite.

Escalier à 3 marches :

```
____
|__|____
|__||__|____
|__||__||__|___
```

Escalier à 8 marches :

```
____
|__|____
|__||__|____
|__||__||__|____
|__||__||__||__|____
|__||__||__||__||__|____
|__||__||__||__||__||__|____
|__||__||__||__||__||__||__|____
|__||__||__||__||__||__||__||__|____
```

Question 1
Créez un programme qui qui affiche l'escalier à `n` marches où `n` est un entier naturel saisi par l'utilisateur.

Exercice 6 Château de briques

On souhaite ici afficher un genre de château de briques à n étages. Chaque ligne représente un étage, et une brique est représentée par la chaîne `!--!`.
Chaque étage est centré sur l'étage du dessous, et présente une brique de moins. L'étage supérieur n'a qu'une seule brique. Les briques sont écartées de manière à laisser un vide entre elles, mais ce vide ne doit pas être trop grand sinon on ne pourrait pas poser de brique dessus.
Voici quelques exemples de l'affichage de ces châteaux.

Le château à 5 étages :

```
        !--!
      !--!  !--!
    !--!  !--!  !--!
  !--!  !--!  !--!  !--!
!--!  !--!  !--!  !--!  !--!
```

Le château à 3 étages :

```
    !--!
  !--!  !--!
!--!  !--!  !--!
```

Le château à 2 étages :

```
  !--!
!--!  !--!
```

À l'inverse de la convention habituelle, on numérote ici les étages de haut en bas, et à partir de 1. Ainsi l'étage 1 correspond à la ligne à afficher en premier, et à la ligne numérotée 1 dans les exemples ci-dessus.

Question 1
Définissez une fonction `affiche_ligne` qui affiche l'étage i d'un château à n étages.

Question 2

Définissez une fonction `affiche_chateau` qui prend en argument un entier naturel `n` et qui affiche un château à `n` étages. Évidemment cette fonction fera appel à la fonction `affiche_ligne`.

Question 3 ✍
Combien de briques a un château à n étages ? Justifiez. En déduire quelle est la hauteur maximale d'un château qu'on peut construire avec 99 briques.

Question 4 🖳 bonus
Créez des fonctions `affiche_ligne_bis` et `affiche_chateau_bis` qui permettent d'inscrire dans les briques l'ordre dans lequel elles ont été posées, en considérant qu'on les pose étage par étage en partant du bas, et de gauche à droite sur chaque étage. Afin d'avoir des numéros qui tiennent sur 2 chiffres, on se limitera à des châteaux ayant moins de 99 briques.

Le château à 5 étages :

```
        !15!
      !13!  !14!
    !10!  !11!  !12!
  !_6!  !_7!  !_8!  !_9!
!_1!  !_2!  !_3!  !_4!  !_5!
```

Le château à 3 étages :

```
    !_6!
  !_4!  !_5!
!_1!  !_2!  !_3!
```

Le château à 2 étages :

```
  !_3!
!_1!  !_2!
```

Il pourra être utile de créer une fonction `brique` qui prend en argument un entier k et affiche la brique numérotée k. Par exemple,

- `brique(1)` affiche `!_1!`
- `brique(2)` affiche `!_2!`
- `brique(12)` affiche `!12!`

Exercice 7 Podium (bonus)

Question 1
En vous inspirant de ce qui a été fait dans les exercices précédents, créez des fonctions permettant d'afficher un podium à n étages.

Plutôt que des mots, voici les podiums à 3 et 4 marches :

```
    _____
____|_1_|
|_2_|___|____
|___|___|_3_|
```

```
        _____
    ____|_1_|
    |_2_|___|____
____|___|___|_3_|
|_4_|___|___|___|
```

Pour préparer le prochain cours

Exercice 8 Choix du type de la variable de boucle

Question 1

Téléchargez le fichier `exo8.c` sur cahier de prépa. Ce fichier contient le programme suivant. Compilez-le puis exécutez-le. L'exécution semble-t-elle normale? Le code vous paraît-il correct ?

```c
#include <stdio.h>

void repete ( char* mot, int n){
//hyp : n>= 0
//affiche n fois la chaine mot suivie d'un retour à la ligne
  char i = 0;
  while(i < n){
    printf("%s\n",mot);
    i=i+1;
  }
}

int main(){
  repete ("MP2I",0);
  printf("---\n");
  repete ("MP2I",2);
  printf("---\n");
  repete ("MP2I",4);
}
```

Question 2

Complétez le programme pour tester la fonction avec l'argument $n = 127$ puis $n = 128$. Compilez-le puis exécutez-le. L'exécution semble-t-elle normale? Le code vous paraît-il correct ? NB : l'exécution d'un programme lancé dans le terminal peut-être interrompue en tapant Ctrl+C.

Question 3

Changez le type de la variable de boucle pour une variable de type `int`. Et refaites les tests précédents. Que constatez-vous ? Proposez de nouveaux tests montrant les limites de ce nouveau code.

Exercice 9 Utiliser le codage binaire pour un tour de magie

Cet exercice est prévu pour être fait après une démonstration dudit tour de magie en classe. À défaut, si vous voulez faire cet exercice avant la démonstration, vous pouvez lire une description rapide de ce tour sur `http://perso.eleves.ens-rennes.fr/~afalq494/fds.html`. On cherche à implémenter une interface qui simule ce tour.

Question 1

Quel paramètre ou quels paramètres faut-il fixer avant de commencer le tour ?
Demandez-vous ce qui doit être fixé pour générer les cartes.

Question 2

Quelles informations doivent être récupérées auprès de la personne à qui on fait le tour ?

Question 3 ✍
Quelles données doivent être calculées par le programme pour faire ce tour ?

Question 4 ✍
Proposer un découpage en fonctions intermédiaires pour l'implémentation de cette interface.