esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

TP n°1 LAB1413

Introduction au langage C

Nicolas Bodin
Arnaud Bannier
Matthieu Le Berre

Ce premier TP vous permet de créer votre premier *vrai* programme. Le but est donc de réfléchir sur un sujet en particulier, construire la solution dans sa tête avant de la coucher sur papier puis de la rédiger en C. Il ne s'agit pas encore d'un TP avec une seule grosse question mais on y viendra ! Chaque exercice nécessite un nouveau programme, donc un nouveau fichier C.

1. Quelques exercices d'introduction

1.1. Rendu de monnaie

Pour le moment, nous allons avoir besoin de bases afin de réaliser correctement les exercices suivants. Tout d'abord, implémentez un programme qui calcule le rendu de monnaie. Pour cela, votre programme doit donner un montant à payer à l'utilisateur (prix de l'objet) et l'utilisateur doit entrer le montant qu'il doit. Le programme répond alors la monnaie à rendre tel que décrit dans l'exemple ci-dessous.

```
bodin@linux:~$ ./monnaie
Montant a payer: 18,75 euros
Vous donnez: 20 euros
Rendu: 1,25 euros
bodin@linux:~$
```

1.2. Série harmonique

Cet exercice consiste à calculer le n-ième terme de la série harmonique définie par $u_n = \sum_{i=1}^{n} \frac{1}{n}$. Votre programme doit donc :

1. demander à l'utilisateur `Rang de la série harmonique?`, et récupérer la valeur saisie dans une variable `n`,
2. calculer le n-ième de la série harmonique
3. afficher à l'écran le résultat du calcul.

1.3. Récupération du plus grand et plus petit élément d'une suite d'entiers

Vous devez créer un programme qui demande à l'utilisateur de saisir 10 entiers, et suite à cette saisie, vous devez afficher le plus grand et le plus petit élément de cette suite. Bien évidemment, il vous est interdit d'utiliser plus de 5 variables.

1.4. Je pense à ...

Dans cet exercice, le but pour l'utilisateur est de trouver à quel nombre pense la machine (entre 1 et 5 pour ne pas faire trop difficile). Attention, il faut limiter le nombre de tentatives à 3 par exemple. Pour cela, voici les étapes à suivre :

1. Générez un nombre aléatoire entre 1 et 5 et stockez-le dans une variable `nombre`. Pour cela, utilisez les fonctions `srand()` et `rand()`.
2. Dans une boucle sur 3 tours, Affichez le message `A quel nombre je pense?` puis récupérez la saisie de l'utilisateur.
   — Si c'est le nombre généré par l'ordinateur, affichez `Gagné!` puis quittez le programme.
   — Sinon affichez le message `Raté` et poursuivez la boucle.
3. Si les trois tentatives ont été utilisées, affichez le nombre recherché et quittez le programme.

2. Suite de Syracuse

La suite de Syracuse[1] est définie pour des entiers strictement positifs. Malgré son apparente simplicité, elle soulève de nombreux problèmes de prévision de comportement (comme par exemple la conjecture de Syracuse qui stipule que la suite de Syracuse de n'importe quel nombre entier naturel atteint 1). Le but de cet exercice est d'étudier quelques propriétés de cette afin d'infirmer ou confirmer cette hypothèse pour certaines valeurs.

La suite de Syracuse est définie par :

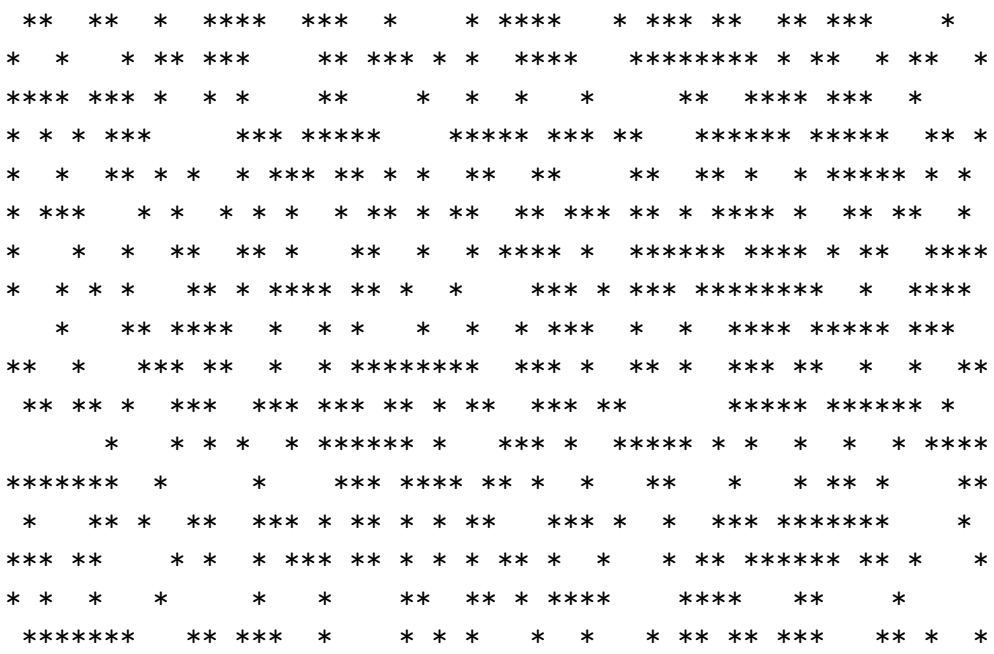
$$\begin{cases} u_0 & = c \\ u_{n+1} & = \begin{cases} u_n/2 \text{ si } u_n \text{ est pair,} \\ 3u_n + 1 \text{ sinon.} \end{cases} \end{cases}$$

Vous devez :
— demander à l'utilisateur la valeur c permettant de définir u_0.
— calculer puis afficher n-ième terme de la suite en ayant demandé à l'utilisateur la valeur n.
— afficher le temps de vol de la suite pour la valeur $u_0 = c$ (plus petit n tel que $u_n = 1$).
— afficher l'altitude maximale de la suite pour la valeur $u_0 = c$ ($\max(u_k), 0 \leq k \leq n$).

1. `http://fr.wikipedia.org/wiki/Conjecture_de_Syracuse`

3. Stars Matrix Like

On se propose dans cet exercice de faire défiler sur la console des lignes d'étoiles aléatoirement remplies du bas vers le haut comme illustré ci-après :



Pour cela, faire un programme qui :

1. fasse un boucle infinie,
2. dans cette boucle infinie pour chaque ligne de largeur `NbCol`, afficher aléatoirement pour chaque colonne soit le caractère `*`, soit le caractère espace.

Petite variante : sur nos machines actuelles, l'affichage des lignes est très rapide. Si vous souhaitez ralentir l'affichage des lignes, il faut pour cela ralentir l'exécution du programme. Vous pouvez utiliser la fonction `usleep`. Voir le `man` pour plus d'explications.

4. Pour les plus rapides

4.1. Le juste prix

Vous connaissez sûrement le jeu de la vitrine dans le Juste Prix : un candidat doit trouver le montant exact de la vitrine sachant que ce dernier est entre 10000 et 50000 euros. Le candidat dispose de 45 secondes et à chaque proposition de prix, l'animateur répond `plus` ou `moins`. En utilisant l'exercice 1.4, implémentez le jeu du juste prix en utilisant la bibliothèque `time.h` pour la gestion du temps et la fonction `time()`.

Vous pouvez également inverser les rôles, c'est à dire, l'utilisateur pense à un nombre et l'ordinateur tente de deviner lequel. Pour accélérer la réponse de l'utilisateur vous pouvez spécifier comme réponse uniquement '`+`' et '`-`'.

4.2. Rendu de monnaie (bis)

Un peu plus difficile que le simple rendu de monnaie, votre programme va devoir également calculer le nombre de chaque pièce et de chaque billet à rendre. Par exemple pour rendre 1,83 euros, vous devez afficher :

- 1 x 1 euros
- 1 x 50 cents
- 1 x 20 cents
- 1 x 10 cents
- 1 x 2 cents
- 1 x 1 cents

Pour simplifier vous pouvez commencer par utiliser uniquement les valeurs 100, 10, 1 euros et 10 et 1 centimes puis ajouter les autres valeurs par la suite.