info719 : Rappels d'algorithmique et programmation C
TDP 0 : premiers pas en C

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/
wiki : http://www.lama.univ-savoie.fr/wiki

Partie 1 : comment ça marche ?

Remarque : des détails complémentaires sont disponibles sur le wiki. Allez les consulter.

Question 0. Pour commencer, il faut redémarrer votre ordinateur pour utiliser le système d'exploitation Linux.

Si vous n'avez jamais utilisé Linux, passez quelques minutes pour explorer l'interface, changer votre fond d'écran ou aller vérifier vos emails.

Question 1. Pour écrire un programme C, il faut utiliser un *éditeur de texte*. Je vous conseille gedit. (N'utilisez pas OpenOffice ou un autre traitement de texte !)

Créez un nouveau fichier appelé `hello.c` contenant

```
/* mon premier programme C */
#include <stdio.h>  /* pour la fonction printf */
int main () {
  printf("Salut...\n")
}
```

Ce fichier est un fichier *texte*. Il faut maintenant transformer ce fichier en un fichier exécutable compréhensible par l'ordinateur. Pour ça, il faut utiliser un *compilateur*. Le compilateur utilisé s'appelle `gcc` ("GNU Compiler Collection").

Ouvrez un terminal et tapez la commande suivante* :

```
> gcc -Wall hello.c -o hello
```

Cette commande se décompose comme suit :

- `gcc` pour invoquer le compilateur
- `-Wall` pour demander d'avoir tous les messages d'erreur, même ceux qui ne sont pas très importants (`-W` comme "warning" et `all` comme "all")
- `hello.c` le nom du fichier que vous voulez compiler
- `-o hello` le nom du fichier exécutable que vous voulez créer. (`-o` comme "output")

Qu'en pensez-vous ? Est-ce ça marche ? Pourquoi ?

Question 2. En suivant les messages de gcc, modifiez le fichier pour obtenir un programme correct.

Question 3. Pour lancer le fichier exécutable, utilisez la commande suivante

```
> ./hello
```

Que se passe-t'il ?

* le "`> `" indique le début de la ligne, et vous ne devez pas le taper

Partie 2 : retour au TD1

Nous allons maintenant reprendre le TD1. Lorsque vous écrivez une fonction, il faut la tester en l'appelant dans la fonction principale (`main`). Par exemple, pour la question 1 :

```
#include <stdio.h>

int bissextile(int a){
  ...
}

int main(){
  int a,b;
  printf("Quelle année ?\n    ");
  scanf("%i",&a);
  b = bissextile(a);
  if (1==b) { printf("L'année %i est bissextile.\n", a);}
  if (0==b) { printf("L'année %i n'est pas bissextile.\n", a);}
  if (-1==b) { printf("L'année était négative !.\n");}
  return(1);
}
```

Les détails concernant la fonction `scanf` seront vu un peu plus tard. Pour le moment, il suffit de savoir que "`scanf("%i",&a) ;`" permet de lire une valeur entière au clavier et de la mettre dans la variable "`a`".

Pour tester les fonctions sur un tableau, vous pouvez déclarer et initialiser le tableau avec

```
int T[10] = { 1 , 2 , 3 , 3 , 5 , -1 , 9 , 42 , 0 , 666 };
```

Question 1. D'après Wikipedia,

> *Depuis l'instauration du calendrier grégorien, sont bissextiles les années :*
> - *divisibles par 4 mais non divisibles par 100*
> - *ou divisibles par 400.*
>
> *...*
>
> *Le calendrier julien, qui avait cours avant le calendrier actuel, ne distinguait pas les fins de siècles (années divisibles par 100). Une année était bissextile tous les 4 ans, sans autre exception.)*

En sachant que le calendrier grégorien a commencé en 1582, écrivez une petite fonction pour décider si une année est bissextile ou non.

Question 2. On suppose que `T` est un tableau de taille `N` > 1 contenant des entiers (`int`) ; écrivez les fonctions suivantes :

- recherche du minimum sur `T`,
- recherche d'un minimum et d'un maximum, et échange de ces deux valeurs dans le tableau,
- recherche simultanée des deux valeurs les plus grandes du tableau,
- recherche des deux valeurs *distinctes* les plus grandes.

Dans le dernier cas, il faut en particulier traiter le cas où toutes les valeurs dans le tableau sont identiques.

Question 3. Écrivez une fonction qui renverse l'ordre des éléments d'un tableau sans le recopier.

Question 4. Écrivez un test pour vérifiez si un tableau est un *palindrome*. Si c'est le cas, il devra renvoyer la valeur `0` ; sinon, il devra renvoyer la plus grande longueur du préfixe qu'on retrouve inversé en suffixe du tableau.

Question 5. On suppose que `T` est un tableau de taille `N` contenant des entiers triés par ordre croissant. Écrivez une fonction qui va vérifier si un entier `n` apparaît dans le tableau.

Votre algorithme doit être le plus efficace possible.