TP1: Premiers programmes en C

ENS-Rennes

Objectifs pédagogiques:

- Savoir éditer et compiler un programme C;
- Connaître la syntaxe de base du C (variables, fonctions, if, for, while);
- Fonctions `printf` et `scanf`;
- Savoir utiliser des tableaux de taille fixe;

★ **Exercice 1: Savoir lire et écrire: la fonction max3.**

Écrivez un programme C qui calcule le maximum de 3 entiers saisis au clavier par l'utilisateur, et qui affiche le résultat. Compilez-le sous le nom `max3`, et testez-le. Quels tests faut-il faire pour s'assurer qu'il fonctionne?

★ **Exercice 2: Tableaux multidimensionnels de caractères: Cadavres exquis.**

L'objectif de cet exercice est de constituer un écrivain automatique. Pour cela, il faut initialiser plusieurs tableaux de chaînes de caractères: un tableau de sujets accompagnés d'un article, un tableau de verbes conjugués à la troisième personne, un tableau de compléments d'objets directs au masculin et un tableau d'adjectifs au masculin.

Ce jeu fut inventé par les surréalistes vers 1954. La première phrase construite sur ce modèle fut «Le cadavre exquis boira le vin nouveau», ce qui donna le nom du jeu.

Sujet	*Adjectif*	*Verbe*	*Complément*	*Adjectif*
Le cadavre	exquis	boira	le vin	nouveau
Le chat	noir	mange	un rat	gris
Le chien	pouilleux	aime	un os	moelleux
La grenouille	gluant	capture	un moustique	agressif

▷ **Question 1:** Le premier problème à résoudre est de savoir comment stocker les morceaux de phrases qui seront combinés ensuite. Le plus simple est d'utiliser des tableaux statiques de chaînes statiques. Vous avez deux possibilités, présentées dans les deux blocs suivants. Choisissez l'une d'entre elles, et écrivez le début de votre programme.

Première approche pour le stockage des données

```
char *nom[]={"le cadavre","le chat","le chien","la grenouille"};
char *adj1[]={"exquis","noir","pouilleux","gluant"};
...
```

Deuxieme approche pour le stockage des données

```
char *elements[5][4]={
  {"le cadavre","le chat","le chien","la grenouille"},
  {"exquis","noir","pouilleux","gluant"},
  ...
  };
```

▷ **Question 2:** Complétez votre programme.

Obtenir un nombre aléatoire en C n'est pas très compliqué. Il faut utiliser la fonction `rand()` qui retourne un entier entre 0 et `RAND_MAX`. Le problème est que la suite pseudo-aléatoire est toujours la même par défaut. Initialiser cette suite à l'heure actuelle est une bonne solution à ce manque de variété.

```
#include <stdlib.h>
#include <time.h>

int main() {
  // Initialise le pseudo aleatoire
  srand(time(NULL));
  // tire un nombre entre 0 et 3
  int nb = rand() % 4;
}
```

★ **Exercice 3: Manipuler des entiers: le triangle de Pascal.**

On souhaite écrire un programme qui affiche le triangle de Pascal de hauteur N, dont les éléments du triangle de Pascal se calculent avec la formule suivante: $P_{i,j} = P_{i-1,j} + P_{i-1,j-1}$

Affichage quand N vaut 6

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10 5  1
1  6  15 20 15 6  1
```

Affichage quand N vaut 7

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10 5  1
1  6  15 20 15 6  1
1  7  21 35 35 21 7  1
```

▷ **Question 1:** Écrivez ce programme. Dans un premier temps, vous écrirez simplement la fonction `int pascal(int i, int j)` et vous calculerez chacune des valeurs de cette façon.

L'approche choisie est relativement inefficace: Afficher le triangle de taille 100 est par exemple très très lent. Le problème vient du fait que de nombreux calculs sont effectués plusieurs fois. Calculer $P_{5,3}$ demande par exemple de calculer $P_{3,2}$ à deux reprises (une fois pour avoir $P_{4,2}$ et une fois pour avoir $P_{4,3}$). Pour résoudre ce problème, nous allons utiliser un tableau pour stocker les calculs déjà faits afin d'éviter de recalculer les choses.

▷ **Question 2:** Ajoutez un tableau d'entiers en global à votre programme en écrivant les lignes suivantes en dehors de toute fonction. Elles créent un tableau à deux entrées de taille 100×100, et on peut augmenter la taille en changeant la valeur de la macro.

```
1 #define MAX 100
2 int tab[MAX][MAX];
```

Ajoutez une double boucle au début de votre fonction `main()` pour remplir le tableau de 0.

▷ **Question 3:** Modifiez votre fonction `pascal()`. Avant de faire le moindre calcul, vous consulterez la case correspondante du tableau. Si cette case contient une valeur non-nulle, la fonction retournera directement cette valeur sans la recalculer. Si la case contient 0, la fonction calculera le résultat comme avant, et stockera le résultat dans la case avant de le retourner en résultat. Comparez la vitesse d'affichage du triangle pour $N = 100$.

▷ **Question 4: (optionnelle)** Il est possible d'optimiser la quantité de mémoire nécessaire en ne stockant qu'une seule ligne du tableau. Il faut alors faire les calculs dans le bon ordre (c'est-à-dire de droite à gauche) pour ne pas effacer des cases dont des calculs futurs auront besoin.

★ Exercice 4: Tableaux: Pendu en 7 coups.

On souhaite écrire un programme C permettant de jouer au pendu, dont un exemple de partie est donné ci-contre.
Une fois que le mot est choisi, l'ordinateur affiche une une série de lignes blanches pour que le joueur ne puisse pas lire le mot, il affiche ensuite une série d'étoiles qui correspond au nombre de lettres du mot à trouver. Le joueur propose des caractères jusqu'à ce qu'il ait trouvé le mot, ou qu'il ait perdu (i.e. son nombre de coups ratés est égal à 7). À chaque fois que le joueur propose un caractère l'ordinateur affiche le mot avec des * et les caractères déjà trouvés.

```
*******
caractere ? E
*******
caractere ? O
*O**O**
caractere ? L
*O**O**
caractere ? N
*ON*O**
caractere ? U
*ON*OU*
caractere ? R
*ON*OUR
caractere ? B
BON*OUR
caractere ? J
BONJOUR
GAGNÉ en 8
```

▷ **Question 1:** Implémentez une première variante de ce jeu où un humain tape le mot à chercher avant que la partie ne commence.

▷ **Question 2: (optionnelle)** Permettez de jouer contre l'ordinateur, qui choisira un mot au hasard dans la liste `/usr/share/dict/words` (du paquet wfrench) ou de la liste `http://www.pallier.org/ressources/dicofr/liste.de.mots.francais.frgut.txt`
Vous aurez besoin du type `wchar_t` pour les accents, comme expliqué sur Internet.

★ Exercice 5: Boucles et conditionnelles: Sapin de noël en ASCII art (optionel).

▷ **Question 1:** Réalisez un programme C qui demande la hauteur de l'arbre avant de le dessiner comme la version 1 ci-contre. Le tronc doit également être proportionnel à la hauteur.

▷ **Question 2: (optionnelle)** Ajoutez des guirlandes, en respectant les règles suivantes:

- On pose des guirlandes une ligne sur deux, en commençant par la seconde;
- On écarte les guirlandes de deux étoiles;
- Si une guirlande ne tient pas sur une ligne, il faut la replier sur la ligne du dessous en commençant par la droite.

```
Nb lignes ? 8

       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************
      |||
      |||
 version 1
```

```
Nb lignes ? 8
Guirlande? 0-0

       *
      0-0
     *****
    0-0**0-
   ********0
  0-0**0-0**0
 ***********0-
0-0**0-0**0-0**
      |||
      |||
 version 2
```

★ Exercice 6: Nombres en virgule flottante: Racines de Newton (optionel).

La suite $x_{n+1} = \frac{1}{2}(x_n + \frac{y}{x_n})$ avec $x_0 = 1$ converge vers $\sqrt{y}$.

▷ **Question 1:** Écrivez un programme qui calcule $\sqrt{2}$ en utilisant cette formule.

▷ **Question 2:** Combien d'itérations sont nécessaires pour obtenir un résultat avec 10 chiffres significatifs?

▷ **Question 3:** Écrire un programme C qui détermine la précision de l'ordinateur (la précision est telle que $1.0 + precision = 1.0$).