TP Programmation C/C++
Les fonctions

© 2017 *tv* <tvaira@free.fr> v.1.1

Sommaire

Les objectifs de ce tp sont de comprendre et mettre en oeuvre la notion essentielle de fonction. Certains exercices sont extraits du site www.france-ioi.org.

Les critères d'évaluation de ce TP sont :

- le minimum attendu : on doit pouvoir fabriquer un exécutable et le lancer !
- le programme doit énoncer ce qu'il fait et faire ce qu'il énonce !
- le respect des noms de fichiers
- le nommage des fonctions et des variables, l'utilisation de constantes
- une fonction doit énoncer ce qu'elle fait et faire seulement ce qu'elle énonce !
- les fonctions ne dépassent pas 15 lignes
- le code est fonctionnel
- la simplicité du code

Exercices : Les joies du code

C'est le soir et la bibliothèque se retrouve fermée. Alors que vous prenez votre repas, vous êtes abordé par des adolescents qui sont intéressés par votre robot. Amateurs de jeux, de codes et d'énigmes, ils souhaiteraient vous voir réaliser quelques programmes. Vous profitez de l'occasion pour avancer dans votre manuel.

☞ Objectif : vous allez découvrir comment la notion de fonction permet de découper le code en différentes parties et de structurer un long programme de manière très efficace et agréable :

- Motif rectangulaire : page 2
- Banquet municipal : page 2
- Journée des cadeaux : page 6
- Jour de marché : page 7
- Recensement : page 7

Motif rectangulaire

À présent, vos spectateurs ont envie que votre robot imprime un motif sur une feuille rectangulaire (dont la taille est de 80 colonnes et 43 lignes maximum), car cela leur sert pour des jeux de géométrie.

☞ Ce que doit faire votre programme :

Votre programme doit lire le nombre de lignes et de colonnes de la feuille, puis le motif à afficher sous la forme d'un caractère. Il doit alors afficher le motif de sorte qu'il remplisse chaque cellule de la feuille. Vous allez créer et utiliser une fonction `afficheMotif` pour cela.

☞ Exemples :

```
$ ./motif-rectangulaire
4
9
F

FFFFFFFFF
FFFFFFFFF
FFFFFFFFF
FFFFFFFFF

$ ./motif-rectangulaire
8
3
P

PPP
PPP
PPP
PPP
PPP
PPP
PPP
PPP
```

Question 1. Écrire le programme `motif-rectangulaire.c`. Tester et valider.

Banquet municipal

Vous commencez à être connu pour vos talents de programmeur, aussi lors de votre passage dans la ville d'Incerto, le maire décide de vous inviter au grand banquet qu'il organise. Le maire se charge lui-même de faire le plan de table mais il change toujours d'avis et les serveurs doivent constamment changer de place les petites affiches sur lesquelles sont indiqués les noms des personnes.

Afin de l'aider, vous lui proposez d'utiliser votre robot pour déterminer la position de chaque personne après tous les changements décidés par le maire. Afin de simplifier le problème, on suppose qu'il n'y a qu'une seule très grande table circulaire.

☞ Ce que doit faire votre programme :

Votre programme lit tout d'abord le nombre total de positions sur la table (au maximum 1000). Il lit ensuite, pour chaque place, le nom de la personne et la position (numérotée à partir de 1) à laquelle elle doit s'installer. Si la position indiquée est déjà occupée, il faudra lui faire une place en décalant les

personnes d'un rang. Une fois toutes les personnes placées, on affiche pour chaque position le nom de la personne qui s'y trouve.

Orthonart, l'un des élèves de la classe, vous a demandé de lui apprendre à utiliser votre petit robot. Après lui avoir expliqué les bases de la programmation, vous lui prêtez le robot afin qu'il améliore le programme que vous avez déjà écrit en le décomposant en fonctions. On limitera la taille de des fonctions à 10 lignes maximum (accolades exclues).

```
// Le programme lit tout d'abord le nombre total de positions sur la table (au maximum 1000)
   . Il lit ensuite, pour chaque place, le nom de la personne et la position (numérotés à
   partir de 1) à laquelle elle doit s'installer. Si la position indiquée est déjà occupée
   par une personne, il faudra décaler les personnes d'une place. On affiche, pour chaque
   position, le nom de la personne qui s'y trouve une fois tous les changements faits.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_POSITIONS 1000
#define MAX_NOM        32

int main()
{
    int nbPositions;
    char nom[MAX_NOM+1];
    int numero;
    int position, positionLibre, positionDepart, positionDestination;
    int i, n, c;

    // saisir le nombre de positions sur la table
    do
    {
        scanf("%d", &nbPositions);
    }
    while(nbPositions > MAX_POSITIONS);

    // vider le tampon
    while ((c = getchar()) != '\n' && c != EOF);

    char *planTable[nbPositions];

    // initialiser le plan de table
    for(i=0;i<nbPositions;i++)
    {
        planTable[i] = (char *)malloc((MAX_NOM+1) * sizeof(char));
        strcpy(planTable[i], ""); // place libre
    }

    // remplir le plan de table
    for(i=0;i<nbPositions;i++)
    {
        // saisir le placement d'une personne
        scanf("%32s%d", &nom[0], &numero);
```



```
        // vider le tampon
        while ((c = getchar()) != '\n' && c != EOF);

        // la place est libre ?
        if(strlen(planTable[numero-1]) == 0)
        {
            // positionner la personne
            strcpy(planTable[numero-1], nom);
        }
        else
        {
            // rechercher la première place de libre
            positionLibre = -1;
            position = numero % nbPositions;
            do
            {
                if(strlen(planTable[position]) == 0)
                {
                    positionLibre = position;
                    break;
                }
                position = (position+1) % nbPositions;
            }
            while(positionLibre == -1);

            // calculer le nombre de décalage à effectuer
            int decalage = ((positionLibre+1-numero) % nbPositions);
            if(decalage < 0)
                decalage = nbPositions + decalage;

            // décaler la ou les personne(s)
            positionDestination = positionLibre;
            for(n=0;n<decalage;n++)
            {
                positionDepart = positionDestination - 1;
                if(positionDepart < 0) positionDepart = nbPositions - 1;
                strcpy(planTable[positionDestination], planTable[positionDepart]);
                positionDestination = (positionDestination-1) % nbPositions;
                if(positionDestination < 0) positionDestination = nbPositions - 1;
            }
            // positionner la personne
            strcpy(planTable[numero-1], nom);
        }
    }

    // afficher le plan de table
    for(i=0;i<nbPositions;i++)
    {
        printf("%d %s\n", i+1, planTable[i]);
    }

    // libèrer le plan de table
    for(i=0;i<nbPositions;i++)
```



```
    {
        free(planTable[i]);
    }

    return 0;
}
```

banquet-municipal.c

☞ Exemples :

```
$ ./banquet-municipal
4
robert 1
roger 2
raymond 3
alphonse 4

1 robert
2 roger
3 raymond
4 alphonse

$ ./banquet-municipal
4
robert 2
roger 2
raymond 2
alphonse 2

1 robert
2 alphonse
3 raymond
4 roger

$ ./banquet-municipal
4
robert 2
roger 1
raymond 2
alphonse 1

1 alphonse
2 roger
3 raymond
4 robert

$ ./banquet-municipal
4
robert 4
roger 4
raymond 1
alphonse 1

1 alphonse
```



```
2 raymond
3 robert
4 roger
```

Question 2. Modifier le programme `banquet-municipal.c`. Tester et valider.

Question 3. Est-ce que la majorité des commentaires sont-ils toujours utiles ?

Journée des cadeaux

Dans la ville de Largition, la tradition veut que, une fois par mois, les habitants les plus riches fassent un cadeau aux habitants les moins riches. Cela permet de maintenir une bonne ambiance malgré les inévitables différences de richesses.

Un habitant est considéré comme riche si sa fortune est plus grande que celle de la moitié de la population. Calculer, chaque mois, qui est riche ou pas est un long travail, aussi lorsque le maire a entendu parlé de vos talents, il vous a demandé votre aide.

☞ Ce que doit faire votre programme :

Il devra lire un premier entier, le nombre d'habitants (au plus 1000) puis, pour chaque habitant il devra lire sa fortune, un entier. Il devra calculer puis afficher une valeur (la **médiane**) permettant de dire facilement si une personne est riche ou pas, simplement en regardant si la fortune de cette personne est plus grande ou plus petite que cette valeur.

Deux cas peuvent se présenter :

– Si le nombre d'habitants est impair, par exemple si leurs fortunes sont 10, 5, 12, 8, 3 alors la valeur recherchée est 8. En effet, il y aura alors 2 personnes "riches" (10 et 12), 2 "moins riches" (3 et 5) et 1 juste au milieu (8) qui ne donnera ni recevra de cadeau.
– Si le nombre d'habitants est pair, par exemple si leurs fortunes sont 10, 5, 12, 8, 3, 9 alors la valeur recherchée est entre 8 et 9. Il y a en effet 3 personnes "riches" (9, 10 et 12) et 3 "moins riches" (3, 5 et 8). Par convention on prendra la valeur 8.5, c'est-à-dire la moyenne de 8 et 9.

✍ *La bibliothèque C standard offre une fonction de tri : `qsort()`. Elle sert à trier un tableau d'éléments à l'aide d'une fonction de comparaison à fournir. Voir le cours sur les tableaux.*

☞ Exemples :

```
$ ./journee-cadeaux
5
10
5
12
8
3

8

$ ./journee-cadeaux
6
10
5
12
8
3
```



```
9

8.5
```

Question 4. Écrire le programme `journee-cadeaux.c`.

Jour de marché

Lorsqu'on organise un marché, certains emplacements sont beaucoup plus intéressants que d'autres. Afin d'éviter les tentations de « pots de vin », la ville dans laquelle vous venez d'arriver a décidé qu'on organiserait à chaque fois un tirage au sort, chacun ayant sa chance !

Chaque marchand met donc son nom dans un grand panier, et des tirages sont effectués. On tire d'abord le nom du marchand qui aura le premier emplacement, puis celui qui aura le second, et ainsi de suite. On décide alors d'afficher par ordre alphabétique, les noms des marchands avec le numéro de leur emplacement.

☞ Ce que doit faire votre programme :

Votre programme devra lire le nombre d'emplacements `nbEmplacements` (au maximum 1000), puis pour chaque emplacement à partir de `0`, le nom du marchand à qui est attribué l'emplacement. Vous allouerez dynamiquement la mémoire pour stocker le nom du marchand.

Ensuite, votre programme devra afficher la liste par ordre alphabétique (cf. `qsort()`) des emplacements attribués : le nom du marchand et le numéro de l'emplacement qui lui est attribué numéroté à partir de `1`.

☞ Exemples :

```
$ ./jour-marche
4
robert
roger
raymond
alphonse

alphonse 4
raymond 3
robert 1
roger 2
```

Question 5. Écrire le programme `jour-marche.c`.

Recensement

Lors de votre passage dans la ville d'Incerto, le maire décide de faire un recensement de la population. Pour cela, il faut se déplacer chez chaque habitant et lui demander son nom et son âge. Au dernier recensement, il y avait 1000 habitants mais toutes les données ont été perdues et la population a augmenté depuis. Afin de l'aider, vous lui proposez d'utiliser votre robot pour effectuer la collecte des données.

☞ Ce que doit faire votre programme :

Votre programme devra lire une suite d'entiers positifs (l'âge d'un habitant) et afficher les statistiques suivantes : le nombre habitants, la moyenne d'âge et la liste des âges par ordre décroissant (du plus âgé

au moins âgé). On ne sait pas combien il y aura d'habitants, et le recensement se terminera par la saisie de la valeur -1 (qui n'est pas un âge, juste un marqueur de fin).

Afin de simplifier le problème, on suppose qu'il y avait seulement 10 habitants au dernier recensement. Vous allouerez dynamiquement la mémoire pour stocker les données de la collecte par tranche de 10 habitants.

☞ Exemples :

```
$ ./recensement
10
51
5
66
23
27
78
1
12
38
40
-1

11
31.9
78
66
51
40
38
27
23
12
10
5
1
```

Question 6. Écrire le programme `recensement.c`.

Bilan

Conclusion : il est indispensable de décomposer un traitement de grande taille en plusieurs parties plus petites jusqu'à obtenir quelque chose de suffisamment simple à comprendre et à résoudre. Cette approche consistant à décomposer une tâche complexe en une suite d'étapes plus petites (et donc plus facile à gérer) ne s'applique pas uniquement à la programmation et aux ordinateurs. Elle est courante et utile dans la plupart des domaines de l'existence.

Descartes (mathématicien, physicien et philosophe français) dans le *Discours de la méthode* :

> « *diviser chacune des difficultés que j'examinerais, en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux résoudre.* »

D'un point de vue théorique, vous pouvez désormais faire tout ce qu'on peut faire avec un ordinateur, le reste n'est que détails ! Cela montre tout de même la valeur des "détails" et l'importance des compétences pratiques parce qu'il est clair que vous débutez à peine votre carrière de programmeur.

