CS-119(a) – ICC-C Série 14

2024-05-27

Important Les exercices 1-3 portent sur le backtracking. Les exercices 4-6 sont des exercices de révision, jetez-y un coup d'oeil aussi !

Exo1 Les permutations

Utilisez le code vu en cours pour écrire un programme qui lit un entier `N` et affiche toutes les permutations de taille `N` à l'écran.

Exo2 Yoda-speak

On nous donne un fichier avec une phrase découpée en séquences de mots, avec un bout de phrase par ligne. Chaque bout de phrase occupe moins de 30 caractères et il y a au plus 10 lignes contenant du texte dans le ficher. La dernière ligne est vide. On aimerait reconstituer la phrase de toutes les manières possibles. Par exemple pour l'entrée

```
que tu aies
il faut
patience
mon jeune Padawan
```

on aimerait obtenir les `4! = 24` phrases suivantes (dans n'importe quel ordre) :

```
patience il faut que tu aies mon jeune Padawan
patience il faut mon jeune Padawan que tu aies
patience que tu aies il faut mon jeune Padawan
patience que tu aies mon jeune Padawan il faut
patience mon jeune Padawan il faut que tu aies
patience mon jeune Padawan que tu aies il faut
il faut patience que tu aies mon jeune Padawan
il faut patience mon jeune Padawan que tu aies
```

```
il faut que tu aies patience mon jeune Padawan
il faut que tu aies mon jeune Padawan patience
il faut mon jeune Padawan patience que tu aies
il faut mon jeune Padawan que tu aies patience
que tu aies patience il faut mon jeune Padawan
que tu aies patience mon jeune Padawan il faut
que tu aies il faut patience mon jeune Padawan
que tu aies il faut mon jeune Padawan patience
que tu aies mon jeune Padawan patience il faut
que tu aies mon jeune Padawan il faut patience
mon jeune Padawan patience il faut que tu aies
mon jeune Padawan patience que tu aies il faut
mon jeune Padawan il faut patience que tu aies
mon jeune Padawan il faut que tu aies patience
mon jeune Padawan que tu aies patience il faut
mon jeune Padawan que tu aies il faut patience
```

Lisez ligne par ligne depuis le fichier avec la fonction `fgets`, et n'oubliez pas d'enlever le dernier caractère `'\n'` de chaque ligne. Rappel : quand le fichier est terminé la fonction `fgets` retourne `NULL`.

Ecrivez les phrases reconstituées dans le fichier `yoda.txt`

Exo3 Les N reines

On lit un entier `N > 1`. Sur un échiquier de taille `N x N` vous devez placer `N` reines de telle façon qu'aucune reine n'attaque aucune autre. Si c'est possible, alors affichez un échiquier possible avec des `'.'` aux emplacements vides et des `'*'` aux emplacements correspondant aux positions des reines. Sinon affichez le mot `"Impossible"`.

Par exemple, pour `N = 5` une réponse possible est

```
* . . . .
. . * . .
. . . . *
. * . . .
. . . * .
```

Indices Il existe des solutions pour tous les nombres naturels, sauf 2 et 3.

Les reines doivent être placées sur des lignes, des colonnes, et des diagonales différentes. Vous pouvez donc décrire la position des reines par un tableau unidimensionnel `col` de taille `N` qui est tel que la `i`-ème reine est placée à la ligne `i` et à la colonne `col[i]`. On peut construire ce tableau commençant par la position `0` avec la méthode "backtracking".

En explorant la `j`-ème position de ce tableau (rempli correctement jusqu'à la position `j-1`) vous pouvez vérifier que la nouvelle reine sur la ligne `j` n'attaque aucune des reines placées aux positions `(1, col[1]), ..., (j-1, col[j-1])`.

Exo4 Stars

Qu'affiche ce code ?

```
    int v[6] = {5, 3, 0, 4, 2, 1};
    for (int *p = v; *p; p = v + *p)
    {
        printf("%d ", *p);
    }
    printf("\n");
```

Exo5 Bug mystère

Votre collègue a écrit le code suivant pour lister les `N` premiers nombres pairs positifs. Il a rencontré un problème en testant ce code et vous demande votre avis.

```
#include <stdio.h>

int* quelques_nombres_pairs(int n)
{
    int tableau[n];
    for (int i=0; i<n; i++)
    {
        tableau[i] = 2*i;
    }
    return tableau;
}

int main()
{
    int combien;
    scanf("%d", &combien);
    int *nombres_pairs = quelques_nombres_pairs(combien);

    for (int i=0; i<combien; i++)
    {
        printf("%d ", nombres_pairs[i]);
    }
    printf("\n");
}
```

Malheureusement, pour l'entrée `25` on voit s'afficher

```
0 32760 0 0 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48
```

Pourquoi ? Comment pourrait-on corriger ce code ?

Exo6 Odd

Souvenez-vous de la cellule d'une liste chaînée :

```
typedef struct _cell_t
{
    int contenu;
    struct _cell_t *next;
} noeud_t;
```

Qu'arrive-t-il à la liste `1 -> 2 -> 4 -> 4 -> 3 -> -5 -> -2` si on appelle la fonction suivante avec le premier élément de la liste en argument ?

```
void pr(cell_t *pcell)
{
    if (pcell == NULL || pcell->next == NULL)
    {
        return;
    }

    if (pcell->next->contenu % 2)
    {
        pr(pcell->next);
    }
    else
    {
        cell_t *a = pcell->next;
        pcell->next = pcell->next->next;
        free(a);

        pr(pcell);
    }
}
```