CS-119(a) – ICC-C Série 5

2024-03-19

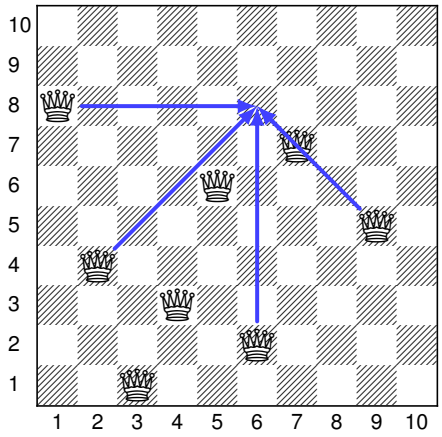
Rappel Pour faire lire des valeurs d'un fichier texte à votre programme il suffit d'utiliser la redirection de l'entrée :

```
./exo < fichier.txt
```

Votre code ne saura pas qu'il lit depuis un fichier. Utilisez tout simplement `scanf` sans intercaler des messages pour l'utilisateur avec `printf`. Affichez seulement le résultat à la fin du programme.

Exo1 M'lady

Sur un échiquier de taille `N x N` on place `k` reines. Pour une position donnée, il faut indiquer combien de reines attaquent cette position. Si deux reines sont alignées avec la position, on compte les deux. (Une reine attaque toutes les cases qui sont sur sa ligne, sur sa colonne, et sur les deux diagonales qui se croisent à sa position.)



Dans cet exemple, on considère un échiquier de taille 10×10 et 8 reines. Selon les règles, la position au croisement de la ligne 8 et de la colonne 6 est attaquée par 5 reines : les reines aux positions `(8, 1)`, `(4, 2)`, `(2, 6)`, `(7, 7)`, `(5, 9)`.

L'entrée contient d'abord l'entier `N` qui donne la taille de l'échiquier. Ensuite il y a deux entiers `r` et `c` (entre `1` et `N`) représentant la ligne et la colonne de la position qui nous intéresse. Ensuite vient l'entier `k` (le nombre de reines) suivi par `k` paires d'entiers indiquant les positions des `k` reines.

Pour l'exemple ci-dessus :

```
10
8 6
8
8 1
4 2
1 3
3 4
6 5
2 6
7 7
5 9
```

Dans ce cas la sortie sera `5`.

Solution de l'exercice 1

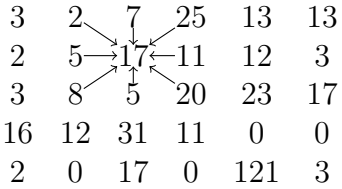
```c
#include <stdio.h>

int main()
{
    int N, n_queens, row, col;
    scanf("%d", &N);
    scanf("%d %d", &row, &col);
    scanf("%d", &n_queens);
    int n_attacks = 0;
    for (int i = 0; i < n_queens; i++)
    {
        int q_row, q_col;
        scanf("%d %d", &q_row, &q_col);
        if (q_row == row ||                // meme ligne
            q_col == col ||                // meme colonne
            q_row - q_col == row - col || // diagonale
            q_row + q_col == row + col)   // anti-diagonale
        {
            n_attacks++;
        }
    }
    printf("%d\n", n_attacks);
}
```

Exo2 Oil

Votre oncle américain Joe s'est retrouvé en charge de l'exploitation pétrolière d'une parcelle rectangulaire de terrain quelque part dans une région désertique. Pour des raisons de sécurité il ne peut pas vous divulguer l'endroit précis, mais il peut vous fournir un plan avec les mesures des gisements. Ces mesures viennent sous la forme d'une matrice avec `M` lignes et `N` colonnes. Joe peut installer son équipement dans une des cellules de la matrice, et il arrivera à extraire la quantité de la cellule et des cellules voisines (gauche, droite, haut, bas, et aussi en diagonale).

Par exemple, si le plan de taille de 5×6 est celui donné ci-dessous, et que Joe choisit la cellule $(2, 3)$, alors il pourra extraire une quantité de 100 unités.



Ce n'est visiblement pas le meilleur endroit, car s'il choisissait la cellule $(4, 4)$ il obtiendrait plus que le double !

Joe a besoin de votre aide pour trouver le meilleur endroit pour placer son équipement, c'est à dire l'endroit qui lui permet d'extraire le plus de pétrole.

On vous donne d'abord les valeurs de `M` et de `N` (entre 3 et 100), suivies de `M` lignes contenant chacune `N` valeurs entières qui correspondent aux mesures de Joe.

Pour l'exemple ci-dessus :

```
5 6
3 2 7 25 13 13
2 5 17 11 12 3
3 8 5 20 23 17
16 12 31 11 0 0
2 0 17 0 121 3
```

Vous devez afficher le meilleur emplacement de l'équipement d'extraction sous la forme de deux entiers `r` et `c` suivis de la quantité de pétrole qui peut être extraite. Les coordonnées sont telles que $1 \leq \texttt{r} \leq M$ et $1 \leq \texttt{c} \leq N$, car Joe n'aime pas la programmation en C, et pour lui les indices commencent à 1.

Pour l'exemple ci-dessus, on devrait voir la sortie `4 4 228`.

Si vous voulez écrire d'autres fonctions que `main`, vous pouvez utiliser une variable globale pour stocker le plan.

Solution de l'exercice 2

```c
#include <stdio.h>

int M, N, map[100][100];

int oil(int i, int j)
{
    if (i < 0 || i >= M || j < 0 || j >= N)
    {
        // Hors de la grille
        return 0;
    }
    return map[i][j];
}

int main()
{
    scanf("%d %d", &M, &N);

    int i, j, r, c;
    int best = 0;
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            scanf("%d", &map[i][j]);

    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)
        {
            int potential = 0;
            for (int k = -1; k <= 1; k++)
                for (int l = -1; l <= 1; l++)
                    potential += oil(i + k, j + l);

            if (potential > best)
            {
                best = potential;
                r = i;
                c = j;
            }
        }
    }
    printf("%d %d %d\n", r+1, c+1, best);
}
```

Exo3 Occurrences

On lit depuis l'entrée standard deux chaînes de caractères sans espaces. On aimerait savoir combien de fois on retrouve la première chaîne dans la deuxième.

Par exemple le mot `"est"` apparaît une seule fois dans le mot `"estuaire"`. Il apparaît aussi une seule fois dans le mot `"vestibule"`.

Attention, les occurrences peuvent se chevaucher ! Par exemple le mot `"aba"` apparaît trois fois dans `"abababa"` (ce n'est pas toujours des vrai mots qu'on nous fournit) : **aba**baba, ab**aba**ba, abab**aba**.

Indice : il vous faudra utiliser le code que vous avez écrit la fois passée pour déterminer la longueur d'une chaîne. Définissez la fonction `strlen_icc` qui prend une chaîne et sa longueur max, et retourne un entier représentant sa vraie longueur. Elle devrait se conformer à la déclaration suivante :

```c
int strlen_icc(const char string[], int taille_max);
```

Solution de l'exercice 3

```c
#include <stdio.h>

const int longueur_max = 100;

int strlen_icc(const char string[], int taille_max)
{
    int longueur;
    for (longueur = 0; longueur <= taille_max; longueur++)
        if (string[longueur] == '\0')
            return longueur;
    return taille_max;
}

int compter_occurrences(const char cle[], const char str[])
{
    int longueur_cle = strlen_icc(cle, longueur_max);
    int longueur_str = strlen_icc(str, longueur_max);

    int occurrences = 0;
    for (int start = 0;
         start + longueur_cle <= longueur_str;
         start++)
    {
        int ok = 1;
        for (int i = 0; i < longueur_cle; i++)
        {
```

```
27              if (cle[i] != str[start + i])
28              {
29                  ok = 0; // ne convient pas
30                  break;
31              }
32          }
33          if (ok)
34          {
35              occurrences++;
36          }
37      }
38      return occurrences;
39  }
40
41  int main()
42  {
43      char cle[longueur_max + 1], str[longueur_max + 1];
44
45      printf("Entrez une chaîne de caractères : ");
46      scanf("%100s", cle);
47
48      printf("Entrez une chaîne de caractères : ");
49      scanf("%100s", str);
50
51      printf("%d\n", compter_occurrences(cle, str));
52  }
```

Exo4 Chercher l'aiguille dans une botte de foin

Dans un fichier texte on nous donne sur la première ligne un mot-clé (sans espaces) d'au plus 30 caractères. A partir de la ligne suivante commence un texte qui pourrait être assez long sur plusieurs lignes. Une ligne contient au plus 100 caractères, sans compter le caractère de fin de ligne. Pour lire le fichier, utilisez la redirection de l'entrée standard.

Déterminez et affichez le nombre d'occurrences du mot-clé donné dans ce texte.

Comme dans l'exercice précédent, il pourrait apparaître comme faisant partie d'un autre mot.

Dans ce programme n'utilisez que des appels à `fgets`, plutôt que `scanf`.

Souvenez-vous que la fonction `fgets(chaine, longueur_max, stdin)` lit une ligne entière de texte dans la variable `chaine` de taille au plus `longueur_max`, y compris le dernier caractère de fin de ligne `'\n'`. Pour vérifier que le texte est fini, inspectez simplement la valeur de retour de `fgets`. Si elle est nulle, c'est que le texte est fini.

Si vous insistez à rentrer le texte manuellement dans le terminal pendant l'exécution du programme sans utiliser un fichier texte (pourquoi ?...), tapez simplement `Ctrl-D` à la fin du texte pour indiquer sa fin.

Indices :

— Il faut éliminer le caractère `'\n'` du mot-clé.

— Si on sait trouver les occurrences dans une ligne, il suffit de répéter pour chaque ligne et de tout sommer pour avoir la réponse finale.

— Utilisez le code de l'exercice précédent - définissez une fonction qui prend deux chaînes et qui compte le nombre d'occurrences de la première dans la deuxième.

Dans ce poème de Lewis Carroll le mot "the" apparaît 15 fois. Il y a 4 occurrences supplémentaires avec "T" majuscule : "The". Vous pouvez les compter aussi si vous utilisez judicieusement la fonction `tolower()` qui se trouve dans `ctype.h`. Cette fonction prend un caractère et le retourne en minuscule.

```
the
Jabberwocky
BY LEWIS CARROLL

'Twas brillig, and the slithy toves
      Did gyre and gimble in the wabe:
All mimsy were the borogoves,
      And the mome raths outgrabe.

“Beware the Jabberwock, my son!
      The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
      The frumious Bandersnatch!”

He took his vorpal sword in hand;
      Long time the manxome foe he sought—
So rested he by the Tumtum tree
      And stood awhile in thought.

And, as in uffish thought he stood,
      The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
      And burbled as it came!

One, two! One, two! And through and through
      The vorpal blade went snicker-snack!
```

```
He left it dead, and with its head
      He went galumphing back.

“And hast thou slain the Jabberwock?
      Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!”
      He chortled in his joy.

’Twas brillig, and the slithy toves
      Did gyre and gimble in the wabe:
All mimsy were the borogoves,
      And the mome raths outgrabe.
```

Solution de l’exercice 4

```c
#include <stdio.h>

const int longueur_max = 101;

int compter_occurrences(const char cle[], const char str[])
{
  ...
        for (int i = 0; i < longueur_cle; i++)
        {
            if (tolower(cle[i]) != tolower(str[start + i]))
            {
                ok = 0; // ne convient pas
                break;
            }
        }
  ...
}

int main()
{
    char cle[longueur_max+1], ligne[longueur_max+1];
    fgets(cle, longueur_max, stdin);
    int longueur_cle = strlen_icc(cle, longueur_max);
    if (cle[longueur_cle-1] == '\n')
    {
        // Enlevons le caractere fin de ligne
        cle[longueur_cle-1] = '\0';
    }
```

```
29      int occurrences = 0;
30      while(fgets(ligne, longueur_max, stdin))
31      {
32          occurrences += compter_occurrences(cle, ligne);
33      }
34      printf("Occurrences: %d\n", occurrences);
35  }
```