Série 6: Fonction (2)

Récursivité, surcharge, valeur par défaut, coût calcul

Lien avec le MOOC Initiation à la Programmation (en C++)
Lien avec ICC-Théorie en complément du MOOC

Les éléments du MOOC sur les fonctions sont répartis sur deux semaines. Cette seconde partie se concentre sur la conception de fonctions récursives.

Le premier exercice complémentaire porte sur la mesure du coût calcul effectif pour une exécution donnée d'un exécutable. Le second exercice complémentaire sert à illustrer la **redirection** des entrées-sorties détaillée à la fin du document dans le complément-projet-2.

Exercices partiels semaine4 du MOOC

- **Document Tutoriel 2ième partie seulement « Somme récursive »**
  - Écriture d'une fonction récursive : ne pas oublier le critère d'arrêt

- **Document Exercices semaine 4 du MOOC : 2ième sélection**

  - **Exercice 14 : nombres de Fibonacci (niveau 1)**
    - Permet de comparer les performances de l'approche récursive avec une solution itérative. Cet exercice est ré-utilisé pour l'exercice complémentaire ExC 1 sur la mesure du temps de calcul en p 2.

- **Document Exercices additionnels semaine 4 du MOOC : 2ième sélection**

  - **Exercice 11 : Recherche dichotomique (niveau 2)**
    - Essayez les deux variantes suivantes de structure de contrôle pour le test du caractère alphanumérique :
      - L'approche avec une cascade de **`if- else if –`**
      - Utilisation d'un **switch** sur le caractère lu au clavier car le type **char** est automatiquement converti en **int** dans l'évaluation d'une expression en C++.

Exercices Complémentaires (ExC)

ExC 1 : Obtenir le temps d'exécution d'un programme avec la commande `time`
Le rendu du projet vous demandera de fournir des temps d'exécution dans le rapport. Pour cela il faut utiliser la commande **`time`** en mode ligne de commande. Par exemple avec l'exécutable **`prog`** écrivez la commande suivante dans le terminal (il faut être dans le répertoire où se trouve l'exécutable) :

```
time ./prog
```

La commande vous affiche 3 durées ; c'est la durée **user** qui nous intéresse car c'est le temps utilisé par le CPU pour exécuter votre code. La durée **sys** est celle des fonctions du système (ex : entrées-sorties) tandis que la durée **real** est la durée totale d'exécution (d'autres programmes peuvent se partager le CPU pendant cette durée, c'est pourquoi elle est généralement supérieure à la somme des deux autres durées).

Action : comparer les temps d'exécutions des exercices précédents en faisant varier la taille N du problème traité. En déduire l'ordre de complexité de votre solution quand N devient grand.

ExC 2 : Codage de César / manipulation du code des caractères alphanumériques
Jules César codait ses messages secrets en décalant les lettres dans l'alphabet d'un nombre fixe de lettres. Nous allons faire une version de ce codage en décalant le code ASCII des caractères d'un texte de **n** caractères dans l'alphabet. Le même programme peut servir pour coder et décoder un message. Pour le décodage il suffit de recoder le message déjà codé mais cette fois en indiquant un décalage de **-n**.

On va se restreindre à des messages contenant des lettres MAJUSCULES, des espaces, le point séparant des phrases et le passage à la ligne. il ne faudra pas coder l'espace ' ', le point '.' et le passage à la ligne '\n' ; ces caractères doivent être récrits en sortie sans transformation.

Travail à effectuer : le programme doit d'abord lire la valeur du décalage = un entier entre -13 et + 13.

Ensuite le programme lit le texte fourni en entrée caractère par caractère avec **cin** et affiche aussitôt avec **cout** chaque caractère codé avec le décalage (ou pas). La détection de tout autre caractère que ceux qui sont prévus agit comme un signal de fin de message et termine le programme.

Remarque sur la lecture des caractères sans filtrage des séparateurs :
Si **c** est une variable de type **char**, alors l'instruction **cin >> c ;** filtre les séparateurs, ce qui ne permet pas de résoudre le problème correctement. Pour pouvoir lire TOUS les caractères, y compris les séparateurs, il faut utiliser l'instruction suivante : **cin.get(c) ;**

Codage : pour le codage, le programme doit veiller à reboucler le code à l'autre bout de l'alphabet en cas de dépassement du début ou de la fin de l'alphabet. Il est demandé de ne pas utiliser les valeurs numériques brutes du code ASCII dans les tests effectués. Travaillez seulement avec les constantes littérales de type char, par exemple **'Z'**.

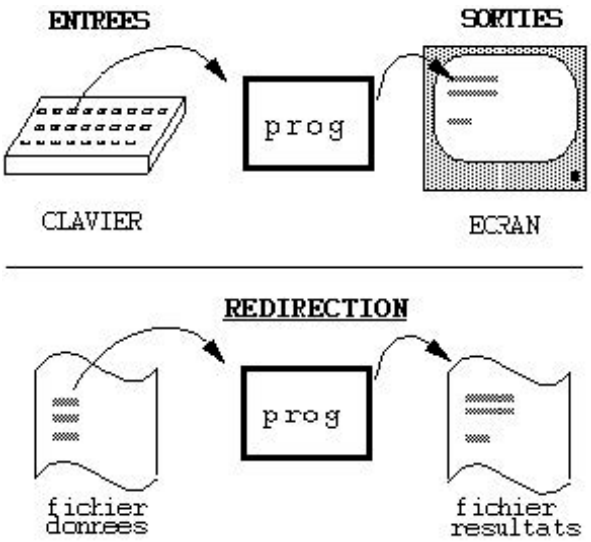
Exemple d'exécution : on donne d'abord 1 comme valeur de décalage puis on entre le texte HAL suivi par Enter ce qui va lancer le codage de la première ligne et afficher IBM. Ensuite on peut quitter le programme en tapant une lettre minuscule puis Enter.

Mise en œuvre du principe d'abstraction : mettez en œuvre une fonction responsable du codage en plus de main. Elle renvoie VRAI si le caractère passé par référence a pu être codé et FAUX s'il faut terminer le programme.

Question subsidiaire : trouver un codage de César intéressant pour GOOGLE

Complément en prévision du Projet (2)

Redirection de l'entrée (clavier) et de la sortie (terminal) par défaut.



Dans les exercices précédents, les données en entrée du programme étaient toujours lues au clavier, et les sorties toujours écrites à l'écran. C'est bien mais on peut faire beaucoup mieux sans changer une seule ligne de notre code source…

En particulier la phase de mise au point d'un projet demande de faire beaucoup de tests différents pour s'assurer que le programme fait ce qu'il doit faire. Entrer manuellement toutes ces données au clavier prend énormément de temps.

La ***redirection de l'entrée*** permet de prendre le contenu d'un fichier texte comme si cela venait du clavier. La méthode de travail est donc d'écrire une fois pour toutes vos fichiers de tests avec geany puis de rediriger le fichier de test qui vous intéresse sur l'entrée par défaut de votre programme exécutable. Supposons qu'il s'appelle **prog** et que les données à rediriger vers l'entrée sont dans le fichier **donnees.txt** ; il suffit de taper la commande suivante dans une fenêtre Terminal :

./prog < donnees.txt

Inversement, la sortie affichée dans le terminal est parfois très longue ; il est plus pratique de la rediriger vers un fichier puis d'ouvrir ce fichier avec geany pour l'examiner en détail. Si vous voulez ***rediriger la sortie*** du programme **prog** vers un fichier **resultats.txt**, il suffit de taper :

./prog > resultats.txt

Vous pouvez aussi combiner ces deux redirections: ./**prog < donnees.txt > resultats.txt**

Remarque: en utilisant le symbole >> au lieu de >, vous ajoutez les résultats à la fin d'un fichier déjà existant.

Travail à effectuer : écrivez un petit fichier texte de test avec geany destiné à votre programme de codage de César (ExC 2). Ce fichier doit commencer par l'entier indiquant le décalage, puis respecter les règles indiquées pour le message et enfin finir avec un caractère qui produit la fin du programme.

Ensuite, depuis une fenêtre terminal, redirigez ce fichier texte vers le programme du codage secret et redirigez la sortie vers un autre fichier texte.

Éditez-le pour indiquer l'opposé du décalage précédent en début de fichier. Recommencez le codage et vérifiez si vous obtenez bien le message original.