AG43

Bases de l'informatique à l'usage de l'ingénieur mécanicien

Examen Médian

Automne 2018

Durée : 1h30

Exercice 1 : Centrage cinématographique

On peut voir au cours des fins de films, défiler sur l'écran les noms et prénoms des collaborateurs. Le positionnement sur l'écran de ces chaînes de caractères est particulier. En effet, l'espace (code ASCII 32) séparant le prénom du nom est placé au centre de la ligne sur l'écran, alors que les traitements de texte font correspondre le milieu de la chaîne de caractères à centrer au centre de la ligne.

Exemple de générique :

⇓ Centre de l'écran

Centrage classique :

ALAIN PROVIST
FELIX LEUCHA
CLAIRE DELUNE
ALBERT MUDA

Centrage "Cinématographique" :

ALAIN PROVIST
FELIX LECHA
CLAIRE DELUNE
ALBERT MUDA

On demande d'écrire une procédure effectuant un centrage "Cinématographique" sur une ligne de 80 caractères. Cette procédure nommée **`pcentrage(texte)`** admet comme argument **`texte`**, une chaîne de 80 caractères maximum, sans compter le [NUL] de fin de chaîne. Au départ, l'argument texte contient la chaîne à "centrer" terminée du caractère [NUL]. Par insertion d'espaces en début de chaîne, on fait coïncider l'espace séparateur au caractère d'indice 40 de **`texte`**. On conserve bien entendu l'intégrité de la chaîne de caractères, et l'indiçage est effectué par rapport à 0.

Les prénoms composés comportent des tirets, et dans le cas des noms à particule, le premier espace est pris comme référence. On ne fera pas de traitement des erreurs possibles. La fonction `strlen(chaine)` qui calcule la longueur d'une chaîne de caractères est disponible.

Exercice 2 : Ascenseur

On se propose d'écrire un programme de gestion d'un ascenseur pour un immeuble de N étages.

Si l'on observe les commandes dans une cabine d'ascenseur, on peut voir les boutons suivants :

- étages (un bouton par étage)
- ouverture porte (maintient la porte ouverte tant qu'il est sollicité, pas utilisé dans notre étude)
- arrêt d'urgence (il s'agit d'un interrupteur qui reste enfoncé et qu'il faut déverrouiller après sollicitation si on veut repartir. Dans ce cas, on attend un appel pour un étage donné pour repartir, pas utilisé dans notre étude)

Fonctionnement désiré :

L'ordinateur reçoit les commandes de deux endroits :

- de l'intérieur de la cabine,
- de l'extérieur (appel à un porte)

Les commandes issues de l'intérieur sont prioritaires.

Supposons l'ascenseur au repos, aucune commande n'est mémorisée. Il se trouve à l'étage x, porte ouverte pour évacuer les odeurs. Une personne appelle la cabine depuis une porte donnée. Ces portes sont repérées par les numéros 0 à N (0 = rez-de-chaussée). L'ascenseur se rend alors directement à l'étage désiré. Cette opération est pilotée par la fonction `aller(etage)`.

Question 1 : Déplacement vers un étage

Ecrire la procédure `aller(etage)` effectuant la gestion du déplacement de l'ascenseur vers l'étage demandé.

Pour écrire cette procédure, on dispose :

- de la variable situation, qui est une variable globale mise à jour automatiquement à chaque passage d'un inter-étages. Le programmeur ne peut que se contenter de lire son contenu pour savoir à chaque instant l'étage où se situe la cabine,
- des fonctions suivantes :
  - `moteur(sens)` : mise en route moteur en montée (sens = 1), en descente (sens = 0)
  - `arret()` : gère la phase de décélération et d'arrêt en face de l'étage courant (pas de paramètre en entrée)
  - `porte(etat)` : ouverture et fermeture porte de l'ascenseur, on ne sort de cette fonction que lorsque la manoeuvre s'est correctement déroulée (etat = 1 pour ouverture, 0 pour fermeture)

Pour faciliter l'écriture de l'algorithme, on ne procède pas à la gestion des incidents pouvant survenir dans chaque fonction, comme par exemple le cas de la porte bloquée.

Question 2 : Lecture des commandes mémorisées

Les commandes de déplacement sont, comme déjà dit, de deux types :

- les appels extérieurs,
- les demandes de déplacement issues de la cabine.

Ces commandes sont enregistrées automatiquement par une fonction du type interruption et sont placées dans leurs tableaux respectifs dans l'ordre de leur apparition. Le premier tableau appelé cc[] contient les commandes issues de la cabine, le second, appelé ce[], les commandes extérieures. La première commande entrée sera la première traitée. Chaque tableau peut contenir au maximum 9 commandes.

Lorsqu'une commande d'un tableau est lue et prise en compte pour le traitement, elle est effacée du tableau et les autres sont décalées de façon à combler le vide laissé.

La structure d'un tableau est la suivante :

```
octet 0 : nombre de commandes enregistrees
octet 1 : numero du premier etage demande
octet 2 : numero du deuxieme etage demande
...
octet 9 : derniere commande prise en compte
```

Exemple :

Les commandes enregistrées sont en fait les numéros des étages, de 0 à N. Supposons qu'une personne dans l'ascenseur appuie dans l'ordre sur les boutons 2, 6, 5, 7. Ces commandes sont placées dans le tableau cc[] de manière automatique (en fait, il s'agit d'une fonction de type interruption appelée automatiquement lors de la sollicitation d'un bouton, si bien que vous supposez toujours avoir des tableaux à jour). Le tableau cc[] contiendra :

4, 2, 6, 5, 7

L'ascenseur devra aller dans l'ordre au 2ème, au 6ème, au 5ème, puis au 7ème.

Dès que la première destination a été lue, elle doit être effacée du tableau et celui-ci sera réorganisé comme suit :

3, 6, 5, 7

Si une nouvelle commande provient de la cabine, celle-ci sera placée à la fin du tableau. Ainsi, tant que ce tableau n'est pas vide, il est lu et la cabine est envoyée à l'étage demandé.

On vous demande d'écrire la fonction `lit_cabine()` qui effectue la lecture du premier étage demandé efface cette destination du tableau et le réorganise. L'étage lu est retourné par cette fonction. Si le tableau est vide, elle retourne la valeur -1. Cette fonction ne prend pas de paramètre.

Elle sera utilisée par exemple comme suit :

```
etage = lit_cabine()
Si etage <> -1
        Alors aller(etage)
Fin si
```

Question 3 : Gestion de fonctionnement

Ecrire la procédure qui gère l'ascenseur. On utilise la fonction `lit_cabine()` que nous avons étudiée, et `lit_appel()`, qui est similaire mais qui gère les appels depuis les paliers (tableau `ce[]`). Les commandes provenant de la cabine seront traitées en priorité. Si une commande extérieure est lancée (prise en compte), elle doit être menée à terme. Cette procédure est simple à écrire. Inutile de chercher des complications. Elle sera appelée `gestion()`, et ne prend pas de paramètre. Elle est constituée d'une boucle `Iterer` sans condition de sortie, ce qui signifie que dès que cette procédure est appelée, on n'en sort plus.