TP n°2 de langage C : tableaux, chaînes, bitwises

Exercices d'acquisition

En cas de difficulté, demandez un indice ou une explication à l'enseignant. Chaque exercice doit faire l'objet d'une **validation expérimentale poussée**.

Exercice 1 (acquisition)

Réalisez une fonction `void setBits(int* x, int masque)` qui met à 1, dans l'entier pointé par x, tous les bits qui sont désignés par les bits du masque ayant la valeur 1.

Exemple :

```
masque              : 11110000
entier pointé par x : 10101010
résultat            : 11111010
```

Exercice 2 (acquisition)

Réalisez une fonction `void resetBits (int* x, int masque)` qui met à 0, dans l'entier pointé par x, tous les bits qui sont désignés par les bits du masque ayant la valeur 1.

Exemple :

```
masque              : 11110000
entier pointé par x : 10101010
résultat            : 00001010
```

Exercice 3 (acquisition)

Réalisez une fonction `int getBit(int x, int k)` qui retourne la valeur du bit situé en position k dans la représentation binaire de l'entier x. La position 0 correspond au bit de poids faible.

Exemple :

```
x         : 10101010
k         : 2
résultat : 0
```

Exercice 4 (acquisition)

Réalisez une fonction `int pow2(int n)` qui retourne 2 à la puissance n. cette fonction ne doit pas comporter de boucle ni utiliser la fonction mathématique **pow**. elle doit reposer sur l'utilisation d'un ou plusieurs opérateurs bitwise.

Exercice 5 (acquisition)

Réalisez une fonction `int min(int t[], int n)` qui retourne le plus petit élément du tableau t, supposé de taille n.

Donnez un exemple de création d'un tableau de 10 entiers et de l'utilisation de la fonction **min** pour rechercher la plus petite des 5 dernières valeurs de ce tableau.

Exercice 6 (acquisition)

Réalisez une fonction `int verif(int t[], int n)` qui retourne un Boolean ayant la valeur vraie si et seulement si les n premiers éléments de t sont classés en ordre croissant.

Exercice 7 (acquisition)

Réalisez une fonction `int getPar(char t[], char p[] int n)` qui place dans le tableau p toutes les valeurs paires contenues dans le tableau t, qui est supposé être de longueur n, et qui retourne le nombre des ces valeurs paires. On supposera que la tableau p est de taille suffisante pour contenir toutes les valeurs paires de t.

Exercice 8 (acquisition)

Sans avoir recours à une fonction prédéfinie, réalisez une fonction `void cap(char* s)` qui remplace chaque lettre minuscule de la chaîne s par la lettre majuscule correspondante.

Exercice 9 (acquisition)

Réalisez une fonction `void verif(char* s)` qui retourne un Booléen valant vrai si et seulement si s ne contient que des lettres (minuscules et majuscules) et des chiffres.

TP n°3 de langage C : tableaux, chaînes, bitwises

Exercices de consolidation

En cas de difficulté, demandez un indice ou une explication à l'enseignant. Chaque exercice doit faire l'objet d'une **validation expérimentale poussée**.

Exercice 1 (consolidation)

Réalisez une fonction `void affBin(char x)` qui affiche les valeurs des 8 bits de l'octet x.

Trouvez un moyen d'utiliser cette fonction, en l'appelant 4 fois, pour afficher les valeurs binaires des 4 octets d'un entier stocké dans une variable z.

Par exemple si z vaut 257, l'affichage produit devrait être :

```
00000000 00000000 00000001 00000001
```

Indices : arithmétique des pointeurs, changement explicite du type d'un pointeur.

Exercice 2 (consolidation)

On souhaite utiliser un tableau de caractères pour représenter un tableau de bits. Par exemple, un tableau de 10 caractères contiendra 80 bits identifiés par des positions allant de 0 à 79. Les bits de position 0 à 7 sont placés dans le premier caractère du tableau, les bits de position 8 à 15 dans le deuxième caractère, etc.

Réalisez une fonction `void setBit(char t[], int i)` qui met à 1 le bit situé en position i dans le tableau de bits t. On supposera que t a une taille suffisante.

Réalisez une fonction `void resetBit(char t[], int i)` qui met à 0 le bit situé en position i dans le tableau de bits t.

Réalisez une fonction int `getBit(char t[], int i)` qui retourne la valeur (0 ou 1) du bit situé en position i dans le tableau de bits t.

Exercice 3 (consolidation)

Réalisez une fonction `int isIn(int t[], int n, int x)` qui, en supposant que t est un tableau de n entiers *triés en ordre croissant*, retourne un Booléen vrai si et seulement si la valeur x est dans le tableau t.

Le principe de la recherche dichotomique doit être utilisé, i.e., à chaque itération, l'intervalle de recherche doit être réduit de moitié.

Exercice 4 (consolidation)

Réalisez une fonction `void randPerm(int t[], int n)` qui place dans le tableau t toutes le valeurs comprises entre 0 et n-1 de sorte que chaque valeur apparaisse exactement une fois, dans un ordre aléatoire.

L'algorithme utilisé doit être efficace et garantir l'équiprobabilité de toutes les séquences possibles.

En utilisant la fonction **randPerm**, réalisez une fonction void `mixTab(double[] t, int n)` qui mélange les n éléments d'un tableau de `double` de manière à ce qu'ils se retrouvent dans un ordre totalement aléatoire et sans rapport (sauf coïncidence) avec leurs positions initiales.

Exercice 5 (consolidation)

Réalisez une fonction `char maxFreq(char* s, int* p)` qui retourne le caractère qui apparaît le plus souvent dans la chaîne s et place dans la variable pointée par p le nombre d'occurrences de ce caractère. Si plusieurs caractères répondent au critère, la fonction doit retourner l'un d'entre eux.

Utilisez un algorithme aussi efficace que possible.

Indice : vous pouvez utiliser un tableau d'entiers pour compter efficacement les occurrences des caractères.

Exercice 6 (consolidation)

On considère que deux chaînes sont similaires si la plus grande des deux peut être obtenue en insérant un caractère (de valeur et à une position appropriée) dans l'autre.

Réalisez une fonction `int simCh(char* s1, char* s2)` qui retourne un Booléen valant vrai si et seulement si s1 et s2 sont similaires.