Licence 1 - section B
TD 9 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 5 décembre au 11 décembre 2016

Pour produire le code source des programmes : écrire d'abord l'algorithme (en mélangeant français et instructions C), puis écrire le programme, sous forme de commentaires. (N'écrivez pas vos commentaires après coup) ! Le but est de remplir le squelette de programme que vous avez écrit avec des commentaires, pour aboutir au texte du programme. Il ne s'agit pas ici de commenter les lignes de votre programme, mais de vous permettre de les produire.

1 Fonctions à argument de type tableau

Pour ces exercices, on pourra faire une version qui utilise un indice i pour parcourir le tableau, ou bien une version qui utilise un pointeur (ou les deux !).

Exercice 1.1 Somme d'un tableau d'entiers.
Donner d'abord le prototype, puis définir la fonction qui retourne la somme des valeurs d'un tableau d'entiers.

Exercice 1.2 Moyenne d'un tableau d'entiers. Donner d'abord le prototype, puis définir la fonction qui calcule la moyenne des valeurs d'un tableau d'entiers.

Exercice 1.3 Nombre d'occurences d'un nombre nb dans un tableau d'entiers.
Donner d'abord le prototype, puis définir la fonction qui retourne le nombre d'occurences d'un nombre nb dans un tableau d'entiers.

2 Tri et recherche d'élément dans un tableau (version pointeurs)

Dans tous ces énoncés, `nb` représente le nombre d'élements du tableau `tab` passé en premier argument à la fonction.
On reprend ici certains exercices des TD précédents qui utilisaient des indices i et j pour parcourir les tableaux, mais on va réécrire ces fonctions avec des pointeurs, et leur prototype est susceptible de changer légèrement.

Exercice 2.1 Recherche d'un élément dans un tableau.

1. Ecrire une fonction `int* searchTabInt(int val, int tab[], int nb);` qui cherche la présence d'une valeur `val` dans un tableau de variables de type `int`, comportant `nb` éléments. Cette fonction retourne le pointeur NULL si la valeur ne figure pas dans le tableau, et sinon un pointeur vers le premier élément du tableau dans lequel cette valeur figure.
2. Même question pour la fonction `char* searchTabChar(char caract, char tab[], int nb)` où le tableau est cette fois un tableau de caractères. La fonction retourne le pointeur NULL en cas d'échec, et un pointeur sur le premier caractère trouvé dans le tableau sinon.

Exercice 2.2 Tri de tableaux
Les questions 1 et 2 ont été traitées en cours.

1. Ecrire une procédure `void imprimeTabInt(int tab[], int nb);` qui affiche un tableau d'entiers sous la forme [] ou [1,3,...,5,9] en utilisant un pointeur pour parcourir le tableau (et pas d'indice i).
2. Ecrire une procédure `void trieTabInt(int tab[], int nb);` qui trie un tableau de variables de type `int`, comportant `nb` éléments, en procédant à des échanges de valeurs entre cases du tableau. On écrira un algorithme qui parcours le tableau avec des pointeurs, et on pourra utiliser la procédure `void echange(int* a, int*b)` vue en cours.
3. Même question pour la procédure `void trieTabChar(char tab[], int nb)` qui trie un tableau de caractères (selon l'ordre des caractères ascii). Il faudra écrire au préalable une procédure `void echangeC(char *p, char*q).`

3 Comparaison de tableaux d'entiers

Exercice 3.1 Cet exercice reprend un exercice de la feuille de TD7.

1. Définir le type énuméré `booleen` comme constitué des deux valeurs constantes VRAI et FAUX, elles mêmes définies avec `#define` par les constantes 1 et 0 .
2. Ecrire une fonction `booleen compare(int tab1[], int tab2[], int nb);`
   qui retourne VRAI si les deux tableaux sont égaux (i.e. s'ils ont les mêmes valeurs), et qui retourne FAUX dans le cas contraire. Pour parcourir les deux tableaux, on utilisera deux pointeurs sur des entiers, et pas d'indice.
3. Modifier la fonction précédente pour écrire une fonction
   `int different(int tab1[], int tab2[], int nb);`
   qui retourne le rang des premiers éléments qui diffèrent dans les deux tableaux (qui sont de même taille), et le nombre zéro si les deux tableaux n'ont que des valeurs identiques. Astuce : pour trouver le rang - sans utiliser d'indice i dans le programme - on peut faire la différence entre un pointeur sur une case du tableau et le pointeur de début du tableau : `pt - tab` (ou encore `pt - &tab[0]`).