Licence 1 - section B
TD 8 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 28 novembre au 4 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 Tri et recherche d'élément dans un tableau

Dans tous ces énoncés, `nb` représente le nombre d'élements du tableau `tab` passé en premier argument à la fonction.

Exercice 1.1 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 -1 si la valeur ne figure pas dans le tableau, et sinon retourne le premier indice i du tableau où cette valeur figure.
2. Même question pour la fonction `int searchTabChar(char caract, char tab[], int nb)` où le tableau est cette fois un tableau de caractères.

Correction

1. L'idée de l'algorithme est de parcourir le tableau dans une boucle avec une variable de boucle i, en testant si la valeur de tab[i] est égale à la valeur `val` passée en argument. Si elle est égale, on retournera l'indice i, mais si on a parcouru tout le tableau sans avoir rencontré la valeur `val` (donc en sortie de boucle), alors on retournera -1.

```
int searchTabInt(int val, int tab[], int nb) {
        for (int i=0; i < nb ; i++)
                if (tab[i] == val)
                        return i;
        return -1;
}
```

2. L'algorithme est le même. Seuls les types sont différents, et la valeur recherchée est de type `char`.

```
int searchTabChar(char caract, char tab[], int nb {
        for (int i = 0;  i < nb ; i++)
                 if (tab[i] == caract)
                         return i;
        return -1;
}
```

Exercice 1.2 Tri de tableaux

Cet exercice sera traité en cours la semaine prochaine. A ne traiter que si le temps le permet.

1. Réfléchir à un algorithme permettant de trier un tableau d'entiers.
2. Ecrire une procédure `void trieTabInt(int tab[], int nb);` qui trie un tableau de variables de type `int`, comportant `nb` éléments.
3. Même question pour la procédure `void trieTabChar(char tab[], int nb)` qui trie un tableau de caractères.

Correction

1. Un algorithme possible est le suivant :

```
Si le nb d elements du tableau est 1 ou 0
   alors (ne rien faire)  et quitter (return)
Sinon
    parcourir le tableau et pour chaque i
          comparer tab[i] avec tab[j] pour j > i
          (tab[i]  doit etre plus petit pour que le tableau soit trie)
          si on trouve un j avec  tab[i] > tab[j]
                alors, echanger les valeurs tab[i] et tab[j]
    quitter la procedure avec return
```

   Pour échanger les valeurs de deux variables, on pourra utiliser la procédure `echange` vue dans le cours.

2. Le code de ces deux procédures en C :

```
void echange(int *a, int *b) {
  int sauve;
  sauve = *a;
  *a  = *b;
  *b = sauve;
}
void trieTab(int tab[], int nb){
  int i, j;

  if ((nb == 0) || (nb == 1))
     return;
  for (i=0; i < nb-1; i++)
      for (j=i+1; j < nb; j++)
          if (tab[i] > tab[j])
               echange(&tab[i], &tab[j]);
  return;
}
```

3. Une procédure qui trie un tableau de caractères suit le même algorithme. Seuls les types changent, mais il faudra réécrire une procédure d'échange des valeurs de deux variables de type `char`.

```
void echangeChar(char *a, char *b) {
  char sauve;
  sauve = *a;
  *a  = *b;
  *b = sauve;
}

void trieTabChar(char tab[], int nb){
 int i, j;

  if ((nb == 0) || (nb == 1))
     return;
  for (i=0; i < nb-1; i++)
     for (j=i+1; j < nb; j++)
         if (tab[i] > tab[j])
           echangeChar(&tab[i], &tab[j]);
  return;
}
```

2 Préparation au TP4

2.1 Structuration du code pour faire de la compilation séparée de fonctions

Dans le but de constituer une petite bibliothèque de fonctions sur les tableaux, on va regrouper dans un même fichier intitulé `tableaux.h` toutes les déclarations des fonctions qui ont été écrites dans ce TD et le précédent. Il suffira alors d'inclure le fichier `tableaux.h` de déclarations avec `#include "tableaux.h"` en tête de programme pour pouvoir utiliser n'importe quelle fonction de cette liste.

Exercice 2.1 Programme prog.c

1. Ecrire un programme dans un fichier prog.c qui
   — permet à l'utilisateur d'initialiser un tableau d'entiers.
   — interroge l'utilisateur pour qu'il entre une valeur entière.
   — affiche tous les indices des variables du tableau d'entiers qui ont cette valeur.
   On définira les fonctions auxilaires du programme (initialiser le tableau d'entiers et afficher les indices) dans des fichiers séparés portant le nom de la fonction qu'ils définissent pour pouvoir les compiler séparément.
2. Que faut-il modifier dans prog.c pour utiliser le fichier `tableaux.h` ?

Correction

1. Fichier prog.c :

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

void readTabInt(int tab[], nb);
void afficheIndices(int val, int tab[], nb);

int main() {
    int tab[MAX] = {0} ;
    int valeur, nombre = 0;

    printf("Entrez le nombre d'elements du tableau:\n");
    scanf("%d", &nombre);
    if (nb <= MAX)
     readTabInt(int tab[], nombre);  /* initialise le tableau jusqu'a nb */
    else {
         printf("Taille maximumm autorisee: %d\n", MAX);
         return -1;
     }

    printf("Entrez une valeur a rechercher dans le tableau: ");
    scanf("%d\n", &valeur);

    afficheIndices(int valeur, int tab[], nombre);

    return EXIT_SUCCESS;
}
```

Fichier readTabInt.c :

```
#include <stdio.h>

readTabInt(int tab[], int nb) {
}
```

Fichier afficheIndices.c :

```
void afficheIndices(int val, int tab[], int nb) {
}
```

2. On rajoute la déclaration des fonctions `readTabInt` et `afficheIndices` au fichier `tableaux.h`. On supprime les déclarations de ces mêmes fonctions dans `prog.c`, et on les remplace par `#include "tableaux.h"`.

   Fichier tableaux.h :

   ```
   void readTabChar(int tab[],int nb);
   void readTabInt(int tab[],int nb);
   ...
   int searchTabInt(int val, int tab[], int nb);
   char searchTabChar(char caract, char tab[], int nb);
   ```

   Début du fichier prog.c :

   ```
   #include <stdio.h>
   #include <stdlib.h>

   #include "tableaux.h"

   #define MAX 100

   int main() {
   ... etc.
   ```