J1MI2013: Algorithmes et Programmes: feuille 4

Arguments en ligne de commandes et tableaux

Travaux pratiques

Récupérer depuis le site de l'UE l'archive `tp04.tar.gz` et désarchivez-le. Placez vous ensuite dans le répertoire `tp04`.

Arguments en ligne de commande

Les arguments d'une ligne de commande sont transmis au programme sous la forme de chaînes de caractères séparées par des caractères d'espacement (espace ou tabulation). Ces arguments vont être récupérés par la fonction `main` par le biais de deux paramètres : l'entier `argc` et le tableau `argv`. Plus précisément, `argc` donne le nombre d'éléments de la ligne de commande, et `argv` contient ces éléments sous la forme d'un tableau de chaînes de caractères (`argv[0]` contient le nom de la commande, `argv[i]`, pour i allant de 1 à `argc-1`, le ième argument).

Exercice 1. Le programme `afficheParam.c` illustre le mécanisme décrit ci-dessus :

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int
5  main(int argc, char *argv[]){
6    printf("on execute %s \n", argv[0]);
7    printf("avec %d parametres: ", argc-1);
8    for( int i=1 ; i<argc ; i++){
9      printf("%s ", argv[i]);
10   }
11   printf("\n");
12   return EXIT_SUCCESS;
13 }
```

Essayer de comprendre ce programme, puis le compiler et l'exécuter avec différentes lignes de commande :

- `./afficheParam b a ba`
- `./afficheParam dix ten 10.0`
- `./afficheParam 2.0 4.0 6.0`

Analyser les résultats.

Exercice 2. Dans les exemples précédents, les constantes 10.0, 2.0, 4.0 et 6.0 sont manipulées par le programme `afficheParam.c` comme étant des chaînes de caractères (remarquer le format d'affichage `%s` et non pas `%g`). Pour pouvoir utiliser ces constantes en tant que nombres réels, il faut

utiliser la fonction de conversion `atof` [1].

- Essayez de compiler le programme `sommeParam.c` contenu dans l'archive fournie, que vous dit le compilateur ?
- Remplacez l'expression `somme + argv[i]` par l'expression `somme + atof(argv[i])` et recompilez.
- Exécutez le programme avec la ligne de commande :`./sommeParam 2.0 4.0 6.0`
- Modifiez le programme pour calculer la somme d'arguments entiers en utilisant `atoi` [2]. Exécutez-le avec la ligne de commande : `./sommeParam 2 4 6`.
- Si vous êtes curieux supprimez `atoi`, recompilez et observez le message d'erreur. En quoi est-il différent du précédent ? Pensez à rétablir une version qui compile.

Vérifications

Exercice 3. Compiler et exécuter le fichier `testAssert.c` contenu dans l'archive fournie.
Quel est le résultat de son exécution ? Le modifier de façon à ne jamais avoir d'échec lorsqu'on invoque `assert`.

Tableaux

Toutes les fonctions demandées à la suite seront écrites et testées dans le fichier `testTableaux.c` fourni, qui contient la définition de la fonction `afficherTableau`.

Exercice 4. Étudier `testTableaux.c`, le compiler et enfin exécuter :

- `./testTableaux`
- `./testTableaux 1 2 3 4 5`
- `./testTableaux 1 2 3 4 5 6 7 8 9 10 11`

Analyser les résultats.

Exercice 5. Écrire une fonction `int minimum(int t[], int n)` qui calcule et renvoie le minimum parmi les premiers `n` éléments d'un tableau `t` d'entiers. Arrêter le programme si `n` vaut 0.

Exercice 6. Écrire une fonction `bool appartient(int x, int t[], int n)` qui renvoie `true` si le tableau `t` contient la valeur `x` parmi ses premiers `n` éléments, et renvoie `false` sinon.

Exercice 7. Écrire une fonction `doubler` qui modifie le tableau `t` d'entiers passé en paramètre en doublant chacun de ses premiers `n` éléments. Quel sera le type de cette fonction ?

Exercice 8. Écrire une fonction `cumuler` qui modifie le tableau `t` passé en paramètre en plaçant dans chacune de ses premières `n` cases la somme des valeurs des termes précédents et du terme

1. dont la déclaration se trouve dans le fichier stdlib.h
2. pour en savoir plus sur `atoi` et `atof` on peut consulter (depuis une fenêtre terminal) le manuel en ligne avec les commandes `man 3 atoi` et `man 3 atof`

courant. Exemple : si avant l'appel `t = [1, -3, 12, -183]`, après l'appel de `cumuler(t,4)` on aura `t = [1, -2, 10, -173]`.

Exercice 9. Écrire une fonction `renverser` qui renverse le contenu du tableau `t` contenant `n` éléments passé en paramètre. Exemple : si avant l'appel `t = [1, 2, 3, 4]` , après l'appel de `renverser` on aura `t = [4, 3, 2, 1]`.

Exercice 10. Écrire et tester toutes les fonctions demandées dans la partie TD.