CPPL1 : TD 1 : C : Fonction et tableau

Nicolas Vansteenkiste Romain Absil Jonas Beleho *
(Esi – He2b)

Année académique 2017 – 2018

Ce TD[1] aborde les fonctions, les tableaux statiques et les chaînes de caractères du langage C dans sa version C11[2] (`ISO/IEC 9899:2011`).

Durée : 3 séances.

1 Fonction

Ex. 1.1 Écrivez la fonction de prototype :

```
bool isPrime(unsigned int number);
```

qui retourne `true`[3] ou `false` selon que son argument est premier ou non.

Répartissez prototype et code dans les fichiers `mathesi.h` et `mathesi.c`.

Ex. 1.2 Arrangez-vous pour produire, à l'aide de la fonction `isPrime(unsigned)` de l'Ex. 1.1, la sortie console suivante :

*Et aussi, lors des années passées : Monica Bastreghi, Stéphan Monbaliu, Anne Rousseau et Moussa Wahid.

1. https://poesi.esi-bru.be/pluginfile.php/1320/mod_folder/content/0/td01_c/td01_c.pdf
2. https://en.wikipedia.org/wiki/C11_(C_standard_revision)
3. http://en.cppreference.com/w/c/types/boolean

```
Les nombres premiers entre 200 et 349 :
   .   .   .   .   .   .   .   .   .   .
   . 211   .   .   .   .   .   .   .   .
   .   .   . 223   .   .   . 227   . 229
   .   .   . 233   .   .   .   .   . 239
   . 241   .   .   .   .   .   .   .   .
   . 251   .   .   .   .   . 257   .   .
   .   .   . 263   .   .   .   .   . 269
   . 271   .   .   .   .   . 277   .   .
   . 281   . 283   .   .   .   .   .   .
   .   .   . 293   .   .   .   .   .   .
   .   .   .   .   .   .   . 307   .   .
   . 311   . 313   .   .   . 317   .   .
   .   .   .   .   .   .   .   .   .   .
   . 331   .   .   .   .   . 337   .   .
   .   .   .   .   .   .   . 347   . 349
```

Pour la mise en forme, consultez la documentation de l'argument `format`[4] de la fonction `printf`[5].

Ex. 1.3 Écrivez la fonction de prototype :

```
void printPrimeFactor(unsigned int number, bool showPower);
```

qui décompose le nombre non signé passé en paramètre en un produit de facteurs premiers et affiche cette décomposition en notant les puissances ou non avant de passer à la ligne.

Par exemple, avec le nombre 126 en entrée et le paramètre `showPower` `false`, la fonction affiche :

```
126 = 2 x 3 x 3 x 7
```

tandis qu'avec le même nombre, mais `showPower` mis à `true`, elle affiche :

```
126 = 2 x 3^2 x 7
```

Répartissez prototype et code dans les mêmes fichiers `mathesi.h` et `mathesi.c` que ceux de l'Ex. 1.1.

Ex. 1.4 Écrivez la fonction de prototype :

```
unsigned gcd(unsigned a, unsigned b);
```

qui calcule le plus grand commun diviseur[6] (*greatest common divisor*) de `a` et `b` en implémentant l'algorithme d'Euclide[7].

4. http://www.cplusplus.com/reference/cstdio/printf/
5. http://en.cppreference.com/w/c/io/fprintf
6. https://fr.wikipedia.org/wiki/Plus_grand_commun_diviseur
7. https://fr.wikipedia.org/wiki/Algorithme_d'Euclide

Pour rappel, cet *algorithme récursif* repose sur les deux propriétés suivantes :

$$\gcd(a, 0) = a$$
$$\gcd(a, b) = \gcd(b, a \bmod b)$$

où a, $b \in \mathbb{N}$. Notez que si $a < b$, alors : $\gcd(a, b) = \gcd(b, a \bmod b) = \gcd(b, a)$.

Répartissez prototype et code dans les mêmes fichiers `mathesi.h` et `mathesi.c` que ceux de l'Ex. 1.1.

Ex. 1.5 Produisez, à l'aide de la fonction `gcd(unsigned, unsigned)` de l'Ex. 1.4, la sortie console suivante :

```
gcd(423, 135) =  9 | gcd(423, 130) =  1 | gcd(423, 125) =  1
gcd(426, 135) =  3 | gcd(426, 130) =  2 | gcd(426, 125) =  1
gcd(429, 135) =  3 | gcd(429, 130) = 13 | gcd(429, 125) =  1
gcd(432, 135) = 27 | gcd(432, 130) =  2 | gcd(432, 125) =  1
gcd(435, 135) = 15 | gcd(435, 130) =  5 | gcd(435, 125) =  5
gcd(438, 135) =  3 | gcd(438, 130) =  2 | gcd(438, 125) =  1
```

2 Tableau statique

Ex. 1.6 Écrivez la fonction de prototype :

```
void arrayIntPrint(const int data [], unsigned nbElem);
```

Elle affiche sur la sortie standard les `nbElem` premiers éléments du tableau [8] d'`int` en argument, séparés par un espace. Un passage à la ligne termine l'affichage. Si `data` vaut `NULL` [9] ou si le tableau contient plus de `nbElem` éléments, la fonction adopte un comportement indéterminé [10].

Ex. 1.7 Écrivez la fonction de prototype :

```
void arrayIntSort(int data [], unsigned nbElem, bool ascending);
```

Elle trie les `nbElem` premiers éléments de `data` dans l'ordre croissant ou décroissant selon que le paramètre `ascending` soit `true` ou `false` [11]. Implémentez l'algorithme de tri [12] de votre choix.

Testez votre fonction de tri et vérifiez son bon fonctionnement avec la fonction d'affichage de l'Ex. 1.6.

8. http://en.cppreference.com/w/c/language/array
9. http://en.cppreference.com/w/c/types/NULL
10. http://blog.llvm.org/2011/05/what-every-c-programmer-should-know.html
11. Un argument d'un type énuméré serait certainement plus explicite que le booléen utilisé ici.
12. https://fr.wikipedia.org/wiki/Algorithme_de_tri

Ex. 1.8 À l'aide de la fonction standard `qsort`[13], triez un tableau d'`int` dans l'ordre :

(a) croissant ;

(b) décroissant ;

(c) croissant modulo 3.

Ex. 1.9 Reprenez votre fonction de tri de l'Ex. 1.7 et modifiez-la pour produire la fonction de prototype :

```
void arrayIntSortGeneric(int data [], unsigned nbElem,
                         int (*comp)(const int *, const int *));
```

Celle-ci trie selon l'algorithme de votre choix les `nbElem` premiers éléments de `data` en utilisant la fonction `comp` pour comparer les éléments comme le fait la fonction standard `qsort`.

Testez votre fonction de tri et vérifiez son bon fonctionnement avec la fonction d'affichage de l'Ex. 1.6 et des fonctions de tri comme dans l'Ex. 1.8.

3 Chaîne de caractères

Ex. 1.10 Écrivez les fonctions réalisant les fonctionnalités de :

— `strlen`[14] retournant une valeur de type `size_t`[15] ;

— `strcmp`[16] ;

— `strcpy`[17] et `strncpy`[18] usant du mot-clé `restrict`[19] du C99[20] ;

— `strcat`[21] et `strncat`[22] ;

— `strtok`[23].

Consultez d'abord la documentation[24] de ces fonctions de manipulation de chaînes de caractères de la bibliothèque standard.

Produire ces fonctions *sans* utiliser l'opérateur d'indexation est un plus.

13. http://en.cppreference.com/w/c/algorithm/qsort
14. http://en.cppreference.com/w/c/string/byte/strlen
15. http://en.cppreference.com/w/c/types/size_t
16. http://en.cppreference.com/w/c/string/byte/strcmp
17. http://en.cppreference.com/w/c/string/byte/strcpy
18. http://en.cppreference.com/w/c/string/byte/strncpy
19. http://en.cppreference.com/w/c/language/restrict
20. https://en.wikipedia.org/wiki/C99
21. http://en.cppreference.com/w/c/string/byte/strcat
22. http://en.cppreference.com/w/c/string/byte/strncat
23. http://en.cppreference.com/w/c/string/byte/strtok
24. http://en.cppreference.com/w/c/string/byte

Ex. 1.11 Écrivez un programme qui attend un argument et affiche sur la sortie standard sa décomposition en produit de facteurs premiers. Le paramètre attendu est un entier non signé, fourni en base décimale. Par défaut, la décomposition est affichée avec exposants (voir Ex. 1.3), mais on peut forcer l'affichage long à l'aide de l'option `-v`.

Si :

— une autre option que `-v` est fournie ;

— l'option `-v` est fournie plusieurs fois ;

— aucune valeur numérique n'est fournie ;

— le paramètre fourni ne correspond pas à un entier non signé en base dix ;

— plus d'une valeur numérique est fournie ;

un message d'erreur accompagné d'un usage du programme est affiché.