Licence 1 - section B
TD 7 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 28 novembre au 2 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 Tableaux et fonctions à argument de type tableau

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

Exercice 1.1 Initialisations de tableaux de `int` ou de `char`

1. Ecrire une procédure `void initTabInt(int tab[], int nb, int val);`
   qui initialise un tableau de variables de type `int`, comportant `nb` éléments, uniformément avec la valeur `val`.
2. Même question pour la procédure `initTabChar`, qui initialise un tableau de caractères uniformément avec une valeur de type `char` nommée `val`.

Correction L'idée de l'algorithme ici est de parcourir le tableau pass en argument, avec une variable de boucle i qu'on prendra soin d'tre toujours < nb ; et pour chaque variable indexe du tableau tab[i], lui affecter la valeur val passe en argument la fonction. D'où

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

        for (int i = 0;  i < nb ; i++)
                tab[i] = val;
        return;
}
```

L'algorithme pour la question 2 est le même. Ce qui change, ce sont les types de données :

```
void initTabChar(char tab[], int nb, char val) {

        for (int i = 0;  i < nb ; i++)
                tab[i] = val;
         return;
}
```

Exercice 1.2 Comparaisons de tableaux d'entiers

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.
3. Modifier la fonction précédente pour écrire une fonction
   `int different(int tab1[], int tab2[], int nb);`
   qui retourne le rang du premier élément qui diffère dans les deux tableaux (de même taille), et le nombre zéro si les deux tableaux n'ont que des valeurs identiques.

Correction On définit ici le type `booleen` comme type *énuméré* avec le mot-clé `enum`. Une énumération est une liste de constantes entre accolades, permettant d'énumérer les valeurs constituant un type.

```
#define VRAI  1
#define FAUX 0
typedef booleen enum = {VRAI, FAUX}
```

L'algorithme pour la fonction compare consiste à faire une boucle, avec une variable de boucle i qui permettra de parcourir simultanment le premier et le deuxième tableau, en comparant à chaque pas les valeurs de tab1[i] et tab2[i]. Si elles sont différentes, c'est que les deux tableaux sont différents ; si elles sont égales, on avance sur les deux tableaux en inspectant l'égalité des éléments suivants.

```
booleen compare(int tab1[], int tab2[], int nb) {
        for (int i = 0;  i < nb ; i++)
                if (tab1[i] != tab2[i])
                        return FAUX;
        /* si on est ici, c'est que tab1 et tab2 ont des valeurs identiques */
        return VRAI;
}
```

Algorithme : on procède comme précédemment : on parcourt les deux tableaux en comparant à chaque pas les valeurs de tab1[i] et tab2[i]. Dès qu'elles sont diffrentes, on connait le premier indice où elles sont différentes. Mais en terme de "rang", il faut ajouter 1, car le premier élément d'indice 0 a le rang 1. Les éléments d'indice i ont donc le rang i+1, et ce sera cette valeur qui sera retournée. Si on franchit la boucle, sans avoir du quitter avec return, c'est que les éléments ´taient toujours ´gaux, donc on retourne 0.

```
int different(int tab1[], int tab2[], int nb) {
        for (int i = 0;  i < nb ; i++)
                if (tab1[i] != tab2[i])
                        return i+1;
        return 0;
}
```

Exercice 1.3 Recherche d'une valeur dans un tableau

1. Ecrire une fonction `int searchValue(int val, int tab[],int nb);` qui recherche la présence d'une valeur `val` dan le tableau `tab` de `nb` éléments. La fonction retourne l'indice de la première varaible du tableau qui a cette valeur, et -1 si le tableau ne contient pas cette valeur.
2. Modifier la fonction précédente pour qu'elle puisse d'appliquer à un tableau de caractères.

Correction

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

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

Exercice 1.4 Tableaux de `char`

1. `void readTabChar(int tab[],int nb);`
   Cette procédure initialise un tableau de `nb` caractères avec une série de caractères qui sont lus au clavier avec `scanf`.

2. `void imprimeChar (char tab[], int nb);`
   Cette procédure imprime un tableau de `nb` caractères en affichant successivement chacun de ses éléments (sans passage à la ligne suivante).
3. `void traduire(short tab[], int nb, char ascii[]);`
   Cette procédure effectue la traduction dn tableau d'entiers (`short`) dans un tableau de caractères de même taille nommé ascii[] dans lequel pour chaque i < nb, ascii[i] sera le caractère (type char) dont le code ASCII est tab[i].

Correction

```
void readTabChar(char tab[],int nb) {
        for (int i = 0;  i < nb ; i++)
                scanf("%c", &tab[i]);
        return;
}

void imprimeChar (char tab[], int nb) {
        for (int i = 0;  i < nb ; i++)
                printf("%c", &tab[i]);
        return;
}

void traduire(short tab[], int nb, char ascii[]) {
        for (int i = 0;  i < nb ; i++)
                ascii[i] = (char) tab[i];
        return;
}
```

Exercice 1.5 Imprimer un tableau de caractères sous la forme [a b z d ...].

1. De manière analogue à ce qui a été vu en cours avec la procédure `imprime` qui prenait en argument un tableau de variables de type `int`, écrire une autre procédure `imprime` qui prend en arguments un tableau de caractères et le nombre d'éléments de ce tableau, puis affiche entre deux crochets les caractères du tableau un à un en les séparant par des caractères espace. Si le tableau passé en argument est vide, on affichera simplement deux crochets ouvrant et fermant [ ].
2. Si vous avez écrit cette fonction en utilisant une boucle `while`, transformez le code source pour utiliser une boucle `for` (et inversement).

Correction

```
1. /* déclaration de fonctionalités supplémentaires */
2. #include <stdlib.h> /* pour EXIT_SUCCESS */
3. #include <stdio.h> /* pour printf */
4.
5. /* déclaration de constantes et types utilisateurs */
6.
7.        /* déclaration de fonctions utilisateurs */
8. void imprime(char tab[], int nb) ;     /* la fonction imprime les elements de tab: [a,
9.        /* fonction principale */
10. int main()
11. {
12.     /* déclaration des variables locales x et y */
13.    char tableau[] = { 'c', 'e', 'c', 'i', ' ', 'e', 's', 't', ' ', '1', ' ', 't',
14. 'a', 'b', 'l', 'e', 'a', 'u' } ;
15.
16.    imprime(tableau, 18);
17.    return EXIT_SUCCESS;
18. }
19.
20.     /* definition de la fonction imprime */
21. void imprime(char tab[], int nb) {/* nb est le nb dlements de tab */
```

```
22. int i;
23. if (nb == 0) {
24.  printf("[ ]");/* affiche: [ ] */
25. return; /* retourne void */
26. }
27. printf("[%c",tab[0]); /* affiche [ tab[0] */
28. i = 1;
29. while (i < nb) {
30. printf(",%c", tab[i]);
31. i = i+1;
32. }
33. /* on a dj affich [ val0, ..., valn-1 */
34. printf("]"); /* affiche ] */
35. return ;
36. }
```