ECUE «Introduction à la programmation » - CC n°3 - **Corrigé**

6 janvier 2011 - Bruno Bouzy
sans document - durée 1 heure 30

Cet énoncé correspond à l'écriture d'un programme C appelé `codeChar.c`.

Question 1 (1 point)

Ecrire les `#include` nécessaires pour utiliser les fonctions `printf`, `strlen`.

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

Question 2 (1 point)

Définir 3 constantes `NBL`, `TAILLE` et `CLE` valant respectivement `26`, `50` et `19`.

```
#define NBL 26
#define TAILLE 50
#define CLE 19
```

Question 3 (1 point)

Ecrire une fonction `codeN` prenant deux entiers `n` et `a` en entrée et retournant le reste de la division entière de `n` fois `a` par `NBL`. Pour la suite, `codeN(n, a)` est le nombre `n` codé avec la clé `a`.

```
int codeN(int n, int a)
{
  return (n*a) % NBL;
}
```

Question 4 (2 points)

Une lettre minuscule correspond à son rang dans l'alphabet moins `1`: `'a'` correspond à `0`, `'b'` à `1`, etc, `'z'` à `25`. Réciproquement un nombre compris entre `0` et `25` correspond à une lettre de l'alphabet en minuscule. Ecrire une fonction `codeChar` prenant un caractère `c` et un entier `a` en entrée. Si le caractère `c` est une lettre minuscule, alors la fonction `codeChar` retourne la lettre minuscule correspondant à `codeN(q, a)` où `q` est le nombre correspondant à `c`. Sinon la fonction retourne le caractère `c`. La fonction `codeChar` appellera la fonction `codeN`.

```
char codeChar(char ch, int a)
{
  if (ch >= 'a' && ch <= 'z') return 'a' + codeN(ch-'a', a);
  return ch;
}
```

Question 5 (3 points)

Ecrire une procédure `codeChaine` transformant une chaîne de caractères `c1` en une chaîne de caractères `c2` codée avec une clé `a`. Pour chaque caractère `c` de `c1`, le caractère de `c2` lui correspondant sera codé avec `codeChar(c, a)`. La déclaration de `codeChaine` est:
`void codeChaine(char * c1, char * c2, int a);`
`c1`, `c2`, `a` sont des paramètres en entrée. `c2` est un paramètre en sortie.

```
void codeChaine(char * c1, char * c2, int a)
{
  int l = strlen(c1);
  int i;
  for (i=0; i<l; i++) *(c2+i) = codeChar(*(c1+i), a);
  *(c2+l) = '\0';
}
```

Question 6 (2 points)

Ecrire un programme principal, affichant la valeur de la constante `CLE`, déclarant un tableau de caractères `ch` de taille `TAILLE` et initialisé avec `"chaine de test"`, affichant `ch`, déclarant un tableau de caractères `cr` de taille `TAILLE`, appelant `codeChaine` avec `ch`, `cr` et `CLE`, affichant `cr`. La sortie du programme correspond à:

```
CLE = 19 , ch = chaine de test , cr = mdawny fy xyex
```

```
int main()
{
  printf("CLE = %d , ", CLE);
  char ch[TAILLE]="chaine de test";
  printf("ch = %s , ", ch);
  char cr[TAILLE];
  codeChaine(ch, cr, CLE);
  printf("cr = %s\n", cr);
  ...
```

Question 7 (2 points)

On veut construire un tableau de caractères `code` de taille `NBL` contenant les codes des lettres de l'alphabet. Déclarer le tableau `code`. Ecrire une boucle `for` remplissant `code` avec des appels à `codeChar` et affichant le contenu du tableau `code`. La sortie du programme correspond à:

```
code(a) = a
code(b) = t
...
code(z) = h
```

```
char code[NBL];
int i; for(i=0; i<NBL; i++) {
  code[i] = codeChar('a'+i, CLE);
  printf("code(%c) = %c\n", 'a'+i, code[i]);
}
```

Question 8 (3 points)

On veut construire un tableau de caractères `decode` de taille `NBL` avec les valeurs inverses de `codeChar` pour toutes les lettres de l'alphabet. Déclarer le tableau `decode`. Ecrire une boucle `for` remplissant `decode` et affichant son contenu. Conseil: pour calculer la valeur d'une case du tableau `decode` correspondant à une lettre `y` on écrira une boucle recherchant la lettre `x` telle que `codeChar(x, CLE)` égale `y`. La sortie du programme correspond à:

```
decode(a) = a
decode(b) = l
...
decode(z) = p
```

```
char decode[NBL];
for(i=0; i<NBL; i++) {
  char a = 'a'+i;
  int j=0;
  while ((j<NBL) && (codeChar('a'+j, CLE)!=a)) j++;
  decode[i] = 'a'+j;
  printf("decode(%c) = %c\n", 'a'+i, decode[i]);
}
```

Question 9 (2 points)

Ecrire une fonction `essai` prenant deux clés `a` et `u` en entrée, retournant `1` si les deux clés sont inverses l'une de l'autre, et `0` sinon. Deux clés `a` et `u` sont inverses l'une de l'autre signifie que, pour toute lettre `L` de l'alphabet, si on code `L` avec `a`, et le résultat avec `u`, on obtient `L`. La déclaration de `essai` sera: `int essai(int a, int u);`

```
int essai(int a, int u)
{
  int i;
  for(i=0; i<NBL; i++) {
    if (codeN(codeN(i, a), u)!=i) return 0;
  }
  return 1;
}
```

Question 10 (2 points)

Ecrire une fonction `cleInverse` prenant deux paramètres: une clé en entrée et une clé en sortie, et retournant `1` si elle a trouvé, `0` sinon. La clé en sortie sera passée par adresse. La fonction `cleInverse` appelle la fonction `essai` pour chaque valeur de clé inverse allant de `0` à `25`. Elle s'arrête dés qu'elle trouve une clé inverse adéquate. La déclaration de `cleInverse` sera:
`int cleInverse(int a, int * u0);`

```
int cleInverse(int a, int * u0) {
  int t=0, u;
  for (u=0; u<NBL && t==0; u++) t = essai(a, u);
  if (t==0) return 0;
  else { *u0 = u-1; return 1; }
}
```

Question 11 (1 point)

Dans le programme principal, en utilisant un un `if` et un appel à la fonction `cleInverse` pour la valeur de `CLE`, tester si `CLE` a une clé inverse et afficher le résultat s'il existe. La sortie du programme correspond à:

```
cle inverse de 19 = 11
```

```
  ...
  int cle2;
  if (cleInverse(CLE, &cle2)==1)
    printf("cle inverse de %d = %d\n", CLE, cle2);
  else
    printf("pas de cle inverse de %d.\n", CLE);

  return 0;
}
```