UNIVERSITÉ CÔTE D'AZUR

Université Côte d'Azur — L2 Informatique
Programmation C

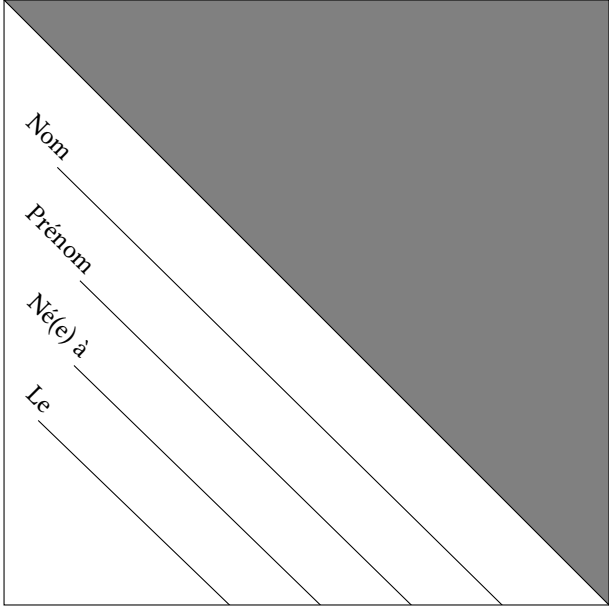
Examen : Programmation C

15 janvier 2024

Durée : 1 heure
Tous documents autorisés. Il est interdit d'accéder à internet.

Note

Toutes les questions sont indépendantes.
Tous les codes devront être écrits en **Langage C ANSI**. La notation est donnée à titre indicatif.
Nombre de pages : 4



Il est de votre responsabilité de rabattre le triangle grisé et de le cacheter au moyen de colle, agrafes ou papier adhésif. Si ne vous le faites pas, vous acceptez implicitement que votre copie ne soit pas anonyme.

Exercices divers (7 points)

1. Écrire une fonction `est_nombre` qui prend en argument une chaîne de caractères et renvoie Vrai si la chaîne ne contient que des chiffres et Faux sinon. C'est à vous de savoir comment coder Vrai et Faux en C.

```c
int est_nombre(char * chaine) {
  int i;
  for (i=0; chaine[i] != '\0'; i++) {
    if (chaine[i] < '0' || chaine[i] > '9') {
      return 0;
    }
  }
  return 1;
}
```

2. Une note est un nombre entier compris entre 0 et 20. Écrire une fonction `int ajouter_deux(int *note)` qui prend en parametre un pointeur sur une note et y ajoute deux. Bien évidemment, la note après ajout ne doit pas dépasser 20. En cas de dépassement, la fonction renverra 1 et la note sera quand même mise à 20.

```c
int ajouter_deux(int * note) {
  *note += 2;
  if (*note > 20) {
    *note=20;
    return 1;
  }
  return 0;
}
```

3. Écrire une fonction `reflet` qui affiche une chaîne en miroir. On pourra utiliser la fonction `strlen` qui renvoie la longueur d'une chaîne. Par exemple, à partir de « `"star"` », elle affichera « `rats` ».

```
int reflet(chaine *c) {
  int i;
  for (i=strlen(c)-1; i>=0; i--) {
    printf("%c", chaine[i]);
  }
}
```

4. Que fait le code suivant ? Réecrivez-le de manière plus lisible (sans les opérateurs `--`, `+=` et sans pointeurs).

```
int mystere(int t[], int n) {
  int x = 0;
  while(n-1)
      x += *(t + --n);
      return x;
}
```

```
// Le code calcul la somme des élements d'un tableau (sauf le premier)
int mystere(int t[], int n) {
  int x = 0;
  while(n>1) { // n-1 != 0
    n = n-1;
    x = x + t[n];
  }
  return x;
}
```

Problème (13 points)

On cherche à implémenter un dictionnaire. Une structure de données qui associe des clefs à des valeurs.

1. Créez une structure `struct couple` contenant deux champs : un champs `clef` de type caractère et un champs `valeur` de type entier.

```
struct couple {
  char clef;
  int valeur;
}
```

2. Définir un type `dico` comme étant un tableau de pointeurs de `struct couple`.

```
typedef struct couple ** dico
```

3. Écrire une fonction `dico nouveau_dico(int n)` qui alloue sur le tas un dictionnaire de taille `n` et renvoie le pointeur correspondant. Par convention, pour indiquer qu'il est vide, on le remplit entièrement avec la valeur `NULL`.

```
dico nouveau_dico(n) {
  int i;
  dico d = malloc(sizeof(struct couple *)*n);
  for (i=0; i<n; i++) {
    d[i] = NULL;
  }
  return d;
}
```

On va distinguer la taille prise en mémoire par le dictionnaire du nombre de cases réellement utilisée. Cette taille est indiquée par la variable `n`. Par contre, dans ce tableau de taille `n`, un pointeur `NULL` indique que la case n'est pas utilisée ainsi que les suivantes : par exemple, le tableau `{c1, c2, c3, NULL, NULL}` est un tableau de taille `n=5` avec trois valeurs.

4. Écrire une fonction `int dico_get(dico d, int n, char c, int *v)` qui cherche dans un dictionnaire de taille `n` un couple contenant la clef `c`. S'il le trouve, il affecte au pointeur `v` la valeur correspondante et renvoie `0` ; si la clef `c` n'est pas trouvée, on renvoie `-1`.

```
int dico_get(dico d, int n, char c, int *v) {
  int i;
  for (i=0 ; i<n; i++) {
    if (d[i] == NULL) return -1;
    if ( d[i]->clef == c) {
      *v = d[i]->valeur;
      return 0;
    }
  }
  return -1;
}
```

5. Écrire une fonction `int dico_set(dico d, int n, char c, int v)` qui modifie la valeur associée à une clef dans le dictionnaire. Elle renverra `0` en cas de succès. Si la clef ne se trouve pas dans le dictionnaire et s'il reste de la place, on crée un nouveau couple `(clef,valeur)` que l'on entre dans le dico à la place de la première case contenant `NULL`, puis on renvoie `1`. S'il n'y a plus de place dans le dictionnaire, on renvoie `-1`.

```
int dico_set(dico d, int n, char c, int v) {
  int i;
  for (i=0 ; i<n; i++) {
    if (d[i] == NULL) {
      d[i] = malloc(sizeof(struct couple *));
      d[i]->clef = c;
      d[i]->valeur = v;
      return 1;
    }
    if ( d[i]->clef == c) {
      d[i]->valeur = v;
      return 0;
    }
  }
  return -1;
}
```

6. Écrire une fonction `void liberer(dico d, int n)` qui libère toute la mémoire associée au dictionnaire.

```
int liberer(dico d, int n) {
  int i;
  for (i=0 ; i<n; i++) free(d[i]) ;
  free(d);
}
```

7. Écrire une fonction `void incremente(dico d, int n, char c)` qui incrémente la valeur associée à `c` dans le dictionnaire et renvoie `0`. Si `c` ne se trouve pas dans le tableau, on le rajoute en l'associant à la valeur `1` et on renvoie `1`. S'il n'y a plus de place dans le tableau on renvoie `-1`. On ne parcourera pas le tableau mais on devra utiliser les fonctions que vous avez écrites dans cet exercice.

```
int incremente(dico d, int n, char c) {
  int v;
  int r;
  r = dico_get(d,n,c,&v);
  if (r==-1) v=0
  return dico_set(d,n,c,v+1);
}
```