Université de Xidian
Polytech'Sophia

Université de Xidian
2017–2018

Examen de Langage C *(V. Granet)*

Durée : 1h30
Aucun document autorisé
Mobiles interdits

▶ 1. En utilisant uniquement la notation de pointeur et à l'exclusion de toute autre fonction, écrivez en C la fonction `nIemeOccurrence` qui renvoie un pointeur sur la n^e occurrence d'un caractère dans une chaîne. Si la n^e occurrence n'existe pas, la fonction renverra `NULL`. Par exemple, l'appel `nIemeOccurrence("abcaaer", 'a', 2)`, renvoie l'adresse du 2ème `a`, et l'appel suivant : `nIemeOccurrence("abcaaer", 'a', 10)` renvoie `NULL`. Cette fonction possède l'en-tête suivant :

```
char *nIemeOccurrence(char *s, const char c, const int n)
```

```
/*
 * Antécédent : n>=0
 * Rôle : renvoie l'adresse de la n\ieme occurrence du caractère c
 *        dans s, ou NULL si non présente
 */
char *nIemeOccurrence(char *s, const char c, const int n) {
  assert(n>=0);
  int i=n;
  while (*s) {
    if (*s==c)
      // on a trouvé la nième occurrence
      if (--i==0) return s;
    s++;
  }
  // pas de nième occurrence
  return NULL;
}
```

.....

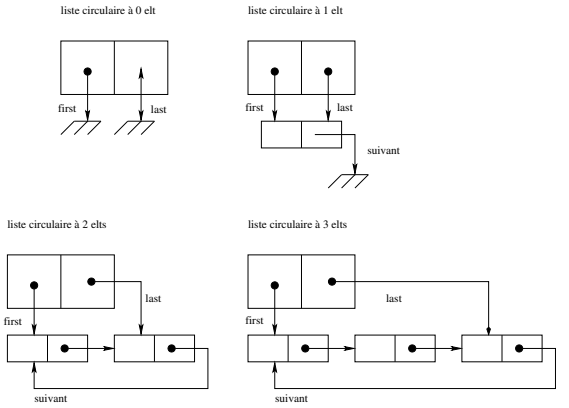
On souhaite représenter une liste d'éléments de type `T` de façon circulaire (le dernier élément est lié au premier) par une structure simplement chaînée. Pour cela, on définit le type `LISTE_CIR` suivant :

```
typedef int T; // par exemple

struct noeud {
  T elt;
  struct noeud *suivant;
};

typedef struct {
  struct noeud *first, *last;
} LISTE_CIR;
```

▶ 2. En utilisant cette déclaration de `LISTE_CIR`, dessinez une liste à 0 élément, une liste à 1 élément, une liste à 2 éléments, et enfin une liste à 3 éléments.



.....

▶ 3. Écrivez la procédure `tourner` qui prend en paramètre une liste `li` et un entier `n` et qui fait `n` fois le tour de la liste de façon circulaire.

```
int estVide(LISTE_CIR l) {
  return l.first == NULL && l.last == NULL;
}

/*
 * Antécédent : n>=0
 */
void tourner(LISTE_CIR l, int n) {
  assert(n>=0);
  if (!estVide(l))
    // faire n tours de la liste circulaire
    for (int i=0; i<n; i++) {
      struct noeud *q = l.first;
      do
        q = q->suivant;
      while (q!= l.first);
    }
}
```

.....

▶ 4. Écrivez la procédure `ajouterEnTete` qui ajoute un élément au début de la liste circulaire. Cette procédure possède l'en-tête suivant (que vous devez respecter) :

```
void ajouterEnTete(LISTE_CIR *li, T e)
```

```
void ajouterEnTete(LISTE_CIR *li, T e) {
  // créer un nouveau noeud
  struct noeud *p = malloc(sizeof(struct noeud));
  p->elt = e;
  if (estVide(*li)) {
    // cas particulier d'une liste circulaire vide
    (*li).first = (*li).last = p;
```

```
    p->suivant = p;
  }
  else {
    // cas général
    p->suivant = (*li).first;
    (*li).first  = p;
    (*li).last->suivant = (*li).first;
  }
}
```

.....

► 5. Donnez la déclaration d'une structure `Date` pour représenter une date formée de 3 entiers : `jour`, `mois` et `année`.

```
typedef struct {
  int jour, mois, annee
} Date;
```

.....

► 6. Un fichier de caractères contient un suite d'entiers séparés par un ou plusieurs espaces. Trois entiers consécutifs représentent une date. Écrivez la fonction `creerFichDates` qui lit un fichier de caractères contenant la suite d'entiers, et qui crée un fichier de dates valides (de type `Date`). Sans l'écrire, vous pourrez utiliser la fonction booléenne `dateValide` qui teste si la date (de type `Date`) passée en paramètre correspond à une date valide ou pas. La fonction `creerFichDates` prend les noms des deux fichiers en paramètre. Son en-tête est le suivant :

```
void creerFichDates(char *in, char *out);
```

```
void creerFichDates(char *in, char *out) {
  FILE *fdIn, *fdOut;
  // ouvrir le fichier d'entrée en lecture
  if ((fdIn = fopen(in, "r")) == NULL) {
    perror(in);
    exit(errno);
  }
  // ouvrir le fichier de sortie en écriture
  if ((fdOut = fopen(out, "w")) == NULL) {
    perror(out);
    exit(errno);
  }
  // ok : les 2 fichiers sont ouverts
  // fabriquer le fichier de Date
  Date d;
  while (fscanf(fdIn, "%d %d %d", &d.jour, &d.mois, &d.annee)==3) {
    // on a lu 3 entiers
    if (dateValide(d))
      fwrite(&d, sizeof(Date), 1, fdOut);
  }
  // fermer les fichiers
  fclose(fdIn);  fclose(fdOut);
}
```

.....