

Examen de Langage C

Durée : 1h30

Aucun document autorisé – Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles, et invariants) dans vos codes C entreront pour partie dans la note finale.

On veut représenter la notion de *durée normalisée* en heures, minutes et secondes. Le nombre d'heures, de minutes et de secondes seront des entiers (`int`), tels que le nombre d'heures est ≥ 0 , le nombre de minutes et de secondes sont compris entre 0 et 59. Par exemple, 1 heure, 30 minutes et 0 secondes est la durée de votre examen.

- 1. Avec un `typedef`, déclarez le type `durée`, une structure avec les 3 champs `heures`, `minutes` et `secondes` de type `int`.

```
typedef struct {
    int heures; // heures ≥ 0
    int minutes; // 0 ≤ minutes ≤ 59
    int secondes; // 0 ≤ secondes ≤ 59
} durée;
```

- 2. Écrivez la fonction booléenne `estNormalisée` qui renvoie `true` si la durée passée en paramètre est normalisée et `false` sinon.

```
/*
 * Rôle : renvoie vrai si la durée d est normalisée
 *       i.e. heures ≥ 0, 0 ≤ minutes ≤ 59 et 0 ≤ secondes ≤ 59
 */
bool estNormalisée(const durée d) {
    return d.heures >= 0 && (d.minutes >= 0 && d.minutes <= 59) &&
           (d.secondes >= 0 && d.secondes <= 59);
}
```

On veut écrire une fonction `initDurée` qui renvoie une durée initialisée à partir de 3 entiers. Pour cela, vous allez d'abord écrire une fonction auxiliaire `normaliser`.

- 3. Écrivez la fonction `normaliser` qui prend en paramètre une durée dont le nombre d'heures, de minutes et de secondes sont supérieurs ou égaux à 0, et qui renvoie une durée *normalisée*, c'est-à-dire dont le nombre d'heures est ≥ 0 , et le nombre de minutes et de secondes sont compris entre 0 et 59. Par exemple, si le nombre d'heures est égal à 3, le nombre de minutes 127, et le nombre de secondes 456, la fonction `normaliser` renverra une durée égale à 5 heures, 14 minutes et 36 secondes. L'en-tête de cette fonction est le suivant :

```
/*
 * Rôle : renvoie une durée normalisée
 */
durée normaliser(const durée d) {

.....

/*
 * Rôle : renvoie la normalisation de la durée d
 */
durée normaliser(const durée d) {
    if (estNormalisée(d)) return d;
    // d n'est pas normalisé => normaliser d
    durée nd=d;
    // normaliser les secondes
    if (nd.secondes >= 60) {
        nd.minutes += nd.secondes / 60;
        nd.secondes %= 60;
    }
    // normaliser les minutes
    if (nd.minutes >= 60) {
        nd.heures += nd.minutes / 60;
        nd.minutes %= 60;
    }
    assert(estNormalisée(nd));
    return nd;
}
```

- 4. À l'aide de la fonction `normaliser` précédente, écrivez la fonction `initDurée` qui renvoie une durée normalisée initialisée à partir de 3 entiers quelconques. Vous vérifiez que les entiers sont ≥ 0 . L'en-tête de cette fonction est le suivant :

```
/*
 * Antécédent : h ≥ 0, m ≥ 0, s ≥ 0
 * Rôle : renvoie une durée normalisée à partir de h, m, s
 */
durée initDurée(const int h, const int m, const int s)
```

```
/*
 * Antécédent : h ≥ 0, m ≥ 0, s ≥ 0
 * Rôle : renvoie une durée normalisée à partir de h, m, s
 */
durée initDurée(const int h, const int m, const int s) {
    assert(h >= 0 && m >= 0 && s >= 0);
    return normaliser((durée) {h, m, s});
}
```

- 5. Écrivez la fonction `cmpDurée` qui compare deux durées normalisées `d1` et `d2`. Cette fonction renvoie 0 si `d1` et `d2` sont égales, une valeur entière négative si `d1 < d2`, et positive si `d1 > d2`.

```

/*
 * Antécédent : d1 et d2 deux durées normalisées
 * Rôle : compare les durées d1 et d2
 */
int cmpDurée(const durée d1, const durée d2) {
    assert(estNormalisée(d1) && estNormalisée(d2));
    if (d1.heures!=d2.heures)
        return d1.heures-d2.heures;
    // nb d'heures égal
    if (d1.minutes!=d2.minutes)
        return d1.minutes-d2.minutes;
    // nb d'heures et de minutes égal
    return d1.secondes-d2.secondes;
}

```

ou bien, autre écriture :

```

/*
 * Antécédent : d est une durée normalisée
 * Rôle : renvoie le nombre de secondes que représente la durée d
 */
int nbSecondes(const durée d) {
    return d.heures*3600+d.minutes*60+d.secondes;
}
/*
 * Antécédent : d1 et d2 deux durées normalisées
 * Rôle : compare les durées d1 et d2
 */
int cmpDurée(const durée d1, const durée d2) {
    assert(estNormalisée(d1) && estNormalisée(d2));
    return nbSecondes(d1) - nbSecondes(d2);
}

```

- 6. Écrivez la fonction moins qui renvoie la soustraction de deux durées normalisées d1 et d2 : d1-d2. La fonction renvoie une durée normalisée. Notez que la soustraction n'est possible que si d1 ≥ d2.

```

/*
 * Antécédent : d1 ≥ d2
 * Rôle : renvoie la durée normalisée d1-d2
 */
durée moins(const durée d1, const durée d2) {
    assert(cmpDurée(d1, d2) >= 0);
    // d1 >= d2
    durée d;
    d.heures = d1.heures-d2.heures;
    //
    d.minutes = d1.minutes-d2.minutes;
    if (d.minutes < 0) {
        d.minutes = d.minutes + 60;
        d.heures--;
    }
    //
    d.secondes = d1.secondes-d2.secondes;
    if (d.secondes < 0) {
        d.secondes = d.secondes + 60;
    }
}

```

```

        d.minutes--;
    }
    return d;
}

```

ou bien, autre écriture :

```

/*
 * Antécédent : s un nombre de secondes ≥ 0
 * Rôle : renvoie une durée normalisée issue de s
 */
durée secondesVersDurée(const int s) {
    return normaliser((durée) {0, 0, s});
}
/*
 * Antécédent : d1 ≥ d2
 * Rôle : renvoie la durée normalisée d1-d2
 */
durée moins(const durée d1, const durée d2) {
    assert(cmpDurée(d1, d2) >= 0);
    // d1 >= d2
    return secondesVersDurée(nbSecondes(d1)-nbSecondes(d2));
}

```

- 7. En utilisant uniquement la notation pointeur, écrivez la fonction duréeMax qui renvoie la plus grande durée contenue dans un tableau t de n durées normalisées. Le tableau et le nombre de durées sont passé en paramètres.

```

/*
 * Antécédent : t tableau de n (>0) durées normalisées
 * Rôle : renvoie la durée max de contenue dans t
 */
durée duréeMax(const durée *t, const int n) {
    assert(n > 0);
    durée max = *t++;
    for (int i=1; i<n; i++) {
        if (cmpDurée(*t, max) > 0)
            // nouvelle durée max
            max = *t;
        //
        t++;
    }
    return max;
}

```

- 8. Écrivez la fonction main qui déclare un tableau initialisé avec 3 durées et qui appelle la fonction duréeMax pour écrire sur la sortie standard la plus grande des durées.

```

/*
 * Antécédent : d est une durée normalisée

```

```

* Rôle : écrit sur la S/S la durée d au format h:mm:ss
*/
void écrireDurée(const durée d) {
    printf("%d:%02d:%02d", d.heures, d.minutes, d.secondes);
}

/*
* Antécédent : d est une durée normalisée
* Rôle : écrit sur la S/S la durée d au format h:mm:ss suivie d'un '\n'
*/
void écrireLnDurée(const durée d) {
    écrireDurée(d);
    printf("\n");
}

int main(void) {

    durée td[] = {
        initDurée(0,127,86), initDurée(3,127,456), initDurée(1,127,456)
    };
    écrireLnDurée(duréeMax(td, 3)); // ⇒ 5:14:36

    return EXIT_SUCCESS;
}

```

.....