

Examen de Langage C

Durée : 1h

Aucun document autorisé

Mobiles interdits Toutes les fonctions que vous écrirez doivent être clairement commentées avec des affirmations significatives (antécédents, conséquents, invariants). Vous prendrez soin de définir les bons paramètres et les bons types des données manipulées. Pensez à définir des fonctions auxiliaires si cela est nécessaire.

- 1. Expliquez l'intérêt de la logique de Hoare.

L'intérêt principal de la *logique de Hoare* est de permettre de prouver de façon formelle et analytique (sans l'exécuter) la validité de la sémantique opérationnelle d'un programme à l'aide d'assertions.

- 2. Expliquez à quoi correspond le triplet : $\{ P \} C \{ Q \}$

P et Q sont l'*antécédent* (ou *pré-condition*), respectivement *conséquent* (ou *post-condition*) du code C à exécuter. P et Q sont assertions vraies (des affirmations) qui doivent être vérifiées avant, respectivement après l'exécution de C . Pour toutes les constructions d'un langage de programmation impératif (comme par exemple C), *Hoare* définit des règles de déduction (*inference rules*) qui permettent de déterminer le conséquent à partir de l'antécédent (et réciproquement). Ces règles de déduction permettent, de proche en proche, de vérifier la validité de l'intégralité du code. On rappelle que les assertions doivent être définies **avant** l'écriture du code.

- 3. Écrivez le type énuméré `monnaie` qui définit un ensemble à deux éléments `EURO` et `YUAN`.

```
enum monnaie { EURO, YUAN };
```

- 4. Écrivez la fonction `convertir` qui renvoie la conversion en euros ou en yuans d'une somme d'argent. Cette fonction possède deux paramètres, une somme d'argent s à convertir, de type `double` supérieure ou égale à 0 (en yuans ou euros), et la monnaie m de type `enum monnaie`, vers laquelle la somme doit être convertie. On considérera le taux de change 1 yuan = 0,13 euro. Vous utiliserez l'énoncé `switch`, et vous testerez la validité des paramètres. En cas d'erreur, vous signalerez l'erreur par un message sur la sortie d'erreur standard et arrêterez le programme.

```
/*
 * Antécédent : s somme d'argent en yuans ou euros
 * Conséquent : renvoie la conversion de s en m (yuans ou euros)
 */
double convertir(double s, enum monnaies m) {
    if (s<=0) {
        fprintf(stderr, "erreur : la somme doit être >=0\n");
        exit(EXIT_FAILURE);
    }
    const double tauxChange = 0.13; // 1 yuan = 0,13 euro
    switch (m) {
        case EURO : return s*tauxChange;
        case YUAN : return s/tauxChange;
        default :
            fprintf(stderr, "erreur : conversion impossible\n");
            exit(EXIT_FAILURE);
    }
}
```

- 5. Écrivez la fonction `main` qui lit sur l'entrée standard une somme d'argent en euros et qui affiche sur la sortie standard sa conversion en yuans à l'aide de la fonction `convertir` précédente.

```
int main(void) {
    double euros;
    printf("donnez une somme en euros : ");
    scanf("%lf" &euros);
    printf("%f € = %f yuans\n", euros, convertir(euros, YUANS));
    return EXIT_SUCCESS;
}
```

Le jury d'une épreuve de gymnastique est formé de 5 membres qui donnent chacun une note entre 0 et 10 à chaque concurrent. La note finale du gymnaste est la moyenne de ses notes sans la meilleure note, ni la plus mauvaise. Par exemple, si un gymnaste a obtenu les notes 5.1, 8.5, 7.2, 9.3 et 7.2, les notes retenues pour le calcul de la moyenne seront 8.5, 7.2 et 7.2.

- 6. Écrivez la fonction `moyenne` qui prend comme paramètres 5 notes (des réels double) et qui renvoie la moyenne des notes selon la règle de calcul donnée ci-dessus. On tiendra compte de la bonne structuration de l'écriture de cette fonction.

Pour calculer la moyenne, il fallait d'abord écrire les fonctions qui calculent le minimum et le maximum des 5 notes. Ces fonctions peuvent être écrites de 2 façons :

Version 1 :

```
/*
 * Rôle : renvoie le minimum des 5 réels a, b, c, d, et e
 */
```

```

*/
double min(double a, double b, double c, double d, double e) {
    int min = a;
    if (b<min) min=b; // min = min(a,b)
    if (c<min) min=c; // min = min(a,b,c)
    if (d<min) min=d; // min = min(a,b,c,d)
    if (e<min)
        // e = min(a,b,c,e)
        return e;
    // min = min(a,b,c,e)
    return min;
}
/*
* Rôle : renvoie le mazimum des 5 réels a, b, c, d, et e
*
*/
double max(double a, double b, double c, double d, double e) {
    int max = a;
    if (b>max) max=b; // max = max(a,b)
    if (c>max) max=c; // max = max(a,b,c)
    if (d>max) max=d; // max = max(a,b,c,d)
    if (e>max)
        // e = max(a,b,c,e)
        return e;
    // max = max(a,b,c,e)
    return max;
}

```

Version 2 :

```

// Rôle : renvoie le minimum de réels
double min(double a, double b) { return a<b ? a : b; }
/*
* Rôle : renvoie le minimum des 5 réels a, b, c, d, et e
*
*/
double min5(double a, double b, double c, double d, double e) {
    return min(a, min(b, min(c, min(d,e))));
}

// Rôle : renvoie le mazimum de réels
double max(double a, double b) { return a>b ? a : b; }
/*
* Rôle : renvoie le mazimum des 5 réels a, b, c, d, et e
*
*/
double max5(double a, double b, double c, double d, double e) {
    return max(a, max(b, max(c, max(d,e))));
}

```

Maintenant, la fonction `moyenne` s'écrit simplement :

```

/*
* Antécédent : n1, n2, n3, n4 et n5, 5 double ∈ [0;10]
* Rôle : renvoie la moyenne des 5 notes n1, n2, n3, n4 et n5
*          sans compter la meilleure et la moins bonne des notes

```

```

*/
double moyenne(double n1, double n2, double n3, double n4, double n5) {
    return (n1+n2+n3+n4+n5-min5(n1,n2,n3,n4,n5)-max5(n1,n2,n3,n4,n5))/3;
}

```

Le produit de deux entiers x et y consiste à sommer y fois la valeur x .

$$x \times y = \underbrace{x + x + \dots + x}_y \text{ fois}$$

Toutefois, on peut améliorer cet algorithme rudimentaire en multipliant x par deux et en divisant y par deux, chaque fois que y est pair. Les opérations de multiplication et de division par deux sont des opérations très efficaces puisqu'elles consistent à faire un décalage de 1 bit vers la gauche ou vers la droite.

- 7. Écrivez la fonction `int produit(int x, int y)` basée sur cet algorithme. Vous indiquerez clairement le/les invariant(s) de boucle et les affirmations qui prouvent la validité de votre fonction.

```

/* antécédent : x et y >=0
* rôle       : renvoie x*y
*/
int produit(int x, int y) {
    // soit x = a, et y = b
    int p = 0;
    // a * b = p + x * y
    while (y>0) {
        // a * b = p + x * y et y>0
        while ((y & 1) == 0) { // y est pair
            // a * b = p + x * y et y = (y / 2) * 2 > 0
            // a * b = p + 2x * (y / 2) et y = (y / 2) * 2 > 0
            y >>= 1;
            // a * b = p + 2x * y
            x <<= 1;
            // a * b = p + x * y
        }
        // a * b = p + (x-1) * y + y et y>0 et y impair
        p += x;
        // a * b = p + x * y-1 et y impair
        y--;
        // a * b = p + x * y
    }
    // y = 0 et a * b = p + x * y = p
    return p;
}

```