Université de Nice-Sophia Antipolis — POLYTECH
ELEC3 — 2016–2017

Examen de Langage C

Durée : 1h
Aucun document autorisé
Mobiles interdits Documents non autorisés

Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives et d'invariant, ainsi que les noms donnés aux variables, et la bonne indentation rentreront pour une part importante dans l'appréciation du travail.

▶ 1. Expliquez ce qu'est un « paramètre formel ».

Question sur 2 pts : Un paramètre formel est un nom qui désigne une entité abstraite, qui représente une donnée ou un résultat, sur laquelle une fonction ou une procédure procède un calcul.

.....

▶ 2. Combien de mode(s) transmission des paramètres existe(nt) en C ? Expliquez de façon synthétique ce ou ces mécanisme(s) de transmission des paramètres.

Question sur 2 pts : En C, il n'exsite qu'un seul mode de transmission des paramètres, la transmission par valeur. Lors d'un appel de fonction ou de procédure, la transmission par valeur consiste à affecter, après évaluation, la valeur du paramètre effectif à son paramètre formel correspondant. Ce mode de transmission est utilisé pour les paramètres données.

.....

▶ 3. Écrivez en C la fonction bissextile qui teste si une année passée en paramètre est bissextile ou pas.

Question sur 1 pt :

```
/* Rôle : vérifie si l'année est bissextile ou pas */
int bissextile(int annee) {
  return annee%4==0 && annee%100!=0 || annee%400==0;
}
```

.....

▶ 4. À l'exclusion de toute autre fonction, écrivez en C la fonction `somme` qui renvoie la somme du plus petit et du plus grand nombre parmi 3 nombres entiers passés en paramètre.

Question sur 4 pts :

```
/*
 * Antécédent : a,b et c 3 entiers quelconques
 * Rôle : retourne la somme du min et du max de a, b et c
 */
int somme(int a, int b, int c) {
  if (a<b)
    if (b<=c)
      /* a < b <= c */
      return a + c;
    else
      /* a < b et c < b */
      if (c>a)
        /* a < c < b */
        return a + b;
      else
        /* c <= a < b */
        return c + b;
  else
    /* a >= b */
    if (c>a)
      /* b <= a < c */
      return b + c;
    else
      /* a >= b et c<=a */
      if (c>b)
        /* b < c <= a */
        return b + a;
      else
        /* c <= b <= a */
        return c + a;
}
```

.....

▶ 5. Déclarez le type énuméré `Couleurs` avec 4 couleurs de votre choix, plus les 3 couleurs BLEU, BLANC et ROUGE, *puis* déclarez la variable `c` de ce type initialisée à `ROUGE`.

Question sur pt :

```
enum Couleurs { VERT, JAUNE, BLEU, ORANGE, BLANC, ROUGE, VIOLET};
enum Couleurs c = ROUGE;
```

.....

▶ 6. Déclarez le type `booleen`, ensemble à deux valeurs `FAUX` et `VRAI`.

Question sur 1 pt :

```
enum booleen { FAUX, VRAI };
```

.....

▶ 7. En utilisant un énoncé **switch**, écrivez la fonction `dansDrapeau` qui renvoie `VRAI` si la couleur passée en paramètre est l'une des 3 couleurs du drapeau français et `FAUX` sinon. Vous traiterez aussi le cas d'erreur.

Question sur 3 pts :

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

```c
enum Couleurs { VERT, JAUNE, BLEU, ORANGE, BLANC, ROUGE, VIOLET};

/*
 * Rôle : retourne VRAI si c est égal à BLEU, BLANC ou ROUGE
 *         et FAUX sinon
 */
enum booleen dansDrapeau(enum Couleurs c) {
  switch (c) {
    case BLEU :
    case BLANC :
    case ROUGE : return VRAI;
    case VERT:
    case JAUNE:
    case ORANGE:
    case VIOLET: return FAUX;
    default : fprintf(stderr, "Couleur inconnue\n");
  }
}
```

.....

▶ 8. On désire calculer la racine carrée d'un nombre réel x par la méthode de Héron l'Ancien [1]. Pour cela, on calcule la suite $r_n = (r_{n-1}+x/r_{n-1})/2$ jusqu'à obtenir une approximation $r_n = \sqrt{x}$ telle que $r_n - x/r_n$ est inférieur à un $\mathcal{E}$ donné (*e.g.* 10^{-15}). Vous pourrez choisir $r_0 = 1$. À l'aide de l'énoncé **do-while**, écrivez la fonction `rac2` qui renvoie la racine carrée de son paramètre (un **double**) selon la méthode de Héron l'Ancien.

Question sur 5 pts :

```c
// algorithme trouvé par Héron d'Alexandrie au Ie siècle avant JC
// f(x) = 1/2(x + a/x)
// u_0 = α > 0, u_{n+1} = f(u_n), u_n ⩾ √a
// v_n = a/u_n
// e_n = u_n − v_n  et 0 ⩽ u_n − √(a) ⩽ e_n
double rac2(double a) {
  const double epsilon = 1e-15;
  double r = 1;
  do
    r = (r + a/r)/2;
    // Invariant : 0 ⩽ u_n − √(a) ⩽ e_n
  while (r-a/r > epsilon);
  return r;
}
```

.....

1. Héron l'Ancien, mathématicien et mécanicien grec du I^{er} siècle.