Université de Nice-Sophia Antipolis — POLYTECH
ELSE3-FISE — 2025–2026

Examen de Info C

Durée : 1h
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.

► 1. Combien de mode(s) transmission des paramètres existe(nt) en C ? Cochez une seule réponse !

- ☐ un seul, la transmission par référence.
- ☐ un seul, la transmission par valeur.
- ☐ deux, transmissions par valeur et par référence.
- ☐ zéro.

Question sur 1 pt

Réponse : **un seul** mode de transmission **par valeur**.

.....

► 2. Dans l'appel de procédure `lire(x)`, `x` est

- ☐ un paramètre *donné* ;
- ☐ un paramètre *résultat* ;
- ☐ un paramètre *donné* et *résultat*.
- ☐ zéro.

Question sur 1 pt

Réponse : `x` est paramètre *résultat*.

.....

► 3. Expliquez de façon claire et synthétique la différence entre les énoncés *tantque*, *répéter* et *pourtout*.

Question sur 2 pts

Ce sont 3 énoncés itératifs qui permettent d'exécuter répétitivement un ou plusieurs énoncés.

Pour les énoncés *tantque* et *répéter* le nombre d'itérations dépend d'un prédicat d'achèvement booléen. On utilise l'énoncé *tantque* lorsque le nombre d'itérations peut être, au minimum, égal à 0, et l'énoncé *répéter* lorsqu'il y a au moins une itération. Pour ces deux énoncés, il faut s'assurer de la finitude la boucle, c'est-à-dire que le prédicat booléen est vérifié. En revanche, l'énoncé *pourtout* est utilisé lorsqu'on connaît statiquement (à l'avance) le nombre d'itérations. Cet énoncé ne possède pas de prédicat d'achèvement booléen, et garantit donc sa finitude. Toutefois, l'énoncé **for** de C, n'est pas un « véritable » énoncé *pourtout*, puisqu'il contient un prédicat d'achèvement booléen, et qu'il faudra donc vérifier sa finitude. Enfin, ces 3 énoncés itératifs possèdent un invariant de boucle qui décrit leur sémantique.

.....

► 4. À partir de l'antécédent suivant :

```
// jour, mois et année : 3 variables de type int qui représentent une date valide
```

écrivez en C le fragment de code qui calcule la date de la veille. Vous pouvez utiliser la fonction `joursDansMois` qui renvoie le nombre de jours d'un mois d'une année donnée.

Question sur 3 pts

```
// jour, mois et année : 3 entiers qui représentent une date valide
if (jour>1)
  jour--;
else
  // 1er jour du mois
  if (mois>1)
    // la veille est le dernier jour du mois précédent
    jour = joursDansMois(--mois, année);
  else {
    // 1er jour de l'année ⇒ la veille est le 31/12 de l'année précédente
    jour = 31;
    mois = 12;
    année--;
  }
// jour, mois et année est la date de la veille
```

.....

► 5. Écrivez en C une fonction `somme` qui lit sur l'entrée standard `n` entiers (**int**) et renvoie leur somme. Cette fonction possède un seul paramètre, le nombre `n` d'entiers > 0.

Question sur 3 pts

```
/*
 * antécédent : n>0
 * rôle : lit n entiers sur l'E/S et renvoie leur somme
 */
int somme(const int n) {
  assert(n>0);
  int som=0;
  for (int i=0; i<n; i++) {
      int x;
      scanf("%d", &x);
      som+=x;
```

```
    // ∀k ∈ [0,i], som = ∑_{k=0}^{i} x_k
  }
  // ∀k ∈ [0,n[, som = ∑_{k=0}^{i=n-1} x_k
  return som;
}
```

.....

▶ 6. Écrivez la fonction booléenne `estPremier` qui teste si son paramètre `n` (un **int** $\geqslant 2$) est premier ou non. *Rappel* : un nombre premier admet uniquement deux diviseurs *distincts* 1 et lui-même. Pensez à minimiser le nombre d'itérations !

Question sur 4 pts

```
/*
 * Antécédent : n⩾2
 * Rôle : renvoie true si n est premier et false sinon
 */
bool estPremier(const int n) {
  assert(n>=2);
  const int rac2N = sqrt(n);
  for (int i=2 ; i<=rac2N ; i++) {
    if ((n%i) == 0)
      // i est un diviseur ⇒ n non premier
      return false;
    // Invariant : ∀k ∈ [2;i], n mod k ≠ 0
  }
  // Invariant : ∀k ∈ [2;⌊√n⌋], n mod k ≠ 0 ⇒ n est premier
  return true;
}
```

.....

En C, un *entier non signé* peut être dénoté de façon décimale (une suite de chiffres de 0 à 9), de façon octale (une suite de chiffres de 0 à 7 préfixée par 0) ou de façon hexadécimale (une suite de chiffres de 0 à 9 et de lettres de A à F ou de a à f préfixée par $0x$ ou $0X$). Les notations suivantes sont valides : 8764 0765 $0xFF$ $0xFa3$ et correspondent aux entiers 8764 501 255 4003.

▶ 7. Écrivez la fonction `lireEntier` qui calcule et renvoie le prochain entier non signé lu sur l'entrée standard. Cet entier peut être précédé par des espaces. Vous pourrez utiliser les fonctions `isspace`, `isdigit` et `isxdigit` qui testent si leur paramètre (un caractère) est, respectivement, un espace, un chiffre décimal (`'0'-'9'`) et un chiffre hexadécimal (`'0'-'9'-'A'-'F'/'a'-'f'`). Utilisez la fonction `assert` pour traiter les cas d'erreur. Cette fonction `lireEntier` possède l'en-tête suivant :

```
/* Antécédent : le caractère courant de l'entrée standard est un espace ou un chiffre [0-9]
 * Rôle : renvoie l'entier non signé lu au format C
 */
int lireEntier(void) {
```

Question sur 6 pts

```
/*
 * Rôle : renvoie true si c est un chiffre octal [0-7] ]et false sinon
 */
```



```
bool isoctal(const int c) {
  return c>='0' && c<='7';
}

/*
* Antécédent : base = 8, 10 ou 16
 * Rôle : renvoie true si le c est un chiffre de la base b et false sinon
 */
bool dansLaBase(const int c, const int b) {
  switch (b) {
     case 8  : return isoctal(c);
     case 10 : return isdigit(c);
     case 16 : return isxdigit(c);
  }
  // pour faire plaisir au compilateur
  return false;
}

/* Antécédent : un entier non-signé, en notaton octale, décimale ou hexadécimle,
 *              éventuellement précédé par des espaces, est disponible sur l'E/S
 * Rôle : renvoie l'entier non signé lu au format C
 */
int lireEntier(void) {
  int base = 10;  // nombre décimal par défaut
  int c;
  // sauter les éventuels espaces
  while (isspace(c=getchar()));
  // c est un chiffre
  if (c=='0')
    // lire un nombre octal ou hexadécimal
    if ((c=getchar()) =='X' || c == 'x') {
      // notation hexadécimale
      base = 16;
      c=getchar();
    }
    else
      // notation octale
      base = 8;
  //
  int n=0;
  // c est un chiffre 0-9/A-F/a-f OU un caractère quelconque
  // i.e. cas où il n'y avait qu'un seul 0 sur l'entrée standard
  while (dansLaBase(c, base)) {
    // le chiffre est valide dans sa base
    n = n*base + (isdigit(c) ?  c-'0' : toupper(c)-'A'+10);
    // passer au caractère suivant
    c=getchar();
  }
  // c n'est pas un chiffre de la base
  // le remettre dans le fichier d'entrée standard
  ungetc(c, stdin);
  return n;
}
```

.....