Université de Nice-Sophia Antipolis — POLYTECH
ELSE3-FISE — 2024–2025

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. Expliquez ce qu'est un langage procédural et citez les noms d'au moins deux tels langages ?

Question sur 1 pt : Le langage C est un langage procédural, c'est-à-dire qu'un programme écrit dans ce langage est structuré autour des actions représentées par des procédures (ou des fonctions).

.....

▶ 2. L'appel de procédure `lire(x)` permet de lire une donnée sur l'entrée standard : Quelle est la nature du paramètre `x` ? Expliquez.

Question sur 1 pt : `x` est un paramètre effectif résultat, il désignera la donnée lue et placée en mémoire centrale.

.....

▶ 3. Soit la procédure C : **void** `p(int x) {x++;}`. Soit le code : **int** `y=1; p(y);`. Quelle est la valeur de `y` après l'appel de `p` ? Expliquez.

Question sur 1 pt : Après l'appel de p, `y` reste égal à 1 car le mode de transmission en C est uniquement par valeur, adaptée aux paramètres *donnée* et pas aux paramètres *résultat*.

.....

▶ 4. Soit le fragment de code algorithmique suivant :

```
si x=1 alors y ← 2 finsi
si x=2 alors y ← 3 finsi
si x≠1 et x≠2 alors y ← 4 finsi
```

Expliquez pourquoi ce code est mal écrit, et récrivez-le correctement.

Question sur 2 pts : ce fragment de code est mal écrit car il fait des tests sur des conditions qui s'excluent mutuellement. Ainsi, par exemple, si `x=1`, le code testera inutilement `x=2`. La bonne écriture doit utiliser la partie **sinon** de l'énoncé conditionnel **si**.


```
si x=1 alors y ← 2
sinon { x≠1 }
   si x=2 alors y ← 3
   sinon { x≠1 et x≠2 }
         y ← 4
   finsi
finsi
```

.....

▶ 5. Dans l'affectation : `{ A }` `v ← e` `{ C }`, qui affecte le résultat de l'évaluation de l'expression `e` à la variable `v`, comment détermine-t-on le conséquent `{C}` à partir de l'antécédent `{A}` ?

Question sur 1 pt : Le conséquent `{C}` est obtenu en remplaçant toutes les apparitions de `e` par `v` dans l'antécédent `{A}`.

.....

▶ 6. Soit la suite d'affections avec l'antécédent initial :

```
{ y = x², d = 2x − 1}
d ← d+2
y ← y+d
d ← d+2
y ← y+d
```

Donnez chaque conséquent qui suit chacune des affectations. Que calcule cette suite d'affectations ?

Question sur 2 pts :

```
{ y = x², d = 2x − 1}
d ← d+2
{ y = x², d = 2x + 1}
{ y + d = (x + 1)², d = 2x + 1}
y ← y+d
{ y = (x + 1)², d = 2x + 1}
d ← d+2
{ y = (x + 1)², d = 2x + 3}
{ y + d = (x + 1)² + 2x + 3, d = 2x + 3}
{ y + d = (x + 2)², d = 2x + 3}
y ← y+d
{ y = (x + 2)², d = 2x + 3}
```

On remarque que cette suite d'affectations permet le calcul de $(x+2)^2$. Si on applique i fois cette suite, on calcule $(x+i)^2$.

.....

On représente les coordonnées d'un point p du plan cartésien par deux réels (**double**) x et y tels que $p = (x, y)$.

▶ 7. Écrivez en C la fonction `distance` qui renvoie la distance entre 2 points. Les coordonnées des 2 points, de type **double**, sont passées en paramètre.

Question sur 2 pts :

```c
/*
 * Rôle : renvoie la distance entre 2 de points du plan
 *        de coordonnées (x1,y1) (x2,y2)
 */
double distance(const double x1, const double y1,
                const double x2, const double y2)
{
  return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
}
```

.....

▶ 8. Écrivez en C la fonction `coefDir` qui calcule et renvoie le coefficient directeur deux points (x_A, y_A) et (x_B, y_B). Cette fonction possède 4 paramètres qui représentent les coordonnées des deux points. On rappelle que le coefficient directeur a d'une droite (AB), non parallèle à l'axe des ordonnées, est $a = y_B - y_A / x_B - x_A$. *

Question sur 2 pts :

```c
/*
 * Antécédent : xA≠xB;
 * Rôle : renvoie le coefficient directeur de la droite qui passe par les 2 points
 *        distincts de coordonnées (xA,yA) et (xB,yB)
 */
double coefDir(const double xA, const double yA,
               const double xB, const double yB)
{
  assert(xA!=xB);
  // calculer et renvoyer coefficient directeur
  return (yB-yA)/(xB-xA);
}
```

.....

▶ 9. On souhaite savoir si 3 points distincts A, B et C de ce plan sont alignés ou non. Écrivez en C la fonction booléenne `sontAlignés` qui teste si trois points du plan de coordonnées (x_A, y_A), (x_B, y_B) et (x_C, y_C) sont alignés ou non. En plus du cas général, vous prendrez en considération les cas particuliers où les points sont alignés verticalement ou horizontalement.

Question sur 4 pts :

Pour vérifier que les 3 points A, B et C sont alignés, on vérifie que les coefficients directeurs des droites qui passent par AB et BC sont égaux .

```c
/* Antécédent : 3 points distincts de coordonnées (xA,yA), (xB,yB) et (xC,yC)
 * Conséquent : renvoie 1 si les 3 points sont alignés et 0 sinon
 */
int sontAlignés(const double xA, const double yA, const double xB,
                const double yB, const double xC, const double yC)
{
  return coefDir(xA, yA, xB, yB) == coefDir(xB, yB, xC, yC);
}
```


.....

▶ 10. Sur l'entrée standard, on a un entier (**int** > 0) égal au nombre coordonnées de points (2 réels **double**) qui le suivent. Écrivez en C le programme qui lit l'entier puis la suite de points, et qui écrit sur la sortie standard le point le plus éloigné du point d'origine $(0, 0)$. Par exemple, si l'entrée standard contient :

```
4
1.9  4.9
-8.7 24.7
12.0 34.4
-1.6 13.8
```

le programme écrira : `Le point max est (12.0,34.4)`

Question sur 4 pts :

```c
int main(void) {
  double x_max=0, y_max=0, dist_max=0.0;
  // lire le nombre de points n>0
  int n;
  scanf("%d", &n);
  if (n<1) {
    fprintf(stderr, "n doit être >0\n");
    return EXIT_FAILURE;
  }
  // n⩾1
  for ( ; n>0; n--) {
    double x, y;
    // lire le point courant de la suite
    scanf("%lf %lf", &x, &y);
    // vérifier si sa distance est
    const double dist_x_y = distance(0,0,x,y);
    if (dist_x_y>dist_max) {
      dist_max = dist_x_y;
      x_max = x;
      y_max = y;
    }
    //∀k ∈ [1, i],distance(x_max, y_max)⩾ distance(x_k, y_k)
  }
  //∀k ∈ [1, n],distance(x_max, y_max)$geqslant distance(x_k, y_k)
  printf("le point max est (%.1lf,%.1lf)\n", x_max, y_max);
  //
  return EXIT_SUCCESS;
}
```