Université de Nice-Sophia Antipolis — POLYTECH
ELEC3 — 2018–2019

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. Soit les quatre affectations suivantes :

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

Appliquez la règle de déduction de l'instruction d'affectation et écrivez la suite d'affirmations nécessaires pour obtenir le conséquent final.

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

.....

▶ 2. Un magasin de reprographie propose un tarif dégressif. De 1 à 20 photocopies, le prix est 10 centimes l'unité, de 21 à 99 le prix est 8 centimes l'unité et au delà de 100 le prix est 6 centimes l'unité. Dans le calcul du tarif final, le prix unitaire est le même quel que soit le nombre de photocopies. Écrivez un programme C qui demande à l'utilisateur le nombre de photocopies qu'il veut réaliser et qui affiche le prix qu'il devra payer.

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



```c
int main(void) {
  int nbPhotocopies, prixUnitaire;

  scanf("%d", &nbPhotocopies);
  if (nbPhotocopies<=0) {
    fprintf(stderr, "erreur : le nombre de photocopies doit être >0\n");
    return EXIT_FAILURE;
  }
  // nbPhotocopies > 0
  if (nbPhotocopies<=20)
    // 1⩽nbPhotocopies⩽20
    prixUnitaire = 10;
  else
    if (nbPhotocopies<100)
      // 20<nbPhotocopies<100
      prixUnitaire = 8;
    else // nbPhotocopies⩾100
      prixUnitaire = 6;
  // écrire le prix final en euros
  printf("prix = %.2lf euros\n", nbPhotocopies*prixUnitaire/100.0);

  return EXIT_SUCCESS;
}
```

.....

On considère les points du plan cartésien. Chaque point possède une coordonnée (x, y).

▶ 3. Écrivez en C la fonction `distance` qui calcule la distance entre 2 points du plan. On rappelle que pour 2 points (x_1, y_1) et (x_2, y_2), la distance qui les sépare est égale à $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

```c
/*
 * Antécédent : (x1, y1) et (x2, y2) deux points du plan cartésien
 * Rôle : renvoie  la distance entre les deux points
 */
double distance(double x1, double y1, double x2, double y2) {
  return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
}
```

.....

On considère qu'un vrai triangle (c'est-à-dire non plat) est représenté par 3 points distincts du plan cartésien. Chaque point possède une coordonnée (x, y).

▶ 4. Écrivez en C la fonction booléenne `estUnVraiTriangle` qui teste si 3 points définissent un vrai triangle. Rappel : la somme des longueurs de deux côtés d'un vrai triangle est toujours strictement supérieure à celle du troisième côté.

```c
/*
 * Antécédent : (x1, y1), (x2, y2) et (x3, y3) 3 points du plan cartésien
 * Rôle : renvoie 1 si les 3 points définissent un vrai triangle, et 0 sinon
 */
int estUnVraiTriangle(double x1, double y1,
                      double x2, double y2,
```

```
                    double x3, double y3)
{
  // calculer les longueurs des 3 cotés du triangle
  double d12 = distance(x1, y1, x2, y2);
  double d13 = distance(x1, y1, x3, y3);
  double d23 = distance(x2, y2, x3, y3);
  // on a un vrai triangle si la somme des longueurs de
  // deux côtés est supérieure à celle du troisième côté
  //
  return d12+d13>d23 && d12+d23>d13 && d13+d23>d12;
}
```

.....

► 5. Expliquez de façon claire et synthétique ce qu'est un invariant de boucle.

Un invariant de boucle est une affirmation toujours vraie quel que soit le nombre d'itérations. Celle qui est placée devant la condition d'arrêt est représentative de l'énoncé itératif, elle décrit sa sémantique.

.....

► 6. Écrivez la fonction `main` qui lit sur l'entrée standard $n \geqslant 0$ triangles, puis écrit sur la sortie standard le nombre triangles vrais lus. Chaque ligne de l'entrée standard contient les coordonnées des 3 points d'un triangle. La toute première ligne de l'entrée standard contient le nombre de triangles à lire.

```
int main(void) {
  int i, n,nbVraisTriangles=0;
  // lire le nombre de triangles à lire
  scanf("%d", &n);
  assert(n>=0);
  i=0;
  while (i<n) {
    // i triangles ont été lus et nbVraisTriangles est le nombre
    // de vrais triangles
    double x1, y1, x2, y2, x3, y3;
    i++;
    scanf("%lf %lf %lf %lf %lf %lf", &x1, &y1, &x2, &y2, &x3, &y3);
    if (estUnVraiTriangle(x1, y1, x2, y2, x3, y3))
      nbVraisTriangles++;
  }
  // n triangles ont été lus et nbVraisTriangle est le nombre
  // de vrais triangles
  printf("%d\n", nbVraisTriangles);

  return EXIT_SUCCESS;
}
```

.....