Final AG43

Durée 1h30
Aucun document autorisé

1) Analyse de code

Soit le code suivant :

```
int globalA = 2;
int globalB = 42;

int operation1(int x, int y){
      return x*y;
}
int operation2(int x, int y){
      globalA = globalA + x;
      return globalA + y;
}

int operation3(int x, int y, int *ptr){
      *ptr = x % y;
      return x / y;
}

int main(){
      int a = 10;
      int b = 5;
      int *ptrA = &a;

      int resultat1 = operation1(a,5);
      int resultat2 = operation2(3, b);
      int resultat3 = operation3(globalB, b, ptrA);
}
```

Donnez l'ensemble des variables modifiées ainsi que leurs valeurs pour chaque appel des fonctions opération 1, 2 et 3. Détaillez votre raisonnement.

2) Application de reconnaissance de quadrilatère

Dans cet exercice vous allez créer un ensemble de fonctions permettant de déduire si un quadrilatère est un parallélogramme, un rectangle, un carré, un losange où est quelconque, à partir des coordonnées de leurs points. Les données à traiter vous serons fournis sous la forme d'une structure détaillée en annexe.

Vous partirez du principe que toutes les bibliothèques dont vous aurez besoin sont déjà incluses dans le programme (stdlib, stdio, math…), les structures de Point et de Quadrilatère sont aussi définies et jointes en annexe.

Il est bien entendu possible de réutiliser les fonctions développer dans les points précédents de l'exercice.

2.1) Création d'un point

Réalisez une fonction permettant la création d'un pointeur vers une structure Point et le retournant, qui prend comme arguments d'entrés ses coordonnées X et Y

2.2) Création d'un quadrilatère

Réalisez une fonction permettant la création d'un pointeur vers une structure Quadrilatère et le retournant, qui prend comme arguments d'entrés ses 4 points A, B, C et D.

2.3) Calcul de la distance entre 2 Points

Réalisez une fonction retournant la distance (de type double) entre 2 structures Points placées en arguments d'entrés.

2.4) Déduction d'un parallélogramme

Réalisez une fonction prenant un Quadrilatère comme paramètre d'entrée et retournant un booléen représentant si le quadrilatère est un parallélogramme.
Propriété géométrique à exploiter : Si les côtés opposés d'un quadrilatère sont de même longueur alors c'est un parallélogramme

2.5) Déduction d'un rectangle

Réalisez une fonction prenant un Quadrilatère comme paramètre d'entrée et retournant un booléen représentant si le quadrilatère est un rectangle.
Propriété géométrique à exploiter : Si les diagonales d'un parallélogramme sont de la même longueur, alors c'est un rectangle.

2.6) Déduction d'un losange

Réalisez une fonction prenant un Quadrilatère comme paramètre d'entrée et retournant un booléen représentant si le quadrilatère est un losange.
Propriété géométrique à exploiter : si un parallélogramme a deux côtés consécutifs égaux, alors c'est un losange

2.7) Déduction d'un carré

Réalisez une fonction prenant un Quadrilatère comme paramètre d'entrée et retournant un booléen représentant si le quadrilatère est un carré.
Propriété géométrique à exploiter : si un quadrilatère est à la fois un losange et un rectangle, alors c'est un carré

2.8) Traitement des données

A l'aides des fonctions DonneeEnAttente, LireDonnee, et EnvoyerDonnee, réalisez la boucle principale de votre programme effectuant le traitement suivant :

Si des données sont en attente de lecture, lire la donnée. A l'aide de vos fonctions créées précédemment, créer une structure DonneeTraite et la remplir avec les informations correspondantes, puis la retourner à l'aide de la fonction EnvoyerDonnee.

Sinon on attend 5 secondes avant de recommencer la boucle principale

Annexe

Les différentes structures préprogrammées sont les suivantes :

```
Struct Point{
      int X;
      int Y;
};
```

```
Struct Quadrilatere
{
   Point *A;
   Point *B;
   Point *C;
   Point *D;
};
```

```
Struct DonneeEntre{
  Int Pts[4][2];
};
```

```
Struct DonneeTraite{
      Quadrilatere *Quad;
      bool Parallelogramme;
      bool Rectangle;
      bool Losange;
      bool Carre;
};
```

Les fonctions suivantes sont déjà programmées :

DonneeEnAttente() : Booleen
Retourne true si des données sont en attente de lecture, false le cas échant.
Exemple :

```
if (DonneeEnAttente()){
     //Des données sont en attente
} else {
     //Pas de données en attente
}
```

LireDonnee() : DonneeEntre*
Retourne les données en attente de traitement
Exemple :

```
DonneeEntre *maDonnee = LireDonnee();
```

EnvoyerDonnee(DonneeTraite* maDonnee) : void
Envoi les données traitées après vos modifications
Exemple :

```
DonneeEntrer *maDonnee = LireDonnee();
DonneeTraite *maDonneeTraite ;
/* Effectuez ici le traitement adéquat */
EnvoyerDonnee(maDonneeTraite);
```

Sleep(int time) : void
Permet d'attendre pendant *time* secondes.