

Examen de Info C

Durée : 1h30
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 Sorte de caractères

- 1. Déclarez le type énuméré `SorteCar` qui définit 4 catégories de caractères : les lettres alphabétiques, les chiffres, les espaces et les autres (le complémentaire). Les valeurs de ce type seront `LETTRE`, `CHIFFRE`, `ESPACE` et `AUTRE`.

```
enum SorteCar { LETTRE, CHIFFRE, ESPACE, AUTRE} ;
```

- 2. Écrivez la fonction `typeDeCar` qui prend en paramètre un caractère et qui renvoie la catégorie du caractère (de type `SorteCar` précédemment défini). Utilisez les fonctions de `ctype.h`.

```
/*
 * Antécédent : c un caractère quelconque
 * Rôle : renvoie la catégorie du caractère c
 */
enum SorteCar typeDeCar(const char c) {
    if (isalpha(c)) return LETTRE;
    if (isdigit(c)) return CHIFFRE;
    if (isspace(c)) return ESPACE;
    return AUTRE;
}
```

Sans utiliser les fonctions de `ctype.h`, on aurait pu écrire :

```
/*
 * Antécédent : c un caractère quelconque
 * Rôle : renvoie la catégorie du caractère c
 */
enum SorteCar typeDeCar(const char c) {
    if (c>='a' && c<='z' || c>='A' && c<='Z')
        return LETTRE;
    if (c>='0' && c<='9') return CHIFFRE;
    if (c==' ' || c=='\n' || c=='\t' || c=='\v' || c=='\f' || c=='\r')
        return ESPACE;
    // tous les autres caractères
}
```

```
return AUTRE;
}
```

2 Plan cartésien

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

- 3. 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.

```
/*
 * 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));
}
```

- 4. Écrivez en C la fonction `pente` qui renvoie la pente d'une droite qui passe par 2 points distincts de coordonnées (x_1, y_1) et (x_2, y_2) .

```
/*
 * Antécédent : x1≠x2;
 * Rôle : renvoie la pente de la droite qui passe par les 2 points
 * distincts de coordonnées (x1,y1) (x2,y2)
 */
double pente(const double x1, const double y1,
            const double x2, const double y2)
{
    assert(x1!=x2);
    // calculer et renvoyer la pente
    return (y2-y1)/(x2-x1);
}
```

- 5. Écrivez en C la fonction `main` qui :
- déclare les variables `Ax`, `Ay`, `Bx`, `By` de type **double** qui représentent les coordonnées de 2 points du plan A et B ;
 - lit sur l'entrée standard 4 réels **double** et les affecte aux variables `Ax`, `Ay`, `Bx`, `By`;
 - écrit sur la sortie standard la distance entre les points A et B ;
 - écrit sur la sortie standard l'équation $y = ax + b$ de la droite qui passe par A et B .
Attention de bien traiter **tous** les cas possibles.

Par exemple, l'exécution de cette fonction main pourra produire (format à respecter) :

A(x,y) ? (3.2,-8)

B(x,y) ? (-5.8,10.23)

distance(A,B) = 20.330590

l'équation de la droite passant par A et B est : $y = -2.03x - 1.52$

```
int main(void) {
    double Ax, Ay, Bx, By;

    // lire les coordonnées du point A
    printf("A(x,y) = ");
    scanf("(%lf,%lf)", &Ax, &Ay);
    // lire les coordonnées du point B
    printf("B(x,y) = ");
    scanf(" (%lf,%lf)", &Bx, &By);

    // écrire sur la S/S standard la distance entre les points A et B
    printf("distance(A,B) = %lf\n", distance(Ax, Ay, Bx, By));

    // déterminer la droite qui passe par A et B
    if (Ax==Bx && Ay==By) {
        fprintf(stderr, "2 points confondus, pas de droite\n");
        return EXIT_FAILURE;
    }
    // les points sont distincts => trouver l'équation de droite
    printf("l'équation de la droite passant par A et B est : ");
    if (Ax==Bx)
        // droite verticale => a=1 et b=Ax
        printf("x = %.2lf\n", Ax);
    else
        // Ax!=Bx
        if (Ay==By)
            // droite horizontale => a=0 et b=Ay
            printf("y = %.2lf\n", Ay);
        else {
            // droite quelconque => calculer la pente a
            double a = pente(Ax, Ay, Bx, By);
            // b = y-ax => calculer b à partir de A(Ax, Ay)
            double b = Ay-a*Ax;
            // écrire l'équation de la droite sur la sortie standard
            printf("y = %.2fx %c %.2f\n", a, (b>0 ? '+' : '-'), fabs(b));
        }
    return EXIT_SUCCESS;
}
```

.....