Année : 2012-2013 Session : 2ème

Programmation impérative

PG108

F. Morandat

Filière : Électronique Année : 1 Semestre : 5
Date de l'examen : 18/04/2013 Durée de l'examen : 2h

Documents : autorisés ☐ non autorisés ☒
Calculatrice : autorisée ☐ non autorisée ☒
Autre : Attention à la clarté du code et l'écriture.
Barème indicatif: 3 ; 4 ; 13

1 Fonctions mystères

Que font ces fonctions ? Les réponses doivent faire une phrase maximum.

```
double mystere1(struct point_t a, struct point_t b) {
    return sqrt(pow(b.x - a.x, 2) + pow(b.y - b.a, 2))
}

int mystere2(const char *a, const char *b) {
    while(*b)
        if(*a++ != *b++)
            return 0;
    return 1;
}

int mystere3(const char *a) {
    int b = 0, c = strlen(a);
    while(b <= c)
        if(a[b ++] != a[c])
            return 0;
    return 1;
}
```

2 Histograme

Nous souhaitons chercher dans un tableau d'octets (0-255) la valeur la plus fréquente. *Si cela vous arrange vous pouvez utiliser un tableau d'entiers, vous devrez cependant expliquer ce qui cela changerait.*

1) Faire une fonction qui permet à l'utilisateur de saisir un tableau de N octets (`void saisir(int N, char tab[])`).
2) Compter le nombre d'occurrence de chaque octet dans un tableau (`void count_occur(int N, char tab[], int occur[])`)
3) Afficher l'occurrence la plus fréquente. (`void print_most_common(int N, int occur[])`)
4) Faire une fonction qui appelle les 3 fonctions précédentes.

3 Problème

SVP lire jusqu'à la fin avant de commencer. Même s'il y a un lien entre les questions, on peut y répondre de façon indépendante (pourvu que les pré-requis soient explicitement écrits). Il vous est ***fortement conseillé*** *de faire des fonctions auxiliaires*

On souhaite modéliser et manipuler des polygones dans le plan réel.

1) On considèrera dans la suite qu'un polygone est en fait une suite de point. Pour simplifier, nous considèrerons que le nombre de points/côtés du polygone est limité à `MAX_POINTS`.
   i) Comment représenter un point (`point_t`) ? Écrire le code correspondant.
   ii) Comment représenter un polygone (`poly_t`) ? Écrire le code correspondant.
   iii) En quoi limiter le nombre de côtés vous simplifie les choses ?
   iv) Définir la constante `MAX_POINTS` à la valeur 10.
2) Étant donnée un polygone quelconque, écrire la fonction `count_point` qui retourne le nombre de points qui le compose.
3) Écrire les fonctions `print_point` et `print_poly` qui affiche respectivement les coordonnés d'un point et et les différentes coordonnés d'un polygone. *Attention* le résultat doit être humainement lisible.
4) Écrire la fonction qui retourne, pour un polygone donné, son périmètre.
5) Écrire la fonction `scan_poly` qui permet de saisir au clavier les coordonnés d'un polygone. Justifiez vos choix. *Attention* cette fonction ne doit autoriser que la saisie d'un polygone valide (par exemple, un segment n'est pas un polygone car il est forcément dégénéré).
6) Écrire la fonction qui **retourne** sous forme de chaine de caractère le type d'un polygone (`"triangle"`, `"quadrilatère"`, ..., `"ennéagone"`, `"décagone"`).
7) Écrire le prédicat `is_equilateral` qui retourne `1` si le polygone est équilatéral (tous les côtés ont la même longueur) et `0` sinon. Cette fonction donne-t-elle toujours un résultat valide ? Justifier.
8) Écrire la fonction qui, étant donné un tableau de polygones, retourne le nombre de polygones équilatéral qu'il contient.
9) Écrire une fonction `main` qui permet de saisir une suite de `n` de polygone (`n` sera passé en argument au programme). La suite de votre programme devra afficher le plus d'information possible sur chacun de ces polygones.