LIFAMI

Contrôle Continu Terminal (Durée : 1h30)

Vendredi 17 mai 2019

NOM :

……
……

PRENOM :

……
……

Numéro Etudiant :

Recommandations : Les documents, calculatrice, téléphone portable sont interdits. La qualité de l'écriture et de la présentation sera prise en compte dans la note finale.

Nombres complexes

Nous allons afficher un polygone régulier. Vous disposez de la structure *Complex* et des fonctions/procédures suivantes, et **de celles-ci uniquement**.

```
struct Complex { float x,y ; } ;
Complex make_complex(float x, float y);
Complex make_complex_exp(float r, float theta); // en radian
Complex operator+(Complex a, Complex b);
Complex operator-(Complex a, Complex b);
Complex operator*(Complex a, Complex b);
```

1. Ecrivez en C++ la fonction *rotate* qui prend en paramètre un *Complex p* à faire tourner, un *Complex* c qui est le centre de rotation, et un réel *angle* **en degré** définissant l'angle de rotation. La fonction calcule et renvoie la rotation du point *p* tourné d'un angle *angle* autour du point *c*.

2. Ecrivez en C++ la procédure *draw_polygon* qui prend en paramètre 4 entiers *cx, cy, r* et *n* et qui dessine un polygone régulier à *n* côtés, centré autour du point *(cx,cy)* et de « rayon » *r* qui est la distance entre le centre et les points (voir Figure 1).
Remarque : utilisez la fonction *rotate* pour faire tourner le point de départ qui sera placé en *(cx+r,cy)*.

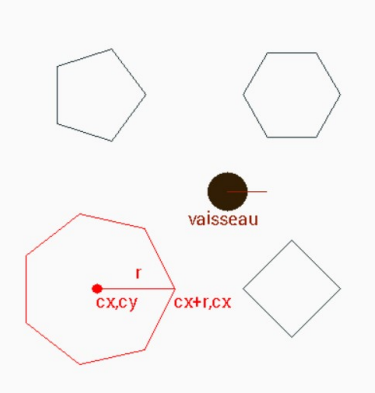


Figure 1. Les polygones de la question 2 et le vaisseau de la question 3

3. Nous voulons déplacer un vaisseau au clavier comme si nous étions assis au poste de pilotage. Le vaisseau avance dans sa direction de déplacement (vecteur) avec une certaine vitesse (scalaire). La flèche gauche fait tourner cette direction d'avancement vers la gauche et la flèche droite fait tourner cette direction vers la droite, les flèches haut et bas faisant accélérer et freiner. Nous utiliserons les nombres complexes et la fonction *rotate* écrite précédemment. Le vaisseau est composé d'un complexe *pos* représentant sa position, un complexe *dir* représentant sa direction de déplacement toujours de norme 1, et un réel *vit* représentant sa vitesse d'avancement. Voir la figure 1. Ecrivez en C++ la structure *Vaisseau* et ses données.

4. Ecrivez en C++ le sous-programme *dessine_vaisseau* qui dessine le vaisseau avec un cercle de rayon 20, de couleur **rouge** et une ligne de couleur **bleu** allant de son centre vers la direction d'avancement *dir*.

5. Ecrivez en C++ le sous-programme *update_vaisseau* qui déplace la position *pos* du vaisseau dans la direction *dir* d'une distance *vitesse*. La physique nous indique que
*Vecteur_nouvelle_position = Vecteur_ancienne_position + scalaire_vitesse * vecteur_direction*
De plus, en fonction des touches appuyées, la fonction fait tourner la direction *dir* à droite ou à gauche ; ainsi qu'accélérer ou freiner en changeant *vitesse*.
Vous utiliserez : *if (isKeyPressed(SDLK_UP))*

Interpolation

6. Soit une barre métallique de longueur *L* dont la température est mesurée aux 2 extrémités A et B. La température en A est notée t_A et la température en B est notée t_B. En supposant que la température suit une fonction linéaire (une droite) entre A et B, écrivez en C++ la fonction qui calcule par interpolation linéaire puis renvoie la température du point P en fonction des paramètres suivants : AP la longueur entre le point A et le point P ; tA et tB les températures de A et B.

Evolution temporelle des systèmes radioactifs

Nous allons nous intéresser à l'évolution temporelle des systèmes radioactifs. On appellera ici n le nombre moyen de désintégrations durant une durée dt. A la date t, l'échantillon radioactif comporte N noyaux non désintégrés. A la date t+dt, l'échantillon contient donc N-n noyaux non désintégrés. La variation du nombre de noyaux radioactifs de l'échantillon se note :

$$\mathrm{dN(t)} = \mathrm{N(t+dt)} - \mathrm{N(t)} = \mathrm{N} - \mathrm{n} - \mathrm{N} = -\mathrm{n}$$

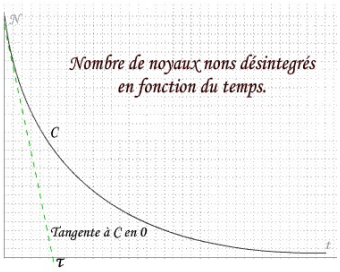
L'activité moyenne A(t) exprimée en becquerel (Bq) représente le nombre moyen de désintégrations que l'échantillon produit par seconde.

$$A(t) = \frac{n}{dt} = -\frac{dN(t)}{dt} = -(\frac{N(t+dt) - N(t)}{dt}) \qquad éq.(a)$$

On montre également que l'activité A(t) est proportionnelle au nombre de noyaux non désintégrés à la date t, c'est à dire à

$$A(t) = \lambda \mathrm{N(t)} \qquad \text{éq. (b)}$$

Cette équation différentielle (qui comporte dans ses termes la fonction et sa dérivée) trouve une solution analytique que vous avez peut-être vue au lycée. Nous allons nous intéresser ici à son calcul numérique par itération.



7. A partir des équations (a) et (b) écrivez le calcul qui conduit à avoir le nombre de noyaux suivant N(t+dt) en fonction du nombre de noyau courant N(t) et de λ.

8. Ecrivez en C++ la fonction *N_suivant* qui calcule le nombre de noyaux de l'itération suivante avec l'équation de la question précédente. Cette fonction prend en paramètre le nombre d'atomes courant *N*, la constante *lambda*, et le pas de temps *dt*.

9. Ecrivez en C++ la fonction *demi_vie* qui calcule la demi-vie d'un type de noyau en fonction de la constante *lambda*. La demi-vie, également appelée période radioactive, est le temps au bout duquel la moitié des noyaux radioactifs d'une source se sont désintégrés. Le pseudo algorithme est le suivant
    1. Initialiser N_init à un grand nombre de noyau ; temps à 0 ; dt à un pas de temps raisonnable et N à N_init
    2. Tant que N est plus grand que la moitié de *N_init* répéter
        Calculer le nombre de noyaux suivant *N* avec la fonction précédente
        Ajouter le *pas de temps* à *temps*
    3. Retourner *temps*