Algorithmique et Base de la programmation

Durée : *1h30*
Documents autorisés : **AUCUN**

Remarques :
- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.

1 Traitement du signal : Filtre IIR (8 points)

« Un filtre à réponse impulsionnelle infinie ou filtre RII (en anglais *infinite impulse response filter* ou *IIR filter*) est un type de filtre électronique caractérisé par une réponse basée sur les valeurs du signal d'entrée ainsi que les valeurs antérieures de cette même réponse.

De façon générale le filtre à réponse impulsionnelle infinie est décrit par l'équation [...] suivante où x représente les valeurs du signal d'entrée et y les valeurs du signal de sortie. »[1]

$$y[i] = \frac{1}{N}\sum_{k=0}^{N} x[i-k] - \frac{1}{M}\sum_{k=1}^{M} y[i-k]$$

NB : On considère que si $i-k$ est « hors » du signal x, alors $x[i-k]$ vaut 0 (idem pour y).

Soit le type `Signal` :
Type Signal = **Structure**
donnees : **Tableau**[1..MAX] **de Reel**
nbDonnees : **Naturel**
finstructure

Proposez l'analyse descendante, la conception préliminaire et la conception détaillée du problème `filtreIRR`, qui calculera un nouveau signal à partir d'un signal et de deux naturels n et m.

Solution proposée :
Analyse :
filtreIIR : Signal × **NaturelNonNul** × **NaturelNonNul** → Signal
sommePartieSignal : Signal × **NaturelNonNul** × **NaturelNonNul** → **Reel**
maxNaturel : **Naturel** × **Naturel** → **Naturel**
Conception préliminaire :
- **fonction** filtreIIR (s : Signal, n,m : **NaturelNonNul**) **:** Signal
- **fonction** sommePartieSignal (s : Signal, d,f : **NaturelNonNul**) **: Reel**
  ⌊**précondition(s)** d≤f et d≤s.nbDonnees et f≤s.nbDonnees
- **fonction** maxNaturel (a,b : **Naturel**) **: Naturel**

Conception détaillée :
fonction sommePartieSignal (s : Signal, d,f : **NaturelNonNul**) **: Reel**
⌊**précondition(s)** d≤f et d≤s.nbDonnees et f≤s.nbDonnees
Déclaration resultat : **Reel**
i **Naturel**
debut
resultat ← 0
pour i ←d **à** f **faire**
resultat ← resultat+s.donnees[i]

[1] source : wikipédia, version de l'équation simplifiée

```
    finpour
    retourner resultat
fin
fonction filtreIIR (s : Signal, n,m : NaturelNonNul) : Signal
    Déclaration  resultat : Signal
                 i,j : Naturel
                 temp : Reel
debut
    resultat.nbDonnees ← s.nbDonnees
    pour i ←1 à s.nbDonnees faire
        resultat.donnees[i] ← sommePartieSignal(s,maxNaturel(1,i-n),i)/n
        -sommePartieSignal(resultat,maxNaturel(1,i-m),i-1)/m
    finpour
    retourner resultat
fin
```

2 Un peu de récursivité (6 points)

Supposons que la procédure suivante permette de dessiner un cercle sur un graphique (variable de type `Graphique`) :

- **procédure** cercle **(E/S** g : Graphique,**E** xCentre,yCentre,rayon : **Reel)**

2.1 Compréhension

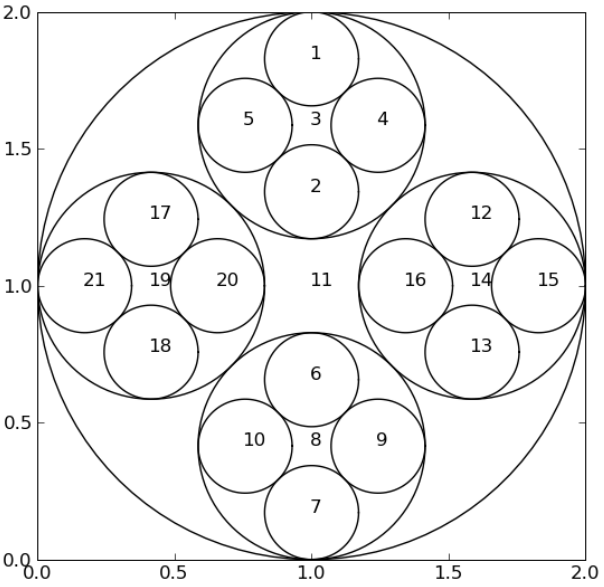
Soit l'algorithme suivant :

```
procédure cercles (E/S g : Graphique,E x,y,r : Reel, n : Naturel)
    Déclaration  rTemp : Reel
debut
    si n>0 alors
        rTemp ← r/(1+racineCarree(2))
        cercles(g,x,y+rTemp*racineCarree(2),rTemp,n-1)
        cercles(g,x,y-rTemp*racineCarree(2),rTemp,n-1)
        cercle(g,x,y,r)
        cercles(g,x+rTemp*racineCarree(2),y,rTemp,n-1)
        cercles(g,x-rTemp*racineCarree(2),y,rTemp,n-1)
    finsi
fin
```

L'instruction `cercles(g,1.0,1.0,1.0,3)` permet d'obtenir le graphique de la figure présentée sur la dernière page (à rendre avec votre copie avec vos nom et prénom).

Numérotez les cercles (numéro à mettre au centre du cercle) suivant leur ordre d'apparition sur le graphique.

Solution proposée :



2.2 Construction

Proposez un autre algorithme de la procédure `cercles` qui, pour la même instruction `cercles(g,1.0,1.0,1.0,3)`, affiche les cercles dans l'ordre proposé par la figure 1.

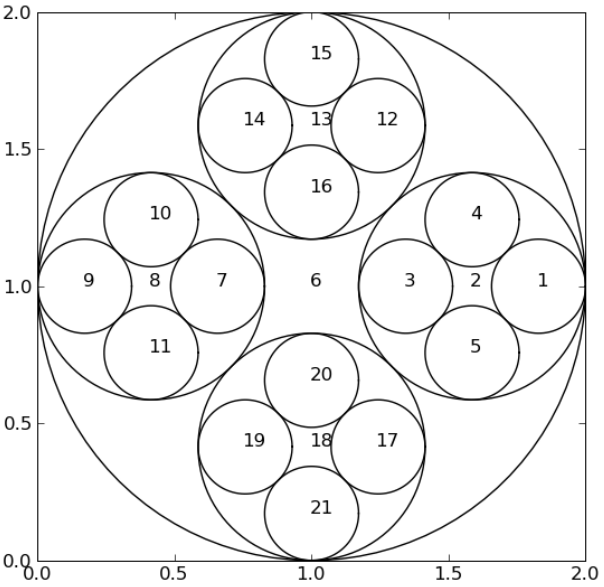


FIG. 1 – Résultat d'un autre algorithme pour `cercles`

Solution proposée :

procédure cercles **(E/S** g : Graphique,**E** x,y,r : **Reel**, n : **Naturel)**

Déclaration rTemp : **Reel**

debut

si n>0 **alors**

rTemp ← r/(1+racineCarree(2))

```
        cercles(g,x+rTemp*racineCarree(2),y,rTemp,n-1)
        cercle(g,x,y,r)
        cercles(g,x-rTemp*racineCarree(2),y,rTemp,n-1)
        cercles(g,x,y+rTemp*racineCarree(2),rTemp,n-1)
        cercles(g,x,y-rTemp*racineCarree(2),rTemp,n-1)
    finsi
fin
```

3 Un peu de C (6 points)

Identifiez les 6 erreurs se trouvant dans le programme ci-dessous. Indiquez brièvement la façon de les corriger.[2]

```
1  #include"stdio.h"
2
3  int main(int argc, char **argv) {
4    char * chaine, motDePasse="Mot2Passe";
5    scanf("%s",chaine);
6    if (chaine == motDePasse)
7      printf("Vous pouvez passer.\n")
8      return 0;
9    else
10     printf("C'est pas ca\n");
11     return 1;
12 }
```

Solution proposée :

- ligne 1 : < et > à la place de ""
- ligne 5 : `chaine` non allouée
- ligne 4 : manque `char *` pour `motDePasse`
- ligne 6-8 : manque les { et }
- ligne 7 : manque ;
- ligne 6 : comparaison qui ne marchera jamais (il faut utiliser `strcmp`)

[2] `http://www.enseirb.fr/~pelegrin/enseignement/enseirb/prog_c/examens/`

Nom :
Prénom :