LIF5 – Algorithmique et programmation procédurale

CC mi-parcours du 16 novembre 2009 - Correction

Exercice 1 : Tri à bulles

1. Tableau à compléter :

Valeur de i	Valeur de j	Etat du tableau juste après que j ait changé de valeur					
		tab[1]	tab[2]	tab[3]	tab[4]	tab[5]	tab[6]
1	6	53.8	26.1	2.5	13.6	8.8	4.0
1	5	53.8	26.1	2.5	13.6	4.0	8.8
1	4	53.8	26.1	2.5	4.0	13.6	8.8
1	3	53.8	26.1	2.5	4.0	13.6	8.8
1	2	53.8	2.5	26.1	4.0	13.6	8.8
1	1	2.5	53.8	26.1	4.0	13.6	8.8
2	6[1]	2.5	53.8	26.1	4.0	13.6	8.8
2	5	2.5	53.8	26.1	4.0	8.8	13.6
2	4	2.5	53.8	26.1	4.0	8.8	13.6
2	3	2.5	53.8	4.0	26.1	8.8	13.6
2	2	2.5	4.0	53.8	26.1	8.8	13.6

2. [Texte à trous] « Lorsqu'on vient de décrémenter j (ligne 11), le **minimum** du sous-tableau tab[**j..n**] se trouve en position **j.** De plus, si j>1, les éléments du sous-tableau tab[**1..j-1**] occupent les mêmes positions qu'avant le démarrage de la boucle sur j. »

3. Quelle est la dernière valeur de j pour laquelle on fait le test du Tant que (ligne 4) ? Déduisez-en la propriété que l'on obtient lorsqu'on a terminé cette boucle interne.

   La dernière valeur de j testée est j = i. C'est la valeur qui rend le test faux et qui va nous faire sortir du Tant que interne. A ce moment là, notre invariant de boucle nous dit que :
   « Le minimum du sous-tableau tab[i..n] se trouve en position i. De plus, si i>1, les éléments du sous-tableau tab[1..i-1] occupent les mêmes positions qu'avant le démarrage de la boucle sur j. »
   Ainsi, la boucle sur j n'affecte pas le sous-tableau tab[1..i-1], mais elle modifie le sous-tableau tab[i..n] se sorte à ce que son minimum se trouve en position i.

4. Calculez le nombre d'affectations de réels réalisées lors du tri à bulles d'un tableau de n réels, dans le cas le plus défavorable.

   Le cas le plus défavorable est celui où l'on réalise la permutation à chaque passage dans la boucle interne. On a alors 3 affectations de réels par passage.

   Le nombre total d'affectations de réels est alors :

$$A = \sum_{i=1}^{n-1}\left(\sum_{j=i+1}^{n} 3\right) = \sum_{i=1}^{n-1}\left(3(n-i)\right) = 3\left(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i\right)$$

$$A = 3\left(n(n-1) - \frac{(n-1)n}{2}\right) = \frac{3}{2}n(n-1)$$

[1] Le tableau est forcément inchangé depuis la ligne précédente car on n'est pas rentré dans le Tant que de la ligne 4 (j était < i+1).

Exercice 2 : Fusion de deux monotonies

```
/* Preconditions :
   - nomfic1 et nomfic2 sont des fichiers binaires qui contiennent
     chacun une sequence triee de nombres au format double (monotonie).
   - les deux sequences peuvent etre de longueur differente, mais
     elles sont au moins de longueur 1.
   Postcondition :
   - un nouveau fichier nomme nomficSortie est cree (s'il existait
     deja, son contenu est ecrase). Ce fichier contient la fusion
     des deux monotonies.  */

void fusionner(const char * nomfic1, const char * nomfic2, const char *
nomficSortie)
{
  double e1, e2;
  int succeslect1 = 1, succeslect2 = 1;
  FILE *fic1, *fic2, *ficSortie;

  fic1 = fopen(nomfic1, "rb");
  if (fic1 == NULL)
    {
      printf("Impossible d'ouvrir le fichier %s en lecture \n",nomfic1);
      exit(EXIT_FAILURE);
    }

  fic2 = fopen(nomfic2, "rb");
  if (fic2 == NULL)
    {
      printf("Impossible d'ouvrir le fichier %s en lecture \n",nomfic2);
      exit(EXIT_FAILURE);
    }

  ficSortie = fopen(nomficSortie, "wb");
  if (ficSortie == NULL)
    {
      printf("Impossible d'ouvrir le fichier %s en ecriture \n",nomficSortie);
      exit(EXIT_FAILURE);
    }

  fread(&e1, sizeof(double), 1, fic1);
  fread(&e2, sizeof(double), 1, fic2);

  while ((succeslect1 != 0) && (succeslect2 != 0))
    {
      if (e1 < e2)
        {
          fwrite(&e1, sizeof(double), 1, ficSortie);
          if (fread(&e1, sizeof(double), 1, fic1) != 1) succeslect1 = 0;
        }
      else
        {
          fwrite(&e2, sizeof(double), 1, ficSortie);
          if (fread(&e2, sizeof(double), 1, fic2) != 1) succeslect2 = 0;
        }
    }

  if (succeslect1 == 0)
    {
      /* On a epuise le fichier1, il reste a recopier la fin du fichier2 */
      fwrite(&e2, sizeof(double), 1, ficSortie);
      while (fread(&e2, sizeof(double), 1, fic2) == 1)
        {
          fwrite(&e2, sizeof(double), 1, ficSortie);
        }
    }
```

```
  else
    {
      /* On a epuise le fichier2, il reste a recopier la fin du fichier1 */
      fwrite(&e1, sizeof(double), 1, ficSortie);
      while (fread(&e1, sizeof(double), 1, fic1) == 1)
        {
          fwrite(&e1, sizeof(double), 1, ficSortie);
        }
    }


  fclose(fic1);
  fclose(fic2);
  fclose(ficSortie);

}
```

Exercice 3 : Trace mémoire d'un programme

Dessin 1 :

VR main		3 987 546 988
monTab[8]	'\0'	3 987 546 987
monTab[7]	'7'	3 987 546 986
monTab[6]	'c'	3 987 546 985
monTab[5]	'1'	3 987 546 984
monTab[4]	'0'	3 987 546 983
monTab[3]	'0'	3 987 546 982
monTab[2]	'0'	3 987 546 981
monTab[1]	'0'	3 987 546 980
monTab[0]	'0'	3 987 546 979
c	0x0	3 987 546 975
k	8	3 987 546 971
monNombre	455	3 987 546 967
Appel à mystere(7, monNombre, monTab)		
i	7	3 987 546 963
nbr	455	3 987 546 959
t	3 987 546 979	3 987 546 955
reste	7	3 987 546 951
Appel à mystere(6, 28, t)		
i	6	3 987 546 947
nbr	28	3 987 546 943
t	3 987 546 979	3 987 546 939
reste	12	3 987 546 935
Appel à mystere(5, 1, t)		
i	5	3 987 546 931
nbr	1	3 987 546 927
t	3 987 546 979	3 987 546 923
reste	1	3 987 546 919
Appel à mystere(4, 0, t)		
i	4	3 987 546 915
nbr	0	3 987 546 911
t	3 987 546 979	3 987 546 907
reste		3 987 546 903

Dessin 2 :

VR main		3 987 546 988
monTab[8]	'\0'	3 987 546 987
monTab[7]	'7'	3 987 546 986
monTab[6]	'c'	3 987 546 985
monTab[5]	'1'	3 987 546 984
monTab[4]	'0'	3 987 546 983
monTab[3]	'0'	3 987 546 982
monTab[2]	'0'	3 987 546 981
monTab[1]	'0'	3 987 546 980
monTab[0]	'0'	3 987 546 979
c	10004	3 987 546 975
k	9	3 987 546 971
monNombre	455	3 987 546 967

TAS	
'\0'	10 012
'7'	10 011
'c'	10 010
'1'	10 009
'0'	10 008
'0'	10 007
'0'	10 006
'0'	10 005
'0'	10 004

2. Indiquez ce qu'affiche le printf du main lorsqu'on exécute ce programme précis.

000001c7

3. Que fait la procédure mystere ? Autrement dit, si vous deviez lui donner un nom plus explicite, lequel choisiriez-vous ?

La procédure mystere remplit le tableau t avec l'écriture hexadécimale du nombre nbr. On pourrait donc la rebaptiser ecritureHexa, par exemple.

4. Si les unsigned int sont codés sur 32 bits, risque-t-on un « buffer overflow » lorsqu'on remplit monTab (tableau de 9 char) avec la procédure mystère ?

Si les unsigned int sont codés sur 32 bits, alors la valeur maximale de nbr est 2^32 – 1. On dispose de 8 positions pour coder ce nombre en hexadécimal, donc on peut aller jusqu'à 16^8 – 1. On risque un overflow si jamais 16^8 -1 était strictement inférieur à 2^32 -1 : en effet, dans ce cas, il nous faudrait plus de 8 positions pour coder le nombre en hexadécimal. Mais on remarque que 2^32 = 2^(4*8) = (2^4)^8 = 16^8. Les deux valeurs maximales sont exactement égales, donc on ne risque pas d'overflow.