Institut Galilée — Algorithmique et programmation C
Ingénieurs 1ère année (MACS/Télécom/Mesures/Energie)

Correction du T.P. 2

Récursivité et tableaux

Pour tous les exercices de cette feuille, il vous est demandé d'écrire l'algorithme correspondant au problème avant son implémentation en langage C.

Récuristvité

Ecrire en C la fonction factorielle et la tester. Ecrire les versions itératives avec boucle `for` et `while` ainsi qu'une version récursive. Pour les versions itératives, donner la complexité des algorithmes proposés.

```
#include <stdio.h>
int factFor(int n)
{
   int j, r =1;
   for(j=1;j<=n;j++)
     r = r * j;
   return r;
}
int factWhile(int n)
{
  int j, r =1;
  while(n>0)
  {
    r = r * n;
    n--;
  }
  return r;
}

int factRec1(int n)
{
  return (n==0) ? 1 : n * factRec1(n-1);
}

int factRec2(int n)
{
  if(n==0)
    return 1;
  else
    return n * factRec2(n-1);
}

int main(void)
{
   int a;
```

```
   printf("Donnez un entier positif : ");
   scanf("%d", &a);
   printf("%d! donne : par for %d par while %d par rec1 %d par rec2 %d \n", a, factFor(a),
   return 0;
}
```

Tableaux

1. Ecrire une fonction qui affiche les éléments d'un tableau d'entiers, ainsi que leurs indices et la tester. Donner la complexité de l'algorithme correspondant.

   ```
   void affiche(int tab[], int n)
   {
     int i;
     printf("Voici le tableau de %d elements:\n", n);
     for(i=0;i<n;i++)
       printf("tab[%d] = %d \n",i, tab[i]);
   }
   ```

2. Ecrire une fonction récursive qui teste si deux tableaux d'entiers sont identiques et la tester. Ecrire une version itérative du problème en évaluant sa complexité.

3. Ecrire une fonction itérative qui teste si deux tableaux d'entiers sont identiques et la tester. Quelle est la complexité de l'algorithme correspondant?

4. Ecrire une fonction qui compte le nombre d'occurrences d'un élélement dans un tableau et la tester.

5. En déduire une fonction qui teste si deux tableaux d'entiers contiennent les mêmes élélments, sans tenir compte de l'ordre dans lequel les éléments sont rangés. Tester cette fonction.

```
#include <stdio.h>
#define MAX 10
#define VRAI 0==0
#define FAUX 0==1
typedef int bool;

bool tablo_ego(int t1[], int t2[], int n, int i)
{
  bool result;
  if(i==n-1)
    result = (t1[i]==t2[i]);
  else
    result = (t1[i]==t2[i]) && tablo_ego(t1, t2, n, i+1);
  return result;
}

void compare_rec(int t1[], int t2[], int n)
```

```
{
  if (tablo_ego(t1, t2, n, 0))
    printf("Les deux tableaux sont egaux.\n");
  else
    printf("Les deux tableaux sont differents.\n");
}

void compare_iter(int t1[], int t2[], int n)
{
  int i;
  bool pas_fini=VRAI;
  for ( i=0; i<n && pas_fini; i++)
    if (t1[i]!=t2[i])
      pas_fini=FAUX;
  if (pas_fini)
     printf("Les deux tableaux sont egaux.\n");
  else
     printf("Les deux tableaux sont differents.\n");
}

int nb_occ(int tab[], int a, int n, int i)
{
  int result;
  if (i==n-1)
  {
    result = (a==tab[i])?1:0;
  }
  else
  {
    printf(" a : %d, tab[%d]: %d, n: %d\n", a, i, tab[i], n);
    if(a==tab[i])
      result = 1 + nb_occ(tab, a, n, i+1);
    else
      result = nb_occ(tab, a, n, i+1);
  }
  printf(" a : %d, tab[%d]: %d, result: %d\n", a, i, tab[i], result);
  return result;
}

void meme_elems( int t1[], int t2[], int n)
{
  int i, j;
  bool pas_fini=VRAI;
  for(i=0; i<n && pas_fini; i++)
    for(j=0;j<n;j++)
      if(t1[i]==t2[j])
  if (nb_occ(t1, t1[i], n, 0) != nb_occ(t2, t2[j], n, 0))
    pas_fini=FAUX;
  if(pas_fini)
    printf("Les deux tableaux contiennent les memes elements.\n");
```

```
  else
    printf("Les deux tableaux ne contiennent pas les memes elements.\n");
}

int main(void)
{
  int my_tab1[MAX]={2,4,6,8,10,12,14,16,18,20};
  int my_tab2[MAX]={1,3,5,7,9,11,13,15,17,19};
  int my_tab3[MAX]={2,6,8,10,4,12,14,16,18,20};

  affiche(my_tab1,MAX);
  affiche(my_tab2,MAX);
  affiche(my_tab3,MAX);
  printf("Version recursive : \n");
  compare_rec(my_tab1, my_tab1, MAX);
  compare_rec(my_tab1, my_tab2, MAX);
  compare_rec(my_tab1, my_tab3, MAX);
  printf("Version iterative : \n");
  compare_iter(my_tab1, my_tab1, MAX);
  compare_iter(my_tab1, my_tab2, MAX);
  compare_iter(my_tab1, my_tab3, MAX);
  printf("Memes elements : \n");
  meme_elems(my_tab1, my_tab1, MAX);
  meme_elems(my_tab1, my_tab2, MAX);
  meme_elems(my_tab1, my_tab3, MAX);
  return 0;
}
```