Université Paul Sabatier TOULOUSE III

S3

Programmation impérative en langage C

feuille de TP n°3 : Tableaux

Objectifs du TP :
- *Savoir manipuler un tableau statique.*
- *Spécifier un programme avec :*
  - *un prédicat d'entrée PE réalisé par les données du programme hormis le type des données*
  - *un prédicat de sortie PS réalisé par les résultats du programme*
  - *des prédicats intermédiaires si nécessaire*

  Ces prédicats prendront la forme de commentaires en français dans le programme et formules de la logique des prédicats sur papier.

On utilisera le langage de la logique des prédicats avec les symboles habituels vus en cours :
$\forall$ *(quantificateur universel)* $\forall I : P_1 \rightarrow P_2$
$\exists$ *(quantificateur existentiel)* $\exists I : P_1 \wedge P_2$
ν *(quantificateur numérique)* $\nu I : (I \in E) \wedge P(I)$ *nombre des* I *dans* E *vérifiant la propriété* P
$\neg$ *(négation)*
$\rightarrow$ *(implication)*
$\wedge$ *(conjonction)*
$\vee$ *(disjonction)*
$\in$ *(appartenance)*
avec le parenthèsage nécessaire.

Pour alléger certaines écritures on pourra utiliser :
· $(T(I..J), <)$ *ou* $(T(I..J), <=)$ *pour exprimer la croissance de* T *entre les indices* I *et* J
· $t = A$ *pour exprimer que les 2 vecteurs* t *et* A *contiennent des valeurs identiques*
· $t = permut(A)$ *pour exprimer que le vecteur* t *contient toutes les valeurs de* A *mais peut-être dans le désordre*

1 Manipulations de vecteurs

Les 4 premiers exercices ne requièrent pas d'imbrication de boucles.

◇ Exercice **1** : Ecrire un programme qui commence par saisir 10 nombres entiers au clavier, les place dans un vecteur puis modifie ce vecteur en deux étapes successives de la manière suivante :
– d'abord en remplaçant les **valeurs impaires** par 0
– puis en remplaçant le contenu de toutes les cases d'**indices impairs** par 0
On affichera le vecteur après chacune des 2 étapes et le nombre de cases non nulles à la fin.
Compléter le cartouche et écrire le programme.

Exo1 : Modification d'un vecteur de 10 entiers
Données : int T[10]
Aprés l'étape 1, Résultat1 : int t[10]
$PS1 = \forall i : ((0 \leq i \leq 9 \wedge T[i] mod 2 = 1) \rightarrow t[i] = 0)$
$\wedge$
$\forall i : ((0 \leq i \leq 9 \wedge T[i] mod 2 = 0) \rightarrow t[i] = T[i]))$
Aprés l'étape 2, Résultat2 : int t[10]
$PS2$ = ...*à compléter*

```
Donnez le tableau: 5 3 8 26 7 26 8 3 5 24
Apres avoir remplacées les valeurs impaires par 0: 0 0 8 26 0 26 8 0 0 24
Apres avoir mis à zero le contenu des cases d'indice impair: 0 0 8 0 0 0 8
0 0 0
Le tableau contient 2 elements non nuls.
```

◇ Exercice **2** : Etant donné un vecteur de N($\leq$ 10) entiers entrés au clavier, ecrire un programme qui vérifie si ce vecteur est un palidrome. Un vecteur est palindrome lorsque ses cases symétriques sont identiques.
Compléter le cartouche et écrire le programme.

Exo2 : Détermination du caractère palindrome d'un tableau
Données : int N , int T[10]
Résultat : bool est_palin, est_palin = ... à compléter

```
Donnez la dimension du tableau: 9
Donnez le tableau: 5 3 8 26 7 26 8 3 5
C'est un palindrome.
```

◇ Exercice **3** : Ecrire un programme qui satisfait à la spécification suivante :

Exo3 : ? ? ? à compléter
Données : int N, int T[10]
Résultat : int t[10], $\forall i : 0 \leq i < N \rightarrow t[i] = T[N-1-i]$

On n'utilisera pas de tableau auxiliaire.

2 Tris d'un vecteur

◇ Exercice **4** : Soit N un entier ($\leq$ 100) saisi au clavier ; étant donné un vecteur de N nombres entiers saisis au clavier, trier ce vecteur de telle sorte que toutes les valeurs paires soient situées au début du tableau et toutes les valeurs impaires à la fin. (Il n'y a pas unicité du résultat, le tri ne doit pas se faire pendant la saisie).
– en utilisant un vecteur supplémentaire
– en n'utilisant pas d'autre vecteur

Exo4 : Tri pair impair
Données : int N, int T[N]
Résultat : int t[N], t=permut(T)$\wedge$*... à compléter*

```
Donnez la dimension du tableau: 5
Donnez le tableau: 1 0 2 3 1
Vecteur trie:0 2 1 3 1
```

◇ Exercice **5** : Tri par insertion :
Etant donné un vecteur contenant des valeurs qu'on peut ordonner (réels, entiers, caractères), on souhaite modifier ce vecteur de telle sorte qu'à la fin du traitement, le vecteur contienne les mêmes valeurs qu'au début mais dans l'ordre croissant :

```
Donnez la dimension du tableau: 5
Donnez le tableau: 1 0 2 3 1
Vecteur trié par insertion:0 1 1 2 3
```

De très nombreux procédés existent ; l'un des plus simples à mettre en œuvre est le tri par insertion.
Méthode :
Les cases du tableau sont indexées de 0 à $N-1$. Supposant les valeurs des $i-1$ premières cases du tableau T bien ordonnées, on remplit la case d'indice i de la manière suivante :
Pour remplir la case d'indice i, on compare la valeur de chaque case d'indice allant de $i+1$ jusqu'au dernier indice : après chaque comparaison, on permute les valeurs si cela est nécessaire.

- Remplissage de la case d'indice 0 :
  Pour chaque case d'indice j allant de 1 à $N-1$
  Début
  Si $T[0] > T[j]$ alors permuter le contenu des cases d'indice 0 et j ;
  Fin

. . . .

- Remplissage de la ième case :
  Pour chaque case d'indice j allant de $i+1$ à $N-1$
  Début
  Si $(T[i] > T[j])$ alors permuter le contenu des cases d'indice i et j ;
  Fin

On peut remarquer qu'une fois les $N-1$ premières cases bien rangées, nécessairement la dernière case du tableau est bien rangée.

3 Liste des parties d'un ensemble

1. Le programme suivant a pour objet l'affichage des nombres binaires sur 2 bits : 00, 01, 10, 11.

```c
//Description: numération en binaire sur 2 bits
//Etat:    pb
#include <stdio.h>
int main (void)
{
 int bits[2]={ 0, 0 };
 int i;
```

```
 //pour chacun des 4 nombres binaires : 00 01 10 11
 for(i=1;i<=4;i++)
    {
      printf("\ni=%2d",i);
    //affichage du nombre binaire
      printf("  %d%d",bits[0],bits[1]);
    //passage au nombre binaire suivant
      //recherche du bit nul de poids le plus faible
      int j=1;
      while (bits[j]==1 ) j--;
      //on le met à 1
      bits[j]=1;
      //mise à zéro des bits de poids plus faible
      int k;
      for (k=j+1;k<2;k++) bits[k]=0;
    }
return (0);
}
```

Sur certains compilateurs, on obtient à l'éxécution :

```
i=1 00
i=2 01
i=3 10
i=4 11
i=2 01
i=3 10
i=4 11
```

Expliquer et corriger le problème (à la ligne 17...). Une fois ce programme corrigé, il pourra vous servir dans les questions suivantes.

2. Un ensemble de n éléments possède 2^n parties qu'on peut numéroter de 1 à 2^n ce qui permet de coder leur numéro sur n bits. Soit l'ensemble $E = \{1, \cdots, n\}$, on peut associer (par une bijection) un nombre binaire codé sur n bits et une partie de E de la manière suivante : chaque bit du nombre binaire dit si l'élément appartient ou non à la partie : le bit de poids 2^i vaut 1 indique que le nombre $i+1$ est dans la partie.
   Par exemple :
   · on associe au nombre binaire 1010 la partie $\{2, 4\}$ numérotée 11(=10+1).
   · on associe au nombre binaire 11011 la partie $\{1, 2, 4, 5\}$ numérotée 28(=27+1).
   Ecrire un programme qui demande à l'auteur un entier $n \geq 0$ puis qui affiche les parties de l'ensemble $E = \{1, \cdots, n\}$.

```
Taper un entier n: 4
partie 1: (0000)
partie 2: (0001) 1
partie 3: (0010) 2
partie 4: (0011) 1,2
partie 5: (0100) 3
partie 6: (0101) 1,3
partie 7: (0110) 2,3
partie 8: (0111) 1,2,3
partie 9: (1000) 4
partie 10: (1001) 1,4
partie 11: (1010) 2,4
partie 12: (1011) 1,2,4
partie 13: (1100) 3,4
partie 14: (1101) 1,3,4
partie 15: (1110) 2,3,4
partie 16: (1111) 1,2,3,4
```

3. Ecrire une deuxième version du programme pour lequel l'ensemble E est constitué de caractères saisis par l'utilisateur.

```
Combien d'elements dans E?: 3
Saisir les elements de E: a m  e
partie 1: (000)
partie 2: (001) a
partie 3: (010) m
partie 4: (011) a,m
partie 5: (100) e
partie 6: (101) a,e
partie 7: (110) m,e
partie 8: (111) a,m,e
```