**TP I1101 nº7**

**Les tableaux en C**

Exercice 1

1. Écrire un programme qui saisit un tableau d'entiers (contient obligatoirement des 0 et 1) et de dimension maximale 50 et qui arrête à saisir lorsque -1 est tapé et qui retourne la longueur de la plus longue séquence.

2.Ajouter les instructions nécessaires pour afficher les indices de début et de fin de la sous-séquence.

Exercice 2

Tri par comptage : Etant donné un tableau T de N entiers (sans répétition), il s’agit de trier T en construisant un autre tableau TC dit de comptage de la façon suivante :

Pour chaque élément T[i], TC[i] = nombre d'éléments < T[i] dans T.

Ainsi l’on peut déterminer la position (indice) de chaque élément dans le tableau résultat.

Indice 0

Exemple :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tableau initial T:** | **52** | **10** | **1** | **25** | **62** | **3** | **8** | **55** |
|  |  |  |  |  |  |  |  |  |
| **Tableau comptage TC:** | **5** | **3** | **0** | **4** | **7** | **1** | **2** | **6** |
|  |  |  |  |  | RES[5] =T[0] |  |  |  |
| **Tableau résultat Res:** | **1** | **3** | **8** | **10** | **25** | **52** | **55** | **62** |

Écrire un programme C permettant de:

*Saisir le tableau T de dimension n;*

*Calculer le tableau de comptage TC;*

*Remplir et afficher le tableau ordonné Res.*

Exercice 3

1. Ecrire un programme qui remplit un tableau T d’une suite de 0 et de 1 d’au plus

MAX éléments, terminée par la valeur -1.

Par ex: 0 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 -1

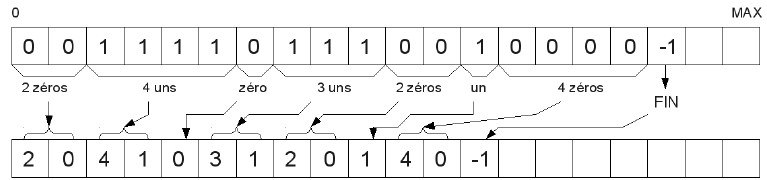
1. Ajouter les instructions nécessaires pour compresser T dans un autre tableau TCOMPRESS.

***Indication :***

L’algorithme de compression :

1. On recherche les groupes d’éléments identiques.
2. Si un groupe d’éléments identiques de T contient un seul élément, il est conservé tel quel dans TCOMPRESS.
3. Si un groupe d’éléments identiques de T contient n éléments avec n .2, il est stocké dans TCOMPRESS sous la forme “ne suivi pas de l’élément répété”.

Par exemple, la séquence ci-dessus sera compressée en : 2 0 4 1 0 3 1 2 0 1 4 0 -1 comme l’illustre le dessin ci-dessous.



**Solution**

Note : Les méthodes ‘expert’ sont des méthodes optionnelles à faire.

Exercice 1

Méthode 1 (simple):

#include <stdio.h>

#define N 50

void main()

{

int i=0,n, counter = 1,val, max = 0, init,T[N];

printf("Donner les elements de tableau (max=50) de la forme:\n 0 et 1 seulement, \n -1 une fois pour s'arreter :\n");

do

{

Remplir d’un tableau conditionné non prelimité (sortie par -1)

scanf("%d", &val);

if (val == 0 || val==1)

{

T[i] = val;

i++;

}

} while (val!=-1);

La limite du tableau remplis aprés est alors n

n = i;

for (i = 0; i < n ; i++)

A ajouter pour que une comparaison entre « T[n-1] » et « T[n] » soit valide.

printf("T[%d]=%d \n", i, T[i]);

T[n] = -1;

for ( i = 0; i < n; i++)

Boucle d’algorithme principale :

Calculer dans « counter » les répétitions des éléments consécutifs du tableau. Lors de la première différence tester si ce nombre de répétitions est le plus grand nombre trouvé, si non ignoré.

Toujours restart le « counter » a un pour compter de nouveau.

(***\*Voir note***  dans la page suivant)

{

if (T[i]==T[i+1])

{

counter++;

}

else

{

if (counter>max)

{

max = counter;

init = i - max +1;

}

counter = 1;

}

}

printf("la longueur maximal est %d et d’indice de %d a %d ",max,init,init+max-1);

}

***\*Note :*** au cas de la dernière séquence on a une comparaison entre « T[n-1] » et « T[n] » mais T[n] est non initialise dans le tableau ni à être comptée, donc on l’initialise par -**1** (par exemple) qui est évidement diffèrent que 0 et 1 : on est par suite capable de prendre en considération la dernière séquence.

**Exemple d’exécution :**

**Donner les elements de tableau (max=50) de la forme:**

**0 et 1 seulement**

**-1 une fois pour s'arreter**

**1 0 4 -2 3 2 1 1 1 0 3**

**0**

**-1**

**T[0]=1**

**T[1]=0**

**T[2]=1**

**T[3]=1**

**T[4]=1**

**T[5]=0**

**T[6]=0**

**la longueur maximal est 3 et d’indice de 2 a 4**

***Note :*** on peut séparer les données saisis par le clavier avec un « **SPACE** » ou « **ENTER** » ou un «**TAB** ».

Méthode 2 (expert):

Une méthode experte est celle qui prend des cas plus généraux que celle du simple qui sont justes pour donner une réponse à la question.

Dans cet exercice, on a un cas spécial : plusieurs séquences peuvent se répéter avec la même longueur maximal ex : “101110001”. On a “111” et “000” de longueur 3, et évidement de différent indice ce qui fallait un plus avancée algorithme réponse sur la partie 2.

(Note : un second tableau est indispensable dans ce cas)

#include <stdio.h>

#define N 50

void main()

{

int i = 0, val, counter = 1, max = 0, n=1, T[N], k=0, TMax[N];

printf("Donner les elements de tableau (max=50) de la forme :\n 0 et 1 seulement \n -1 une fois pour s'arreter\n");

do

{

scanf("%d", &val);

if (val == 0 || val==1)

{

T[i] = val;

i++;

}

} while (val!=-1);

n = i; T[n] = -1;

for (i = 0; i < n; i++)

{

if (T[i] == T[i + 1])

counter++;

else

{

if (counter > max)

{

max = counter;

k = 0;

TMax[k++] = i – counter + 1;

}

else

if (counter == max)

TMax[k++] = i - counter + 1;

counter = 1;

}

}

for (i = 0; i < n; i++)

printf("T[%d]=%d \n", i, T[i]);

printf(" la longueur maximal est %d\n", max);

for (i = 0; i < k; i++)

printf("le %d eme sequence a d'indices de %d a %d \n",i+1,TMax[i], TMax[i]+max-1);

}

Exemple d’exécution Méthode 2 ☺

**Donner les elements de tableau (max=50) de la forme :**

**0 et 1 seulement**

**-1 une fois pour s'arreter**

**1 1 0 0 1 1 0 1 1 0 -1**

**T[0]=1**

**T[1]=1**

**T[2]=0**

**T[3]=0**

**T[4]=1**

**T[5]=1**

**T[6]=0**

**T[7]=1**

**T[8]=1**

**T[9]=0**

**la longueur maximal est 2**

**le 1 eme sequence a d'indices de 0 a 1**

**le 2 eme sequence a d'indices de 2 a 3**

**le 3 eme sequence a d'indices de 4 a 5**

**le 4 eme sequence a d'indices de 7 a 8**

***Note :*** on peut séparer les données saisis par le clavier avec un « **SPACE** » ou « **ENTER** » ou un «**TAB** ».

Exercice 2

#include <stdio.h>

#define N 50

void main()

{

int n, i, j, T[N], TC[N], Res[N];

printf("Entrer le nombre d'element du tableau ");

scanf("%d", &n);

printf("Remplir les elements du tableau sans repetition :\n");

Remplir le tableau T normal

Et annuler tout élément dans TC afin d’ajouter directement après des “++” ou ‘+1’ a des cases déjà initialise par 0.

for (i = 0; i < n; i++)

{

scanf("%d", &T[i]);

TC[i] = 0;

}

for (i = 0; i < n; i++)

{

for ( j = 0; j < n; j++)

Boucle d’algorithme principale.

if (T[j]<T[i])

TC[i]++;

Res[TC[i]] = T[i];

}

for (i = 0; i < n; i++)

printf("Res[%d]=%d \n", i, Res[i]);

}

Exemple d’exécution :

**Entrer le nombre d'element du tableau 8**

**Remplir les elements du tableau sans repetition :**

**52 10 1 25 62 3 8 55**

**Res[0]=1**

**Res[1]=3**

**Res[2]=8**

**Res[3]=10**

**Res[4]=25**

**Res[5]=52**

**Res[6]=55**

**Res[7]=62**

Autre exemple d’exécution :

Entrer le nombre d'element du tableau 5

Remplir les elements du tableau sans repetition :

5 3 2 7 4

Res[0]=2

Res[1]=3

Res[2]=4

Res[3]=5

Res[4]=7

Exercice 3

#include <stdio.h>

#define N 50

void main()

{

int i = 0, n, numgrp=1,k=0, T[N], TCOMPRESS[N],val;

printf("Donner les elements de tableau (max=50) de la forme:\n 0 et 1 seulement \n -1 une fois pour s'arreter\n");

do

{

scanf("%d", &val);

if (val == 0 || val == 1)

{

T[i] = val;

TCOMPRESS[i] = 1;

i++;

}

} while (val != -1);

T[i] = -1;

n = i + 1;

for (i = 0; i < n-1; i++)

{

if (T[i] == T[i + 1])

Calcule de redondance dans « numgrup ».

numgrp++;

else

if (numgrp==1)

TCOMPRESS[k++] = T[i];

Lors de la première différence entre « T[i] » et « T[i+1] » c’est indispensable à remplir « TCOMPRESS[]» selon le cas.

else

{

TCOMPRESS[k++] = numgrp;

TCOMPRESS[k++] = T[i];

numgrp = 1;

}

}

TCOMPRESS[k] = -1;

printf(" Forme compressée: ");

for (i = 0; i < k+1; i++)

printf("%d ", TCOMPRESS[i]);

}

Exemple d’exécution :

**Donner les elements de tableau (max=50) de la forme:**

**0 et 1 seulement**

**-1 une fois pour s'arreter**

**0 0 1 0 0 0 0 0 0 0 -1**

**Forme compressée: 2 0 1 7 0 -10**