

# **Solution des exercices du Cours 5**

Les tableaux

# Qu'affichent ces programmes ?

**A:**

```
int T[5];

for(int i = 0; i < 5; i++)
    T[i] = 5 * i;
for(int i = 4; i >= 0; i--)
    cout << T[i] << endl;
```

**B:**

```
bool D[4];

D[0] = true;
D[1] = false;
D[2] = false;
D[3] = true;

for(int i = 0; i < 5; i++)
    if (D[i])
        cout << "*";
    else
        cout << "-";
cout << endl;
```

**C:**

```
int T[4];

T[0] = 1;
T[1] = 3;
T[2] = 2;
T[3] = 2;

cout << T[ T[0] ] << endl;

int a = 0;
for(int i = 0; i < 100; i++)
    a = T[a];
cout << a << endl;
```

**A:**

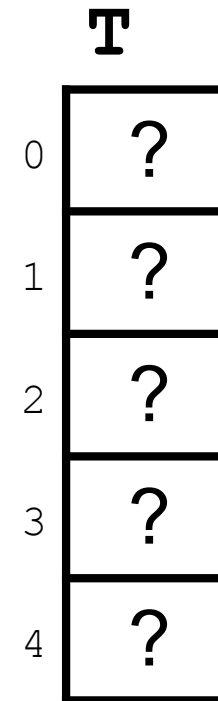
→ `int T[5];`

```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

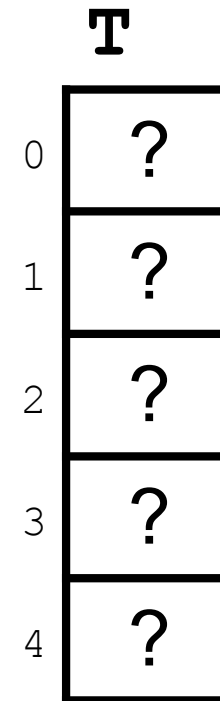
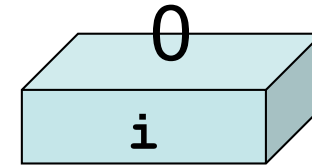
```
    cout << T[i] << endl;
```



**A:**

```
int T[5];
```

```
→ for(int i = 0; i < 5; i++)  
    T[i] = 5 * i;  
for(int i = 4; i >= 0; i--)  
    cout << T[i] << endl;
```



**A:**

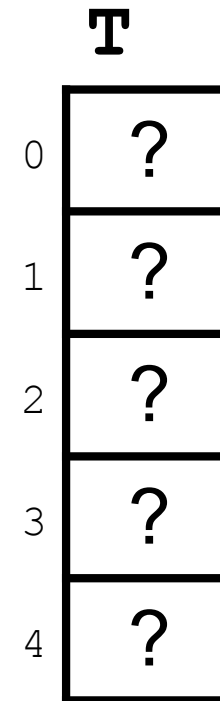
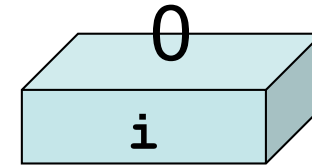
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
→ T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```



**A:**

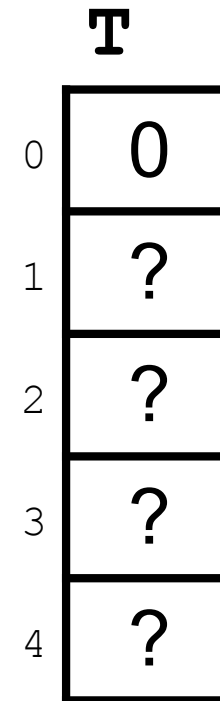
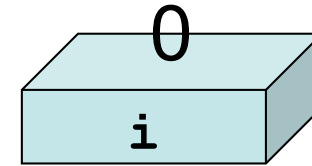
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
→ T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

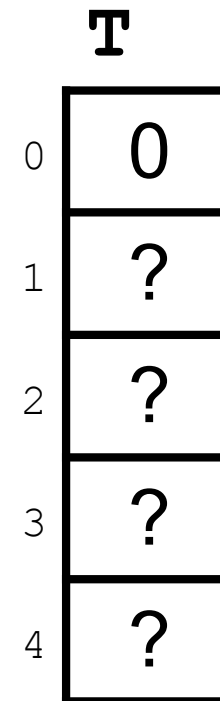
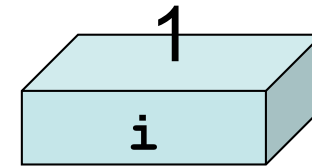
```
    cout << T[i] << endl;
```



**A:**

```
int T[5];
```

```
→ for(int i = 0; i < 5; i++)  
    T[i] = 5 * i;  
for(int i = 4; i >= 0; i--)  
    cout << T[i] << endl;
```



**A:**

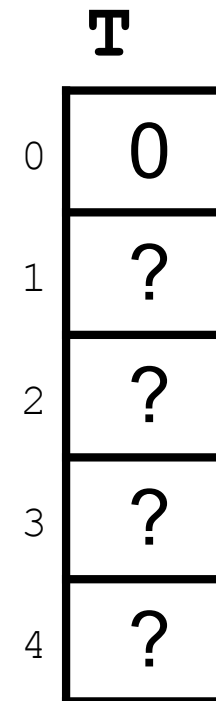
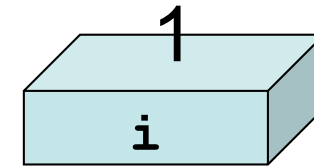
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
→ T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```



**A:**

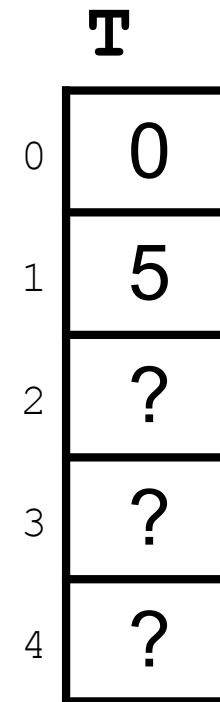
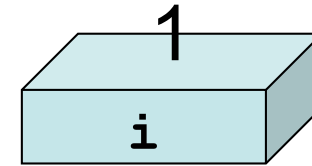
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
→ T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```



**A:**

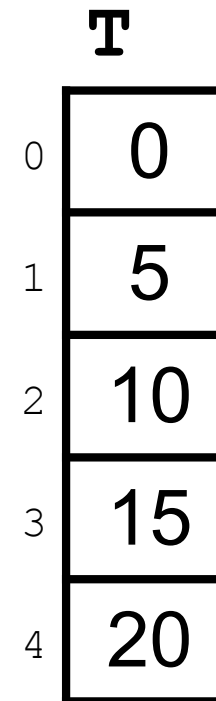
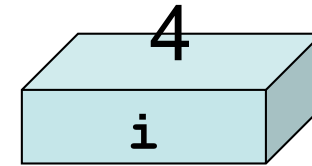
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
→ for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```



**A:**

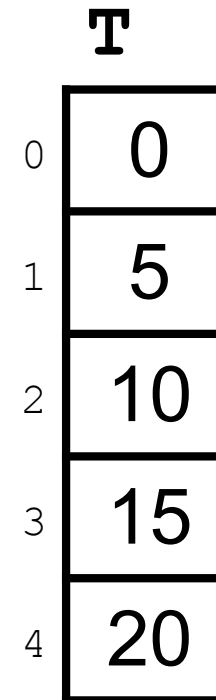
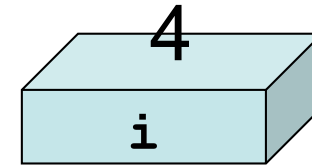
```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
→ cout << T[i] << endl;
```



**A:**

```
int T[5];
```

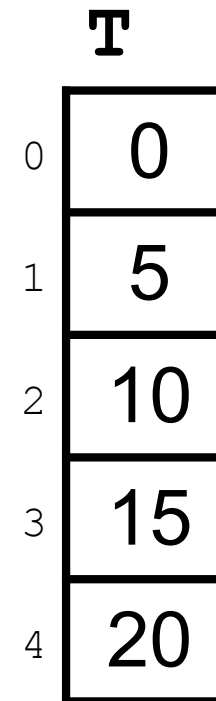
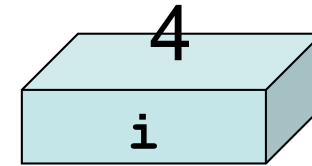
```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
→ cout << T[i] << endl;
```

20



**A:**

```
int T[5];
```

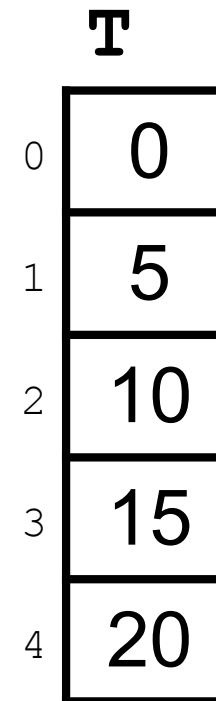
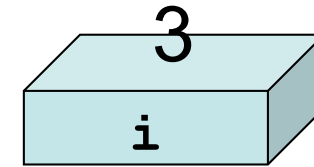
```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
→ for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```

20



**A:**

```
int T[5];
```

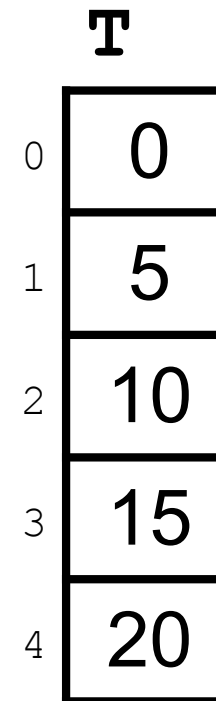
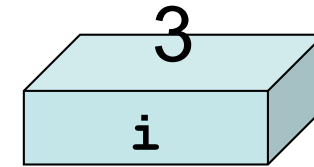
```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
→ cout << T[i] << endl;
```

20



**A:**

```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

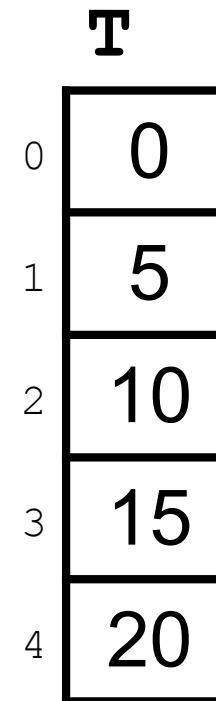
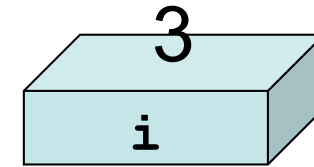
```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
→ cout << T[i] << endl;
```

20

15



**A:**

```
int T[5];
```

```
for(int i = 0; i < 5; i++)
```

```
    T[i] = 5 * i;
```

```
for(int i = 4; i >= 0; i--)
```

```
    cout << T[i] << endl;
```



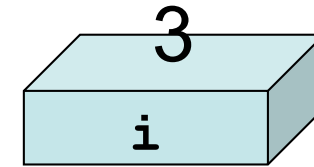
20

15

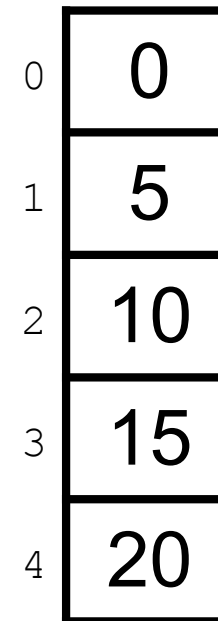
10

5

0



**T**



**B:**

```
bool D[4];
```

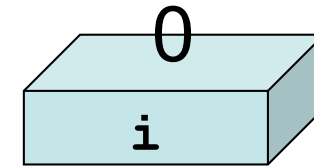
```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
→ for(int i = 0; i < 5; i++)  
    if (D[i])  
        cout << "*";  
    else  
        cout << "-";  
cout << endl;
```



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



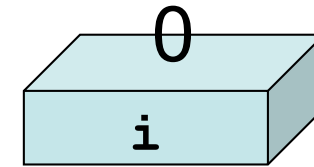
```
    if (D[i])
```

```
        cout << "*";
```

```
    else
```

```
        cout << "-";
```

```
cout << endl;
```



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```

```
    if (D[i])
```

```
        cout << "*";
```

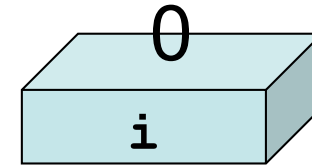
```
    else
```

```
        cout << "-";
```

```
cout << endl;
```



\* ◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

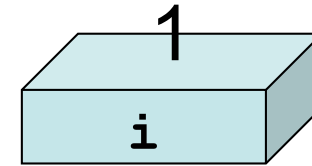
```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
→ for(int i = 0; i < 5; i++)  
    if (D[i])  
        cout << "*";  
    else  
        cout << "-";  
cout << endl;
```

\* ◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



```
    if (D[i])
```

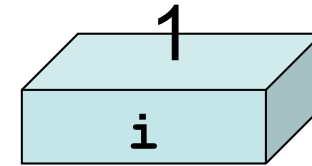
```
        cout << "*";
```

```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```

\* ◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```

```
    if (D[i])
```

```
        cout << "*";
```

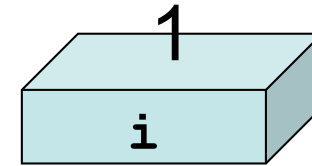
```
    else
```

```
        cout << "-";
```

```
cout << endl;
```



\* ◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```

```
    if (D[i])
```

```
        cout << "*";
```

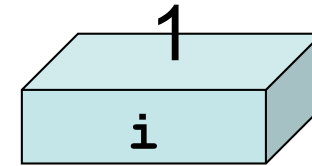
```
    else
```

```
        cout << "-";
```

```
cout << endl;
```



\* - ◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

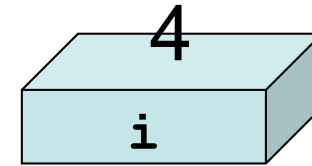
```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
→ for(int i = 0; i < 5; i++)  
    if (D[i])  
        cout << "*";  
    else  
        cout << "-";  
cout << endl;
```

\*--\*◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



```
    if (D[i])
```

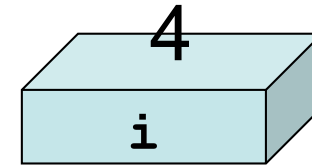
```
        cout << "*";
```

```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```

\*--\*◆



**D**

0	true
1	false
2	false
3	true

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



```
    if (D[i])
```

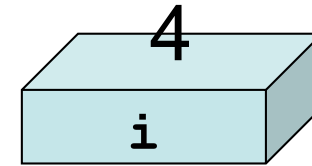
```
        cout << "*";
```

```
    else
```

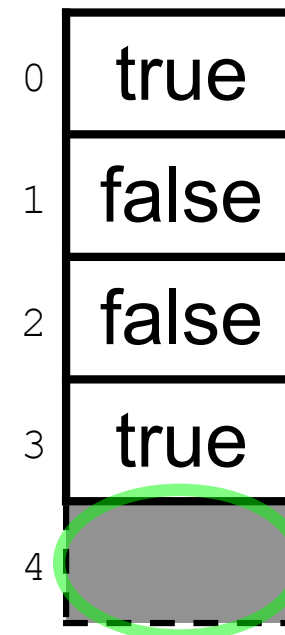
```
        cout << "-";
```

```
    cout << endl;
```

\*--\*◆



**D**



**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



```
    if (D[i])
```

```
        cout << "*";
```

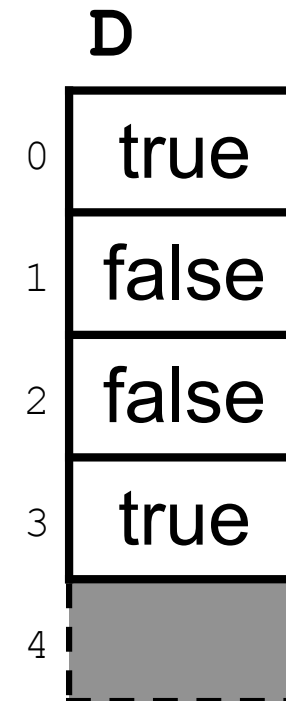
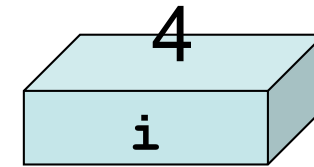
```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```

Si la mémoire en D[4] appartient au programme et  
contient true:

\*--\*◆



**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```

```
    if (D[i])
```

```
        cout << "*";
```

```
    else
```

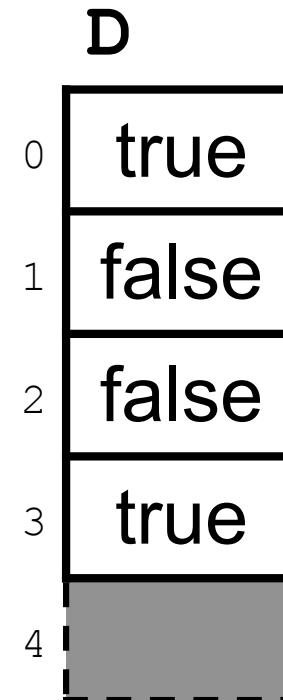
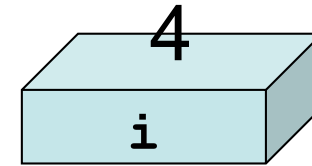
```
        cout << "-";
```

```
    cout << endl;
```



Si la mémoire en D[4] appartient au programme et contient true:

\*--\*\*◆



**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



```
    if (D[i])
```

```
        cout << "*";
```

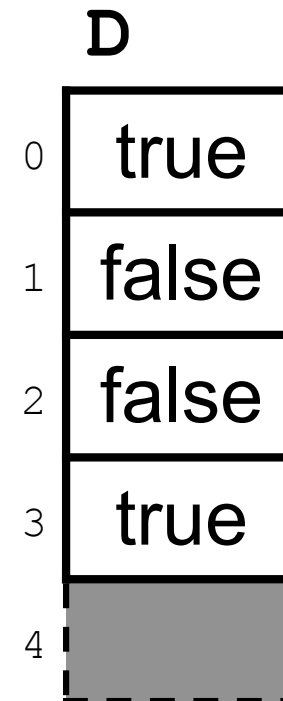
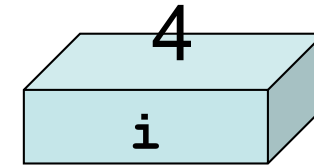
```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```

Si la mémoire en D[4] appartient au programme et  
contient false:

\*--\*◆



**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```

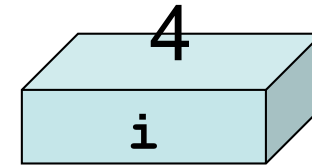
```
    if (D[i])
```

```
        cout << "*";
```

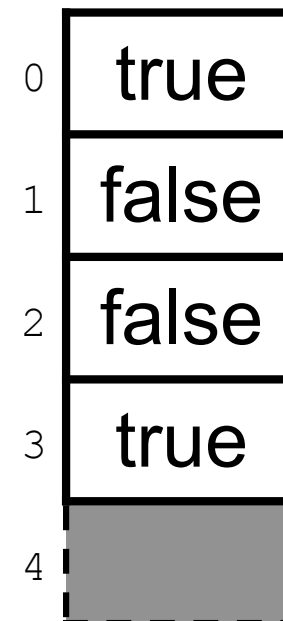
```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```



**D**



Si la mémoire en D[4] appartient au programme et contient false:

\*--\*--◆

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



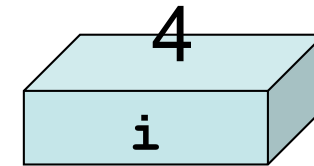
```
    if (D[i])
```

```
        cout << "*";
```

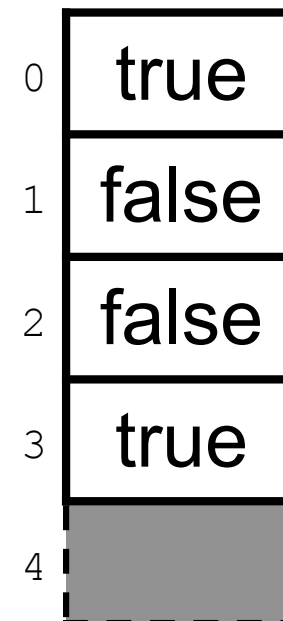
```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```



**D**



Si la mémoire en D[4] n'appartient pas au programme:

\*--\*◆

**B:**

```
bool D[4];
```

```
D[0] = true;
```

```
D[1] = false;
```

```
D[2] = false;
```

```
D[3] = true;
```

```
for(int i = 0; i < 5; i++)
```



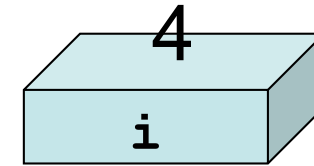
```
    if (D[i])
```

```
        cout << "*";
```

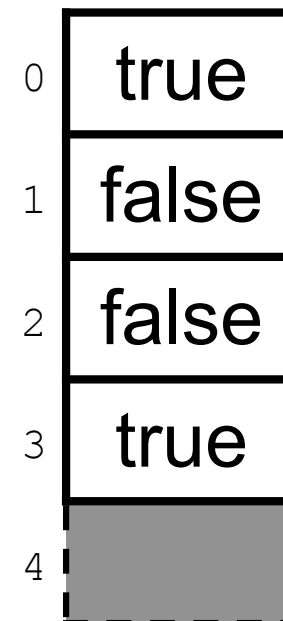
```
    else
```

```
        cout << "-";
```

```
    cout << endl;
```



**D**



Si la mémoire en D[4] n'appartient pas au programme:

\*--\*Segmentation fault  
et le programme s'arrête.

**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

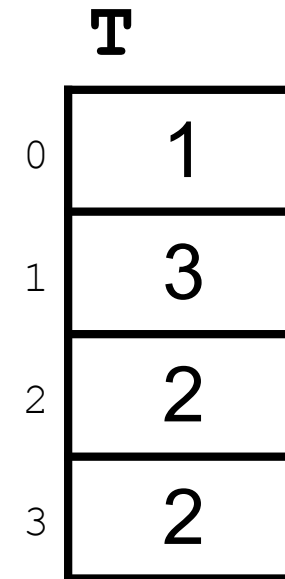
```
→ cout << T[ T[0] ] << endl;
```

```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
    a = T[a];
```

```
cout << a << endl;
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
→ cout << T[ T[0] ] << endl;
```

```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
    a = T[a];
```

```
cout << a << endl;
```

```
3
```

<b>T</b>	
0	1
1	3
2	2
3	2

**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

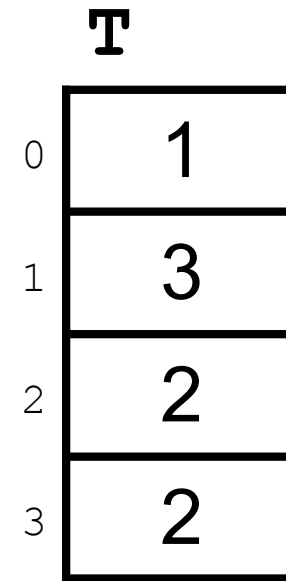
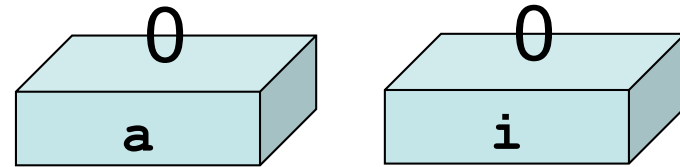
```
int a = 0;
```

```
→ for(int i = 0; i < 100; i++)
```

```
    a = T[a];
```

```
    cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

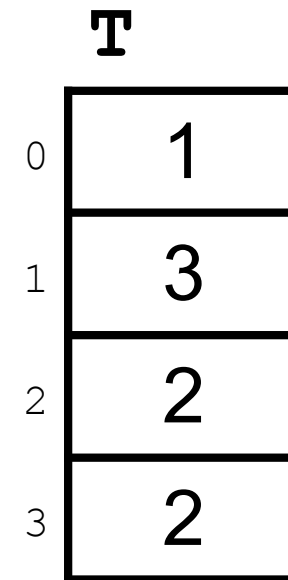
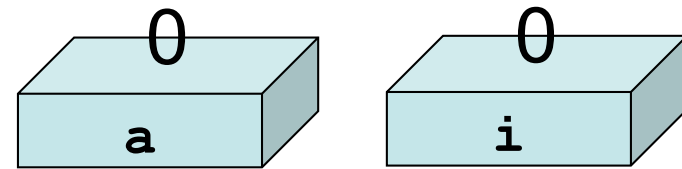
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

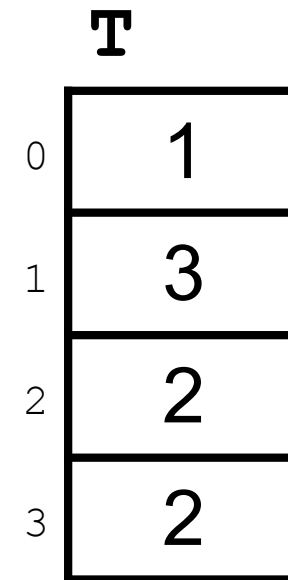
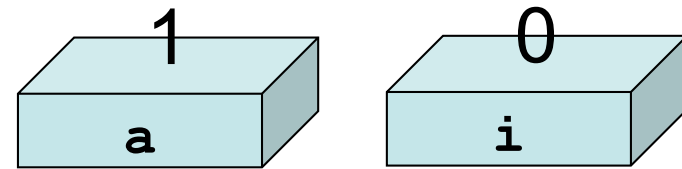
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

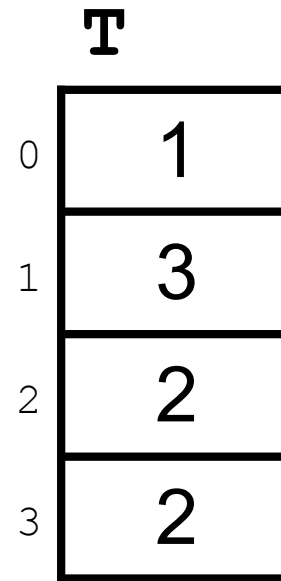
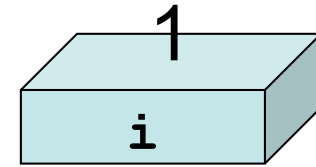
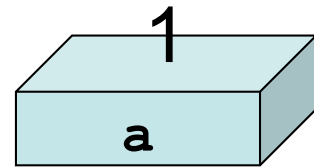
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

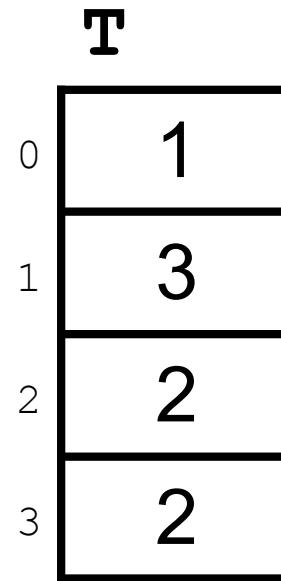
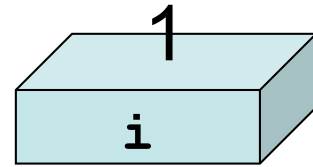
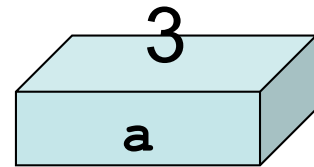
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

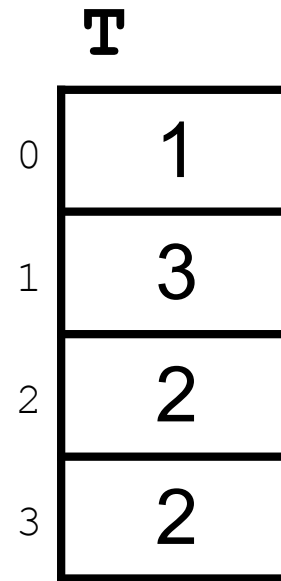
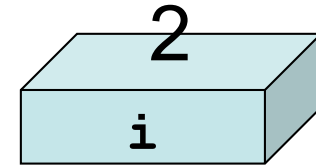
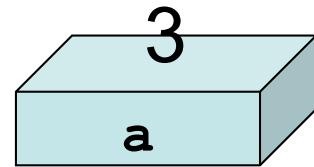
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

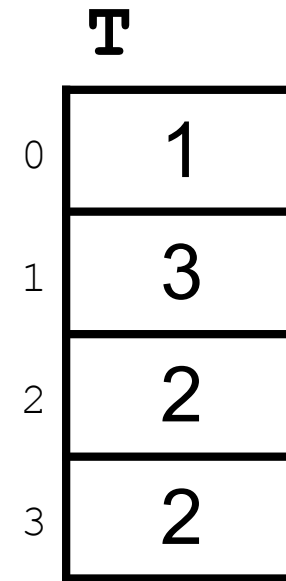
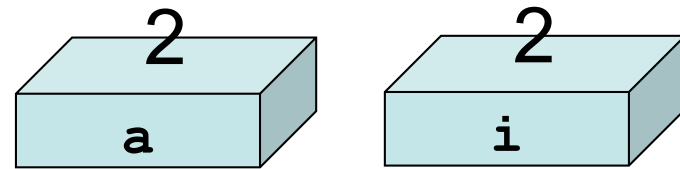
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

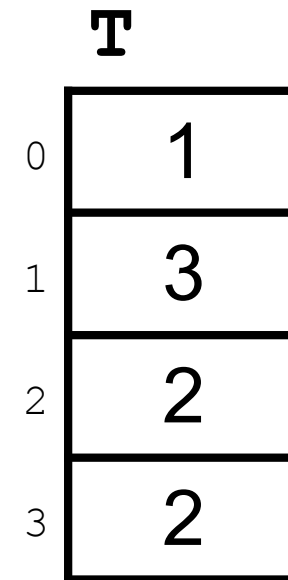
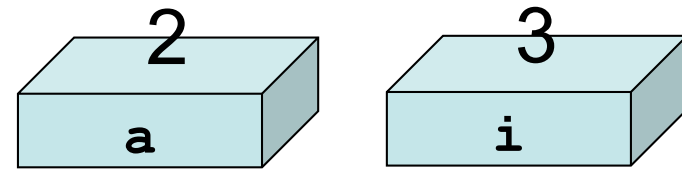
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

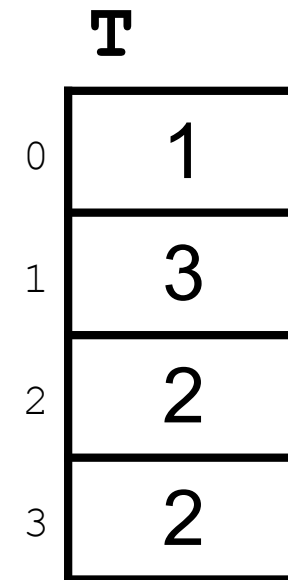
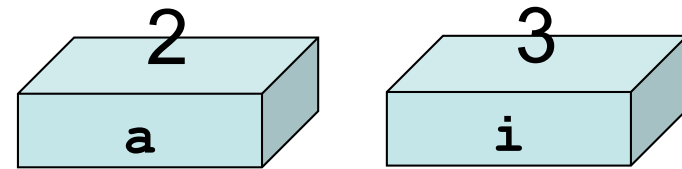
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

```
cout << T[ T[0] ] << endl;
```

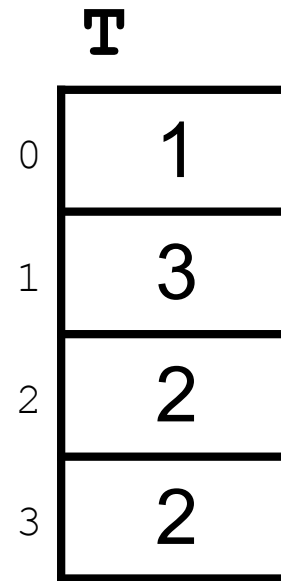
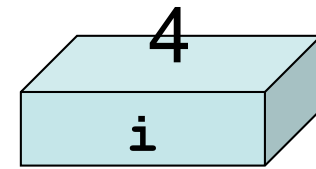
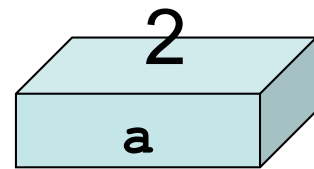
```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
→ a = T[a];
```

```
cout << a << endl;
```

```
3
```



**C:**

```
int T[4];
```

```
T[0] = 1;
```

```
T[1] = 3;
```

```
T[2] = 2;
```

```
T[3] = 2;
```

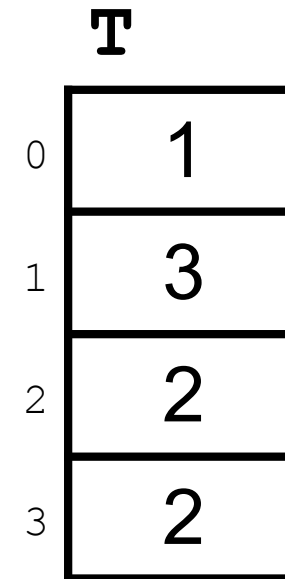
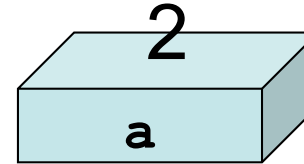
```
cout << T[ T[0] ] << endl;
```

```
int a = 0;
```

```
for(int i = 0; i < 100; i++)
```

```
    a = T[a];
```

```
→ cout << a << endl;
```



# Exercices

1. Décalage dans l'autre sens:

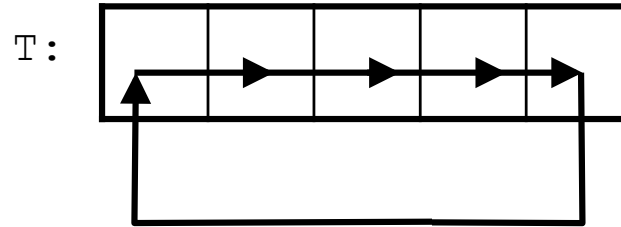
0 1 2 3 4 devient 4 0 1 2 3.

2. Écrire le code qui inverse les éléments du tableau:

0 1 2 3 4 devient 4 3 2 1 0

# Décalage vers la droite

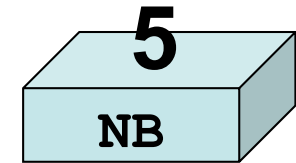
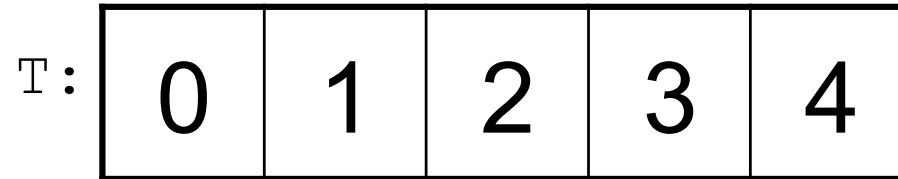
Écrire le code qui décale les éléments de  $T$  vers la droite:



En adaptant le programme précédent:

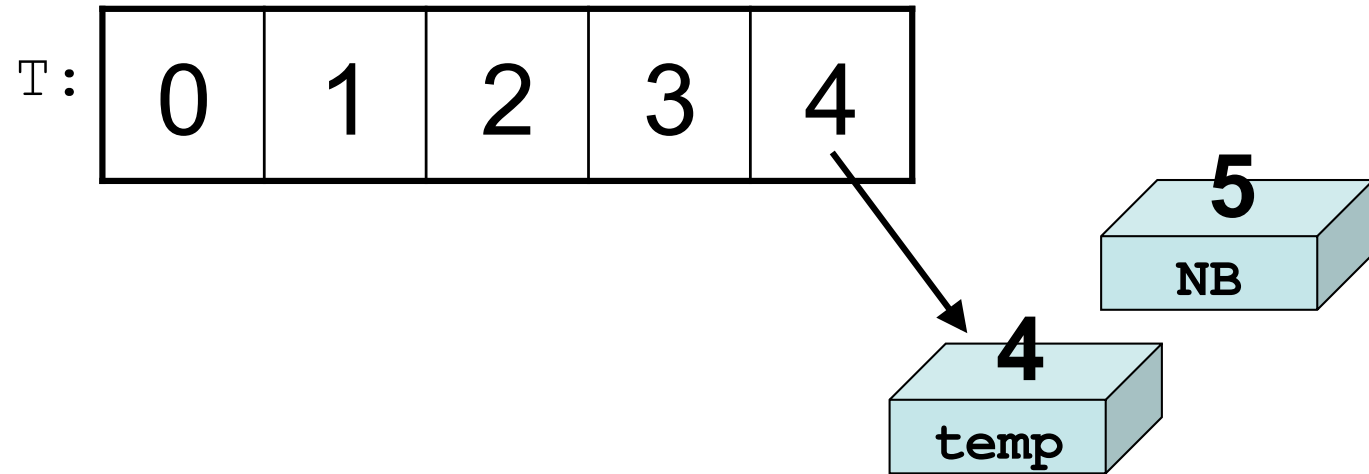
```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
    T[i+1] = T[i];  
T[0] = temp;
```

# Vérifier le code sur un exemple



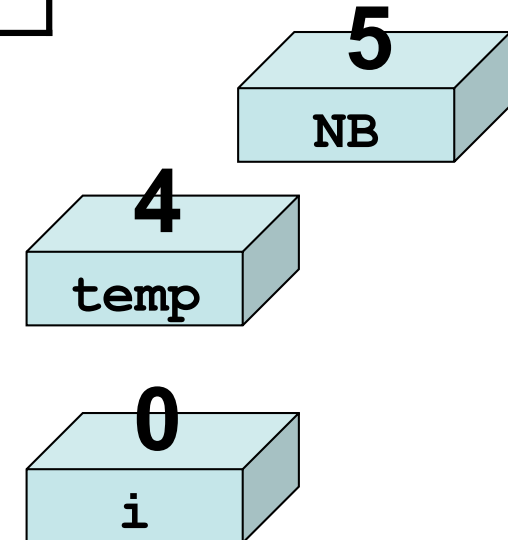
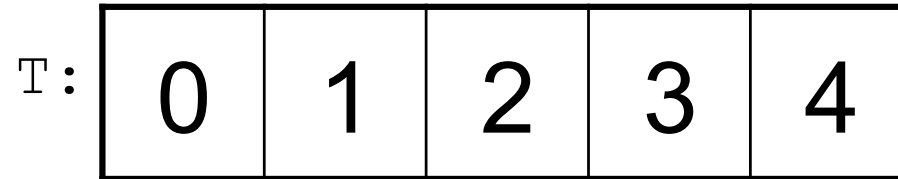
```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
    T[i+1] = T[i];  
T[0] = temp;
```

# Vérifier le code sur un exemple



```
→ int temp = T[NB-1];  
   for(int i = 0; i < NB - 1; i++)  
       T[i+1] = T[i];  
   T[0] = temp;
```

# Vérifier le code sur un exemple



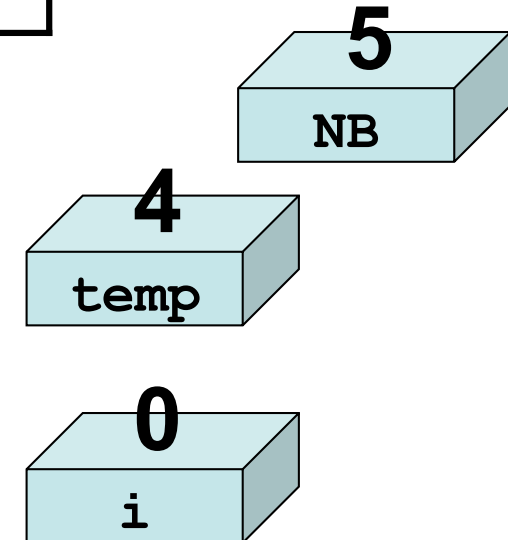
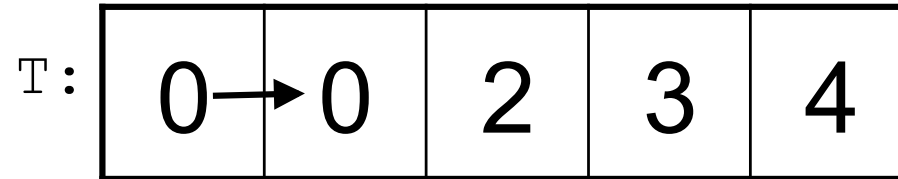
```
int temp = T[NB-1];
```

```
for(int i = 0; i < NB - 1; i++)
```

```
→ T[i+1] = T[i];
```

```
T[0] = temp;
```

# Vérifier le code sur un exemple

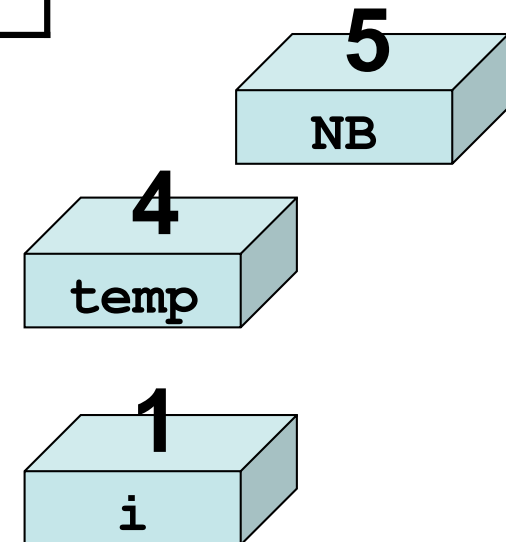


```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
→ T[i+1] = T[i];  
T[0] = temp;
```

# Vérifier le code sur un exemple

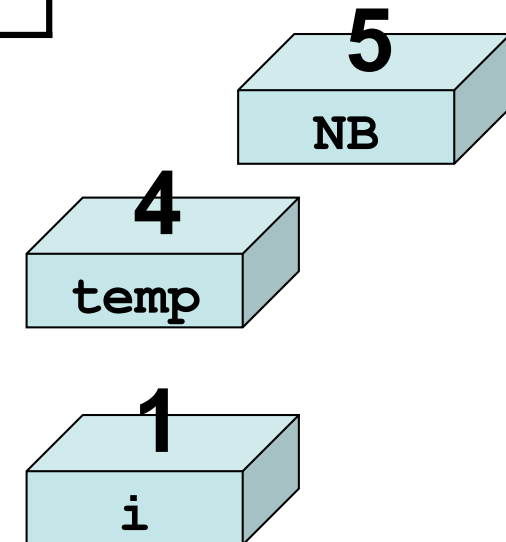
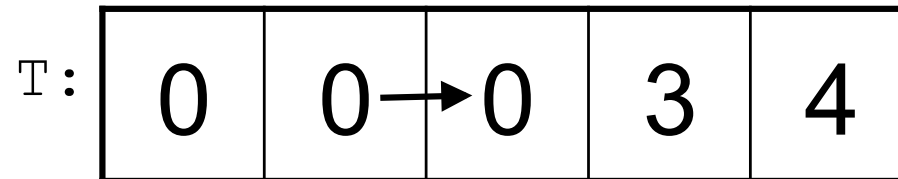
T: 

0	0	2	3	4
---	---	---	---	---



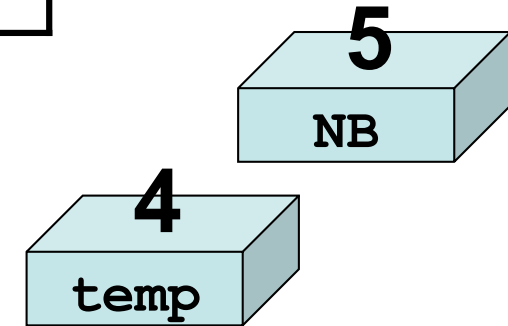
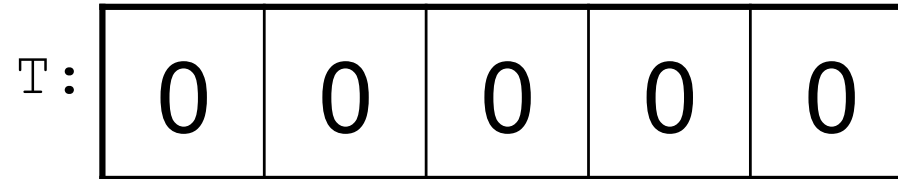
```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
→ T[i+1] = T[i];  
T[0] = temp;
```

# Vérifier le code sur un exemple



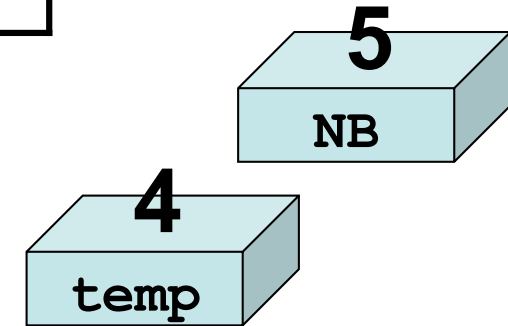
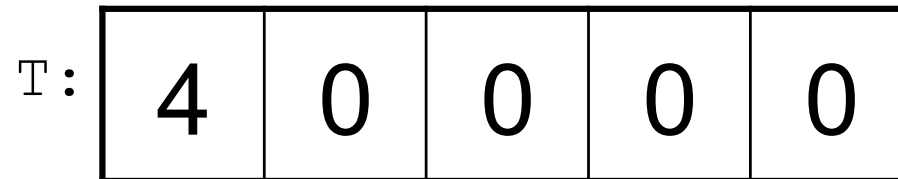
```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
→ T[i+1] = T[i];  
T[0] = temp;
```

# Vérifier le code sur un exemple



```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
    T[i+1] = T[i];  
→ T[0] = temp;
```

# Vérifier le code sur un exemple



```
int temp = T[NB-1];  
for(int i = 0; i < NB - 1; i++)  
    T[i+1] = T[i];  
→ T[0] = temp;
```

# Vérifier le code sur un exemple

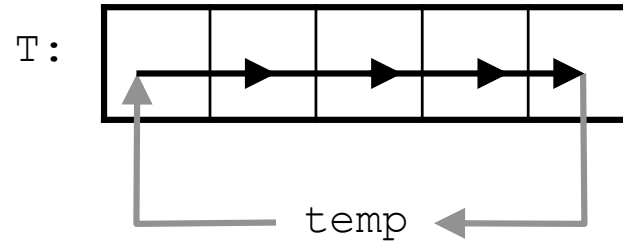
T: 

4	0	0	0	0
---	---	---	---	---

Au lieu de

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

# Décalage vers la droite



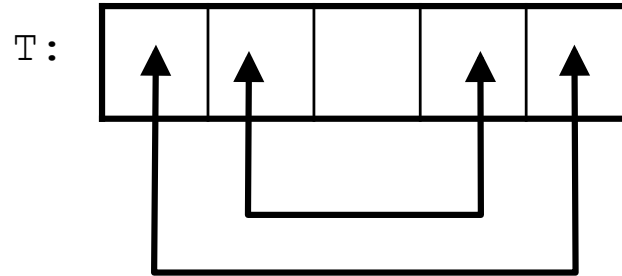
Nouvelle tentative:

```
int temp = T[NB-1];  
for(int i = NB-1; i > 0; i--)  
    T[i] = T[i-1];  
T[0] = temp;
```

Vérifiez sur l'exemple que cette version est correcte !

# Inverser les éléments du tableau

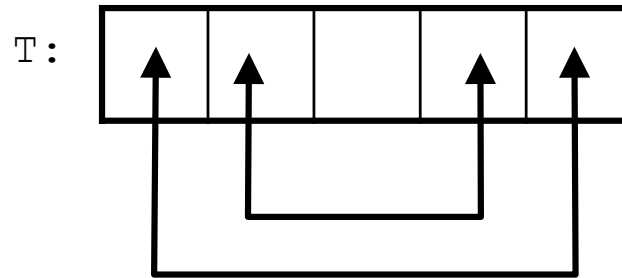
On peut procéder en échangeant les éléments de  $\mathbb{T}$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

Pour un tableau de 5 éléments:

```
T[0] = T[4];
```

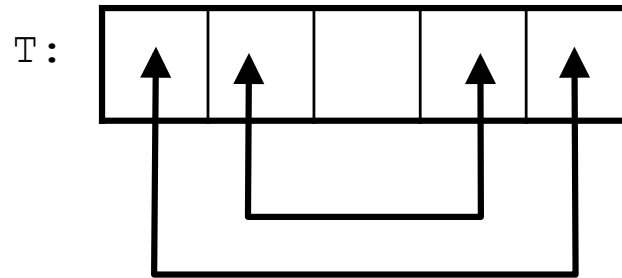
```
T[4] = T[0];
```

```
T[1] = T[3];
```

```
T[3] = T[1];
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

Pour un tableau de 5 éléments:

```
temp = T[0];
```

```
T[0] = T[4];
```

```
T[4] = temp;
```

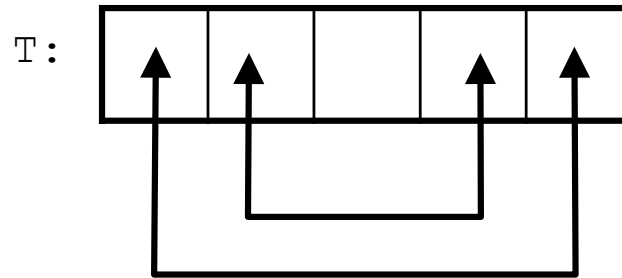
```
temp = T[1];
```

```
T[1] = T[3];
```

```
T[3] = temp;
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

Pour un tableau de NB éléments:

```
temp = T[0];
```

```
T[0] = T[4];
```

```
T[4] = temp;
```

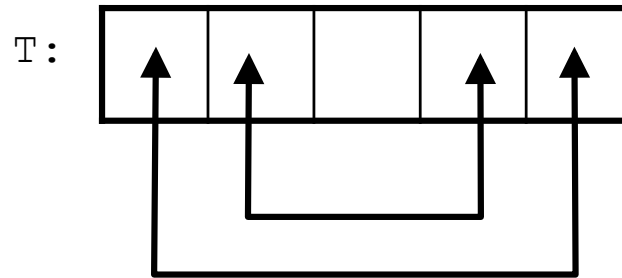
```
temp = T[1];
```

```
T[1] = T[3];
```

```
T[3] = temp;
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

Pour un tableau de  $NB$  éléments:

```
temp = T[0];
```

```
T[0] = T[NB-1];
```

```
T[NB-1] = temp;
```

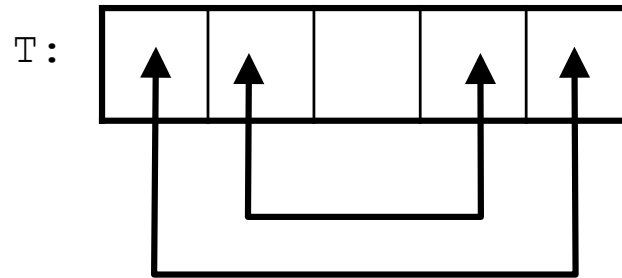
```
temp = T[1];
```

```
T[1] = T[3];
```

```
T[3] = temp;
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



On échange le premier élément avec le dernier, le deuxième avec l'avant dernier, etc...

Pour un tableau de  $NB$  éléments:

```
temp = T[0];
```

```
T[0] = T[NB-1];
```

```
T[NB-1] = temp;
```

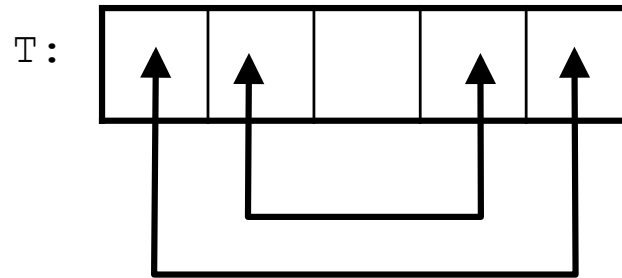
```
temp = T[1];
```

```
T[1] = T[NB-2];
```

```
T[NB-2] = temp;
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



```
temp = T[0];  
T[0] = T[NB-1];  
T[NB-1] = temp;
```

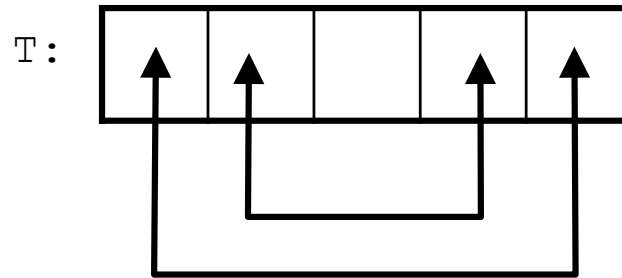
```
temp = T[1];  
T[1] = T[NB-2];  
T[NB-2] = temp;
```



```
for(int i = 0; i < ??; i++)  
{  
    int temp = T[i];  
    T[i] = T[??];  
    T[??] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



```
temp = T[0];  
T[0] = T[NB-1];  
T[NB-1] = temp;
```

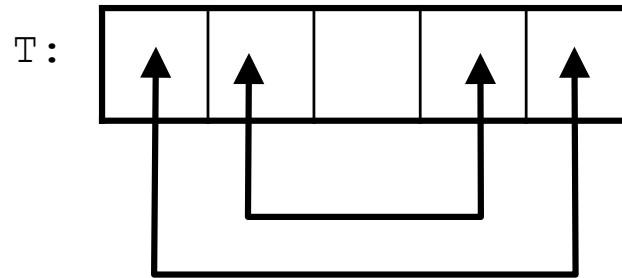
```
temp = T[1];  
T[1] = T[NB-2];  
T[NB-2] = temp;
```



```
for(int i = 0; i < ??; i++)  
{  
    int temp = T[i];  
    T[i] = T[??];  
    T[??] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



```
temp = T[0];  
T[0] = T[NB-1];  
T[NB-1] = temp;
```

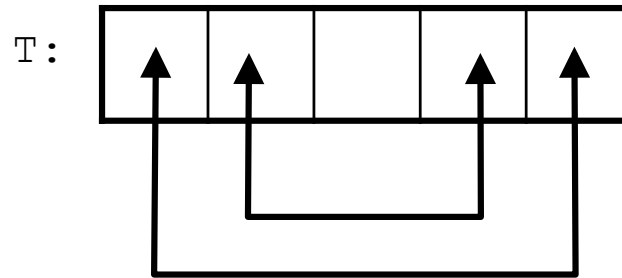
```
temp = T[1];  
T[1] = T[NB-2];  
T[NB-2] = temp;
```



```
for(int i = 0; i < ??; i++)  
{  
    int temp = T[i];  
    T[i] = T[NB-i-1];  
    T[NB-i-1] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



```
temp = T[0];  
T[0] = T[NB-1];  
T[NB-1] = temp;
```

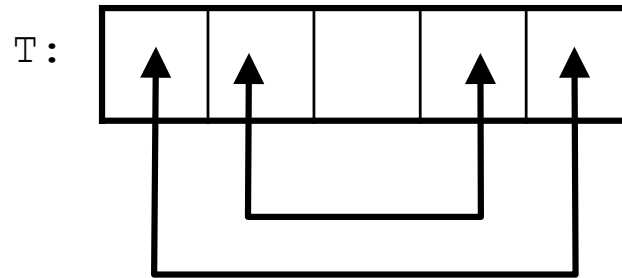
```
temp = T[1];  
T[1] = T[NB-2];  
T[NB-2] = temp;
```



```
for(int i = 0; i < ??; i++)  
{  
    int temp = T[i];  
    T[i] = T[NB-i-1];  
    T[NB-i-1] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



```
temp = T[0];  
T[0] = T[NB-1];  
T[NB-1] = temp;
```

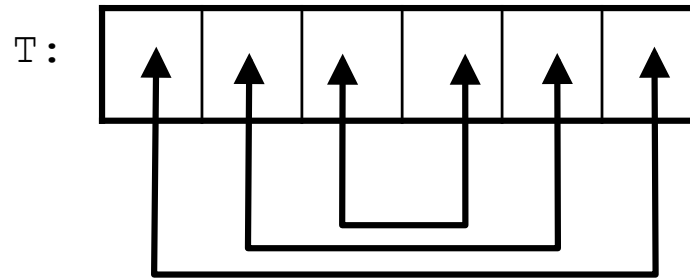
```
temp = T[1];  
T[1] = T[NB-2];  
T[NB-2] = temp;
```



```
for(int i = 0; i < NB/2; i++)  
{  
    int temp = T[i];  
    T[i] = T[NB-i-1];  
    T[NB-i-1] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:

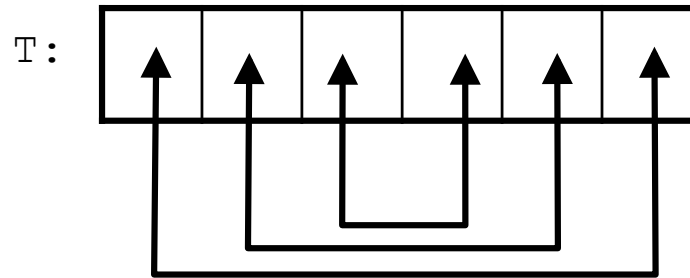


Que se passe-t-il si NB est pair ?

```
for(int i = 0; i < NB/2; i++)  
{  
    int temp = T[i];  
    T[i] = T[NB-i-1];  
    T[NB-i-1] = temp;  
}
```

# Décomposer

On peut procéder en échangeant les éléments de  $T$  2 par 2:



Que se passe-t-il si on fait ?

```
for(int i = 0; i < NB; i++)  
{  
    int temp = T[i];  
    T[i] = T[NB-i-1];  
    T[NB-i-1] = temp;  
}
```