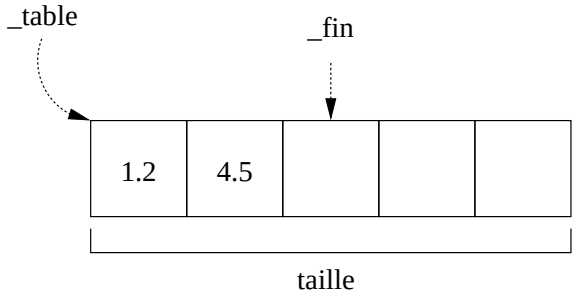
TD/TP 4

La classe TABLE

Exercice 1 Définition de la classe

La classe TABLE définit un tableau de `double`.
- `_table` : L'adresse du tableau
- `_taille` : La taille du tableau maximum que l'objet peut contenir (le tableau peut être plus petit)
- `_fin` : L'adresse de la case après la dernière case effectivement occupée.



Définir les attributs de la classe TABLE. (fichier table.h)

Exercice 2 Constructeurs, destructeurs par défaut

Lorsqu'un objet TABLE est construit on veut pouvoir préciser la taille max du tableau, par défaut cette taille sera de 16 cases. C'est à dire on veut pouvoir déclarer un objet TABLE de la façon suivante :

```
TABLE t1;
TABLE t2(32);
TABLE * t3 = new TABLE(64);
```

Question 1

Ecrire le constructeur et le destructeur par défaut. (fichier table.cpp)

Question 2

Ajouter dans les 2 fonctions précédentes une trace d'exécution sur le flux `cerr`. On affichera à chaque fois qu'une table est crée le nombre de case et l'adresse du premier élément ; à chaque fois qu'une table est détruite on affichera qu'il y a eu destruction.

Question 3

Dans un fichier main.cpp écrire une fonction main qui test la création et la destruction des objets.

Exercice 3 Compteur de TABLE

Question 1

Ajouter un compteur qui donne le nombre d'objet table en cours d'utilisation. On pensera à modifier le constructeur et le destructeur.

Question 2

Modifier la trace d'exécution sur `cerr` pour afficher ce nombre.

Exercice 4 Fonction d'accès

Question 1

Ecrire la fonction `bool ajout_elem(double element)` qui ajoute un élément à la fin du tableau si c'est possible. Cette fonction retourne vrai si tout s'est bien passée, faux sinon.

Question 2

Ecrire la fonction `double recup_elem(bool recupOK, unsigned int num)` qui retourne l'élément de la case num. Cette fonction met `recupOK` à vrai si tout s'est bien passée, faux sinon.

Question 3

Ecrire une fonction `size_t nb_element()` qui retourne le nombre d'éléments contenus effectivement dans la table.

Question 4

Ecrire une fonction `size_t taille_max()` qui retourne la taille max d'un tableau.

Question 5

Ecrire une fonction `double* end()` qui retourne un pointeur sur la case après le dernier élément.

Question 6

Ecrire une fonction `double* begin()` qui retourne un pointeur sur le premier element de la table.

Question 7

Testez vos fonctions dans le main.

Exercice 5 Constructeurs complémentaires

On veut pouvoir construire une table de la manière suivante :

```
TABLE t1;
TABLE t2(t1);
list<double> l;
...// rempli la liste
TABLE t3(l);
```

Question 1

Ecrire un constructeur par copie d'une table.

Question 2

Ecrire un constructeur par conversion qui construit une table à partir d'une liste.

Question 3

Tester vos fonction dans le main.

Exercice 6 Surcharge de l'opérateur «

On veut surcharger l'opérateur pour pouvoir écrire la ligne suivante :

```
TABLE t;
... //remplissage de la table
cout << t << endl;
```

et avoir le résultat :

```
Table = 14.3 | 23 | 25
```

La surcharge ne fait pas partie de la classe puisque l'opérateur appartient à la classe `stream`. Pour surcharger l'opérateur la déclaration se fait de la façon suivante :

```
ostream& operator<< (ostream& out, const TABLE& t)
```