Algorithmique et Structures de Données

Cours 2 - Les structures de données séquentielles

H. Djerroud

LIASD - Université Paris 8

Hiver 2020

Plan de cours

Partie 1 :

- Les tableaux et implémentation en C
- Opérations de base
- Algorithmes de Tri
- Algorithmes de recherche
- Implémentation des tableaux en C++

Partie 2 : STL C++

- les chaînes de caractères : `basic_string`, `string` et `wstring`
- les tableaux : `vector`
- Queue à double entrée : `deque`
- Les algorithmes : l'en-tête `<algorithm>`

Définition d'un tableau

Définition

Un tableau est une structure de donnée T qui permet de stocker un certain nombre d'éléments $T[i]$ repérés par un index i. Les tableaux vérifient généralement les propriétés suivantes :

- Les éléments ont le même type
- Les éléments stockés en mémoire sont adjacents
- Le nombre d'éléments stockés est fixé
- l'accès et la modification de l'élément numéro i est en temps constant $O(1)$, indépendant de i et du nombre d'éléments dans le tableau.

Tableaux en C

5	10	45	2	10	7	14	13
0	1	2	3	4	5	6	7

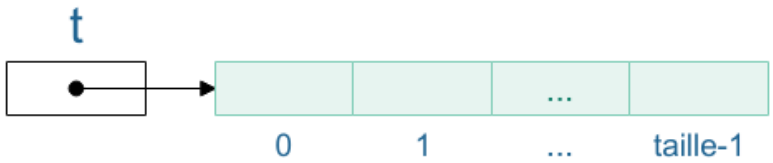
En *C* deux types de tableaux :

- Tableaux statiques :
  - Taille est connue à la compilation (depuis la *norme C99* il est possible déclarer la taille du tableau à l'exécution, le tableau est alors créé sur la pile voir : **Variable Length Array (VLA)**)
- Tableaux dynamiques :
  - Définition dynamique en deux temps :

Tableaux statiques

Tableaux statiques :

```
elem t[taille];
```



l'adresse de $t[i]$ est noté $t + i$. Calculée comme suite :

$$Addr(t[i]) = Addr(t[0]) + sizeof(elem) * i$$

Tableaux dynamiques

Définition dynamique en deux temps :

- Déclaration

```
#include <stdlib.h>
elem *t;
...
```

- Allocation

```
t = (elem*)malloc(taille*sizeof(elem));
```

Opérations de base

Soit :

- Un tableau de taille *Taille*
- éléments 0 $\leq$ i ¡ taille $\leq$ max_taille initialisés

Algorithme de recherche dans un tableau

Définition

Un algorithme de recherche

- Recherche par force brute
- Recherche dichotomique

Algorithmes de tri

Définition

Un algorithme permet d'organiser une collection d'objets selon une relation d'ordre déterminée.

Comparaison des algorithmes de tri

Algorithme	**Moyenne**	**Pire cas**	**Espace**	**Remarque**
Bulle	N^2	*en place*	*en place*	*stable*
Sélection	N^2	N^2	*en place*	–
Insertion	N^2	N^2	*en place*	*stable*
Peigne	$N \log N$	N^2	*en place*	*pas stable*
Rapide	$N \log N$	N^2	*en place*	*pas stable*
Fusion	$N \log N$	N^2	$x2$	*stable*
Tas	$N \log N$	$N \log N$	*en place*	*stable*

Tri sélection (principe)

```
void tri_selection(int *tab, int size) {
     int en_cours, plus_petit, j, temp;
     for (en_cours = 0; en_cours < size - 1; en_cours++){
         plus_petit = en_cours;
         for (j = en_cours 1; j < size; j++)
              if (tab[j] < tab[plus_petit])
                   plus_petit = j;
          temp = tab[en_cours];
          tab[en_cours] = tab[plus_petit];
          tab[plus_petit] = temp;
     }
}
```

Tri bulle (principe)

```
void tri_bulle(int* tableau, int size){
    int passage = 0;
    bool permutation = true;
    int en_cours;
    while ( permutation) {
        permutation = false;
        passage ++;
        for (en_cours=0;en_cours<size-passage;en_cours++) {
            if (tableau[en_cours]>tableau[en_cours+1]){
                permutation = true;
                // on echange les deux elements
                int temp = tableau[en_cours];
                tableau[en_cours] = tableau[en_cours+1];
                tableau[en_cours+1] = temp;
            }
        }
    }
}
```

Tri insertion (principe)

```
void tri_insertion(int* t)
{
    int i, j;
    int en_cours;

    for (i = 1; i < 20; i++) {
        en_cours = t[i];
        for (j = i; j > 0 && t[j - 1] > en_cours; j--) {
            t[j] = t[j - 1];
        }
        t[j] = en_cours;
    }
}
```

Tri rapide (code)

```
int tri_rapide(int* tab, int size){
      if (size == 0){ return 0; }
      int pivot = tab[size-1];
      int sel = 0,pos = 0, tmp = 0;
      while (pos < size){
        if (tab[pos] <= pivot){
          tmp = tab[sel];
          tab[sel] = tab[pos];
          tab[pos] = tmp;
          sel++;
        }
        pos++;
      }
      tri_rapide(tab, sel-1);
      tri_rapide(tab+sel, size - sel);
}
```

Introduction C++

Introduction C++

Plan

- Les templates
- La bibliothèque standard
- Le type vector
- Les algorithmes

Plan

- Les templates (modèles en français, ou patrons)
- Un modèles génériques de code qui permettent de créer automatiquement des fonctions (dans le cas de fonctions templates)
- Fournir un paramètre à un modèle générique s'appelle la spécialisation

Exemple template

```
#include <iostream>

template <typename T>
T add (T a, T b){
    T c = a+b;
    return c;
}

int main (int argc, char* argv[]){
    std::cout << "Hello " << std::endl;
    int a = 5, b = 6, c;
    c = add (a,b);
    std::cout << c << std::endl;
    return 0;
}
```