Analyse et programmation

Thème 8 — le type «vector»

Gabriel Girard

Département d'informatique


UNIVERSITÉ DE SHERBROOKE

5 novembre 2015

Synthèses des problèmes des tableaux en C++

Difficultés identifiées

- Taille fixe déterminée statiquement.
- Ne connaît pas sa taille.
- Pas de mécanisme de vérification au niveau des indices
  - Risque de dépasser la taille.
  - Risque d'indice négatif.
- Pas de transmission en valeur de retour.

Solution : implanter un nouveau type ...

Exemple

```
struct vecteur_t
{
    float elements[MAX];
    int taille;
};
```

Solution : implanter un nouveau type ...

Exemple

```
using taille_t =  unsigned int;
using element_t = float;

struct vecteur_t
{
    element_t elements[MAX];
    taille_t taille;
};
```

Solution : implanter un nouveau type avec des opérations

Exemple

```
void init(vecteur_t& v)

void ajouter(element_t ele, vecteur_t& v)

element_t lire(vecteur_t v, taille_t ind)
```

Le type `vector` : généralité

Nouveau type

- Implante un tableau à une dimension
- Nécessite `#include <vector>`
- À utiliser de préférence à un tableau
- L'interface peut être le même qu'un tableau
  ... avec les mêmes problèmes
- Fiabilité possible et facilité d'utilisation
- Comparable à `string`, mais reste général

Définition

Syntaxe

```
vector<type> nom_vecteur (dimension);
vector<type> nom_vecteur ;
```

Exemples

- `vector<double> notes (10);`
- `vector<int> indices;`
- `vector<char> ligne(80);`
- `vector< vector<int> > matrice(5,5);`
- `vector<string> noms;`

Accès

Syntaxe

un_vecteur[*un_indice*]

- aucun vérification (en général) ;
- ne devrait être utilisé qu'exceptionnellement.

un_vecteur.**at**(*un_indice*)

- « protège » d'un dépassement des limites du vecteur ;
- devrait toujours être utilisé.

Exemple

```
vector<string> liste_des_noms (4) ;
liste_des_noms.at(3) = "Benoit" ;
cout << liste_des_noms[3] << endl;
```

Accès

Syntaxe

```
un_vecteur.front()
un_vecteur.back()
```

- lit la première valeur
- lit la dernière valeur

Taille d'un vecteur

Généralité

- On peut obtenir la taille d'un vecteur (size)
- Un vecteur défini sans dimension est vide (taille = 0)
- Des opérations existent pour changer dynamiquement la dimension d'un vecteur
  - changer la taille (`resize`)
  - ajouter un élément (`push_back, insert`)
  - retirer un élément (`pop_back, erase`)

Obtenir la taille d'un vecteur

Syntaxe

un_vecteur.**size**()

- retourne la taille courante du vecteur

Exemple

```
vector<int> m (3,1); // initialise avec des 1
cout << m.size() << endl;
cout << m.at(0) << "," << m.at(1) << endl;
```

Modifier la taille d'un vecteur

Syntaxe

un_vecteur.**resize**(*une_taille*)

- change la taille courante du vecteur

Exemple

```
vector<int> m (3);
m.resize(10);
cin >> m.at(9) ;
```

Modifier la taille d'un vecteur

Ajouter un élément

```
un_vecteur.push_back(un_element)
un_vecteur.insert(un_element,un_indice)
un_vecteur.insert(un_vecteur.begin()+i,n,x)
```

- ajoute un élément à la fin du vecteur
- insère une valeur entre deux indices
- insère n fois x à partir de i

Exemple

```
vector<string> liste_des_noms (1) ;
liste_des_noms.at(0) = "Achille" ;
liste_des_noms.push_back ("Hector") ;
liste_des_noms.push_back ("Maxence") ;
cout << liste_des_noms.at(2) << endl ;
```

Modifier la taille d'un vecteur

Retirer un élément

```
un_vecteur.pop_back(un_indice)
un_vecteur.erase(un_vecteur.begin()+i)
```

- retire la dernière valeur et diminue la taille du vecteur
- efface la cellule d'indice *i*

Exemple

```
vector<string> liste_des_noms (1) ;
liste_des_noms.at(0) = "Achille" ;
liste_des_noms.push_back ("Hector") ;
liste_des_noms.pop_back () ;
cout << liste_des_noms.at(1) << endl ;
```

Autres commandes

Syntaxe

```
un_vecteur.clear()
un_vecteur.assign(n,x)
un_vecteur.swap(un_autre_vecteur)
```

- vide un vecteur
- vide le vecteur et assigne n nouveaux éléments x
- échange le contenu de deux vecteurs

Itération

Syntaxe

- Avec les vecteurs on utilise une nouvelle forme d'itération

```
std::vector<int> maListe;
// inserer des valeurs....
//
for (int element : maListe)
{
    cout << element << endl;
}
```

Vecteur à plusieurs dimensions

Exemple 1

```
vector< vector<double> > matrice_identite (3,3);
matrice_identite.at(0).at(0) = 1.0;
matrice_identite.at(1).at(1) = 1.0;
matrice_identite.at(2).at(2) = 1.0;
```

Exemple 2

```
vector< vector<double> > m (3,vector<double>(3,-1.0));
```