Cours de C++

Utilisations des conteneurs

Cécile Braunstein
cecile.braunstein@lip6.fr

Introduction

Containers - Why ?

- Help to solve messy problems
- Provide useful function and data structure
- Consistency between containers

Containers

- Collection of objects
- Each `containers` type is optimized for a specific use (access/modification).
- Main containers :
  `list, vector, stack, queue, map`

Container's properties

- Containers have their own elements
- Elements of a container have to support the copy and assigment instruction (=)
- All containers have a method `empty()` and `size()` in constant time
- All containers have a method `begin()` and `end()`

Container's type

`list`

- Insert and remove anywhere in constant time
- Automatic memory management

`vector`

- General purpose
- Fast access by index (constant time)
- Insert and remove an element at the end in constant time
- Other insert and remove in linear time

`set, map`

- Access an element by a key in constant time
- Fast search of an element

How to choose ?

What is the purpose ?

- How we want ot access the element (randomly, in one order ...)
- Which modification on the collection of data (add/remove elements, sort ...)

Programm performance

- Access time/ Modification time
- Time depends on the number of elements
- Types of times : linear, log, exponential ...
- Memory usage ...

How to access element ?

Iterator Purpose

- Pointer generalization
- Use for a sequential access to elements
- Optimisation regarding the container's type

Iterator Definition

An iterator is a value that

- Identifies a container and an element in the container
- Lets us examine the value stored in that element
- Provides operations for moving between elements in the container
- Restricts the available operations to correspond to what the container can handle efficently

First example

```
vector<double> hw;
read_hw(cin,hw);

vector<double>::size_type
  i;

for(i = 0; i != hw.size();
    ++i)
{
   cout << hw[i] << endl;
}
```

```
vector<double> hw;
read_hw(cin,hw);

vector<double>::iterator
    iter;

for(iter = hw.begin();
  iter != hw.end(); ++iter)
{
  cout << *iter << endl;
}
```