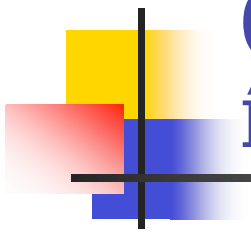
SEMINARIO C++

STL: Standard Template Library

v. 20100915

Programación orientada a objetos
Curso 10-11



C++ - Librería STL

ÍNDICE

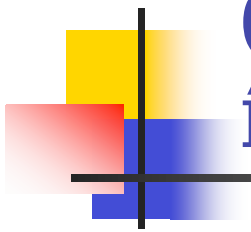
C++ - Librería STL

OBJETIVO

- La **STL** (Standard Template Library – Biblioteca Estándar de Plantillas) es una biblioteca de C++ que básicamente está compuesta de:
  - ***Contenedores***. Permiten almacenar objetos.
  - **Algoritmos** de uso frecuente.
  - ***Iteradores*** para recorrer los elementos incluídos en los contenedores.

- Dentro de la biblioteca podemos encontrar diferentes tipos de contenedores:
  - Vectores, Listas, Colas, Pilas…

- Nuestro objetivo es aprender el uso básico de uno de los tipos de datos contenedores: el tipo **Vector**.



C++ - Librería STL

ÍNDICE

C++ - Librería STL

El contenedor VECTOR

- Permite almacenar cero o más elementos del mismo tipo, pudiendo acceder a ellos *individualmente* mediante un índice (acceso aleatorio).
- El número de elementos del vector puede variar dinámicamente.
- La gestión de memoria se hace de manera totalmente transparente al usuario.
- Va a poder albergar elementos de cualquier tipo.
- Para usarlo, simplemente:

```
#include <vector>
```

Declaración más común:

```
vector<TIPO> miVector;
```

Donde TIPO puede ser cualquier tipo de los que ofrece C++, así como cualquiera definido por un usuario.

C++ - Librería STL

El contenedor VECTOR

- Así, podríamos declarar los siguientes vectores:
  - `vector<double>  vectorReales;` // De números reales.
  - `vector<string> vectorCadenas;` // De cadenas de caracteres.
  - `vector<MiStruct> vectorObj;` // Contendrá elementos de un tipo definido por un usuario.
- Crear un vector de 10 enteros:
  - *`vector<int> vectorEnteros(10);`*
- Si deseamos inicializar los elementos del vector a algún valor en concreto, entonces añadimos un argumento más.
  - *`vector<int> vectorEnteros(10,-1);`*

C++ - Librería STL

El contenedor VECTOR

- **Algunas operaciones**

```
vector<int> miVector;
```

  - La mayor parte de las operaciones usa esta sintáxis:

```
miVector.operación(argumentos)
```

  - Obtener el número de elementos almacenados en el vector:

```
mivector.size()
```

  - Almacenar un elemento al final del vector:

```
mivector.push_back(17);
```

  - Acceso aleatorio a los elementos de un vector:

```
cout << mivector[6];
mivector[6] = 17;     // sin comprobación de rango
mivector.at(6) = 17; // con comprobación de rango
```

C++ - Librería STL

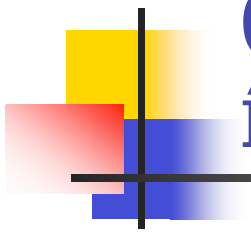
El contenedor VECTOR

- Para cualquier sub-índice **n**, lo siguiente debe ser verdadero:

```
0 <= n < size()
```

- Ejemplo:

```
vector<int> mivector;
mivector[0] = 25;        // ¡ERROR!.

mivector.push_back(15);
mivector[0] = 25;        // OK
```



C++ - Librería STL

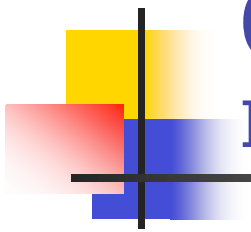
ÍNDICE

C++ - Librería STL

OBJETIVO

- Los contenedores pueden ser "recorridos" utilizando **iteradores.**

- Estos nos permiten:
  - Colocarnos en el *primer elemento* de un contenedor.
  - *Avanzar* al siguiente elemento.
  - *Ver* el elemento (o modificarlo).
  - Comprobar si ya hemos llegado hasta el *final*



C++ - Librería STL

ITERADORES

- Un **iterador** es una especie de puntero utilizado por un **algoritmo** para recorrer los elementos almacenados en un **contenedor**.

- Un iterador tiene que ser del tipo de dato que posee la lista.
  - Si tenemos por ejemplo un vector que posee datos de tipo entero (**int**) entonces el iterador será algo similar a un puntero a entero.

C++ - Librería STL
ITERADORES

- Sintaxis general para declarar un iterador
  - **Contenedor<tipo>::iterator iter;**

Tipo del iterador

Nombre de variable

Ejemplo 1:

```
vector<int>::iterator el_iterador;
```

*Obtiene un iterador para **vector** de tipo **int.***

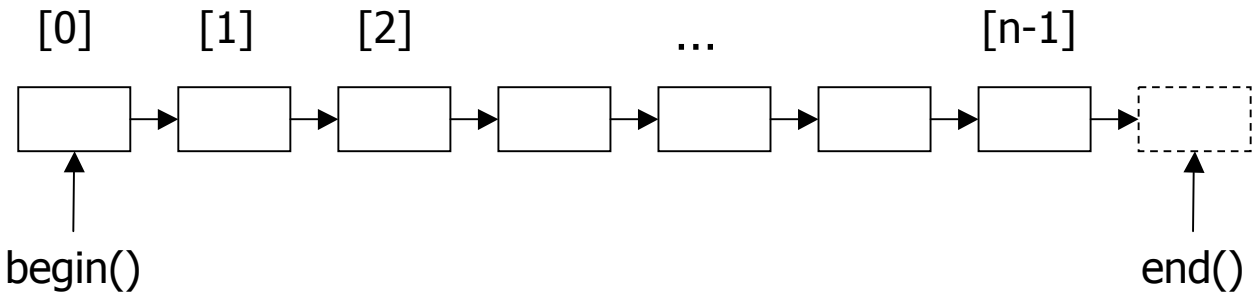
C++ - Librería STL
ITERADORES

- Operaciones aritméticas permitidas entre iteradores
  - (equivalentes a la aritmética de punteros)

Operación	Resultado	Comentario
`It++`	Iterador	Desplazamiento ascendente (1 elemento).
`It--`	Iterador	Desplazamiento descendente (1 elemento).
`It + n`	Iterador	Desplazamiento ascendente (n elementos).
`It - n`	Iterador	Desplazamiento descendente (n elementos).
`It1 == It2`	bool	Igualdad entre iteradores.
`It1 != It2`	bool	Desigualdad de iteradores.
`*It`	elemento	Elemento al que apunta el iterador

C++ - Librería STL
ITERADORES

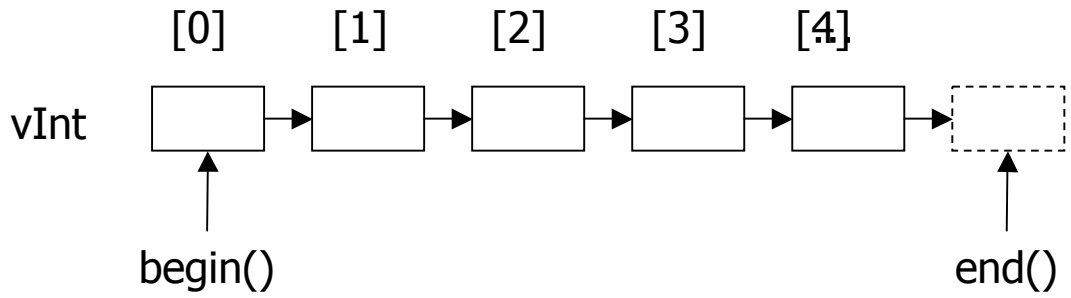
- Operaciones para obtener un iterador
  - **iterator begin();**
    Devuelve un iterador que referencia el comienzo del vector.
  - **iterator end();**
    Devuelve un iterador que referencia la posición siguiente al final del vector.



C++ - Librería STL
ITERADORES

- Un ejemplo

```
vector<int> vInt(5,3);
vector<int>::iterator it;

for (it = vInt.begin(); it != vInt.end(); it++) {
    cout << *it << endl;
}

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



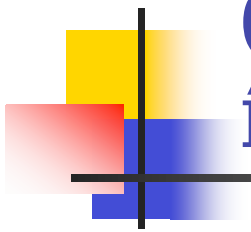
C++ - Librería STL
ITERADORES

- Cuando el contenedor sobre el que se quiere iterar es constante (no puede ser modificado):
  - `Contenedor<tipo>::`***`const_iterator`*** `iter;`

Ejemplo 2:

```
const vector<int> v(5,3);
vector<int>::const_iterator it;
...
```

*Obtiene un iterador a un **vector** de **int** constante.*



C++ - Librería STL

ÍNDICE

C++ - Librería STL

Operaciones del tipo 'vector'

- **size_type size();**
  Devuelve el número de elementos almacenados en el vector. El tipo size_type es un entero sin signo.

- **bool empty();**
  Devuelve *true* si el número de elementos es cero y *false* en caso contrario.

- **size_type capacity();**
  Devuelve el tamaño (en número de elementos) del espacio reservado en memoria en ese momento por el vector. Siempre es mayor o igual que size.

C++ - Librería STL

Operaciones del tipo 'vector'

- Funciones para insertar
  - **void push_back(const T& x);**
    - Inserta un elemento **x** al final del vector. Aumenta el tamaño del vector en 1. **T** es el tipo de datos que almacena el vector.

Ejemplo 1:

```
vector<int> a;
a.push_back(5);
```

Ejemplo 2:

```
double x;
vector<double> trabajos;
while (cin >> x)
  trabajos.push_back(x);
```

El bucle while trabaja leyendo valores de la entrada estándar y almacenándolos en el vector. Así hará hasta que encuentre end.of.file o una entrada que no sea double.

C++ - Librería STL

Operaciones del tipo 'vector'

- Ejemplo 1

Creamos un contenedor tipo **vector** para contener enteros:

```
vector<int> vInt;
for (int i=0; i < 100; i++)
  vInt.push_back(i);
```

Y lo recorremos mediante un iterador:

```
vector<int>::iterator iter;
for (iter = vInt.begin(); iter != vInt.end(); iter++) {
        cout << *iter << endl;
}
```

C++ - Librería STL

Operaciones del tipo 'vector'

- *EJEMPLO 2*

```
vector<int> vInt;
for (int i=0; i < 100; i++) vInt.push_back(i);

cout << "Tamaño actual:   " << vInt.size() << endl;
cout << "Capacidad actual: " << vInt.capacity() << endl;
```

SALIDA

Tamaño actual: 100
Capacidad actual: 256 (depende de la implementación)

C++ - Librería STL

Operaciones del tipo 'vector'

- Función **insert.**
  - **iterator insert(iterator loc, const T& val);**
    - inserta ***val*** antes de ***loc.*** Devuelve un iterador que apunta al elemento insertado.
  - **void insert(iterator loc, size_type num, const T& val);**
    - inserta ***num*** copias de ***val*** antes de ***loc***.
  - **void insert(iterator loc, input_iterator start, input_iterator end);**
    - inserta los elementos entre ***start*** y ***end*** antes de ***loc*** . Start y end son iteradores que especifican un rango de elementos, a menudo de otro vector. Copias de los elementos en el rango [start,end] son insertados antes de la posición *loc*.

C++ - Librería STL

Operaciones del tipo 'vector'

- Ejemplo inserción 1:

```
// Crea un vector, almacenando los primeros 10 caracteres del alfabeto

vector<char> alphaVector;
for( char c='A'; c < 'K'; c++ ) {
   alphaVector.push_back( c );
}

 // Inserta cuatro C's en el vector
vector<char>::iterator theIterator = alphaVector.begin();
alphaVector.insert( theIterator, 4, 'C' );

 // Muestra el vector
for ( theIterator = alphaVector.begin();
      theIterator !=   alphaVector.end(); theIterator++ )    {
   cout << *theIterator;
}
// La salida será : CCCCABCDEFGHIJ
```

C++ - Librería STL

Operaciones del tipo 'vector'

EJEMPLO INSERCIÓN 2 (HACER TRAZA)

```
vector<int> mivector (3,100);
vector<int>::iterator it;

it = mivector.begin();
it = mivector.insert ( it , 200 );

mivector.insert (it,2,300); // devuelve void
it = mivector.begin();

vector<int> otrovector (2,400);
mivector.insert (it+2,otrovector.begin(),otrovector.end());

int miarray [] = { 501,502,503 };
mivector.insert (mivector.begin(), miarray, miarray+3);
cout << "mivector contiene:";
for (it=mivector.begin(); it<mivector.end(); it++)
  cout << " " << *it; cout << endl;
```

La salida será :

mivector contiene: 501 502 503 300 300 400 400 200 100 100 100

C++ - Librería STL

Operaciones del tipo 'vector'

Moraleja ejemplo inserción 2:

En general, si se modifica el contenedor, los iteradores que apuntaban a elementos del contenedor ya no son válidos. En particular **`insert()`** invalida el iterador que se usa como punto de inserción.

C++ - Librería STL

Operaciones del tipo 'vector'

- Funciones para borrar
  - **void erase(iterator first, iterator last);**
    Borra los elementos del vector que estén situados entre los iteradores *first* y *last*. Incluye el elemento apuntado por *first* pero no el apuntado por *last*.
  - **iterator erase (iterator position) ;**
    Borra el elemento que está en la posición indicada por *position*.
  - **void pop_back();**
    Elimina el último elemento. Reduce el tamaño en 1.
  - **void clear();**
    Borra todos los elementos de un vector.

Ejemplo 1:

```
vector<int> a;
...
a.erase(a.begin(),a.end());
```

// Se borran todos los elementos entre la primera y la última posición, es decir, todos los elementos de un vector.

C++ - Librería STL

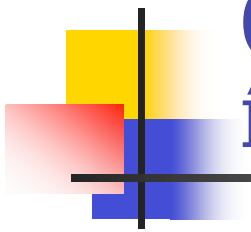
Operaciones del tipo 'vector'

EJEMPLO DE BORRADO

```
int i;
vector<int> mivector;

// Insertamos valores (Del 1 al 10)
for (i=1; i<=10; i++) mivector.push_back(i);

// Borramos el 6º elemento
mivector.erase (mivector.begin()+5);

// Borramos los primeros 3 elementos
mivector.erase (mivector.begin(),mivector.begin()+3);

cout << "mivector contiene:";
for (i=0; i<mivector.size(); i++)
  cout << " " << mivector[i]; cout << endl;
```

La salida será :

mivector contiene: 4 5 7 8 9 10



C++ - Librería STL

ÍNDICE

C++ - Librería STL

Operadores del tipo 'vector'

- Algunos operadores que se pueden usar con `vector`:
  - **El operador de asignación (=)** sustituye el contenido de un vector por el de otro.

```
vector<int> a;
vector<int> b;

a.push_back(5);
a.push_back(10);

b.push_back(3);

b = a; // El vector b contiene dos elementos: 5 y 10 (los
mismos que contenía el vector a).
```

  - **El operador comparación (==)** comprueba si dos vectores contienen los mismos elementos. Para ello lleva a cabo una comparación elemento a elemento.

C++ - Librería STL

Operadores del tipo 'vector'

- **El operador de subíndice** devuelve una referencia a un elemento del vector. Una referencia con un subíndice igual a cero devuelve el primer elemento del vector. Así, el rango del subíndice debe estar entre cero y size()-1.

```
vector<double> vec;
vec.push_back(1.2);
vec.push_back(4.5);

vec[1] = vec[0] + 5.0;
vec[0] = 2.7; // El vector contiene ahora los
// elementos 2.7, 6.2
```

C++ - Librería STL

Ejemplo (uso de 'vector')

- **Otro ejemplo (de todo un poco)**

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
  vector<int> v(5, 1);
  int i;

  cout << "Size = " << v.size() << endl;
  cout << "Contenido Original:" << endl;
  for(i = 0; i <v.size(); i++)    cout << v[ i ] << " ";
  cout << endl << endl;

  vector<int>::iterator p = v.begin();
  p += 2;

  v.insert(p, 10, 9);

  cout << " Tamanyo despues de insert  = " << v.size() << endl;
  cout << " Contenido despues insert \n";
  for(i = 0; i <v.size(); i++)  cout << v[ i ] << " ";
  cout << endl << endl;
```

C++ - Librería STL

Ejemplo (uso de 'vector')

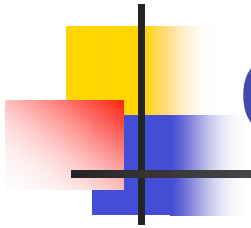
- **Continúa EJEMPLO**

```
p = v.begin();
p += 2;
v.erase(p, p+10);

cout << " Tamanyo despues de borrado  = " << v.size() << endl;
cout << " Contenido despues borrado \n";
for(i = 0; i <v.size(); i++)     cout << v[ i ] << " ";
cout << endl;
}
```

- ***SALIDA:***

Size = 5
Contenido Original:
1 1 1 1 1
Tamaño después de insert = 15
Contenido después de insert
1 1 9 9 9 9 9 9 9 9 9 1 1 1

Tamaño después de borrado = 5
Contenido después de borrado
1 1 1 1 1



C++ - Librería STL

- Ejercicio
  - Usando el tipo 'vector' de la librería STL:
    - Crea un vector con que contenga los enteros del 1 al 10 (en ese orden)
    - Elimina los números pares
    - Invierte el orden de los elementos en el vector
    - Inserta los números pares en el lugar que les corresponde en el nuevo orden.
    - Imprime el contenido del vector y el número de elementos que contiene tras cada modificación de su contenido.