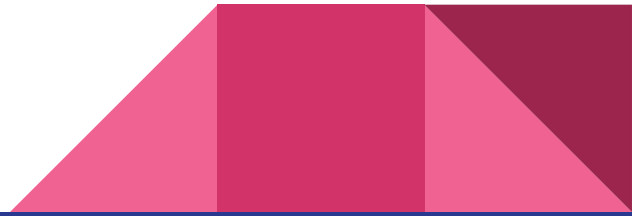
Herramientas de C++: Standar Template Library (STL)

Informática II - R2004
2018

STL

La STL (Standard Template Library) de C++ es una conjunto de plantillas de clases y funcione que permite a los programadores implementar fácilmente estructuras estándar de datos como colas (queues), listas (lists), y pilas (stacks). A su vez posee clases para manejo de strings y streams así como también incluye dentro del namespace std los headers standard de C,



Templates

Es una herramienta que permite utilizar una misma clase ya diseñada para manejar distintos tipos de datos. Por ejemplo, teniendo:

```
class vector {

private:
        int * comienzo;
        int size;
        int err;

public:

        vector ( int );
        vector ();

        ~vector();

        int & operator [] ( int );

        void operator + (int);

}
```

```
...

vector a(3);

...

a[0] = 54;
a[1] = 30;
a[2] = 33;

...

a + 90;
```

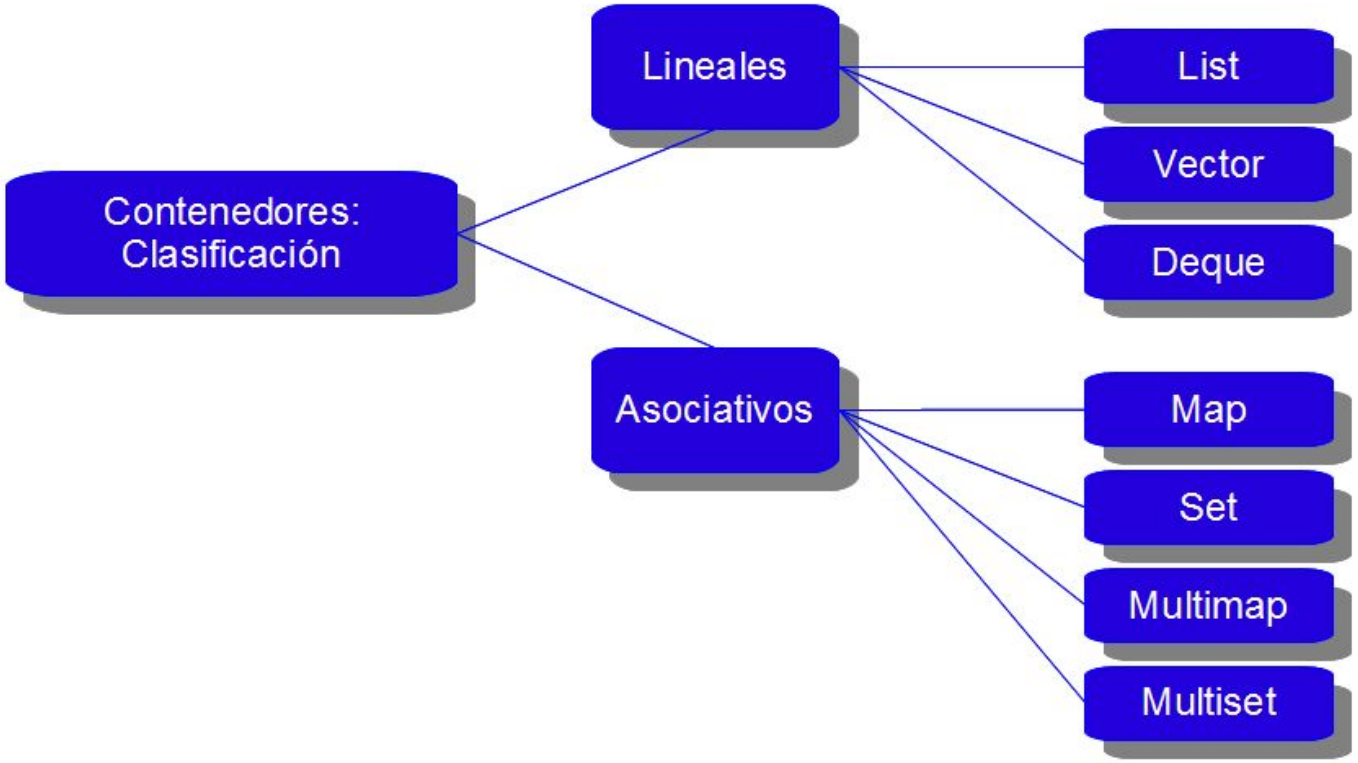
¿Cómo hago si ahora quiero un vector de chars, de floats, o de doubles?

Template (II)

```
template <class comodin> class vector {

private:
        comodin * comienzo;
        int size;
        comodin err;

public:

        vector ( int );
        vector ();

        ~vector();

        comodin & operator [] ( int );

        void operator + ( comodin );

}
```

```
vector <int> a(3);

...

a[0] = 54;
a[1] = 30;
a[2] = 33;

...

a + 90;
```

Librería STL - Contenedores



Instancia de un contenedor

```
vector<int> miVector;

vector<char> miVector2(tamanioInic);

list<misDatos_t> miLista;
```

Métodos clases contenedoras

Tabla 1: operaciones comunes de contenedores

`X::size()`	Devuelve la cantidad de elementos que tiene el contenedor como un entero sin signo
`X::max_size()`	Devuelve el tamaño máximo que puede alcanzar el contenedor antes de requerir más memoria
`X::empty()`	Retorna verdadero si el contenedor no tiene elementos
`X::swap(T & x)`	Intercambia el contenido del contenedor con el que se recibe como parámetro
`X::clear()`	Elimina todos los elementos del contenedor

`v == w` `v != w`
`v < w` `v > w`
`v <= w` `v >= w`
	Supóngase que existen dos contenedores del mismo tipo: *v* y *w*. Todas las comparaciones se hacen lexicográficamente y retornan un valor booleano.

Tabla 2: operaciones comunes de contenedores lineales

`S::push_back(T & x)`	Inserta un elemento al final de la estructura
`S::pop_back()`	Elimina un elemento del final de la estructura
`S::front()`	Devuelve una referencia al primer elemento de la lista
`S::back()`	Devuelve una referencia al último elemento de la lista

Ejemplo vector

```
#include <vector>

using namespace std;

int main(void)
{
    vector<int> v; //Observar que no fué inicializado su tamaño.

    //cargo un vector de 30 elementos enteros con valores consecutivos
    for(unsigned int i = 0; i < 30; i++)
        v.push_back(i);

    cout << "Muestro el vector de 30 elementos enteros con valores consecutivos'
    for(unsigned int i = 0; i < v.size(); i++)
        cout << v[i] << ", ";

    //multiplico por 10 cada elemento
    for(unsigned int i = 0; i < v.size(); i++)
        v[i] = v[i] * 10; // asignación

    cout << "\n\n\n" << "Muestro el mismo vector pero multiplicando por 10 cada
    for(unsigned int i = 0; i < v.size(); i++)
        cout << v[i] << ", ";

    cout << endl << endl << endl;
}
```

Iteradores

Los contenedores proveen **iteradores** para que sean utilizados por los algoritmos. Estos componentes genéricos están diseñados para trabajar en conjunto y así producir un resultado óptimo.

Para instanciar un operador debemos:

```
list<int> ::iterator inicio;
```

O podemos hacer uso de sus constructores:

```
list<double> valores( 10,0 );

list<double>::iterator inicio( valores.begin() );
```

Iteradores (II)

```
int main ()
{
  list<int> first;                                    // lista vacia
  int dato;

  cin >> dato;
  for (; dato != 0 ; )
  {
    first.push_back(dato);
    cin >> dato;
  }

  cout << "The contents of first are: ";
  for (list<int>::iterator it = first.begin(); it != first.end(); it++)
    cout << *it << ' ';

  cout << '\n';

  return 0;
}
```

Librería STL - strings

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
int main()
{
    string str("informatica II");
    string str1("probando la clase string");

    // Sumamos las dos strings y ponemos blancos en medio
    string str2=str + " " + str1;
    cout << str << endl << str1 << endl << str2 << endl ;

    //con la funcion size() podemos saber el numero de caracteres de la linea
    cout << "caracteres de: " << str2 << " >>>> " << str2.size() << endl;

    //Convertimos str2 a un string de c llamada str3
    const char *str3 = str2.c_str();

    //cambiamos las 'a' en str2 por '*'
    int size =str2.size();
    for (int ix=0; ix<size;++ix)
        if (str2[ix]=='a')
            str2[ix] = '*';

    cout<<str2 << endl;
    return 0;
}
```

Librería STL - streams

Classes

Narrow characters (char)

ifstream	Input file stream class (class)
ofstream	Output file stream (class)
fstream	Input/output file stream class (class)
filebuf	File stream buffer (class)

Librería STL - Headers C

- <cstdio>
- <cstdlib>
- <cmath>
- <cstring>
- <ctime>
- <csignal>
- etc.

Cada cabecera de la biblioteca estándar de C está incluida en la biblioteca estándar de C++ con diferente nombre, generado eliminando la extensión *.h* y añadiendo una 'c' al inicio