Curso C++ - Clase 4

Juan Antonio Zubimendi
azubimendi@lifia.info.unlp.edu.ar

LIFIA

22 de noviembre de 2010

Templates

Los templates nos permiten escribir código generico que puede ser usado con varios tipos de datos. Sin templates, deberiamos reescribir, o sobrecargar muchas veces las mismas funciones o clases.

Introducción

Veamos un ejemplo:

```
int maximo(int e1, int e2) {
    return (e1 < e2) ? e2: e1;
}

int maximo(float f1, float f2) {
    return (f1 < f2) ? f2: f1;
}
```

Introducción

Con templates podriamos escribir:

```
template <typename T>
T maximo(T a, T b)
{
    return (a < b) ? a : b;
}
```

```
int x,y;
float a,b;
...
std::cout << "m1: " << maximo(x,y);
std::cout << "m2: " << maximo(a,b) << std::endl;
```

Luego podriamos utilizar la función máximo con cualquier tipo de datos que pueda comparar por menor. El compilador

Introducción

Para definir un template, debemos anteponer la función o clase con una indicación de que vamos a usar templates:

```
template <typename T1, typename T2... >
```

Asi definimos los tipos de datos que nuestro template toma como parámetros.

swap

Otro ejemplo lo tenemos con la función swap, que intercambia los valores de dos variables:

```
template <typename T>
void maximo(T &a, T &b)
{
    T c = a;
    a = b;
    b = c;
}
```

Array

Veamos un ejemplo simple de una clase que almacena como un arreglo cualquier tipo de datos:

Excepciones

Namespaces

Standard Template Library

La Standard Template Library es una librería que forma parte del Estándar de C++, por lo que podemos contar con ella para programar de manera portable y eficente. Todos los compiladores de C++ deberían implementar esta librería, si bien la eficiencia de la implementación podría variar.
Esta compuesta por templates y cubren aspectos como Entrada / Salida, contenedores, iteradores...

Subclases

Subclases

Subclases

Librería Boost

Boost::filesystem