

Introducción (Genericidad).

Plantillas de funciones o funciones genéricas.

Sintaxis de plantillas.

Ejemplos de declaraciones.

A tener en cuenta ...

Un ejemplo de plantilla de funciones: Máximo de un vector.

Sobrecarga de plantillas.

Especialización de funciones genéricas.

Plantillas de funciones (Templates)

Genericidad

En muchos casos, se necesita que una determinada operación sea válida para diferentes tipos de datos.

Uso de la sobrecarga

Una posible
solución

Pero requiere definir la función tantas
veces como tipos se quiera aplicar

Las plantillas permiten definir funciones genéricas.

Las plantillas hacen posible una mayor reutilización de código.

Plantillas de funciones o funciones genéricas

Podemos tener definida una función que calcule el máximo de dos valores **enteros**. Si queremos además una función que calcule el máximo de dos valores **reales**, tenemos que repetir el código de la función (sobrecarga).

```
int maximo(int a, int b )
{
    if (a<b)
        return b;
    return a;
}
```

```
float maximo(float a, float b )
{
    if (a<b)
        return b;
    return a;
}
```

Esta técnica requiere un trabajo adicional para cada tipo de dato nuevo que se quiera disponer.

Pérdida de tiempo

Posibilidad de error

Plantillas de funciones (Templates)

Plantillas de funciones o funciones genéricas

```
int maximo(int a, int b )
{
    if (a<b)
        return b;
    return a;
}
```

```
float maximo(float a, float b )
{
    if (a<b)
        return b;
    return a;
}
```

C++ permite definir funciones genéricas mediante el uso de plantillas (Templates).

Se puede observar que ambas funciones **maximo()** tienen un cuerpo idéntico, pero como los argumentos son diferentes, **se diferencian en los prototipos**.

Sería muy interesante tener una función genérica capaz de calcular el máximo de dos valores de cualquier tipo.

```
template <class UNTIPO>
UNTIPO maximo ( UNTIPO a, UNTIPO b )
{
    if (a<b)
        return b;
    return a;
}
```

Plantilla que sustituye a las dos funciones anteriores

Plantillas de funciones o funciones genéricas

```
template <class UNTIPO>
UNTIPO maximo ( UNTIPO a, UNTIPO b )
{
    if (a<b)
        return b;
    return a;
}
```

```
...
int a=5, int b=8, c;
float d=3.0, e=5.1, f;
c=maximo(a,b);
f=maximo(d,e);
...
```

- Una **función genérica** define un conjunto de operaciones que se aplicarán a diferentes tipos de datos.
- Una **plantilla de funciones** es como un patrón.
- Una plantilla especifica un conjunto infinito de funciones que pueden ser aplicadas a distintos tipos de datos.
- Una plantilla describe las propiedades genéricas de una función.

Plantillas de funciones (Templates)

Sintaxis

Las plantillas de funciones o funciones genéricas tienen la siguiente sintaxis:

```
template <lista de tipos genéricos>
declaración de función
```

La *lista de tipos* contiene los tipos genéricos a los que precede la palabra **class**

- ➔ La palabra reservada **template** indica que se va a declarar una plantilla.
- ➔ Las plantillas se declaran normalmente en un archivo de cabecera.
- ➔ La *lista de tipos* contiene los tipos genéricos separados por comas.
- ➔ Aunque la notación anteponga **class** al tipo, se refiere a cualquier tipo, sea clase o no (enteros, reales, etc.).

Ejemplos de declaraciones

Algunas posibles declaraciones podrían ser:

```
template <class TIPO1>
TIPO1 funcion ( TIPO1 a )
{
    // cuerpo de la función.
}
```

```
template <class TIPO1>
TIPO1 funcion ( TIPO1 a, TIPO1 b )
{
    // cuerpo de la función.
}
```

```
template <class TIPO1>
float funcion ( int cont, TIPO1 a )
{
    // cuerpo de la función.
}
```

```
template <class TIPO1, class TIPO2 >
TIPO2 funcion ( TIPO1 a, TIPO2 b )
{
    // cuerpo de la función.
}
```

Varios tipos genéricos con nombres distintos

Plantillas de funciones (Templates)

A tener en cuenta ...

- Una plantilla de función o función genérica define un conjunto ilimitado de funciones sobrecargadas.
- Las plantillas **no generan código directamente**, sino que la generación de código se realiza en el momento en que se necesita, con el tipo correspondiente.
- Las plantillas **siguen las mismas normas que la sobrecarga**, solo se mira la lista de parámetros, no el retorno.

```
template <class TIPO>
TIPO funcion ( int a )
{
    // cuerpo de la función.
}
```

Error !!

Los tipos de la lista **template**, deben aparecer como mínimo una vez en la lista de parámetros de la función.

(Error de compilación)

Un ejemplo de plantilla de funciones

Supongamos que se tiene un vector y queremos calcular el máximo valor.

Si el vector contiene elementos de tipo entero:

```
int max_vector ( int v[ ] , int dim )
{
    int aux;
    aux = v[0];
    for (int i =0; i<dim; i++)
    {
        if ( v[i]>aux )
            aux = v[i];
    }
    return aux;
}
```

Si el vector contiene elementos de tipo **long int**, tenemos que sobrecargar la función `max_vector()`

```
long int max_vector ( long int v[ ] , int dim )
{
    long int aux;
    aux = v[0];
    for (int i =0; i<dim; i++)
    {
        if ( v[i]>aux )
            aux = v[i];
    }
    return aux;
}
```

Plantillas de funciones (Templates)

Un ejemplo de plantilla de funciones

Supongamos que se tiene un vector y queremos calcular el máximo valor.

La mejor solución es declarar una **plantilla** para dicha función:

```
template <class TIPOg>
TIPOg max_vector( TIPOg v[ ] , int dim )
{
    TIPOg aux;
    aux = v[0];
    for (int i =0; i<dim; i++)
    {
        if ( v[i]>aux )
            aux = v[i];
    }
    return aux;
}
```

Se genera una función con **TIPOg = int**

```
void main()
{
    int w[5]={1,7,3,4,5};
    float k[4]={4.6,7.8,3.2,4.0};
    char c[]={ 'f' , 'e' , 't' , 'h' };
    int a;
    float b;
    char m;
    a = max_vector(w, 5);
    b = max_vector(k, 4);
    m = max_vector(c, 4);
}
```

Se genera una función con **TIPOg = char**

Sobrecarga de plantillas de funciones

Las plantillas de funciones se pueden definir para distintas combinaciones de parámetros, es decir, se pueden sobrecargar.

Por ejemplo, para la función `maximo()`, serían válidas las siguientes plantillas sobrecargadas:

```
template <class T>  
T maximo ( T a, T b );
```

Máximo de dos parámetros de tipo T

```
template <class T>  
T maximo ( T a, T b, T c );
```

Máximo de tres parámetros de tipo T

```
template <class T>  
T maximo ( T v[ ], int dim);
```

Máximo de un vector de tipo T

Plantillas de funciones (Templates)

Sobrecarga de plantillas de funciones

```
template <class T>  
T maximo ( T a, T b );
```

```
template <class T>  
T maximo ( T a, T b, T c );
```

```
template <class T>  
T maximo ( T v[ ], int dim);
```

Las llamadas serán:

```
void main()  
{  
    char c[5]={5, 6, 3, 2, 0};  
    int a; float b;  
    double m;  
    a = maximo(5,10);  
    b = maximo(4.5, 5.0, 9.0);  
    m = maximo(c, 5);  
}
```

Especialización de funciones genéricas

La existencia de una plantilla no impide que se pueda definir una función normal que prevalezca sobre la definición de la plantilla.

Por ejemplo, podemos declarar la siguiente plantilla para calcular el máximo de dos valores:

```
template <class T>
T maximo ( T a, T b )
{
    if (a<b)
        return b;
    return a;
}
```

Esta plantilla se comporta bien para tipos de datos enteros, reales, etc.

Pero se comporta mal si le pasamos cadenas (char *)

```
char * maximo ( char* s1, char* s2 )
{
    if (strcmp(s1,s2)>0)
        return s1;
    return s2;
}
```

Para resolverlo, añadimos una función normal

Plantillas de funciones (Templates)

Especialización de funciones genéricas

De ésta forma, se utiliza la plantilla para todos los tipos excepto para (char *)

```
template <class T>
T maximo ( T a, T b )
{
    if (a<b)
        return b;
    return a;
}
```

```
char * maximo ( char* s1, char* s2 )
{
    if (strcmp(s1,s2)>0)
        return s1;
    return s2;
}
```

Este ejemplo muestra que se puede definir una función genérica con templates y otras funciones normales para determinados tipos.

Es el compilador el que determina qué función utilizar.

¿Cómo?

Especialización de funciones genéricas

Las reglas que se utilizan para encontrar la función sobrecargada correcta son:

1. Busca coincidencia exacta de funciones, es decir, funciones específicas.
2. Busca una plantilla de función (template).
3. Utiliza las reglas de sobrecarga normal.

Si no encuentra ninguna coincidencia, se genera un error.