
UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

CS1103
Programación Orientada a Objetos 2
Unidad 2 - Semana 4 - Templates y Metaprogramming

Rubén Rivas

Function Templates

Ejercicio # 2

Generar una función template **split_range,** que reciba como parámetros un **contenedor** y un número **n** y permita dividir el contenedor en **n** contenedores, si el valor **n** no es múltiplo del tamaño del contenedor, los ítems restantes de la división deberán ser almacenados en el último contenedor. el resultado o valor de retorno deberá ser un contenedor que contenga los contenedores devueltos, siendo **vector** el contenedor por defecto, el template tendrá la posibilidad de personalizar en contenedor de resultado.

Ejemplo:

```
dequeue<int> v1 = {1, 2, 3, 4, 5, 6, 7};
auto resultado = split_range(v1, 3};
     // resultado[0] = {1, 2};
     // resultado[1] = {3, 4};
     // resultado[2] = {5, 6, 7};
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ejercicio # 3

Escribir una función template **sumar_rango** que permita la suma de los valores de 2 contenedores, por ejemplo.

```
vector<int> v1 = {1, 3, 4};
vector<int> v2 = {4, 5, 6};
auto v3 = sum_rango(v1, v2}; // {5, 8, 10}
```

si uno de los contenedores es de menor tamaño, el contenedor de menor tamaño se sumará repetidamente con otro contenedor hasta completar el tamaño del mayor. ejemplo

```
list<int> v1 = {1, 2, 3, 4, 5};
list<int> v2 = {10, 20};
auto v3 = sum_rango(v1, v2}; // {11, 22, 13, 24, 15}
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ejercicio # 4

Escribir una función template **`delete_items`** que permita eliminar un valor específico o una lista de valores de un contenedor secuencial:

Ejemplo # 1:

```
vector<int> v1 = {1, 3, 4, 1, 3, 2, 3, 4, 6, 5};
auto v3 = delete_items(v1, 1); // {3, 4, 3, 2, 3, 4, 6, 5}
```

Ejemplo # 2:

```
list<int> v1 = {1, 3, 4, 1, 3, 2, 3, 4, 6, 5};
auto v3 = delete_items(v1, {1, 4});    // {3, 3, 2, 3, 6, 5}
```

Ejercicio # 5

Escribir una función template **delete_duplicated** que permita eliminar todos los valores duplicados:

Ejemplo # 1:

```
vector<int> v1 = {1, 3, 4, 1, 3, 2, 3, 4, 6, 5};
auto v3 =delete_duplicated(v1);   // {1, 3, 4, 2, 6, 5}
```

Ejemplo # 2:

```
list<int> v1 = {1, 1, 1, 1, 3, 2, 2, 2, 2, 5};
auto v3 = delete_duplicated(v1);  // {1, 3, 2, 5}
```

Ejercicio # 8

Extender la función **unpack** del ejercicio #7 para la estructura std:tuple (investigar el funcionamiento de tuple):

```
std::tuple<int, string, string, double> t1 = {1321, "Jose", "Perez", 1.68};
```

El acceso no tiene valores significativos.

```
std::cout <<std::get<0>(t1) << " "
<< std::get<1>(t1) << " " << std::get<2>(t1) << " "
<< std::get<3>(t1);
```

De otro lado lenguajes como python brindan un mecanismo conocido como **unpack** que permite asignar valores de un contenedor a variable con nombres significativos.

Ejemplo de unpack usando tuple:

```
int key; string first_name; string last_name; double height;
unpack(key, first_name, last_name, height) = t1;
std::cout << key <<  " "<< first_name << last_name << height;
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ejercicio # 11

En python existe una función denominada zip que permite recibir un número variado de contenedores del mismo tipo y agrupar los valores de una misma fila generando tuplas por cada fila con los valores de cada contenedor.
Generar un function template similar, pero en vez de tuplas genere un vector con los valores correspondientes de cada fila

Ejemplo:

```
list<int> v1 = { 11, 12, 13 };
    list<int> v2 = { 21, 22, 23 };
    list<int> v3 = { 31, 32, 33 };
    auto result = zip(v1, v2, v3);
    for (const auto& row : result) {
            for (const auto& value : row)
                    cout << value << " ";
            cout << endl;
    }
```

El tipo de **result** sería un **list<vector<int>>** y lo impreso sería:

```
11 21 31
12 22 32
13 23 33
```

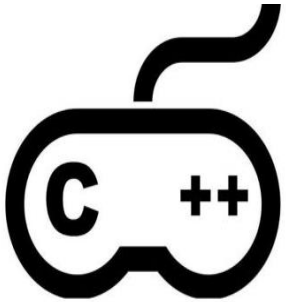
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Explorando lo aprendido

- ¿Qué es el variadic template?
- ¿Por qué es útil?
- ¿Que es metaprogramación?
- ¿Cuales son los mecanismos de la metaprogramación?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Bibliografía:



- **C++ Templates, The Complete Guide**; 2018; David Vandevoorde, Nicolai M. Josuttis, Douglas Gregor
- **C++ Primer, Fifth Edition; 2013**; Stanley B. Lippman, Josée Lajoie, Barbara E. Moo