CS2013 - Programación III

Simulacro B de Práctica Calificada #2 (PC2)

2023 - 2

Profesor: Rubén Rivas

Librería Estándar – 7 puntos

Diseñar y desarrollar el template de función **join_containers** que permita unir varios contenedores (en una cantidad variada de contenedores) y que genere un contener que por default sea vector.
Los contenedores podrian ser: **list**, **vector**, **std::deque**, **std::forward_list**.

Si se tiene los siguientes containers :
list lst = {1, 2, 3, 4, 5}
vector vec = {6, 7, 8, 9, 10, 11, 12}
deque deq = {13, 14}

Y si se llama:
*result = **join_containers**(lst, vec, deq)*

result contendria:
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Casos de uso #1

```
vector v1 = {1, 2, 3};
list l1 = {4, 5, 6};
vector v2 = {7, 8, 9};
deque d1 = {10, 11, 12};
auto res = join_containers(v1, l1, v2, d1);
copy(begin(res), end(res), ostream_iterator<int>(cout, " "));
```

Casos de uso #2

```
vector v = {1, 2, 3};
list<int> l;
forward_list f = {7, 8, 9};
auto res = join_containers(v, l, f);
copy(begin(res), end(res), ostream_iterator<int>(cout, " "));
```

Complejidad Algorítmica – 6 puntos

Dada la siguiente función determinar la complejidad algoritmica, incluir en su respuesta el procedimiento.

```
bool recursive_binary_search(
        std::vector<int>::iterator start,
        std::vector<int>::iterator end, int value) {
    if (start >= end) {
        return false;
    }

    std::vector<int>::iterator mid = start + (end - start) / 2;

    // Búsqueda binaria anidada
    if (std::binary_search(start, mid, value)) {
        return true;
    }

    // Recursión en la mitad superior
    return recursive_binary_search(mid + 1, end, value);
}
```

Programación Concurrente – 7 puntos

Escribir la función template **parallel_variance** que retorne la variancia de los elementos del contenedor.

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

Donde:

- $\boldsymbol{\sigma^2}$: variancia
- $\boldsymbol{\mu}$: promedio
- $\boldsymbol{N}$: cantidad de elementos
- $\boldsymbol{X_i}$: elemento i

El contenedor puede almacenar valores enteros o doubles.

El contenedor podrian ser cualquier tipo secuencial y su recorrido será a traves de sus interadores.

Si se tiene el container con los siguientes valores:
cnt1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Y si se llama:
*result = **parallel_variance**(begin(cnt1), end(cnt1))*

result contendria:
11.916

Caso de uso #1

```
int n = 0; cin >> n;
vector<double> vec(n);
for(auto& item: vec) cin >> item;
auto res = parallel_variance(begin(vec), end(vec));
cout << res;
```

Caso de uso #2

```
int n = 0; cin >> n;
vector<int> vec(n);
for(auto& item: vec) cin >> item;
auto res = parallel_variance(begin(vec), end(vec));
cout << res;
```

Caso de uso #3

```
int n = 0; cin >> n;
deque<int> deq(n);
for(auto& item: deq) cin >> item;
auto res = parallel_variance(begin(deq), end(deq));
cout << res;
```