CS1103

Programación Orientada a Objetos II

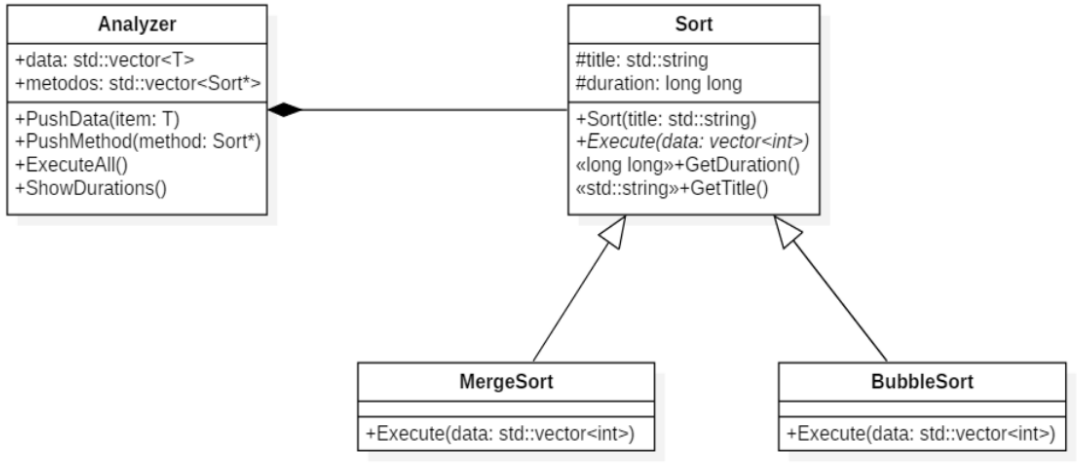
Set de Problema Calificado 2

2018 - 2

Profesor: Rubén Rivas

Alumno:

Desarrollar el siguiente diagrama de clases:



1. (8 puntos) Desarrollar la relación de herencia entre las clases derivadas MergeSort, BubbleSort y la base Sort, implementando los métodos y atributos que se muestran en el diagrama, implementando el polimorfismo.

2. (4 puntos) Desarrollar los algoritmos de ordenamiento BubbleSort y MergeSort sugiriéndose revisar los algoritmos al final.

3. (4 puntos) Desarrollar la Clase de nombre Analyzer (deberá ser desarrollado como template) que debe incluir los siguientes métodos:
   a. PushData, para agregar valores en el vector data
   b. PushMethod, para agregar puntero a objetos del tipo Sort
   c. ExecuteAll, ejecutara los métodos de ordenamiento que hayan sido agregados al vector metodos.
   d. ShowDuration, cada método debe almacenar en el atributo duration de la clase Base Sort el tiempo que demoro el ordenamiento y deberá mostrar la siguiente información:
      i. GetTitle() << ‘ ’ << GetDuration() << ‘\n’

4. (4 puntos) Desarrollar el programa principal que permita agregar aleatoriamente 1000 datos y los 2 métodos y que Muestre cuanto tiempo dura cada método.

```
int main()
{
    Analyzer<int> a;
    int n = 1000;
    std::random_device r;
    for (int i = 0; i < n; ++i)
        a.PushData(r() % n);

    a.PushMethod(new BubbleSort<int>("Bubble Sort"));
    a.PushMethod(new MergeSort<int>("Merge Sort"));

    a.ExecuteAll();
    a.ShowDurations();
    return 0;
}
```

```
Bubble Sort 353376
Merge Sort 5390
```

Ordenamiento Merge:

```cpp
#include <vector>
#include <iostream>
#include <algorithm>

template<class Iter>
void merge_sort(Iter first, Iter last)
{
    if (last - first > 1) {
        Iter middle = first + (last - first) / 2;
        merge_sort(first, middle);
        merge_sort(middle, last);
        std::inplace_merge(first, middle, last);
    }
}

int main()
{
    std::vector<int> v{8, 2, -2, 0, 11, 11, 1, 7, 3};
    merge_sort(v.begin(), v.end());
    for(auto n : v) {
        std::cout << n << ' ';
    }
    std::cout << '\n';
}
```

Ordenamiento Burbuja:

```cpp
template<typename T>
long long bubble_sort(std::vector<T> data) {

    // Capturando tiempo inicial
    using tp = typename std::chrono::steady_clock::time_point;
    tp start = std::chrono::steady_clock::now();

    // Ordenamiento de burbuja
    for (size_t i = 0; i < data.size() - 1; ++i)
        for (size_t j = i + 1; j < data.size(); ++j)
            if (data[i] > data[j])
                std::swap(data[i], data[j]);

    // Capturando tiempo final y calculando duración
    tp end = std::chrono::steady_clock::now();
    return std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
}
```

Barranco, 24 de octubre del 2018