Emilio Chica Jiménez y Julián Torices Hernández 06-11-2014

PRACTICA 2.1

Tiempo empleado 10h
Recursos utilizados: librería stream, fstream, vector

Complicaciones asociadas al proyecto:

1. Comprensión de la práctica a realizar por horrible documentación de la misma.
2. Dedicación demasiado extensa para unas practicas de 1h a la semana.
3. Depuración de lo archivos entregados por los profesores, ya que había errores y fallos que no se encuentran facilmente. Preferiblemente hubiera empezado yo la practica desde 0.
4. Realización de la función load de lectura del archivo .csv. Pensamos que esta función no era relevante para este proyecto y se lleva mas tiempo del que realmente se debería para que funcione correctamente.

Diferencias:

1. En esta prática utiliza iteradores y nosotros al no tener conocimiento de ellos era imposible de que supieramos que teniamos que usarlos, sin embargo nuestra solución funciona perfectamente ya que recorrer un vector por medio de indices es totalmente válido y se puede usar nuestra solución como una implemtacion correcta para el problema del diccionario, ya que nos funcionan las versiones 1 y 2 correctamente.
2. EL metodo de chec_rep no lo implementa en la versión 1 del diccionario que se no ha entregado como solucion, nosotros si lo hemos implementado.
3. Al realizar el metodo de redifinicion del operator[] no tuvimos en cuenta el devolver dic.back().second, y en su lugar nos creamos una variable static para que el valor de esta referencia no se destruyera al acabar el metodo.
4. En el operator= no comprobamos si eran iguales los diccionarios.
5. En el metodo insert nosotros no hacemos uso de la funcion find que habiamos creado, pero en su lugar hacemos una busqueda igualmente correcta.
6. En el opertaror[] no salimos del programa, ya que en el PDF entregado no lo especificaba.
7. El constructor vacio de la practica entregada no hace lo que se especificaba en el PDF de la practica, ya que esta vacio, el nuestro al menos define una entrada vacía.
8. En diccionarioV2 nuestro find no hace busqueda binaria en la solucion si, pero sólo optimiza la velocidad no indica que este mal.
9. La insercción en el operator[] de diccionarioV2 utiliza la busqueda e inserción binaria en el vector en la versión entregada, la nuestra despues de insertar ordena el vector con sort.
10. Lo mismo ocurre en insert.
11. Nosotros hemos sobreescrito el metodo ostream<< en el principal.cpp para usar la llamada a D[“Granada”] por ejemplo ya que no se especificaba que se tuviese que hacer dos sobreescrituras en meteorito del metodo ostream<<.
12. El metodo de load en la versión entregada solo carga el fichero y llama a un metodo string2meteorito para que desglose las entradas del meteorito, el nuestro lo hace directamente en el metodo.
13. Hemos tenido que escribir un metodo de comparación entre dos entradas para poder usar el metodo sort.

En definitiva creemos que nuestra solución es perfectamente válida y solo la mejoraría con los

iteradores para practicarlos.
Y en lugar de usar el método sort en la segunda version de de diccionario, realizariamos una busqueda binaria para practicarla porque la eficiencia creeemos que sería la misma para ambos casos.