Ejercicios de C++

1. Strings

1. Declare dos variables de tipo *string*. La primera utilizando el *constructor* de la clase *string* y la segunda utilizando el operador de asignación `=`.
2. Compare ambos *string* e imprima el tamaño de cada uno por consola.
3. Elimine los primeros 3 caracteres del primer *string* y los últimos 3 del segundo.
4. Inserte 3 caracteres en la mitad del primer *string*. Si no tiene un número par de caracteres, aproxime la posición de inserción al entero superior.
5. Inserte 3 caracteres al final del segundo *string*.
6. Obtenga la representación C de ambos *string* y compare nuevamente las cadenas de texto utilizando `strcmp()` e imprima el largo de las cadenas con `strlen()`.
7. Cuente la aparición de vocales en el primer *string* e imprima por pantalla.
8. Cuente la aparición de consonantes en el primer *string* e imprima por pantalla.
9. Elimine el contenido de ambos *strings* y verifique si están vacíos usando la función miembro `empty()`.
10. Dada la variable `string cadena = "Este es un ejemplo";`, utilice iteradores normales y reversos para recorrer sus elementos e imprimirlos por pantalla.

2. STL

1. Llene desde teclado un *vector* y lo muestre por teclado **haciendo uso de iteradores**. Utilice la directiva `<vector>`.
2. Llene desde teclado un *vector* y muestre primero las posiciones **pares** y luego las **impares**. Utilice la directiva `<vector>` y el operador mod `%`.
3. Llene desde teclado un *vector* y lo muestre en **orden inverso**. Utilice la directiva `<vector>`.
4. Llene desde teclado un *vector* y lo ordene de **mayor a menor** para luego mostrarlo por pantalla. Utilice la directiva `<vector>`.
5. Llene desde teclado dos *vectores* de **palabras** y busque cada palabra del primero en el segundo. Para cada palabra del primer elemento debe decir si fue encontrada o no en el segundo por pantalla. Utilice la directiva `<vector>`.
6. Llene un *mapa* desde teclado que relacione un **índice random** y un **valor random** y muestre ambos por pantalla. Utilice la directiva `<map>`.
7. Llene un *mapa* desde teclado que relacione un **índice random** y un **valor random** y muestre ambos por pantalla en **orden inverso**. Utilice la directiva `<map>`.
8. Escriba un programa en C++ que tome el apellido (índice) de una persona y su nombre (valor) y los muestre en **orden alfabético de apellido**. Utilice la directiva `<map>`.
9. Escriba un programa en C++ que tome el apellido (índice) de una persona y su nombre (valor) y los muestre en **orden alfabético de nombre**. Utilice la directiva `<map>`.

10. Escriba un programa en C++ que tome el apellido (índice) de una persona y su nombre (valor) y los muestre en **orden alfabético de apellido de 3 en 3 cada vez**. Utilice la directiva `<map>`.

3. Algorithm

1. Compare e imprima el **mínimo** entre dos números utilizando la función `min()`.
2. Compare e imprima el **máximo** entre dos números **decimales** utilizando la función `max()`.
3. Dado el array `int enteros[] = {5, 1, 8, 0, 2, 7, 9, 3}`, encuentre el número más pequeño y más grande dentro del rango, utilizando las funciones `min_element()` y `max_element()`.
4. Copie los valores del array `int enteros[]` al vector `vector<int> vec(8)`. Utilice la función `copy()`.
5. Dado el array `int numeros[] = {10, 20, 10, 20, 20, 30, 30, 10}`, cuente e imprima las veces que se repiten los números: *10, 20 y 30*. Utilice la función `count()`.
6. Realice lo solicitado en el *ejercicio 5*, pero esta vez utilizando `vector<int>`.
7. Dado la variable `vector<int> temp(10)`, llene con **1s** desde la posición 0 a la 4 y con **2s** desde la posición 5 hasta 2 posiciones antes del fin del rango. Imprima los valores del vector por pantalla. Utilice la función `fill()`.
8. Utilice la función `equal()` para verificar si dos rangos de valores son idénticos.
9. Utilice la función `sort()` para ordenar en forma **ascendente** los valores de un vector `vector<int> v1` y un array `int a1[]`. Inicialice ambas variables con valores a su gusto.
10. Realice lo mismo del *ejercicio 9*, pero esta vez ordene los elementos en forma **descendente**. Para ello deberá crear una función auxiliar que compare dos números y retorne `true` si es mayor o no. El nombre de la función deberá ser el tercer argumento de la función `sort()`.
11. Invierta los valores de un vector utilizando la función `reverse()`.
12. Utilizando la función `find()`, verifique si existe el valor *10* entro del array `int a2[] = {20, 30, 10, 24, 60, 15}`. Realice la misma operación utilizando un `vector<int>`.
13. Tome las variables del *ejercicio 12*, **ordénelas** usando `sort()` y luego busque el valor que desee utilizando la función `binary_search()`.