PROYECTO C++

"FORMULATRÓN"

ISMAEL ÁNGEL SORIA RIVERO

PROGRAMACIÓN

1º DAW

CURSO 2023/2024

Índice

Introducción

La formulación de óxidos es parte fundamental de la química inorgánica, y para muchas personas es un auténtico quebradero de cabeza. Los oxidos son compuestos binarios formados por la unión de oxigeno con otro elemento, pudiendo originarse multitud de combinaciones posibles, existiendo, a su vez varias formas de nombrarlos.

Por ello se debe desarrollar un programa que sirva de herramienta, para estudiantes o cualquier persona en general, y que nos permita de manera agil y sencilla, formular todos los óxidos posibles.

Índice de tablas y figuras

Especificaciones de Requisitos

- Requisitos Funcionales

A continuación, se detallan los requisitos funcionales del proyecto

Código	RF-001	**Nombre**	Menú Inicial
Descripción	Crear un menú inicial con tres opciones: Formula a Compuesto, Compuesto a Fórmula y Salir del Programa		
Desarrollo	Desarrollar una pequeña interfaz de usuario que permita el usuario de manera clara elegir entre las opciones.		
Restricciones	Ninguna.		
Grado de necesidad		**Alto**	

Código	RF-002	**Nombre**	Registro de Fórmula
Descripción	En la opción "Formula a Compuesto", debe permitir al usuario registrar una fórmula química.		
Desarrollo	Desarrolla una forma que registre la entrada del usuario y valide que cumple con las reglas básicas de los óxidos.		
Restricciones	El segundo elemento sea oxígeno. Las valencias de los índices deben ser factibles.		
Grado de necesidad		**Alto**	

Código	RF-003	**Nombre**	Identificación del elemento
Descripción	Identificar el elemento a partir de su símbolo en la opción "Formula a Compuesto".		
Desarrollo	Desarrollar la forma concreta en la que se asocie el símbolo introducido con el elemento correspondiente.		
Restricciones	Solo reconocerá los elementos que puedan formar óxidos.		
Grado de necesidad		**Alto**	

Código	RF-004	**Nombre**	Comprobación de Índices
Descripción	Comprobar que los índices introducidos por teclado son correctos y se corresponden al elemento		
Desarrollo	Se debe verificar que los índices introducidos son correctos, puedan pertenecer a ese elemento y puedan conformar un óxido.		
Restricciones	Los índices sólo tienen unos valores concretos.		
Grado de necesidad		**Alto**	

Código	RF-005	**Nombre**	Menú de Nomenclaturas
Descripción	Abrir un menú que permita formular en las nomenclaturas Stock, Sistemática y Tradicional. También, debe permitirnos guardar el contenido en un fichero, introducir otra fórmula o volver al menú principal.		
Desarrollo	Crear un menú que ofrezca las opciones al usuario después de registrar una fórmula.		
Restricciones	Que sea meridianamente, claro y sencillo de entender para el usuario. Asegurar que se guardan correctamente en el fichero.		
Grado de necesidad		**Alto**	

Código	RF-006	**Nombre**	Registro de Compuesto
Descripción	En la opción "Compuesto a Fórmula", permitir al usuario registrar un compuesto químico.		
Desarrollo	Desarrolla una forma que registre la entrada del usuario y valide que cumple con las reglas básicas de los óxidos.		
Restricciones	Contiene palabra oxido/anhidrido. Contenga un elemento registrado		
Grado de necesidad		**Alto**	

Código	RF-007	**Nombre**	Identificación de Nomenclatura
Descripción	Identificar en qué nomenclatura está escrita la fórmula del compuesto introducido.		
Desarrollo	con el compuesto que nos registre el usuario, desarrollar el modo que se identifque en que nomenclatura esta escrito el compuesto.		
Restricciones	Las reglas propias de las nomenclaturas		
Grado de necesidad		**Alto**	

Código	RF-008	**Nombre**	Cálculo de Valencias
Descripción	Calcular que las valencias sean correctas y se ajusten al elemento introducido en la opción "Compuesto a Fórmula".		
Desarrollo	Desarrollar un sistema que consiga identificar el valor de los índices, y con ello comprobar que las valencias son correctas.		
Restricciones	Cada nomenclatura registra los índices de una manera concreta.		
Grado de necesidad		**Alto**	

Código	RF-009	**Nombre**	Mostrar y Guardar Fórmula
Descripción	Debe mostrar la fórmula del compuesto y permitir al usuario guardarla. Finalmente nos permitirá introducir otra fórmula o volver al menú principal		
Desarrollo	Debe mostrarse una interfaz donde el usuario vea de forma clara la respuesta, y pueda elegir las opciones.		
Restricciones	Asegurar que la fórmula se muestre correctamente y se pueda guardar de manera efectiva.		
Grado de necesidad		**Alto**	

Complicaciones y soluciones

1. No definir la estructura general del programa.
    a. Comencé a programar teniendo claro lo que quería que mi programa hiciese y como, aunque era un esbozo bastante general, pero no tuve en cuenta la manera de organizar la estructura general que tendría todo el conjunto. Cuando llevaba, aproximadamente el 40% del trabajo, tuve que parar y diseñar la estructura, aunque al menos pude reutilizar bastante del código ya escrito.
2. Uso de ficheros y funciones.
    a. No había utilizado ficheros y funciones hasta que me puse a desarrrolar el programa, así que me ha pasado de todo un poco, porque apenas tenia idea de como funcionaban. Asi que para solucionarlo me puse con los apuntes y a probar.
3. Ficheros
    a. Para poder borrar un fichero anteriormente y no se queden guardados anteriores.
        - Uso de la funcion: **remove(nombre_archivo)** que necesita de la libreria **<cstdio>**. Se utiliza para cuando se abre el programa el fichero formulacion.txt y nomenclaturas.txt, se queden vacios y así se registren sólo las fórmulas o compuestos que añadamos durante ese momento.
4. Comprobar que un string es un int.
    a. Al introducir la fórmula, por comodidad para posteriores actuaciones, registraba los índices como string, pero no sabía como comprobar que sólo me pudieran dar números y no se pudieran introducir letras.
        - Uso de la funcion: **isdigit(nombre_string)** que necesita de la libreria **<cstdio>**. Devuelve un booleano.
5. Transformar el elemento introducido en mayúsculas o minúscula.
    a. Para poder comparar si el elemento era correcto, introduje en el fichero todos los símbolos de los elementos, debía comparar los que el usuario introducía por teclado con los del fichero. Debían estar todos en mayúsculas o minúsculas, para evitar errores,
        - Uso de la funcion: **toupper(nombre_string)** que necesita de la libreria **<cctype>**. Transforma el string en mayúsculas. Podía haber usado la

función **tolower(nombre_string)**, que si la dimos en clase y usa la misma libreria, pero convierte en minúsculas.

6. Transformar string en int
    a. Cuando he usado los índices, varias veces he tenido que ir cambiandolos de tipo, por exigencias de mi programa.
        - Uso de la funcion: **stoi(nombre_string)** que necesita de la libreria **<string>**. Transforma un string en int.
7. Transformar int en string
    a. Lo mismo que en el apartado anterior, cambio de tipos por exigencias del programa.
        - Uso de la función: **to_string**(nombre int) que necesita de la libreria <string>. Transforma un string en int.
8. Identificar la primera palabra y la última de los compuestos introducidos por el teclado, para determinar las nomenclaturas.
    a. Una vez que el usuario metía el compuesto por teclado, tenía que identificar si la primera palabra era compatible con un óxido, y despúes la ultima palabara del compuesto siempre hace referencia al elemento utilizado. He utilizado dos funciones.
        - Uso de la función: **cadena.find(subcadena)** que necesita la libreria **<string>**. La primera aparición de una subcadena en una cadena más grande. Si se encuentra devuelve la posición y si no se encuentra devuelve **string::npos**.
        - Uso de la función: cadena_**find_last_of(subcadena)** que necesita la libreria **<string>**. La última aparicion de un subcadena en una cadena más grande. Si se encuentra devuelve la posición y sino se encuentra devuelve string::npos.
        - Uso de la función**: cadena.substr(posición_inicial,longitud)** que necesita de la libreria **<string>** extrae una subcadena de una cadena.

9. Como identificar partes concretas de una cadena y utilizarlas.
    a. Cuando se meten compuestos en nomenclatura sistemática o stock, la subcadena del elemento es necesaria dividirla en dos partes más pequeñas, para poder extraer las valencias.
    b. Ejemplo:
        - Monoxido de dihierro= necesitamos separar mono(indica valencia elemento)-oxido(indica que es un compuesto válido) y di(valencia oxígeno)-hierro(elemento del óxido)..
        - Oxido de hierro(III)=hierro(elemento)-(III)(valencia del hierro).
        - Uso de la función: cadena**.erase(posicion_inicial, longi_subcad_borrar)** que necesita de la libreria **<string>**. Esta función toma la posición desde donde se comienza a eliminar y longitud de la subcadena a eliminar.(necesario usar otra función .size() o .length()).

Bibliografía

He utilizado ChatGPT, tanto para buscar funciones o comprobar algunas partes del código que no me funcionaban correctamente. He intentado comprender las sugerencias de ChatGPT y no utilizar funciones que no comprendía su funcionamiento.

Para la parte de documentación sobre óxidos he utilizado la siguiente página web https://www.formulacionquimica.com/inorganica/.