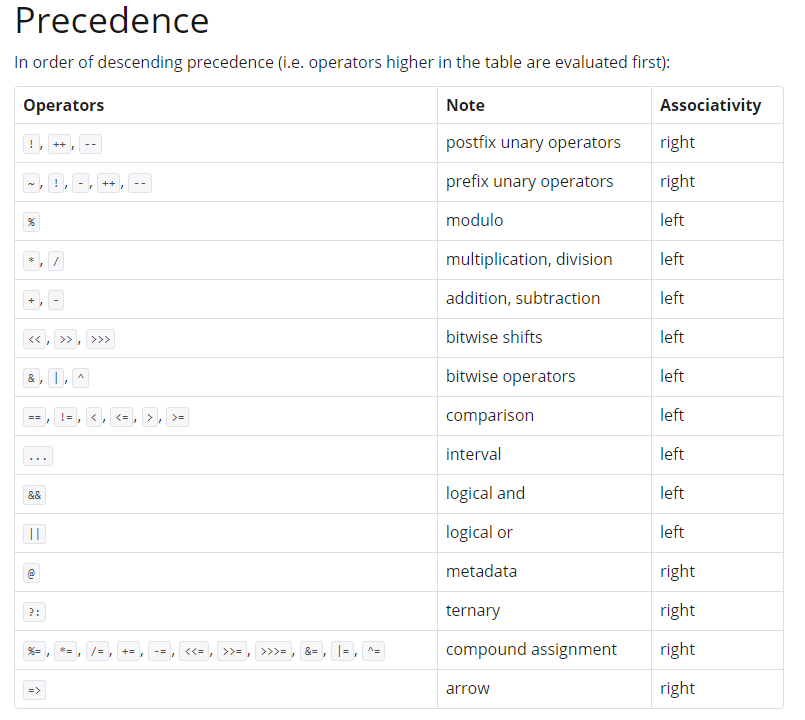
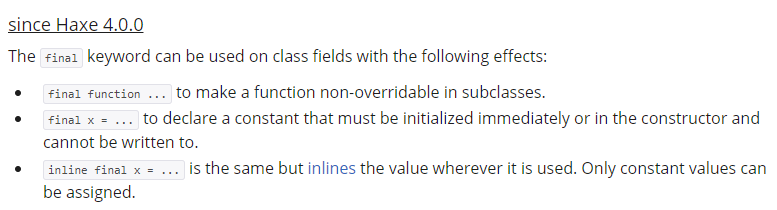


El lenguaje escogido por mí es Haxe. Es un lenguaje de alto nivel, fuertemente tipado que puede ser compilado a multiples programas. Como JavaScript, Python, Php, entre otros. Es orientado a objetos también, permitiendo la creación de clases, interfaces, clases abstractas, herencias (Solo se puede heredar de una clase). Posee tipos de datos, float, int y bool. Dentro de las clases, solo se permite un constructor, los cual puede ser inconveniente en muchas ocasiones, también permite en cierto tipo de estructura llamadas abstract hacerle overload a los operadores. La tabla de precedencia es la siguiente:



El lenguaje posee asociación dinámica de métodos, y ofrece algo parecido a una asociación dinámica con la posibilidad de colocarle a las variables y métodos la etiqueta final que permite que una vez asignado un valor a dicha variable, o definido el método no se pueda cambiar más.



Por más de ser un lenguaje de uso no tan a gran escala ofrece una muy buena cantidad de documentación. Esta contiene ejemplos detallados, así como explicaciones de casi todas las funciones y o propiedades del lenguaje. <https://haxe.org/manual/introduction.html>

Desde hace poco Haxe implementó como sistema de concurrencia los Threads y Process, aunque al ser tan reciente la documentación provista no es tan a gran escala. Aun así me fue posible desarrollar los algoritmos. Para declarar los Threads es necesario llamarlos como aparece en la imagen:



Asi creamos un Thread (**Sys.thread.Thread.create**), estos comparten todas las variables y memoria del programa principal, además de ejecutarse como deamons haciendo que por defecto el programa principal no espere por ellos. Para eso es necesario utilizar un Lock. Creamos una variable con las propiedades de lock y necesitamos llamar Lock.wait() por cada thread que queramos esperar. Y dentro de cada thread es necesario llamar a lock.release() para poder decirle al programa principal que puede proseguir.

Como se puede ver a la imagen, el thread ejecuta todo lo que esta en su espacio (Entre los parentesis), es decir que podemos definir con mucha exactitud que queremos que haga nuestro thread. Lo cual de cierta manera es muy convenite, desconozco si se le puede llamar con una función nada más, pero si puedes llamar a funciones definidas en cualquier lugar como si del main se refiriera. Para sincronización entre threads tenemos los mutex los cuales funcionan como siempre. Un hilo puede tener un mutex y los demás deben esperar a que este se libere. El programa principal no espera a que el mutex sea liberado, por lo que se debe jugar con los locks y con los mutex. Con respecto a los Process no conseguí mucha información al respecto, se que se pueden definir más no entendí a fondo como se hace.

En mi experiencia codeando en este lenguaje, pienso que es un lenguaje con mucho potencial que de verdad es cómodo de programar y es bastante intuitivo. Posee Iteradores y en conjunto a estos unas maneras muy interesantes de generar programas estilo Haskell, si bien no funcionaran exactamente igual tiene un estilo de diseño bastante funcional (al menos de manera visual).



Un pequeño inconveniente que me conseguí a la hora de realizar el reto especial, es que el sistema solo provee int32. Si bien no esta nada mal para el usuario promedio, puede que en algunos casos no sea suficiente. Existe un tipo de librería que ofrece unos int64 pero tiene poca documentación y no funciona de manera tan intuitiva. Le invito a revisar con calma el lenguaje, de verdad quedé encantado. Posee integración con al menos Visual Studio así que el IDE también aporta su ayuda para poder desarrollar código cómodo y legible. Sin más que decir esta es mi opinión sobre el lenguaje