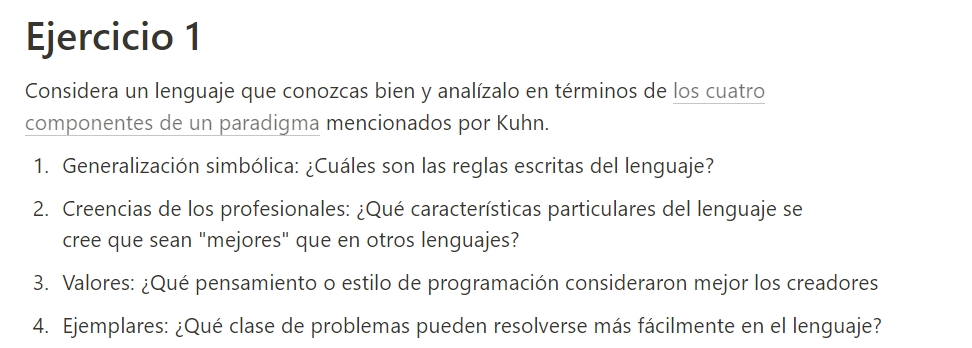
Alumno: Rodriguez Daglio Andrés



1. Reglas del Lenguaje C:

* **Estructura básica de un programa C:** Todo programa en C sigue una estructura básica compuesta por funciones, siendo la más importante la función main, que es el punto de entrada de cualquier programa en C
* **Declaración de variables:** Las variables deben declararse antes de ser utilizadas y su tipo debe especificarse. Las variables pueden declararse en cualquier parte del bloque de código, pero siempre antes de ser usadas.
* **Tipos de datos básicos:**

int (Enteros)

float (números decimales)

double (números decimales de mayor tamaño)

char (caracteres individuales).

* **Operadores:**

Aritméticos (+, -, \*, /, %)

Relacionales (==,!=, >, <, >=, <=)

Lógicos (&&, ||, !)

Asignación (=, +=, -=, \*=, /=, %=)

* **Estructuras de control:**

Condicionales (if, else if, else, switch)

Bucles (for, while, do while).

* **Funciones:** Las funciones en C se declaran especificando el tipo de retorno, el nombre de la función y sus parámetros (si los hay). Toda función debe devolver un valor si no es de tipo void. Si la función no devuelve un valor, se usa void.
* **Punteros:** Los punteros son una característica clave de C. Un puntero es una variable que almacena la dirección de otra variable.
* **Estructuras de datos:** C ofrece soporte para crear tipos de datos compuestos mediante estructuras (struct).
* **Entrada y salida** (C utiliza funciones estándar para la entrada y salida) : Principalmente printf () para salida y scanf () para entrada.
* **Comentarios** (C admite dos tipos de comentarios):

De una sola línea: usando //.

De múltiples líneas: usando /\* \*/.

* **Alcance de variables** (C tiene dos principales tipos de alcance de variables): **Local**: dentro de funciones o bloques de código, donde solo es visible en ese contexto.

**Global**: declarada fuera de las funciones, accesible desde cualquier parte del programa.

* **Errores de compilación:**

El compilador de C es estricto con la sintaxis y arroja errores si las reglas no se siguen correctamente. Algunos errores comunes incluyen:

Falta de punto y coma (;) al final de cada una de las instrucciones.

Declaración de variables no utilizadas.

Tipos de datos incorrectos en las operaciones.

1. Características particulares del lenguaje C:

El lenguaje C destaca por:

* **Eficiencia**
* **Control sobre la memoria**
* **Portabilidad**
* **Simplicidad**
* **Compatibilidad**
* **Longevidad**
* **Versatilidad**.

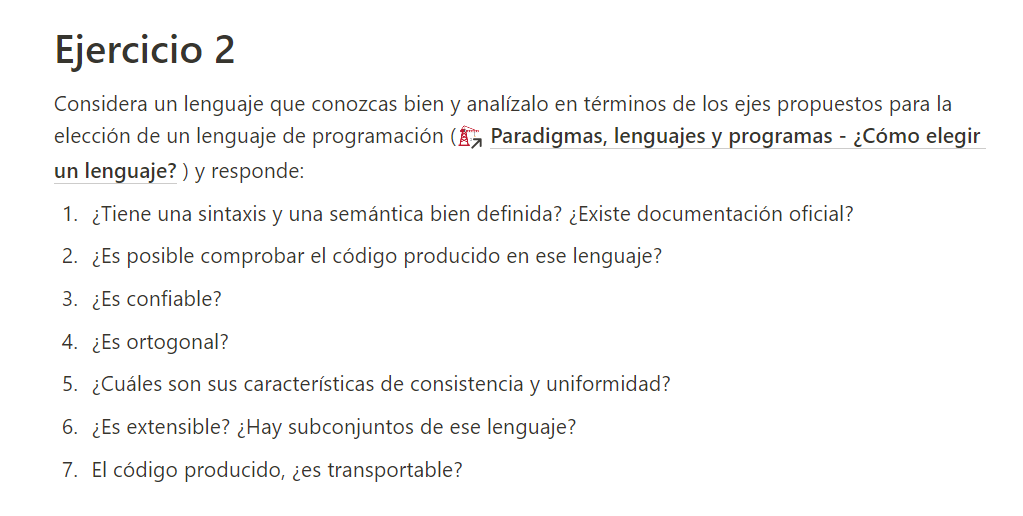
1. Valores de los creadores del lenguaje C (Dennis Ritchie):

* **Eficiencia**: El código debe ser rápido y optimizado.
* **Simplicidad**: Mantener el lenguaje y el código claro y directo.
* **Control**: Dar al programador el control total sobre los recursos del sistema, especialmente la memoria.
* **Portabilidad**: Poder escribir código que se ejecute en diversas plataformas.
* **Transparencia**: El programador debe entender cómo funciona cada parte del código sin comportamientos ocultos.
* **Flexibilidad**: El lenguaje no debe imponer un estilo o paradigma de programación.

Estos valores promovieron un estilo de programación enfocado en la eficiencia, la simplicidad y el control directo sobre el hardware, características que han hecho de C un lenguaje fundamental en el desarrollo de sistemas y software de alto rendimiento.

1. Problemas que pueden resolverse más fácilmente con C:

* **Sistemas operativos**
* **Sistemas embebidos**
* **Aplicaciones de alto rendimiento**
* **Controladores de dispositivos**
* **Compiladores**
* **Aplicaciones de red**
* **Algoritmos complejos como**: (gráficos, compresión y cifrado).



Ejercicio N°2 (Lenguaje C)

1. El lenguaje C tiene una **sintaxis y semántica bien definidas** gracias a su estandarización oficial a través de ANSI e ISO. La documentación oficial está contenida en estos estándares y asegura que los compiladores y herramientas sigan las mismas reglas, haciendo que el código en C sea predecible, consistente y portable. Además, existen varios libros y recursos bien documentados que facilitan el aprendizaje y el dominio del lenguaje.
2. Sí, es posible comprobar el código en C a través de varias técnicas y herramientas. Algunas de las más importantes son las advertencias del compilador (warnings, errors), el uso de depuradores, las pruebas unitarias, análisis de memoria con Valgrind, análisis estático con herramientas como Cppcheck y Clang, y el uso de sanitizers para detectar errores en tiempo de ejecución.
3. El código en C **puede ser confiable**, pero depende significativamente del **enfoque del programador** y el **uso de herramientas adecuadas** para detectar y corregir errores. La flexibilidad y el poder que ofrece C para controlar directamente el hardware y la memoria también implican mayores riesgos. Sin embargo, con una metodología de desarrollo rigurosa, uso de herramientas de análisis y comprobación, el código en C puede ser tan confiable como el de cualquier otro lenguaje, aunque el proceso para garantizar esa confiabilidad requiere más esfuerzo y cuidado.
4. El lenguaje C **no es completamente ortogonal**, ya que existen varias inconsistencias en cómo se combinan ciertos tipos de datos, operadores y estructuras. Aunque algunas áreas del lenguaje siguen principios ortogonales (como las estructuras de control y ciertos operadores), otras, como la combinación de punteros y arreglos, la conversión de tipos y el manejo de punteros void, no lo son.

C tiene un diseño que prioriza la **eficiencia y flexibilidad** sobre la ortogonalidad, lo que le permite ser un lenguaje de alto rendimiento, aunque con algunas excepciones que los programadores deben manejar con cuidado para evitar comportamientos indefinidos o inesperados.

1. El lenguaje C presenta una buena **consistencia y uniformidad** en varias áreas clave, como operadores, estructuras de control, funciones, punteros y manejo de memoria. Estas características hacen que el lenguaje sea **predecible** y **comprensible** para los programadores, a pesar de sus limitaciones en ortogonalidad y algunos casos donde las reglas no se aplican de manera completamente uniforme. Las áreas más consistentes de C, como su manejo de operadores y control de flujo, contribuyen a su popularidad y eficacia en programación de sistemas y aplicaciones de alto rendimiento.
2. El lenguaje C es **extensible** en el sentido de que se pueden crear bibliotecas y funciones personalizadas, y se han desarrollado varios **subconjuntos** y estándares a lo largo del tiempo para satisfacer diversas necesidades de programación. La extensibilidad de C permite a los desarrolladores adaptarlo a diferentes contextos y aplicaciones, mientras que la existencia de subconjuntos garantiza la compatibilidad y la consistencia en el uso del lenguaje en diferentes entornos. Esto ha llevado a que C siga siendo uno de los lenguajes de programación más utilizados y versátiles en la actualidad.
3. El código producido en C es **transportable**, pero su portabilidad puede verse afectada por varios factores, incluyendo dependencias del sistema operativo, variaciones en tipos de datos, y el uso de extensiones específicas del compilador. Al seguir prácticas recomendadas y adherirse a los estándares del lenguaje, los programadores pueden maximizar la transportabilidad de su código C entre diferentes plataformas y entornos.