Cuando se solicita ingresar información través de un programa por parte del usuario que está utilizando el programa, independientemente el lenguaje que esté realizado; es importante considerar la validación de la información no solo en su tamaño sino también en los tipos de datos, formatos soportados lo cual nos permite asegurar la calidad de la información que recibimos, almacenamos y procesamos.

Dentro de este contexto en la programación bash para cumplir con este objetivo se utiliza expresiones regulares, las cuales son básicamente cadenas de caracteres que definen un patrón de búsqueda que se valida frente a una información específica para asegurar que cumple la validación definida.

Se necesita conocer ciertos criterios utilizados en las expresiones regulares que son los siguientes:

- ^.- Caracter que representa el inicio de la expresión regular.
- $.- Caracter que representa el final de la expresión regular.
- *.- Caracter que representa cero o más ocurrencias de la expresión
- +.- Caracter que representa una o más ocurrencias de la expresión.
- {n}.-Representa n veces de una expresión.
- [ ] .- Representa un conjunto de caracteres, por ejemplo: [a-z] representa las letras del abecedario de la a a la z.

Tomando en cuenta estos criterios se realizará un programa que valida la siguiente información:
Número de Identificación de un tamaño de 10 números. Ejemplo: 1717836520
País de Origen denotado por dos letras en un rango específico. Ejemplo: EC, CO, US
Fecha de Nacimiento en el formato yyyyMMDD. Ejemplo: 20181222

Primero se definirá las expresiones regulares y se solicitará la información del usuario:

```
#! /bin/bash
# Programa para ejemplificar como capturar la información del usuario y validarla utilizando expresiones regulares
# Autor: Marco Toscano Freire - @martosfre

identificacionRegex='^[0-9]{10}$'
paisRegex='^EC|COL|US$'
fechaNacimientoRegex='^19|20[0-8]{2}[1-12][1-31]$'

echo "Expresiones regulares"
read -p "Ingresar una identificacion:" identificacion
read -p "Ingresar las iniciales de un país [EC, COL, US]:" pais
read -p "Ingresar la fecha de nacimiento [yyyyMMdd]:" fechaNacimiento
```

Luego se validará cada expresión regular comenzando con la identificación, para lo cual para cada validación se utilizará la sentencia condicional if y para comparar la expresión se debe utilizar el siguiente formato especial if [[$variable =~ $expresionRegular]] como se muestra a continuación.

```
#Validación Identificación
if [[ $identificacion =~ $identificacionRegex ]]; then
    echo "Identificación $identificacion válida"
else
    echo "Identificación $identificacion inválida"
fi
```

```
#Validación País
if [[ $pais =~ $paisRegex ]]; then
    echo "País $pais válido"
else
    echo "País $pais inválido"
fi
```

```
#Validación Fecha Nacimiento
if [[ $fechaNacimiento =~ $fechaNacimientoRegex ]]; then
    echo "Fecha Nacimiento $fechaNacimiento válida"
else
    echo "Fecha Nacimiento $fechaNacimiento inválida"
fi
```

Se realizará la ejecución de la aplicación con los dos escenarios el correcto y el incorrecto como se muestra a continuación:

```
MacBook-Pro-de-Marco:PlatziProgramacionShellGit martosfre$ ./8_regularExpression.sh
Expresiones regulares
Ingresar una identificacion:1718234512
Ingresar las iniciales de un país [EC, COL, US]:EC
Ingresar la fecha de nacimiento [yyyyMMdd]:19831112
Identificación 1718234512 válida
País EC válido
Fecha Nacimiento 19831112 válida
MacBook-Pro-de-Marco:PlatziProgramacionShellGit martosfre$
```

```
MacBook-Pro-de-Marco:PlatziProgramacionShellGit martosfre$ ./8_regularExpression.sh
Expresiones regulares
Ingresar una identificacion:1718652341111
Ingresar las iniciales de un país [EC, COL, US]:PE
Ingresar la fecha de nacimiento [yyyyMMdd]:18001230
Identificación 1718652341111 inválida
País PE inválido
Fecha Nacimiento 18001230 inválida
MacBook-Pro-de-Marco:PlatziProgramacionShellGit martosfre$
```

Finalmente el código fuente lo pueden encontrar en el repositorio de GitHub en el branch 7.ValidarInformacion