ESTUDIOS
GENERALES
CIENCIAS

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

FUNDAMENTOS DE PROGRAMACIÓN
LABORATORIO 9
PROPUESTAS DE SOLUCIÓN
SEMESTRE ACADÉMICO 2022-2

Horarios: Todos los horarios

Elaborado por Mag. Jennifer Zárate

INDICACIONES:

- Debe utilizar variables descriptivas, comentarios y mensajes descriptivos.
- El orden y la eficiencia de su implementación serán considerados en la calificación.

RESULTADOS ESPERADOS:

- Al finalizar la sesión, el alumno construirá programas usando diseño estructurado.

CONSIDERACIONES:

- La solución presentada para cada problema corresponde a una propuesta de solución por parte del autor.
- En programación pueden existir muchas soluciones para un mismo problema pero debe cumplir con todo lo solicitado, incluyendo las restricciones brindadas.

Desarrolle los siguientes problemas en lenguaje C:

1. Curiosa Propiedad de Números - 1

En los números existe una curiosa propiedd que se da en algunos números. La propiedad indica que si multiplicamos dos números, que no tienen dígitos repetidos y tampoco sus dígitos se repiten entre ellos, el resultado que se obtiene contiene exactamente los mismo dígitos de ambos números.

Por ejemplo:

Los números 204 y 615 cumplen con esta curiosa propiedad porque:

- El número 204 no tiene dígitos repetidos.
- El número 615 no tiene dígitos repetidos.
- Ningún dígito del número 204 se encuentra en el número 615.
- Y el resultado de la múltiplicación de 204 y 615 es 125460 y todos los dígitos de 125460 se encuentran en los números 204 y 615.

Se pide elaborar un programa en Lenguaje C que, utilizando el paradigma de la programación modular, permita leer dos números y determine si dichos números cumplen con la curiosa propiedad.

Para su solución deberá implementar, además del módulo main(), solo los siguientes módulos:

- Un módulo que permita validar si los números no tienen dígitos repetidos y que los dígitos de un número no se encuentran en el otro número. Para ello debe recibir como parámetros los dos números y debe devolver si son válidos teniendo en cuenta lo indicado.
- Un módulo que permita validar si un dígito se encuentra en un número. Para ello debe recibir como parámetros un dígito, un número y debe devolver si el dígito se encuentra en el número.
- Un módulo que permita validar si un número no tiene dígitos repetidos. Para ello debe recibir como parámetros un número y debe devolver si el número no tiene dígitos repetidos. Además, debe elaborar una tabla de datos para este módulo considerando como parámetro inicial el número 33578.
- Un módulo que permita validar si dos números cumplen con la curiosa propiedad. Para ello deben recibir como parámetros los dos números, el resultado de la multiplicación de ambos y debe devolver si los números cumplen con la curiosa propiedad.

Considere que no deberá añadir algún módulo adicional y que en su solución debe utilizar todos los módulos indicados. En caso añada un módulo adicional y/o no utilice alguno de los módulos, su solución no será considerada válida.

Casos de prueba para verificar la solución:

Debe usar los mensajes que se muestran para el desarrollo del programa.

```
Ingrese los dos números a multiplicar: 88 106
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 87 166
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 187 146
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 204 615
Los numeros 204 y 615 = 125460 cumplen con la curiosa propiedad.
```

```
Ingrese los dos números a multiplicar: 12 387
Los numeros 12 y 387 = 4644 no cumplen con la curiosa propiedad.
```

```
Ingrese los dos números a multiplicar: 435 87
Los numeros 435 y 84 = 37845 cumplen con la curiosa propiedad.
```

Programa 1: Propuesta de solución - Curiosa Propiedad de Números - 1

```
1  #include <stdio.h>
2
3  int validarDatos(int num1,int num2);
4  int validarCumplePropiedad(int resultado,int num1,int num2);
5  int buscarDigito(int digito,int num);
6  int validarRepetidos(int num);
7
8  int main(){
9          int num1, num2, datosValidos, resultado, cumplePropiedad;
10         printf("Ingrese los dos números a multiplicar: ");
11         scanf("%d %d",&num1,&num2);
12         datosValidos = validarDatos(num1,num2);
13         if (datosValidos){
```

```
14                      resultado = num1*num2;
15                      cumplePropiedad = validarCumplePropiedad(resultado,num1,num2);
16                      if (cumplePropiedad)
17                              printf("Los números %d y %d = %d cumplen con la curiosa propiedad",num1,num2,resultado);
18                      else
19                              printf("Los números %d y %d = %d no cumplen con la curiosa propiedad",num1,num2,resultado);
20              }
21              else
22                      printf("Los números ingresados no son válidos");
23              return 0;
24      }
25
26      int validarDatos(int num1,int num2){
27              int sinRepetidos1, sinRepetidos2, valido, digito, esta;
28              sinRepetidos1 = validarRepetidos(num1);
29              sinRepetidos2 = validarRepetidos(num2);
30              valido = 1;
31              while (num1>0){
32                      digito = num1 % 10;
33                      num1 = num1 / 10;
34                      esta = buscarDigito(digito,num2);
35                      if (esta){
36                              valido = 0;
37                              break;
38                      }
39              }
40              return sinRepetidos1 && sinRepetidos2 && valido;
41      }
42
43      int validarCumplePropiedad(int resultado,int num1,int num2){
44              int cumple=1, digito, esta;
45              while (resultado>0){
46                      digito = resultado % 10;
47                      resultado = resultado / 10;
48                      esta = buscarDigito(digito,num1);
49                      if (!esta){
50                              esta = buscarDigito(digito,num2);
51                              if (!esta){
52                                      cumple=0;
53                                      break;
54                              }
55                      }
56              }
57              return cumple;
58      }
59
60      int buscarDigito(int digito,int num){
61              int esta=0, digitoNum;
62              while (num>0){
63                      digitoNum = num % 10;
64                      num = num / 10;
65                      if (digitoNum==digito){
66                              esta = 1;
67                              break;
68                      }
69              }
70              return esta;
71      }
72
73      int validarRepetidos(int num){
74              int sinRepetidos, digito, copiaNum, digito1;
75              sinRepetidos = 1;
76              while (num>0){
77                      digito = num % 10;
78                      num = num / 10;
79                      copiaNum = num;
80                      while (copiaNum>0){
```

```
81                              digito1 = copiaNum  % 10;
82                              copiaNum = copiaNum / 10;
83                              if (digito==digito1){
84                                      sinRepetidos = 0;
85                                      break;
86                              }
87                      }
88                      if (sinRepetidos==0)
89                              break;
90              }
91              return sinRepetidos;
92      }
```

Desarrolle los siguientes problemas en lenguaje C:

2. Curiosa Propiedad de Números - 2

En los números existe una curiosa propiedd que se da en algunos números. La propiedad indica que si multiplicamos dos números, que no tienen dígitos repetidos, tampoco sus dígitos se repiten entre ellos y que entre los dos tienen todas los dígitos del 1 al 9, el resultado que se obtiene contiene exactamente los mismo dígitos de ambos números.

Por ejemplo:

Los números 8745231 y 96 cumplen con esta curiosa propiedad porque:

- El número 8745231 no tiene dígitos repetidos.
- El número 96 no tiene dígitos repetidos.
- Ningún dígito del número 8745231 se encuentra en el número 96.
- Los números 8745231 y 96 tienen, entre los dos, todos los dígitos del 1 al 9.
- Y el resultado de la múltiplicación de 8745231 y 96 es 839542176 y todos los dígitos de 839542176 se encuentran en los números 8745231 y 96.

Se pide elaborar un programa en Lenguaje C que, utilizando el paradigma de la programación modular, permita leer dos números y determine si dichos números cumplen con la curiosa propiedad.

Para su solución deberá implementar, además del módulo main(), solo los siguientes módulos:

- Un módulo que permita validar si los números no tienen dígitos repetidos y que los dígitos de un número no se encuentran en el otro número. Para ello debe recibir como parámetros los dos números y debe devolver si son válidos teniendo en cuenta lo indicado.
- Un módulo que permita validar si un dígito se encuentra en un número. Para ello debe recibir como parámetros un dígito, un número y debe devolver si el dígito se encuentra en el número. Además, debe elaborar una tabla de datos para este módulo considerando como parámetro inicial el dígito 3 y el número 365789.
- Un módulo que permita validar si un número no tiene dígitos repetidos. Para ello debe recibir como parámetros un número y debe devolver si el número no tiene dígitos repetidos.
- Un módulo que permita validar si entre todos los dígitos de los dos números tienen todos los dígitos del 1 al 9. Para ello debe recibir como parámetros los dos números y debe devolver si entre todos los dígitos de los dos números tienen todos los dígitos del 1 al 9.

- Un módulo que permita validar si dos números cumplen con la curiosa propiedad. Para ello deben recibir como parámetros los dos números, el resultado de la multiplicación de ambos y debe devolver si los números cumplen con la curiosa propiedad.

Considere que no deberá añadir algún módulo adicional y que en su solución debe utilizar todos los módulos indicados. En caso añada un módulo adicional y/o no utilice alguno de los módulos, su solución no será considerada válida.

Casos de prueba para verificar la solución:

Debe usar los mensajes que se muestran para el desarrollo del programa.

```
Ingrese los dos números a multiplicar: 88 106
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 87 166
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 187 146
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 204 615
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 12 387
Los números ingresados no son válidos.
```

```
Ingrese los dos números a multiplicar: 8745231 96
Los numeros 8745231 y 96 = 839542176 cumplen con la curiosa propiedad.
```

Programa 2: Propuesta de solución - Curiosa Propiedad de Números - 2

```c
#include <stdio.h>

int validarDatos(int num1,int num2);
int validarCumplePropiedad(int resultado,int num1,int num2);
int buscarDigito(int digito,int num);
int validarDigitos1al9(int num1,int num2);
int validarRepetidos(int num);

int main(){
        int num1, num2, datosValidos, resultado, cumplePropiedad;
        printf("Ingrese los dos números a multiplicar: ");
        scanf("%d %d",&num1,&num2);
        datosValidos = validarDatos(num1,num2);
        if (datosValidos){
                resultado = num1*num2;
                cumplePropiedad = validarCumplePropiedad(resultado,num1,num2);
                if (cumplePropiedad)
                        printf("Los numeros %d y %d = %d cumplen con la curiosa propiedad",num1,num2,resultado);
                else
                        printf("Los numeros %d y %d = %d no cumplen con la curiosa propiedad",num1,num2,resultado);
        }
        else
                printf("Los números ingresados no son válidos");
        return 0;
}
```

```
26
27  int validarDatos(int num1,int num2){
28          int sinRepetidos1, sinRepetidos2, valido, digito, esta, validoDigitos;
29          sinRepetidos1 = validarRepetidos(num1);
30          sinRepetidos2 = validarRepetidos(num2);
31          validoDigitos = validarDigitos1al9(num1,num2);
32          valido = 1;
33          while (num1>0){
34                  digito = num1  % 10;
35                  num1 = num1 / 10;
36                  esta = buscarDigito(digito,num2);
37                  if (esta){
38                          valido = 0;
39                          break;
40                  }
41          }
42          return sinRepetidos1 && sinRepetidos2 && valido && validoDigitos;
43  }
44
45  int validarCumplePropiedad(int resultado,int num1,int num2){
46          int cumple=1, digito, esta;
47          while (resultado>0){
48                  digito = resultado  % 10;
49                  resultado = resultado / 10;
50                  esta = buscarDigito(digito,num1);
51                  if (!esta){
52                          esta = buscarDigito(digito,num2);
53                          if (!esta){
54                                  cumple=0;
55                                  break;
56                          }
57                  }
58          }
59          return cumple;
60  }
61
62  int buscarDigito(int digito,int num){
63          int esta=0, digitoNum;
64          while (num>0){
65                  digitoNum = num  % 10;
66                  num = num / 10;
67                  if (digitoNum==digito){
68                          esta = 1;
69                          break;
70                  }
71          }
72          return esta;
73  }
74
75  int validarRepetidos(int num){
76          int sinRepetidos, digito, copiaNum, digito1;
77          sinRepetidos = 1;
78          while (num>0){
79                  digito = num  % 10;
80                  num = num / 10;
81                  copiaNum = num;
82                  while (copiaNum>0){
83                          digito1 = copiaNum  % 10;
84                          copiaNum = copiaNum / 10;
85                          if (digito==digito1){
86                                  sinRepetidos = 0;
87                                  break;
88                          }
89                  }
90                  if (sinRepetidos==0)
91                          break;
92          }
```

```
93          return sinRepetidos;
94  }
95
96  int validarDigitos1al9(int num1,int num2){
97          int valido = 1, digito, esta;
98          for (digito=1; digito<=9; digito++){
99                  esta = buscarDigito(digito,num1);
100                 if (!esta){
101                         esta = buscarDigito(digito,num2);
102                         if (!esta){
103                                 valido = 0;
104                                 break;
105                         }
106                 }
107         }
108         return valido;
109 }
```

Desarrolle los siguientes problemas en lenguaje C:

3. Fechas Capicúas - 1

Un número capicúa es el que se lee exactamente igual de izquierda a derecha, que de derecha a izquierda. Teniendo en cuenta esta premisa, podemos trabajar lo que son las fechas capicúas. Para que una fecha en el formato dd/mm/yyyy se considere capicúa se debe cumplir que se lee exactamente igual de izquierda a derecha, que de derecha a izquierda.

Por ejemplo:

La fecha 12/02/2021 es considerada una fecha capicúa porque el número 12022021, formado por el día, mes y año en formato ddmmyyyy, es un número capicúa.

Otro ejemplo:

La fecha 02/02/2020 es considerada una fecha capicúa porque el número 02022020, formado por el día, mes y año en formato ddmmyyyy, es un número capicúa.

Se pide elaborar un programa en Lenguaje C que, utilizando el paradigma de la programación modular, permita leer un año y un mes y determine la siguiente fecha capicúa más próxima a partir de dicho año y mes. Debe validar que el año debe ser mayor que 2010 y el mes debe ser válido, si no se cumple dicha validación el programa termina y no se determina nada.

Para su solución deberá implementar, además del módulo main(), solo los siguientes módulos:

- Un módulo que permita validar si el año y mes son válidos. Para ello debe recibir como parámetros el año, el mes y debe devolver si son válidos teniendo en cuenta lo indicado en el enunciado.
- Un módulo que permita encontrar la siguiente fecha capicúa. Para ello debe recibir como parámetros el año, el mes y debe devolver, utilizando parámetros que modifican su valor, el año, mes y día de la siguiente fecha capicúa.
- Un módulo que permita obtener la cantidad de días que tiene un mes. Para ello debe recibir como parámetros el año, el mes y debe devolver la cantidad de días que tiene ese mes. Recuerde que los meses de Enero, Marzo, Mayo, Julio, Agosto, Octubre y Diciembre tienen 31 días, los meses de Abril, Junio, Setiembre y Noviembre tienen 30 días, el mes de Febrero tiene 28 días, a excepción de los años bisiestos donde tiene 29 días.
- Un módulo que permita validar si una fecha es capicúa. Para ello debe recibir como parámetros el día, mes, año y debe devolver si la fecha es capicúa, de acuerdo a lo descrito en el enunciado. Además, debe elaborar una tabla de datos para este módulo considerando como parámetros iniciales el día=2, mes=2 y año=2020.

Considere que no deberá añadir algún módulo adicional y que en su solución debe utilizar todos los módulos indicados. En caso añada un módulo adicional y/o no utilice alguno de los módulos, su solución no será considerada válida.

Casos de prueba para verificar la solución:

Debe usar los mensajes que se muestran para el desarrollo del programa.

```
Ingrese el año: 2009
Ingrese el mes: 10
Los datos ingresados no son correctos.
```

```
Ingrese el año: 2015
Ingrese el mes: 15
Los datos ingresados no son correctos.
```

```
Ingrese el año: 2020
Ingrese el mes: 1
La siguiente fecha capicúa es 12/02/2021.
```

```
Ingrese el año: 2015
Ingrese el mes: 8
La siguiente fecha capicúa es 02/02/2020.
```

Programa 3: Propuesta de solución - Fechas Capicúas - 1

```c
#include <stdio.h>

int validarDatos(int anho,int mes);
void encontrarSiguienteFecha(int anho,int mes,int *anho1,int *mes1,int *dia1);
int obtenerDiaMaximo(int mes,int anho);
int validarFechaCapicua(int dia,int mes,int anho);

int main(){
	int anho, mes, datosValidos, anho1, mes1, dia;
	printf("Ingrese el a: ");
	scanf(" %d",&anho);
	printf("Ingrese el mes: ");
	scanf(" %d",&mes);
	datosValidos = validarDatos(anho,mes);
	if (datosValidos){
		encontrarSiguienteFecha(anho,mes,&anho1,&mes1,&dia);
		printf("La siguiente fecha capica es  %02d/ %02d/ %04d",dia,mes1,anho1);
	}
	else {
		printf("Los datos ingresados no son correctos");
	}
	return 0;
}

int validarDatos(int anho,int mes){
	return anho>2010 && mes>0 && mes<13;
}

void encontrarSiguienteFecha(int anho,int mes,int *anho1,int *mes1,int *dia1){
	int centinela, encontreCapicua, centinela1, centinela2, diaMaximo, dia, esCapicua;
	centinela = 1;
	encontreCapicua = 0;
	while (centinela){
		centinela1 = 1;
		while (centinela1){
```

```
                        diaMaximo = obtenerDiaMaximo(mes,anho);
                        dia = 1;
                        centinela2 = 1;
                        while (centinela2){
                                esCapicua = validarFechaCapicua(dia,mes,anho);
                                if (esCapicua){
                                        *anho1 = anho;
                                        *mes1 = mes;
                                        *dia1 = dia;
                                        centinela2 = 0;
                                        encontreCapicua = 1;
                                }
                                dia++;
                                if (dia>diaMaximo){
                                        centinela2 = 0;
                                }
                        }
                        mes++;
                        if (mes>12){
                                centinela1 = 0;
                        }
                        if (encontreCapicua){
                                centinela1 = 0;
                        }
                }
                mes = 1;
                anho++;
                if (encontreCapicua){
                        centinela = 0;
                }
        }
}

int obtenerDiaMaximo(int mes,int anho){
        int diaMaximo, esBisiesto;
        if (mes==1 || mes==3 || mes==5 || mes==7 || mes==8 || mes==10 || mes==12){
                diaMaximo = 31;
        }
        else {
                if (mes==4 || mes==6 || mes== 9 || mes==11){
                        diaMaximo = 30;
                }
                else {
                        esBisiesto = (anho  % 4 == 0) && (!(anho  % 100==0) || (anho %400==0));
                        if (esBisiesto){
                                diaMaximo = 29;
                        }
                        else {
                                diaMaximo = 28;
                        }
                }
        }
        return diaMaximo;
}

int validarFechaCapicua(int dia,int mes,int anho){
        int valor, copia, invertido, digito, i;
        valor = (dia*100 + mes)*10000 + anho;
        copia = valor;
        invertido = 0;
        for (i=1; i<=8; i++){
                digito = valor %10;
                valor = valor/10;
                invertido = invertido*10 + digito;
        }
        return (invertido==copia);
}
```

Desarrolle los siguientes problemas en lenguaje C:

4. Fechas Capicúas - 2

Un número capicúa es el que se lee exactamente igual de izquierda a derecha, que de derecha a izquierda. Teniendo en cuenta esta premisa, podemos trabajar lo que son las fechas capicúas. Para que una fecha en el formato dd/mm/yyyy se considere capicúa se debe cumplir que se lee exactamente igual de izquierda a derecha, que de derecha a izquierda.

Por ejemplo:

La fecha 12/02/2021 es considerada una fecha capicúa porque el número 12022021, formado por el día, mes y año en formato ddmmyyyy, es un número capicúa. Además, se puede decir que la fecha 12/02/2021 se encuentra en base 5 porque todos sus dígitos son menores que 5.

Otro ejemplo:

La fecha 02/02/2020 es considerada una fecha capicúa porque el número 02022020, formado por el día, mes y año en formato ddmmyyyy, es un número capicúa. Además, se puede decir que la fecha 02/02/2020 se encuentra en base 3 porque todos sus dígitos son menores que 3.

Se pide elaborar un programa en Lenguaje C que, utilizando el paradigma de la programación modular, permita leer un año, un mes, una base y determine la siguiente fecha capicúa más próxima a partir de dicho año y mes y que cumpla con pertenecer a la base indicada. Debe validar que el año debe ser mayor que 2010, el mes debe ser válido y que la base debe ser mayor o igual que 2 y menor o igual que 10, si no se cumple dicha validación el programa termina y no se determina nada.

Para su solución deberá implementar, además del módulo main(), solo los siguientes módulos:

- Un módulo que permita validar si el año, mes y la base son válidos. Para ello debe recibir como parámetros el año, el mes, la base y debe devolver si son válidos teniendo en cuenta lo indicado en el enunciado.
- Un módulo que permita encontrar la siguiente fecha capicúa que se encuentre dentro de una base. Para ello debe recibir como parámetros la base, el año, el mes y debe devolver, utilizando parámetros que modifican su valor, el año, mes y día de la siguiente fecha capicúa que se encuentre dentro de la base indicada.
- Un módulo que permita obtener la cantidad de días que tiene un mes. Para ello debe recibir como parámetros el año, el mes y debe devolver la cantidad de días que tiene ese mes. Recuerde que los meses de Enero, Marzo, Mayo, Julio, Agosto, Octubre y Diciembre tienen 31 días, los meses de Abril, Junio, Setiembre y Noviembre tienen 30 días, el mes de Febrero tiene 28 días, a excepción de los años bisiestos donde tiene 29 días.
- Un módulo que permita validar si una fecha es capicúa y se encuentra en una base. Para ello debe recibir como parámetros la base, el día, mes, año y debe devolver si la fecha es capicúa y se encuentra dentro de la base, de acuerdo a lo descrito en el enunciado. Además, debe elaborar una tabla de datos para este módulo considerando como parámetros iniciales la base=3, el día=2, mes=2 y año=2020.

Considere que no deberá añadir algún módulo adicional y que en su solución debe utilizar todos los módulos indicados. En caso añada un módulo adicional y/o no utilice alguno de los módulos, su solución no será considerada válida.

Casos de prueba para verificar la solución:

Debe usar los mensajes que se muestran para el desarrollo del programa.

```
Ingrese el año: 2009
Ingrese el mes: 10
Ingrese la base: 2
Los datos ingresados no son correctos.
```

```
Ingrese el año: 2015
Ingrese el mes: 15
Ingrese la base: 3
Los datos ingresados no son correctos.
```

```
Ingrese el año: 2015
Ingrese el mes: 10
Ingrese la base: 11
Los datos ingresados no son correctos.
```

```
Ingrese el año: 2028
Ingrese el mes: 9
Ingrese la base: 4
La siguiente fecha capicúa que está en base 4 es 03/02/2030.
```

```
Ingrese el año: 2060
Ingrese el mes: 10
Ingrese la base: 7
La siguiente fecha capicúa que está en base 7 es 16/02/2061.
```

Programa 4: Propuesta de solución - Fechas Capicúas - 2

```c
#include <stdio.h>

int validarDatos(int anho,int mes, int base);
void encontrarSiguienteFecha(int base, int anho,int mes,int *anho1,int *mes1,int *dia1);
int obtenerDiaMaximo(int mes,int anho);
int validarFechaCapicua(int base, int dia,int mes,int anho);
int validarBase(int numero, int base);

int main(){
	int anho, mes, datosValidos, anho1, mes1, dia,base;
	printf("Ingrese el año: ");
	scanf(" %d",&anho);
	printf("Ingrese el mes: ");
	scanf(" %d",&mes);
	printf("Ingrese la base: ");
	scanf(" %d",&base);
	datosValidos = validarDatos(anho,mes,base);
	if (datosValidos){
		encontrarSiguienteFecha(base,anho,mes,&anho1,&mes1,&dia);
		printf("La siguiente fecha capicúa que está en base  %d es  %02d/ %02d/ %04d",base,dia,mes1,anho1);
	}
	else {
		printf("Los datos ingresados no son correctos");
	}
	return 0;
}

int validarDatos(int anho, int mes, int base){
	return anho>2010 && mes>0 && mes<13 && base>1 && base<=10;
}

void encontrarSiguienteFecha(int base, int anho,int mes,int *anho1,int *mes1,int *dia1){
	int centinela, encontreCapicua, centinela1, centinela2, diaMaximo, dia, esCapicua;
```

```
34          centinela = 1;
35          encontreCapicua = 0;
36          while (centinela){
37                  centinela1 = 1;
38                  while (centinela1){
39                          diaMaximo = obtenerDiaMaximo(mes,anho);
40                          dia = 1;
41                          centinela2 = 1;
42                          while (centinela2){
43                                  esCapicua = validarFechaCapicua(base,dia,mes,anho);
44                                  if (esCapicua){
45                                          *anho1 = anho;
46                                          *mes1 = mes;
47                                          *dia1 = dia;
48                                          centinela2 = 0;
49                                          encontreCapicua = 1;
50                                  }
51                                  dia++;
52                                  if (dia>diaMaximo){
53                                          centinela2 = 0;
54                                  }
55                          }
56                          mes++;
57                          if (mes>12){
58                                  centinela1 = 0;
59                          }
60                          if (encontreCapicua){
61                                  centinela1 = 0;
62                          }
63                  }
64                  mes = 1;
65                  anho++;
66                  if (encontreCapicua){
67                          centinela = 0;
68                  }
69          }
70  }
71
72  int obtenerDiaMaximo(int mes,int anho){
73          int diaMaximo, esBisiesto;
74          if (mes==1 || mes==3 || mes==5 || mes==7 || mes==8 || mes==10 || mes==12){
75                  diaMaximo = 31;
76          }
77          else {
78                  if (mes==4 || mes==6 || mes== 9 || mes==11){
79                          diaMaximo = 30;
80                  }
81                  else {
82                          esBisiesto = (anho  % 4 == 0) && (!(anho  % 100==0) || (anho %400==0));
83                          if (esBisiesto){
84                                  diaMaximo = 29;
85                          }
86                          else {
87                                  diaMaximo = 28;
88                          }
89                  }
90          }
91          return diaMaximo;
92  }
93
94  int validarFechaCapicua(int base, int dia,int mes,int anho){
95          int valor, copia, invertido, digito, i, estaBase;
96          valor = (dia*100 + mes)*10000 + anho;
97          copia = valor;
98          invertido = 0;
99          estaBase = 1;
100         for (i=1; i<=8; i++){
```

```
101                     digito = valor %10;
102                     valor = valor/10;
103                     invertido = invertido*10 + digito;
104                     if (digito>=base){
105                             estaBase = 0;
106                     }
107             }
108             return (invertido==copia && estaBase);
109     }
```

Puede usar cualquier estructura selectiva por lo que el uso de la estructura selectiva múltiple queda a su criterio

Puede usar cualquier estructura anidada de ser el caso