```
 1  using System;
 2
 3  namespace ResumenFunciones
 4  {
 5      public class FunctionsClaseMarin
 6      {
 7
 8          #region Funciones Principiantes
 9          // FUNCION 1 ·
10          // Funcion que devuelve el menor de dos numeros.
11          public static int GetMinor(int number1, int number2)
12          {
13              if (number1 <= number2)
14                  return number1;
15              return number2;
16          }
17
18          // FUNCION 2 ·
19          // Funcion que devuelve verdadero si un numero es par.
20          public static bool IsEven(int number)
21          {
22              return (number % 2 == 0);
23          }
24
25          // FUNCION 3 ·
26          // Funcion que devuelve el menor de tres numeros.
27          public static int GetMinest(int number1, int number2, int
              number3)
28          {
29              if (number1 <= number2 && number1 <= number3)
30                  return number1;
31              if (number2 <= number3)
32                  return number2;
33              else
34                  return number3;
35          }
36
37          // FUNCION 4 ·
38          // Serie que imprime por consola la serie 0, -2, 4, -6, 8.
39          public static void PrintSerie(int number)
40          {
41              int result;
42
43              for (int i = 0; i <= number / 2; i++)
44              {
45                  if (IsEven(i))
46                      result = (2 * i);
47                  else
48                      result = (-2 * i);
49                  Console.WriteLine(result);
50              }
51          }
52
```

```
53          // FUNCION 5 ·
54          // TODO: [EXAMEN] Funcion 'bool' si un numero es primo.
55          public static bool IsPrime(int number)
56          {
57              for (int i = 2; i < number; i++)
58              {
59                  if (number % i == 0)
60                      return false;
61              }
62              return true;
63          }
64
65          // FUNCION 6 ·
66          // Funcion que imprime en consola los valores de la serie
              Collatz.
67          public static void Collatz(int number)
68          {
69              int result = number;
70              Console.WriteLine(result);
71
72              while (result != 1)
73              {
74                  if (IsEven(result))
75                      result /= 2;
76                  else
77                      result = (result * 3) + 1;
78                  Console.WriteLine(result);
79              }
80          }
81
82
83          // FUNCION 7 ·
84          // Funcion que devuelve el sumatorio de un numero.
85          public static int GetSummatory(int number)
86          {
87              int result = 0;
88
89              for (int i = 1; i <= number; i++)
90              {
91                  result += i;
92              }
93              return result;
94          }
95
96
97          // FUNCION 8 ·
98          // Funcion que devuelve el productorio de un numero.
99          public static int GetProductory(int number)
100         {
101             int result = 1;
102             for (int i = 1; i <= number; i++)
103             {
104                 result *= i;
```

```
105             }
106             return result;
107         }
108         #endregion
109
110         #region Funciones Elementales
111
112         // FUNCION 9 ·
113         // Devuelve un 'bool' si un número es MENOR que otro.
114         public static bool IsMinor(int number1, int number2)
115         {
116             return number1 < number2;
117         }
118
119         // FUNCION 10 ·
120         // Devuelve un 'bool' si un número es MAYOR que otro.
121         public static bool IsMajor(int number1, int number2)
122         {
123             return number1 > number2;
124         }
125
126         // FUNCION 11 ·
127         // Funcion auxiliar que devuelve un 'bool' si un número PAR.
128         public static bool IsEvenAlternative(int number)
129         {
130             if (number % 2 == 0)
131                 return true;
132             return false;
133         }
134
135         // FUNCION 12 ·
136         // Funcion auxiliar que devuelve un 'bool' si un numero es
              IMPAR.
137         public static bool IsOdd(int number)
138         {
139             return (number % 2 != 0);
140         }
141         #endregion
142
143         #region Funciones de bucles
144         // FUNCION 13 ·
145         // Funcion Sumatory con un bucle WHILE.
146         public static int GetSumatoryWhile(int number)
147         {
148             int result = 0;
149             int i = 1;
150
151             while (i <= number)
152             {
153                 result += i;
154                 i++;
155             }
156             return result;
```

```
157         }
158
159         // FUNCION 14 ·
160         // Funcion Sumatory con un bucle FOR.
161         public static int GetSumatoryFor(int number)
162         {
163             int result = 0;
164             for (int i = 1; i <= number; i++)
165             {
166                 result += i;
167             }
168             return result;
169         }
170
171         // FUNCION 15 ·
172         // Funcion Secuencia con un bucle WHILE.
173         public static void GetSequenceWhile(int number)
174         {
175             int count = 0;
176             int result = 0;
177             int resultCounter = 0;
178
179             while (resultCounter < number)
180             {
181                 if (IsEven(count))
182                 {
183                     result = count * 2 * (1);
184                 }
185                 else
186                 {
187                     result = count * 2 * (-1);
188                 }
189                 Console.WriteLine(result);
190                 resultCounter = Math.Abs(result);
191                 count++;
192             }
193         }
194
195         // FUNCION 16 ·
196         // Funcion Secuencia con un bucle FOR.
197         public static void GetSequenceFor(int number)
198         {
199             int result;
200
201             for (int i = 1; i <= number / 2; ++i)
202             {
203                 if (IsEven(i))
204                 {
205                     result = i * 2 * (1);
206                 }
207                 else
208                 {
209                     result = i * 2 * (-1);
```

```
210                 }
211
212                 Console.WriteLine(result);
213             }
214         }
215         #endregion
216
217         #region Funciones Repaso
218         // FUNCION 17 ·
219         // Devuelve el menor de dos numeros.
220         public static int GetMinorAlternative(int number1, int number2)
221         {
222             if (number1 <= number2)
223                 return number1;
224             return number2;
225         }
226
227         // FUNCION 18 ·
228         // Funcion anterior con operador ternario.
229         public static int GetMinorTernary(int number1, int number2)
230         {
231             return number1 < number2 ? number1 : number2;
232         }
233
234         // FUNCION 19 ·
235         // Devuelve el menor de tres numeros.
236         public static int GetMinestAlternative(int number1, int
              number2, int number3)
237         {
238             if (number1 <= number2 && number1 <= number3)
239                 return number1;
240             if (number2 <= number3)
241                 return number2;
242             else
243                 return number3;
244         }
245
246         // FUNCION 20 ·
247         // Devuelve un 'bool' si un numero es primo.
248         public static bool IsPrimeFirst(int number)
249         {
250             for (int i = 2; i < number; i++)
251             {
252                 if (number % i == 0)
253                     return false;
254             }
255             return true;
256         }
257
258         //FUNCION 21 ·
259         //Imprime en consola los valores de la serie Collatz.
260         public static void CollatzV2(int number)
261         {
```

```
262             int result = number;
263             Console.WriteLine(result);
264 
265             while (result != 1)
266             {
267                 if (IsEven(result))
268                     result /= 2;
269                 else
270                     result = (result * 3) + 1;
271                 Console.WriteLine(result);
272             }
273         }
274 
275         // FUNCION 22 ·
276         // Devuelve el sumatorio de un numero.
277         public static int GetSummatoryAlternative(int number)
278         {
279             int result = 0;
280             for (int i = 1; i <= number; i++)
281             {
282                 result += i;
283             }
284             return result;
285         }
286 
287         //FUNCION 23 ·
288         //Devuelve el productorio de un numero.
289         public static int GetProductoryAlternative(int number)
290         {
291             int result = 1;
292             for (int i = 1; i <= number; i++)
293             {
294                 result *= i;
295             }
296             return result;
297         }
298         #endregion
299 
300         #region Funciones series
301         // FUNCION 24 ·
302         // Funcion Secuencia con un metodo reservado 'MATH.POW'.
303         public static void GetSequence3(int number)
304         {
305             int result = -2, printNumber, count = 0;
306 
307             while (result < number)
308             {
309                 result += 2;
310                 printNumber = result * (int)Math.Pow(-1, count);
311                 Console.WriteLine(printNumber);
312                 count++;
313             }
314         }
```

```
315
316         // FUNCION 25 ·
317         // Devuelve 'bool' si un numero es Primo.
318         public static bool IsPrimeAlternative(int number)
319         {
320             for (int i = 2; i < number; i++)
321             {
322                 if (number % i == 0)
323                     return false;
324             }
325             return true;
326         }
327         // FUNCION 26 ·
328         // Devuelve la serie Collatz.
329         public static List<int> CollatzAlternative(int number)
330         {
331             List<int> list = new List<int>();
332             int result = number;
333
334             list.Add(number);
335
336             while (result != 1)
337             {
338                 if (IsEven(result))
339                     result /= 2;
340                 else
341                     result = (result * 3) + 1;
342                 list.Add(result);
343             }
344
345             return list;
346         }
347
348         // FUNCION 27 ·
349         // Serie de Fibonacci con un numero determinado de resultados
350         // TODO: [EXAMEN] Serie Fibonacci.
351         public static string FibonacciFOR(int number)
352         {
353             string result = "0,1";
354             int n1 = 0;
355             int n2 = 1;
356             int sumNumbers;
357
358             for (int i = 0; i < number - 2; i++)
359             {
360                 sumNumbers = n1 + n2;
361                 result += "," + sumNumbers;
362                 n1 = n2;
363                 n2 = sumNumbers;
364             }
365             return result;
366         }
367
```

```
368         // FUNCION 28 ·
369         // Serie de Fibonacci mientras sea menor que un numero de resultados.
370         public static string FibonacciWHILE(int number)
371         {
372             string result = "0,1";
373             int n1 = 0;
374             int n2 = 1;
375             int sumNumbers = n1 + n2;
376             while (sumNumbers <= number)
377             {
378                 sumNumbers = n1 + n2;
379                 result += "," + sumNumbers;
380                 n1 = n2;
381                 n2 = sumNumbers;
382             }
383             return result;
384         }
385         #endregion
386
387         #region Ordenar listas
388         // FUNCION 29 ·
389         // Funcion que ejecuta un 'SWAP' entre dos elementos.
390         // TODO: [EXAMEN] Funcion Swap.
391         public static void Swap(List<int> list)
392         {
393             int aux;
394             aux = list[0];
395             list[0] = list[1];
396             list[1] = aux;
397         }
398
399         // FUNCION 30 ·
400         // Funcion que ejecuta un metodo 'SORT' una lista.
401         // TODO: [EXAMEN] Funcion Sort.
402         public static void Sort(List<int> list)
403         {
404             if (list == null || list.Count == 0)
405                 return;
406
407             int aux;
408             int n1 = list.Count - 1;
409             int n2 = list.Count;
410
411             for (int i = 0; i < n1; i++)
412             {
413                 for (int j = i + 1; j < n2; j++)
414                 {
415                     if (list[i] > list[j])
416                     {
417                         aux = list[i];
418                         list[i] = list[j];
419                         list[j] = aux;
```

```
420                     }
421                 }
422             }
423         }
424
425         // FUNCION 31 ·
426         // Devuelve los elementos pares de una lista.
427         public static List<int> GetListOdd(List<int> list)
428         {
429             if (list == null)
430                 return null;
431
432             if (list.Count == 0)
433                 return new List<int>() { 0 };
434
435             List<int> result = new List<int>();
436
437             for (int i = 0; i < list.Count; i++)
438             {
439                 if (IsEven(list[i]))
440                     result.Add(i);
441             }
442             return result;
443         }
444
445         // FUNCION 32 ·
446         // Devuelve los elementos pares de un array.
447         // [EXAMEN] Posible ejercicio de arrays.
448         public static int[] GetListOdd(int[] array)
449         {
450             if (array == null)
451                 return new int[0];
452
453             int[] result;
454             int count = 0;
455
456             // Buscamos los numeros pares y guardamos el conteo total
457             for (int i = 0; i < array.Length; i++)
458             {
459                 if (IsEven(array[i]))
460                     count++;
461             }
462
463             // Creamos un array cuya longitud es igual al numero de
                  pares encontrados
464             result = new int[count];
465
466             // Introducimos los elementos pares dentro de un nuevo
                  array
467             int contador = 0;
468             for (int i = 0; i < array.Length; i++)
469             {
470                 if (IsEven(array[i]))
```

```
471                 {
472                     result[contador] = array[i];
473                     contador++;
474                 }
475             }
476 
477             return result;
478         }
479         #endregion
480 
481         #region Funciones Varias
482         public static bool IsLeapYear(int year)
483         {
484             if (year % 4 == 0)
485             {
486                 if (year % 100 == 0)
487                 {
488                     if (year % 400 == 0)
489                         return true;
490                     return false;
491                 }
492                 return true;
493             }
494             return false;
495         }
496 
497         public static string FunctionTernary(int number)
498         {
499             string result = "";
500             result = (number > 5) ? "Mas de cinco" : "Menos de cinco";
501             return result;
502         }
503 
504         public enum DayWeek
505         {
506             MONDAY,
507             TUESDAY,
508             WEDNESDAY,
509             THURSDAY,
510             FRIDAY,
511             SATURDAY,
512             SUNDAY
513         }
514 
515         public static int GetDayWeek(DayOfWeek dayWeek) => (int)
              dayWeek;
516 
517         public DayWeek dia;
518         public int numero;
519 
520         #endregion
521     }
522 }
```