```
1: ////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreo
7: //
8: ////////////////////////////////////////////////////////////////////////
9:
10: /*
11: 24. [Mínimo de varios valores] Realice un programa que lea enteros desde teclado y calcule
12: cuántos se han introducido y cual es el mínimo de dichos valores (pueden ser
13: positivos o negativos). Se dejará de leer datos cuando el usuario introduzca el valor 0.
14: Realice la lectura de los enteros dentro de un bucle sobre una única variable llamada
15: dato. Es importante controlar los casos extremos, como por ejemplo, que el primer
16: valor leído fuese ya el terminador de entrada (en este caso, el cero).
17: Ejemplo de entrada: 0
18:    Salida correcta: Introducidos: 0. Mínimo: 0
19: Ejemplo de entrada: 1 3 -1 2 0
20:    Salida correcta: Introducidos: 4. Mínimo: -1
21: Ejemplo de entrada: 1 3 1 2 0
22:    Salida correcta: Introducidos: 4. Mínimo: 1
23: Una vez hecho el programa, indique qué cambios debería realizar si los valores a leer
24: son enteros negativos y el final de la entrada de datos lo marca la introducción de
25: cualquier valor positivo.
26: */
27: #include <iostream>
28: #include <cmath>
29:
30: using namespace std;
31:
32: int main(){
33:
34:    int dato, min_dato, i=-1;
35:    const int terminador = 0;
36:
37:    cout << "Introduce una serie de numeros: ";
38:    cin >> dato;
39:    i++;
40:    min_dato = dato;
41:    do{
42:       cout << " ";
43:       cin >> dato;
44:       if(dato < min_dato && dato != 0)
45:          min_dato = dato;
46:       i++;
47:    }
48:    while(dato != terminador);
49:
50:    cout << "---------------------------------------" << endl;
51:    cout << "\tEl numero mas pequenio es el " << min_dato << endl
52:         << "\t y hay " << i << " numeros" << endl;
53: }
54:
```

```
1: ////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreo
7: //
8: ////////////////////////////////////////////////////////////////////////
9:
10:    /* [Aproximación de PI por Gregory-Leibniz] En el siglo XVII el matemático alemán Gottfried
11:    Leibniz y el matemático escocés James Gregory introdujeron una forma de calcular
12:    PI a través de una serie, es decir, de una suma de términos:
13:
14:    pi/4 = pow(-1,i)/(2*n +1)
15:
16:    Esta es una serie infinita, pues realiza la suma de infinitos términos. Como en Programación
17:    no podemos realizar un número infinito de operaciones, habrá que parar en
18:    un índice dado, llamémosle tope, obteniendo por tanto una aproximación al valor de
19:    PI. Usaremos el símbolo "aprox" para denotar esta aproximación:
20:
21:    pi/4 aprox pow(-1,n)/(2*n +1) ... + pow(-1,tope)/(2*tope +1)
22:
23:    Construya un programa que lea el valor tope obligando a que esté entre 1 y cien mil,
24:    calcule la aproximación de PI mediante la anterior serie e imprima el resultado en
25:    pantalla.
26:    Resuelva este problema de tres formas distintas (no hace falta que entregue tres ejercicios:
27:    baste con que incluya las dos primeras soluciones dentro de un comentario):
28:
29:    a) Use la función pow (potencia) de cmath para implementar (-1)^n
30:
31:    b) Para cada valor de n, calcule (-1)^n con un bucle, tal y como hizo en el ejercicio
32:    de la potencia (problema 18 [Factorial y Potencia] )
33:
34:    c) De una forma más eficiente que las anteriores. Por ejemplo, observe que el valor
35:    de (-1)^n es 1 para los valores pares de i y -1 para los impares
36:
37:    Recuerde que, para visualizar 15 cifras decimales, por ejemplo, debe incluir la sentencia
38:    cout.precision(15); antes de realizar la salida en pantalla.
39:
40:    Ejemplo de entrada: 1000 -- Salida correcta: 3.14259165433954
41:    Ejemplo de entrada: 100000 -- Salida correcta: 3.14160265348972
42:    */
43: #include <iostream>
44: #include <cmath>
45:
46: using namespace std;
47:
48: int main(){
49:
50:
51:    int tope;
52:    int n;
53:    double sumando, suma;
54:    double pi_aprox;
55:    do
56:    {
57:      cout << "Introduce el tope de calculo: ";
58:      cin >>tope;
59:    }
60:     while(tope<0 || tope>1e+5);
61:
62:    /* //Metodo (a):
63:    suma = 0;
64:    n = 0;
65:
66:    while(n<=tope){
67:
68:       sumando = pow(-1,n)/(2*n +1);
69:       // o directamente suma = suma + pow(-1, n) / (2*n + 1);
70:       suma = suma + sumando;
71:       n++;
72:    }
73:
74:    pi_aprox = 4 * suma;             //Es porque se iguala a pi/4 por lo que para obtener pi, se le multi
plica por 4
75:    cout.precision(15);
76:    cout << pi_aprox << endl;
77:
78:    return 0;
79:    */
80:
81:
82:    /* //METODO (B)
83:    suma = 0;
84:    int signo = 1;
85:    const int CAMBIO_SIGNO = -1;
```

```
   86:    for(n=0;n<tope;n++){
   87:       sumando = signo/(2.0*n +1);
   88:       suma = suma + sumando;
   89:       signo= signo * CAMBIO_SIGNO;
   90:    }
   91:    pi_aprox = 4*suma;                //Es porque se iguala a pi/4 por lo que para obtener pi, se le mult
iplica por 4
   92:    cout.precision(15);
   93:    cout << pi_aprox << endl;
   94:
   95:    return 0;
   96:    */
   97:    //METODO (c)
   98:    suma = 0;
   99:    int signo;
  100:
  101:    for(n=0;n<tope;n++){
  102:       if(n%2==0)
  103:          signo=1;
  104:       else
  105:          signo=-1;
  106:
  107:       sumando = signo/(2.0*n +1);
  108:       suma = suma + sumando;
  109:    }
  110:    pi_aprox = 4*suma;                //Es porque se iguala a pi/4 por lo que para obtener pi, se le mult
iplica por 4
  111:    cout.precision(15);
  112:    cout << pi_aprox << endl;
  113:
  114:    return 0;
  115: }
  116:
  117:
  118:
  119:
  120:
  121:
  122:
```

```cpp
////////////////////////////////////////////////////////////////////////
//
// Fundamentos de Programación
// ETS Informática y Telecomunicaciones
// Universidad de Granada
// // Don Oreo
//
////////////////////////////////////////////////////////////////////////

/* [Aproximación de PI por Wallis] Otra aproximación de PI introducida en el siglo XVII por
el matemático inglés John Wallis viene dada por:

   pi/2 =2/1 • 2/3 • 4/3 • 4/5 • 6/5 • 6/7 ...

   Construya un programa que lea el valor tope obligando a que esté entre 1 y cien mil,
   calcule la aproximación de PI mediante la anterior fórmula (multiplicando un total de
   tope fracciones) e imprima el resultado en pantalla.
   Debe resolver este problema de dos formas distintas, a saber:
      --Observe que el numerador y el denominador varían de forma alternativa (aunque
   ambos de la misma forma, a saltos de 2). Cuando a uno le toca cambiar, el otro
   permanece igual. Este comportamiento se puede controlar con una única variable
   de tipo de dato bool.
      --Otra forma de implementar los cambios en el numerador y denominador es observando
   que en cada iteración, el numerador es el denominador de la iteración
   anterior más 1 y el denominador es el numerador de la iteración anterior más 1.
   Ejemplo de entrada: 1000 -- Salida correcta: 3.1400238186006
   Ejemplo de entrada: 100000 -- Salida correcta: 3.14157694582286
*/

#include <iostream>
#include <cmath>

using namespace std;

int main(){
   int tope;
   int i = 0.0;
   double numerador = 0.0;
   double denominador = 1.0;
   double serie;
   double acumulador = 1.0;
   double pi_aprox;
   bool cambio;

   do{
      cout << "Introduce el tope de calculo: ";
      cin >> tope;
   }
   while(tope<0 || tope>1e+5);

   for(i=0;i<tope;i++){
      cambio = (i%2==0);
      if(cambio){
         numerador = numerador + 2;
         serie = numerador/denominador;
      }
      else{
         denominador = denominador + 2;
         serie = numerador/denominador;
      }
   acumulador = serie * acumulador;

   }

   pi_aprox = 2*acumulador;         //Es porque se iguala a pi/2 por lo que para obtener pi, se le multi
plica por 2
   cout.precision(15);
   cout << "Pi aproximado segun wallis es: " << pi_aprox << endl;

}
```

```
1: ////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreo
7: //
8: ////////////////////////////////////////////////////////////////////////+
9:
10:    /*[Aproximación de PI por Madhava sin usar pow] En el siglo XIV el matemático indio
11:    Madhava of Sangamagrama calculó el arco tangente a través de un desarrollo de
12:    Taylor (este tipo de desarrollos se ve en la asignatura de Cálculo)
13:
14:    arctan(x) = ((-1)^i•x^(2i+1))/(2i+1)
15:
16:    Usando como x el valor 1, obtenemos la serie de Leibniz vista en el ejercicio 26:
17:
18:    arctan(x)= pi/4 = ((-1)^i)/(2i+1)
19:
20:    Usando como x el valor 1/sqrt(3), obtenemos:
21:
22:    arctan(1/sqrt(3))= pi/6 = ((-1)^i•(1/sqrt(3))^(2i+1))/(2i+1)
23:
24:    Por lo tanto, podemos usar la siguiente aproximación:
25:
26:    pi/6 = ((-1)^i•(1/sqrt(3))^(2i+1))/(2i+1)
27:
28:       Construya un programa que lea el valor tope obligando a que esté entre 1 y cien
29:    mil, calcule la aproximación de PI mediante la anterior serie e imprima el resultado en
30:    pantalla.
31:    Importante: En la implementación de esta solución NO puede usar la función pow ni
32:    ningún condicional if. Se le pide expresamente que para el cómputo de cada término,
33:    intente aprovechar los cómputos realizados en la iteración anterior.
34:    Ejemplo de entrada: 1000 -- Salida correcta: 3.14159265358979
35:    Ejemplo de entrada: 100000 -- Salida correcta: 3.14159265358979
36:    */
37:
38: #include <iostream>
39: #include <cmath>
40:
41: using namespace std;
42:
43: int main(){
44:    int tope;
45:    int i=0;
46:    int signo = 1;
47:    double numerador,denominador;
48:    double serie = 0;
49:    double pi_aprox;
50:    const int CAMBIO_SIGNO=-1;
51:    const double raiz= (1/sqrt(3));
52:
53:    do{
54:    cout << "Introduce el tope: ";
55:    cin >> tope;
56:    }
57:    while(tope<0 || tope>1e+5);
58:
59:    numerador= raiz;
60:    denominador=1.0;
61:
62:    for(i=0;i<=tope;i++){
63:
64:          serie += signo*numerador/denominador;
65:
66:          signo *= CAMBIO_SIGNO;
67:          denominador+= 2;
68:          numerador*=raiz*raiz;
69:    }
70:    pi_aprox= 6*serie;
71:
72:    cout.precision(15);
73:    cout << "La aproximacion de PI por madhava es: " << pi_aprox << endl;
74: }
75:
76:
```

```
1: ////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreo
7: //
8: ////////////////////////////////////////////////////////////////////////
9:
10: /* ENUNCIADO:
11:    29. [Secuencia de temperaturas] Construya un programa que calcule cuándo se produjo
12:    la mayor secuencia de días consecutivos con temperaturas crecientes. El programa
13:    leerá una secuencia de reales representando temperaturas. Una temperatura es correcta
14:    si su valor está en el intervalo [-90, 60] (los extremos representan las temperaturas
15:    extremas registradas en la Tierra). La entrada de datos terminará cuando
16:    se introduzca una temperatura fuera del rango anterior. El programa debe calcular la
17:    subsecuencia de números ordenada, de menor a mayor, de mayor longitud.
18:    El programa nos debe decir la posición donde comienza la subsecuencia y su longitud.
19:    Por ejemplo, ante la entrada siguiente:
20:
21:    17.2 17.3 16.2 16.4 17.1 19.2 18.9 100
22:
23:    El programa nos debe indicar que la mayor subsecuencia empieza en la posición 3 (en
24:    el 16.2) y tiene longitud 4 (termina en 19.2)
25:    Considere los siguientes consejos:
26:       -Tendrá que leer sobre dos variables anterior y actual, para así poder comparar
27:       el valor actual con el anterior.
28:       -Se recomienda que use la técnica de lectura anticipada, por lo que tendrá que
29:       leer un primer valor y comprobar si está en el rango adecuado:
30:
31:    cin >> anterior;
32:    final_entrada_datos = anterior < MIN_TEMP
33:                                   ||
34:                                   anterior > MAX_TEMP;
35:    while (! final_entrada_datos){
36:       cin >> actual;
37:       ......
38:    }
39:
40:    Dentro del cuerpo del bucle tendrá que comparar el valor actual con los extremos
41:    del rango de temperaturas, tal y como se hizo antes de entrar al bucle
42:    con el valor anterior. Esto hace que repitamos un código muy parecido. Lo
43:    resolveremos cuando veamos las funciones.
44:    Ejemplo de entrada: 17.2 17.3 16.2 16.4 17.1 19.2 18.9 100
45:       -Salida correcta: Inicio: 3 Longitud: 4
46:    Ejemplo de entrada: 17.2 17.3 16.2 16.4 17.1 19.2 100
47:       -Salida correcta: Inicio: 3 Longitud: 4
48:    Ejemplo de entrada: 17.2 17.3 100
49:       -Salida correcta: Inicio: 1 Longitud: 2
50:    Ejemplo de entrada: 17.2 15.3 100
51:       -Salida correcta: Inicio: 2 Longitud: 1
52:    Ejemplo de entrada: 17.2 100
53:       -Salida correcta: Inicio: 1 Longitud: 1
54:    Ejemplo de entrada: 100
55:       -Salida correcta: Inicio: 1 Longitud: 0
56:    Finalidad: Trabajar con bucles que comparan un valor actual con otro anterior. Dificultad
57:    Media.
58:    */
59:
60: #include <iostream>
61: #include <cmath>
62:
63: using namespace std;
64:
65: int main(){
66:    /*Intervalo válido de temperatura [-90,60]
67:    Límite de registro de temperaturas de 100
68:    El terminador será true y false en función del intervalo [-90,60]
69:    */
70:    double temperatura;
71:    const int MIN_TEMP=-90, MAX_TEMP=60;
72:    const int LIM_TEMP=100;
73:    double registro_temp[LIM_TEMP];
74:    bool terminador;
75:    double minimo=MAX_TEMP;
76:
77:
78: //Entrada de datos...
79:
80:    cout << "Introduce las temperaturas: ";
81:
82:    /*sale del bucle cuando el numero de temperaturas excede el limite del vector o cuando una temperatur
a sale del rango [-90,60]
83:    Cada vez que metemos una temperatura, comprobamos que sea valida([-90,60]) con el bool terminador
84:    */
85:    int i=0;
```

```
   86:    int contador=-1;
   87:    int posicion;
   88:
   89:    while(contador<=LIM_TEMP && terminador==false){
   90:       cin >> temperatura;
   91:       terminador = (temperatura< MIN_TEMP || temperatura > MAX_TEMP);
   92:       registro_temp[i]=temperatura;
   93:       contador++;                        //Cuenta el numero de temperaturas para guardarlo en la variable
util después
   94:       i++;
   95:       if(temperatura<minimo){            //con el condicional guardamos la menor temperatura y su posicion
 gracias al contador
   96:          minimo = temperatura;
   97:          posicion = contador + 1 ;
   98:       }
   99:    }
  100:
  101:
  102: //Cálculo de datos...
  103:    /*guardamos el valor del contador en la variable util*/
  104:
  105:    int util = contador;
  106:    int longitud=1;
  107:    int anterior, actual;
  108:    //anterior y actual son los valores que utilizamos para comparar las variables del vector(en el bucle
 posterior)
  109:
  110:    //si util es 0 es porque hemos metido una temperatura invalida y entonces su longitud y posicion son
0
  111:    if(util>0){
  112:       for(i=posicion;i<=util;i++){
  113:          anterior=registro_temp[i];
  114:          actual=registro_temp[i+1];
  115:          if(anterior<actual){
  116:             longitud++;
  117:          }
  118:       }
  119:    }
  120:    else
  121:    {
  122:       longitud=0;
  123:       posicion=0;
  124:    }
  125:
  126: //Salida de datos...
  127:
  128:    cout << endl << "Inicio: " << posicion << " Longitud: " << longitud << endl;
  129:
  130:
  131:
  132:
  133: }
  134:
  135:
```

```cpp
////////////////////////////////////////////////////////////////////////
//
// Fundamentos de Programación
// ETS Informática y Telecomunicaciones
// Universidad de Granada
// // Don Oreo
//
////////////////////////////////////////////////////////////////////////

/* 2. [Divisores] Recupere la solución del ejercicio 15 [Divisores de un entero] de la Relación
de Problemas II que puede encontrar en el siguiente enlace:
http://decsai.ugr.es/jccubero/FP/II_Divisores.cpp
Modifíquelo para separar los cómputos de las entradas y salidas de datos. Para ello,
se pide que cada vez que encuentre un divisor lo guarde en un vector divisores.
Una vez construido el vector, en un bucle aparte, debe imprimir sus componentes en
pantalla.
Ejemplo de entrada: 16 -- Salida correcta: 2, 4, 8
*/

#include <iostream>
using namespace std;

int main(){
   int entero, ultimo_divisor,divisor,util;
   const int LIMITE_DIVISORES=1000;
   int divisores[LIMITE_DIVISORES];

   cout << "Divisores de un entero\n\n";

//Entrada de datos...

   do{
      cout << "Introduce un numero entero mayor estricto que 0: ";
      cin >> entero;
   }while (entero <= 0);

//Cálculo de datos...

   ultimo_divisor = entero / 2;
   int i=0;

   for(divisor=2;divisor <= ultimo_divisor;divisor++){
      if(entero%divisor==0){
         divisores[i]= divisor;
         i++;
      }
   }
   util=i;

//Salida de datos...
   cout << "Los divisores son: ";
   for(int i=0;i<util;i++)
      cout << divisores[i] << " ";




}
```

```
1: ////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreo
7: //
8: ////////////////////////////////////////////////////////////////////////
9:
10:
11: /*[Mayor nota media] (Examen Prácticas Noviembre 2019)
12: Se quiere calcular la máxima nota media de evaluación continua de un conjunto de
13: alumnos. Para ello, se anota en un fichero un número entero con el código del alumno
14: y las notas que ha conseguido. El número de notas puede variar de un alumno a otro,
15: por lo que se terminará la introducción de las notas con un -1. La entrada de datos
16: finaliza con el código de alumno 0.
17: Cree un programa que lea las notas desde la entrada por defecto, y calcule el alumno
18: con mayor nota media. Puede suponer que los datos de entrada son siempre correctos.
19: Por ejemplo, para el siguiente registro de entradas, el alumno con máxima nota es el
20: que tiene identificador 17 con una nota media de 9.5
21:
22: 11    8 7 6 -1
23: 14    3 -1
24: 7     9 9 8 7 -1
25: 17    10 9 -1
26: 8     9 9 -1
27: 15    6 7 5 -1
28: 5     8 -1
29: 0
30:
31: */
32:
33: #include <iostream>
34: #include <cmath>
35:
36: using namespace std;
37:
38: int main(){
39:
40:    int identificador=1;
41:    int i;
42:    const int terminador_identificador=0;
43:    const int terminador_notas=-1;
44:    double notas;
45:    double acumulador;
46:    double media;
47:    double mejor_alumno=0;
48:    double mejor_nota=0;
49:
50:    cout << "Introduce el identificador del alumno: ";
51:    cin >> identificador;
52:
53:    while(identificador!= terminador_identificador){
54:       //inicializamos los datos
55:       acumulador=0.0;
56:       i=-1;
57:       notas=1.0;
58:
59:       cout << "Introduce las notas del alumno: ";
60:       cin >> notas;
61:
62:       while(notas!=terminador_notas){
63:          cin >> notas;
64:          acumulador+= notas;
65:          i++;
66:       }
67:
68:       media = acumulador/i;
69:       cout << identificador << "\t" << media << endl;
70:
71:       if(media>mejor_nota){
72:          mejor_nota=media;
73:          mejor_alumno=identificador;
74:       }
75:       cout << "Introduce el identificador del alumno: ";
76:       cin >> identificador;
77:    }
78:    cout << "---------------------------------------" << endl;
79:    cout << mejor_alumno << "\t" << mejor_nota << endl;
80: }
81:
82:
83:
84:
85:
```