```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int main()
{
    cout << "Este programa calcula el alcance de un proyectil: ";

    const long double PI = 3.1415;
    const long double g = 9.806;

    double vel_lanz, angulo;
    double tiempo_vuelo, h_max, alcance;

    cout << "\nIntroduzca velocidad de lanzamiento(m/s): ";
    cin >> vel_lanz;
    cout << "\nIntroduzca angulo de elevacion en grados (0-90): ";
    cin >> angulo;
    const double angulo_deg = angulo*PI/180;                //La funcion sin está en radianes, por lo qu
e hay pasarla a grados.
    tiempo_vuelo = (2*vel_lanz*sin(angulo_deg))/g;

    h_max = ((vel_lanz*sin(angulo_deg))*(vel_lanz*sin(angulo_deg)))/(2*g);

    alcance = (vel_lanz*vel_lanz*sin(2*angulo_deg))/g;

     cout << "Tiempo de vuelo:" << tiempo_vuelo  << " s" << endl
          << "Altura maxima:" << h_max << " metros" << endl
          << "Alcance:" << alcance << " metros" << endl;

}
```

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

#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int main()
{
   const long double PI = 3.1415;
   const long double g = 9.806;

   int n, x0, y0, vel, elev, rot;                          //valores de entrada
   double tiempo_vuelo, h_max, alcance, xf, yf;            //valores de calculo y salida

   cout << "Este programa calcula el alcance de un proyectil: " << endl;

//Entrada de datos...
   cout << "Introduce las coordenadas del canion(x0,y0): ";
   cin >> x0 >> y0;
   cout << "Introduce la velocidad de lanzamiento: ";
   cin >> vel;                                             //Expresado en m/s
   if(vel<0){
      cout << "Error: La elevacion debe ser positiva";
      exit(-1);
   }
   cout << "Introduce la elevacion del canion: ";
   cin >> elev;                                            //Expresado en grados
   if(elev>=90){
      cout << "Error: La elevacion debe estar entre 0-90 grados";
      exit(-1);
   }
   cout << "Introduce la rotacion del canion: ";
   cin >> rot;                                             //Angulo 0 (en grados) es igual al Norte(Eje
 Y)
   if(rot>180 || rot<-180){
      n=rot/180;                                           //truncamiento de decimales por ser variable
 entera
      rot = rot - 360*n;
      cout << "El angulo de rotacion rectificado es: " << rot << endl;
   }

   const double elev_deg =elev*PI/180;                     //La funcion sin está en radianes, por lo qu
e hay pasarla a grados.
   const double rot_deg = rot*PI/180;                      //***

//Cálculos...
   tiempo_vuelo = (2*vel*sin(elev_deg))/g;
   h_max = ((vel*sin(elev_deg))*(vel*sin(elev_deg)))/(2*g);
   alcance = (vel*vel*sin(2*elev_deg))/g;                  //Se repite vel a proposito para no tener qu
e usar pow(más tiempo de calculo)
   xf = x0 + alcance*sin(rot_deg);
   yf = y0 + alcance*cos(rot_deg);

//Salida de datos...

   cout << "----------Resultado de Disparo----------"
        << "\nTiempo de vuelo:" << tiempo_vuelo  << " s" << endl
        << "Altura maxima:" << h_max << " metros" << endl
        << "Alcance:" << alcance << " metros" << endl
        << "Impacto:" << xf <<"," << yf << " metros" << endl;

//-----------------------Exito de Disparo-------------------------------------------------------------
----

//Entrada de datos...

   cout << "----------Comprobacion de Objetivo----------";

   double x, y, d, d0, d1;

   cout << "\nIntroduzca la posicion del objetivo(x,y): ";
   cin >> x >> y;
   cout << "Introduzca distancias de impacto(dos valores): ";
   cin >> d0 >> d1;
   if(d0<0 || d1<0){
      cout << "Error: Las distancias de impacto no deben ser negativas";
      exit(-1);
   }
```

```
   82:
   83: //calculo de datos...
   84:
   85:    d = sqrt((x-xf)*(x-xf) + (y-yf)*(y-yf));            //Se repiten expresiones a proposito para no ten
er que usar pow(más tiempo de calculo)
   86:
   87: //Salida de datos...
   88:
   89:    cout << "La distancia al objetivo es: " << d;
   90:    if(d<d0 && d<d1)
   91:       cout << "---->Impacto Directo";
   92:    else if(d>d0 && d>d1)
   93:       cout << "---->No hay Impacto";
   94:    else
   95:       cout << "---->Impacto Parcial";
   96:    return 0;
   97: }
```

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

//Introduce la frecuencia y la temperatura e imprime las temperaturas en función de su frecuencia

#include <iostream>
#include <cmath>

using namespace std;

int main(){
   const int CENTINELA=0;
   const int TERMINADOR=-300;
   double frecuencia, temperatura;
   bool error;

   cin >> frecuencia;
   cin >> temperatura;

   while(frecuencia!=CENTINELA && error==false){

      for(int i=0; i<frecuencia; i++)
         cout << temperatura << " ";

      cin >> frecuencia;

      if(frecuencia!=trunc(frecuencia)){
         cout << "Error inesperado. Frecuencia decimal." << endl;
         error=true;
      }
      else if(frecuencia<0){
         cout << "Error inesperado. Frecuencia sin sentido." << endl;
         error=true;
      }
      else if(frecuencia!=CENTINELA)
         cin >> temperatura;
   }

   cout << TERMINADOR << " ";
}

```

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

//Introduce las temperaturas e imprime su frecuencia

#include <iostream>
#include <cmath>

using namespace std;

int main(){
   const double TERMINADOR = -273.15;
   const int CENTINELA = 0;
   double temperatura;
   double frecuencia = 1;        //Al menos se van a repetir 1 vez
   double copia;

   cin >> temperatura;
   copia = temperatura;

   while(temperatura > TERMINADOR){
      copia = temperatura;
      cin >> temperatura;

      while(temperatura == copia){
         frecuencia++;
         cin >> temperatura;
      }
      if(temperatura != copia){
         cout << frecuencia << " " << copia << " ";
         frecuencia=1;
      if(temperatura < TERMINADOR)
         cout << " " << CENTINELA << endl;
      }
   }

   return 0;

}

```

```
/**
 * @file ahorcado.cpp
 * @brief Programa para jugar al ahorcado en consola
 *
 * @author Yeray López Ramírez y Mario Guisado García
 * @date Enero-2020
 *
 * El programa implementa una versión básica del juego del ahorcado como
 * ejercicio de uso de clases simples.
 *
 * El objetivo que se pretende es programar el juego en base a una clase "Ahorcado"
 * que resuelva las operaciones que se tienen que realizar. El programa principal
 * deberá declarar un objeto de esta clase y llamar a los distintos métodos hasta
 * el final de la partida.
 *
 * Para hacerse una idea de qué componentes formarán parte de la clase:
 *
 * - Un objeto de esta clase, deberá tener información sobre:
 *    - La frase o palabra que hay que acertar.
 *    - La plantilla con el estado actual de la partida (la palabra con guiones).
 *    - El número de intentos que quedan.
 *    - Las letras que ya se han dicho hasta el momento.
 *
 * - Para que la clase controle el proceso de la partida, deberá tener operaciones para:
 *    - Consultar el estado actual de la plantilla (la cadena de letras/guiones actual)
 *    - Consultar el número de intentos que quedan.
 *    - Consultar la cadena de letras que ya se han usado.
 *    - Saber si la partida se ha terminado.
 *    - Saber si una letra se puede decir o no.
 *    - Saber si se ha ganado la partida.
 *    - Procesar una nueva letra seleccionada por el jugador, teniendo en cuenta que si
 *      se da en mayúscula también debe funcionar. La función devolverá si se ha procesado
 *      correctamente, es decir, si la letra tenía sentido y no se había usado antes. Por
 *      ejemplo, si le da un carácter que no es una letra no se puede procesar.
 *
 * Para hacer el programa más interesante, el juego debería "inventarse" una palabra. Para
 * resolver esto, creamos una clase con la responsabilidad de seleccionar una palabra
 * aleatoria. El diseño que se propone es crear una nueva clase "Bolsa" que nos hace de
 * generador aleatorio.
 *
 * Para hacerse una idea de qué componenetes formarán parte de la clase, tenga en cuenta
 * que deberá tener múltiples palabras y nos debería permitir "sacar palabras" en un orden
 * arbitrario. Para ello, puede considerar
 *     - Deberá contener un vector privado con las palabras que hay en la bolsa.
 *     - El constructor debería cargar ese vector privado con múltiples palabras en un
 *       orden aleatorio.
 *     - Debería tener un método para seleccionar una nueva palabra.
 *
 * Ya que es una bolsa, podemos realizar el siguiente comportamiento:
 *     - Cuando se declara un objeto de la bolsa, se cargan las palabras y se barajan.
 *     - Se puede pedir la siguiente palabra, dando lugar a una secuencia de palabras que
 *       surgen con un orden aleatorio según hayan quedado ordenadas al construir la bolsa.
 *     - Si se llegan a pedir todas las palabras, pedir la siguiente palabra implicará volver
 *       a barajar la bolsa y comenzar con la primera de ellas.
 *
 * Para simplificar el problema sin entrar en soluciones que impliquen pedir palabras desde
 * cin, puede declarar un vector con un contenido predeterminado en el constructor y que nos
 * permite inicializar la bolsa. Si quiere, puede usar:
 *       "caballero",  "Dulcinea",   "historia",   "escudero",
 *       "rocinante",  "adelante",   "gobernador", "andantes",
 *       "voluntad",   "capitulo",   "menester",   "doncella",
 *       "caballeria", "castillo",   "Fernando",   "finalmente",
 *       "aventura",   "hermosura",  "palabras",   "gobierno",
 *       "intencion",  "cardenio",   "pensamientos", "Luscinda",
 *       "lagrimas",   "aposento",   "aventuras",  "quisiera",
 *       "libertad",   "desgracia",  "entendimiento", "pensamiento",
 *       "licencia",   "Mercedes",   "semejantes", "silencio",
 *       "valeroso",   "doncellas",  "labrador",   "caballerias",
 *       "cristiano",  "cristianos", "discreto",   "hicieron",
 *       "llegaron",   "quisiere",   "espaldas",   "muestras",
 *       "escuderos",  "discurso",   "grandeza",   "altisidora",
 *       "princesa",   "haciendo",   "renegado",   "provecho",
 *       "quedaron",   "resolucion", "presente",   "encantadores",
 *       "enamorado",  "valiente",   "encantado",  "molino",
 *       "licenciado", "necesidad",  "responder",  "discrecion",
 *       "ejercicio",  "hacienda",   "posadero",   "Rocinante",
 *       "presencia",  "historias",  "presentes",  "verdadero"
 *
 * Observe que una vez que tenga las dos clases, puede declarar una bolsa de palabras y después
 * inicializar un objeto de la clase Ahorcado con una palabra aleatoria, ya que la palabra se pide
 * al objeto "Bolsa".
 *
 * Para programar el juego, puede definir la clase "Ahorcado" e inicializar el objeto con una palabra
 * fija y conocida (por ejemplo, en el constructor). Una vez que ya lo ha depurado y obtenido
 * una solución que funciona, puede añadir la clase bolsa y crear un programa que juega varias
 * partidas del ahorcado.
```

```
   87:    *
   88:    * En concreto, el programa pedirá cuántas palabras quiere adivinar y repetirá el juevo con un
   89:    * bucle que permita al usuario jugar varias partidas. Note que declarará una Bolsa al principio del
   90:    * main y el bucle que repite las partidas pedirá a dicha bolsa una nueva palabra para cada nueva parti
da.
   91:    *
   92:    */
   93:
   94: //Juego del ahorcado
   95:
   96: #include<iostream>
   97: #include<string>
   98: #include<cstdlib>
   99: #include <stdlib.h>
  100: #include <time.h>
  101:
  102: using namespace std;
  103:
  104: void Separador(){
  105:    cout << "\n";
  106:    for(int i = 0; i < 25; i++)
  107:       cout << "*";
  108: }
  109:
  110: class Ahorcado{
  111: private:
  112:
  113:         string inicial, palabra, car_introducidos;
  114:    int intentos = 5, tamanio;
  115:
  116: public:
  117:
  118:         void SeleccionarPalabra(string candidato){
  119:         palabra = candidato;
  120:         tamanio = palabra.size() + 1;
  121:    }
  122:
  123:    string Inicial(){
  124:       return inicial;
  125:    }
  126:
  127:    string Palabra(){
  128:       return palabra;
  129:    }
  130:    string Car_Introducidos(){
  131:       return car_introducidos;
  132:    }
  133:
  134:    int Intentos(){
  135:       return intentos;
  136:    }
  137:
  138:    void ReiniciarIntentos(){
  139:         intentos = 5;
  140:    }
  141:
  142:    string Inicio(){
  143:       for(int i = 1; i <= tamanio ; i++)
  144:          inicial.push_back('_');
  145:
  146:         return inicial;
  147:    }
  148:
  149:    void ImprimeInicial(){
  150:
  151:                cout << endl;
  152:
  153:                for(int i = 0; i <= tamanio; i++)
  154:                cout << inicial[i] << " ";
  155:
  156:                cout << endl;
  157:    }
  158:
  159:    bool LetrasIntroducidas(char car){
  160:
  161:           bool introducida;
  162:
  163:                if(car_introducidos.find(car) <= car_introducidos.size())
  164:         introducida = true;
  165:       else{
  166:          car_introducidos.push_back(tolower(car));
  167:          car_introducidos.push_back(toupper(car));
  168:       }
  169:    return introducida;
  170:    }
  171:
```

```
172:    bool EsLetra(char car){
173:       bool esletra;
174:       if  ((car >= 'A' && car <= 'Z') || (car >= 'a' && car <= 'z') )
175:          esletra = true;
176:       return esletra;
177:    }
178:
179:    void Ronda(char car){
180:
181:                bool encontrado = false, introducido = false, es_valido;
182:       int pos_letra = 0;
183:       es_valido = EsLetra(car);
184:
185:                if(es_valido)
186:          introducido = LetrasIntroducidas(car);
187:
188:                while(pos_letra < tamanio && es_valido){
189:          if(tolower(car) == palabra[pos_letra] || toupper(car) == palabra[pos_letra]){
190:             inicial[pos_letra] = palabra[pos_letra];
191:             encontrado = true;
192:          }
193:          pos_letra++;
194:       }
195:
196:                if(!encontrado && !introducido && es_valido){
197:          intentos--;
198:          cout << "No está!";
199:       }
200:       if(!es_valido)
201:          cout << "No es una letra!";
202:       if(introducido)
203:          cout << "La letra ya ha sido introducida";
204:    }
205:
206:    bool Victoria(){
207:       bool win = true;
208:       for(int i = 0; i < tamanio && win; i++)
209:          if(inicial[i] != palabra[i])
210:             win = false;
211:
212:       if(win)
213:          cout << endl << "HAS GANADO!";
214:
215:    return win;
216:    }
217:
218:    bool Derrota(){
219:       bool game_over;
220:
221:       if(intentos == 0){
222:          game_over = true;
223:          cout << endl << "HAS MUERTO" << endl;
224:       }
225:    return game_over;
226:    }
227:
228: };
229:
230: class Bolsa{
231:
232: private:
233:         string palabras[76] =  { "Dulcinea",   "historia",   "escudero",
234:                  "rocinante",  "adelante",   "gobernador", "andantes",
235:       "voluntad",   "capitulo",   "menester",   "doncella",
236:        "caballeria", "castillo",   "Fernando",   "finalmente",
237:        "aventura",   "hermosura",  "palabras",   "gobierno",
238:        "intencion",  "cardenio",   "pensamientos", "Luscinda",
239:        "lagrimas",   "aposento",   "aventuras",  "quisiera",
240:        "libertad",   "desgracia",  "entendimiento", "pensamiento",
241:       "licencia",   "Mercedes",   "semejantes", "silencio",
242:        "valeroso",   "doncellas",  "labrador",   "caballerias",
243:       "cristiano",  "cristianos", "discreto",   "hicieron",
244:        "llegaron",   "quisiere",   "espaldas",   "muestras",
245:        "escuderos",  "discurso",   "grandeza",   "altisidora",
246:       "princesa",   "haciendo",   "renegado",   "provecho",
247:        "quedaron",   "resolucion", "presente",   "encantadores",
248:        "enamorado",  "valiente",   "encantado",  "molino",
249:       "licenciado", "necesidad",  "responder",  "discrecion",
250:       "ejercicio",  "hacienda",   "posadero",   "rocinante",
251:       "presencia",  "historias",  "presentes",  "verdadero" };
252:
253:         string palabra_seleccionada;
254:         int numero_aleatorio;
255:
256: public:
257:
```

```
258:         void PalabraAleatoria(){
259:                 numero_aleatorio = rand () % 75;
260:                 palabra_seleccionada = palabras[numero_aleatorio];
261:         }
262:
263:         string GetPalabra(){
264:                 return palabra_seleccionada;
265:         }
266:
267: };
268:
269: int main(){
270:
271:         bool victoria = false, derrota = false;
272:    char car;
273:
274:    cout << "\nJuego del Ahorcado." << endl;
275:
276:    Ahorcado juego1;
277:         Bolsa bolsa1;
278:
279:         int partidas;
280:
281:         cout << "\nIntroduzca el número de partidas que desea jugar: ";
282:         cin >> partidas;
283:
284:         srand(time(NULL));
285:
286:         for(int i = 0; i < partidas; i++){
287:
288:                 bolsa1.PalabraAleatoria();
289:                 juego1.SeleccionarPalabra(bolsa1.GetPalabra()), juego1.Inicio(), juego1.ImprimeInicial()
;
290:
291:                 while(!victoria && !derrota){
292:
293:       cout << "\n\nQuedan " << juego1.Intentos() << " intentos";
294:       cout << endl << "\nIntroduce una letra: ";
295:       cin >> car;
296:
297:                 juego1.Ronda(car), juego1.ImprimeInicial();
298:
299:       victoria = juego1.Victoria();
300:       derrota = juego1.Derrota();
301:
302:       Separador();
303:         }
304:
305:         if (derrota)
306:                 i = partidas;
307:         else{
308:                 juego1.ReiniciarIntentos();
309:                 victoria = false;
310:         }
311:
312:    }
313:
314: }
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
```

```
343:
344:
345:
346:
```

```cpp
/*
  Autores: Don Oreo y el compa mamado
  Fecha: Enero-2020
 */

//Juego del ahorcado

#include<iostream>
#include<string>
#include <stdlib.h>
#include <time.h>

using namespace std;

void Separador(){
   cout << "\n";
   for(int i = 0; i < 25; i++)
      cout << "*";
}

//Crea un minijuego donde hay que introducir letras para acertar una palabra aleatoria

class Ahorcado{

/*
  El formato del juego es:

   _ _ _ _ _ _ //Cada raya representa una letra de la palabra a acertar

   Nº de intentos:_
*/

private:
        string inicial, palabra, car_introducidos;
   int intentos = 5, tamanio;

public:
   //Para quitar los "warnings". Esto no es estrictamente necesario hacerlo
   Ahorcado()
   :inicial(""),palabra(""),car_introducidos(""),tamanio(0)
   {
   }

   //Toma la palabra a acertar de la clase bolsa
        void SeleccionarPalabra(string candidato){
        palabra = candidato;
        tamanio = palabra.size();
   }

   //Inicia la partida
   string Inicio(){
      /*Inicializa todos las variables que se usaran en la partida
        Crea la plantilla inicial con los guiones necesarios
      */
      intentos = 5;
      inicial = "";
      car_introducidos = "";
      for(int i = 1; i <= tamanio ; i++)
         inicial.push_back('_');

        return inicial;
   }

   /*Metodos que devuelven los valores de los miembros de la clase*/

   string Inicial(){
      return inicial;
   }

   string Palabra(){
      return palabra;
   }

   string Car_Introducidos(){
      return car_introducidos;
   }

   int Intentos(){
      return intentos;
   }

   /*Metodos que imprimen el estado actual del juego*/

   void ImprimeInicial(int num_partida){
                cout << "\nPartida " << num_partida << endl;

```

```
87:       for(int i = 0; i < tamanio; i++)
88:          if(palabra[i] != ' ')
89:             cout << inicial[i] << " ";
90:          else
91:             cout << "\t";
92:
93:                 cout << endl;
94:    }
95:
96:    void ImprimeLetrasIntroducidas(){
97:       cout << car_introducidos << endl;
98:    }
99:
100:   /*Metodos para comprobar la validez del caracter introducido*/
101:
102:   bool LetrasIntroducidas(char car){
103:           bool introducida;
104:
105:                 if(car_introducidos.find(car) <= car_introducidos.size())
106:          introducida = true;
107:       else{
108:          car_introducidos.push_back(tolower(car));
109:          car_introducidos.push_back(toupper(car));
110:       }
111:
112:       return introducida;
113:    }
114:
115:    bool EsLetra(char car){
116:       bool esletra = false;
117:       if ((car >= 'A' && car <= 'Z') || (car >= 'a' && car <= 'z') || car == 'ñ'){
118:          esletra = true;
119:       }
120:
121:       return esletra;
122:    }
123:
124:    //Actualiza el estado de la partida segun el caracter introducido por el jugador
125:    void Ronda(char car){
126:                 bool encontrado = false, introducido = false, es_valido;
127:       int pos_letra = 0;
128:       es_valido = EsLetra(car);
129:
130:                 if(es_valido)
131:          introducido = LetrasIntroducidas(car);
132:
133:                 while(pos_letra < tamanio && es_valido){
134:          if(tolower(car) == palabra[pos_letra] || toupper(car) == palabra[pos_letra]){
135:             inicial[pos_letra] = palabra[pos_letra];
136:             encontrado = true;
137:          }
138:          pos_letra++;
139:       }
140:
141:       //En nuestro caso, NO penalizaremos si la letra ya habia sido introducida o no era válida
142:                 if(es_valido && !encontrado && !introducido){
143:          intentos--;
144:          cout << "No está!";
145:       }
146:
147:       if(es_valido && introducido)
148:          cout << "La letra ya ha sido introducida";
149:
150:       if(!es_valido)
151:          cout << "No es una letra!";
152:
153:    }
154:
155:    /*Métodos que comprueban el estado final de la partida*/
156:
157:    bool Victoria(){
158:       bool win = true;
159:
160:       for(int i = 0; i < tamanio && win; i++){
161:          if(palabra[i] == ' ')
162:             i++;
163:          if(inicial[i] != palabra[i])
164:             win = false;
165:       }
166:
167:    return win;
168:    }
169:
170:    bool Derrota(){
171:       bool game_over = false;
172:
```

```
173:       if(intentos == 0)
174:          game_over = true;
175:
176:
177:    return game_over;
178:    }
179:
180: };
181:
182: class Bolsa{
183:
184: private:
185:         static const int TAMANIO = 76;
186:         string palabras[TAMANIO] = {
187:            "caballero",  "Dulcinea",   "historia",   "escudero",
188:                 "Rocinante",  "adelante",   "gobernador", "andantes",
189:       "voluntad",   "capitulo",   "menester",   "doncella",
190:       "caballeria", "castillo",   "Fernando",   "finalmente",
191:       "aventura",   "hermosura",  "palabras",   "gobierno",
192:       "intencion",  "cardenio",   "pensamientos", "Luscinda",
193:       "lagrimas",   "aposento",   "aventuras",  "quisiera",
194:       "libertad",   "desgracia",  "entendimiento", "pensamiento",
195:       "licencia",   "Mercedes",   "semejantes", "silencio",
196:       "valeroso",   "doncellas",  "labrador",   "caballerias",
197:       "cristiano",  "cristianos", "discreto",   "hicieron",
198:       "llegaron",   "quisiere",   "espaldas",   "muestras",
199:       "escuderos",  "discurso",   "grandeza",   "altisidora",
200:       "princesa",   "haciendo",   "renegado",   "provecho",
201:       "quedaron",   "resolucion", "presente",   "encantadores",
202:       "enamorado",  "valiente",   "encantado",  "molino",
203:       "licenciado", "necesidad",  "responder",  "discrecion",
204:       "ejercicio",  "hacienda",   "posadero",   "rocinantes",
205:       "presencia",  "historias",  "presentes",  "verdadero"
206:       };
207:
208: /*   Nota:Rocinante = rocinante en nuestro codigo ya que segun la linea 31-32, el codigo debe aceptar ma
yusculas y minusculas
209:      Por tanto, hemos cambiado Rocinante por rocinantes para evitar una repeticion inesperada de esta pa
labra
210: */
211:
212:         string vector_aleatorio[TAMANIO];
213:         int num_generados[TAMANIO];
214:
215: public:
216:    Bolsa()
217:    //Barajamos la bolsa cada vez que creamos un objeto bolsa
218:    {
219:       Barajar();
220:    }
221:
222:    //Genera un vector aleatorio con las palabras de la bolsa
223:         void Barajar(){
224:            int num_aleatorio;
225:       bool ya_generado = true;
226:
227:       srand(time(NULL)); //Inicializa la funcion random
228:
229:       for(int i = 0; i < TAMANIO; i++){
230:          while(ya_generado){
231:             ya_generado = false;
232:             num_aleatorio = rand () % TAMANIO;
233:             for(int j = 0; j < i+1 && !ya_generado; j++)
234:                if(num_aleatorio == num_generados[j])
235:                   ya_generado = true;
236:          }
237:          vector_aleatorio[i] = palabras[num_aleatorio];
238:          num_generados[i] = num_aleatorio;
239:          ya_generado = true;
240:       }
241:       //Muestra el vector aleatorio en pantalla. Solo uso de depuracion
242:       //for( int i = 0; i < TAMANIO; i++)
243:       //   cout << vector_aleatorio[i] << endl;
244:         }
245:
246:         //Devuelve una palabra del vector aleatorio segun la posicion
247:         string Seleccionar_palabra(int pos){
248:            while(pos>=TAMANIO)
249:          pos-=TAMANIO-1;
250:
251:                return vector_aleatorio[pos];
252:         }
253:
254:         int Tamanio(){
255:       return TAMANIO;
256:         }
```

```
  257:
  258:
  259: };
  260:
  261: int main(){
  262:
  263:         bool victoria = false, derrota = false;
  264:         char confirmacion, car;
  265:
  266:    cout << "\nJuego del Ahorcado." << endl;
  267:
  268:    Ahorcado juego;
  269:    Bolsa bolsa;
  270:
  271:         int partidas;
  272:
  273:         cout << "\nIntroduzca el número de partidas que desea jugar: ";
  274:         cin >> partidas;
  275:
  276:         for(int i = 0; i < partidas && !derrota; i++){
  277:       victoria = false;
  278:
  279:                 juego.SeleccionarPalabra(bolsa.Seleccionar_palabra(i)),juego.Inicio(), juego.ImprimeInic
ial(i);
  280:
  281:                 while(!victoria && !derrota){
  282:       cout << "\n\nQuedan " << juego.Intentos() << " intentos";
  283:       cout << endl << "\nIntroduce una letra: ";
  284:       cin >> car;
  285:
  286:                 juego.Ronda(car), juego.ImprimeInicial(i);
  287:
  288:       victoria = juego.Victoria();
  289:       derrota = juego.Derrota();
  290:
  291:       Separador();
  292:         }
  293:       if(victoria)
  294:          cout << endl << "HAS GANADO!" << endl;
  295:
  296:         if (derrota){
  297:                 cout << endl << "HAS PERDIDO!" << endl;
  298:                 cout << endl << "La palabra era: " << juego.Palabra() << endl;
  299:         }
  300:
  301:       if(i % (bolsa.Tamanio() - 1) == 0 && i !=0){
  302:          //Si bolsa es recorrida completamente tienes dos opciones:
  303:          //-Seguir jugando barajando la bolsa otra ver
  304:          //-Salir del juego
  305:
  306:          cout << "\n¿Quieres seguir jugando? Se volverá a barajar la bolsa con las mismas palabras\n"
  307:               << "Escribe Y/y para seguir, N/n para salir." << endl;
  308:
  309:          while(confirmacion != 'y' && confirmacion != 'n'){
  310:             cin >> confirmacion;
  311:             confirmacion = tolower(confirmacion);
  312:          }
  313:
  314:          if(confirmacion == 'y')
  315:             bolsa.Barajar();
  316:          else
  317:             derrota = true;
  318:
  319:          confirmacion = ' ';
  320:       }
  321:    }
  322:
  323:    return 0;
  324: }
```

```
/**
 * @file ahorcado.cpp
 * @brief Programa para jugar al ahorcado en consola
 *
 * @author Fulanito...
 * @date Enero-2020
 *
 * El programa implementa una versión básica del juego del ahorcado como
 * ejercicio de uso de clases simples.
 *
 * El objetivo que se pretende es programar el juego en base a una clase "Ahorcado"
 * que resuelva las operaciones que se tienen que realizar. El programa principal
 * deberá declarar un objeto de esta clase y llamar a los distintos métodos hasta
 * el final de la partida.
 *
 * Para hacerse una idea de qué componentes formarán parte de la clase:
 *
 * - Un objeto de esta clase, deberá tener información sobre:
 *    - La frase o palabra que hay que acertar.
 *    - La plantilla con el estado actual de la partida (la palabra con guiones).
 *    - El número de intentos que quedan.
 *    - Las letras que ya se han dicho hasta el momento.
 *
 * - Para que la clase controle el proceso de la partida, deberá tener operaciones para:
 *    - Consultar el estado actual de la plantilla (la cadena de letras/guiones actual)
 *    - Consultar el número de intentos que quedan.
 *    - Consultar la cadena de letras que ya se han usado.
 *    - Saber si la partida se ha terminado.
 *    - Saber si una letra se puede decir o no.
 *    - Saber si se ha ganado la partida.
 *    - Procesar una nueva letra seleccionada por el jugador, teniendo en cuenta que si
 *      se da en mayúscula también debe funcionar. La función devolverá si se ha procesado
 *      correctamente, es decir, si la letra tenía sentido y no se había usado antes. Por
 *      ejemplo, si le da un carácter que no es una letra no se puede procesar.
 *
 * Para hacer el programa más interesante, el juego debería "inventarse" una palabra. Para
 * resolver esto, creamos una clase con la responsabilidad de seleccionar una palabra
 * aleatoria. El diseño que se propone es crear una nueva clase "Bolsa" que nos hace de
 * generador aleatorio.
 *
 * Para hacerse una idea de qué componenetes formarán parte de la clase, tenga en cuenta
 * que deberá tener múltiples palabras y nos debería permitir "sacar palabras" en un orden
 * arbitrario. Para ello, puede considerar
 *     - Deberá contener un vector privado con las palabras que hay en la bolsa.
 *     - El constructor debería cargar ese vector privado con múltiples palabras en un
 *       orden aleatorio.
 *     - Debería tener un método para seleccionar una nueva palabra.
 *
 * Ya que es una bolsa, podemos realizar el siguiente comportamiento:
 *     - Cuando se declara un objeto de la bolsa, se cargan las palabras y se barajan.
 *     - Se puede pedir la siguiente palabra, dando lugar a una secuencia de palabras que
 *       surgen con un orden aleatorio según hayan quedado ordenadas al construir la bolsa.
 *     - Si se llegan a pedir todas las palabras, pedir la siguiente palabra implicará volver
 *       a barajar la bolsa y comenzar con la primera de ellas.
 *
 * Para simplificar el problema sin entrar en soluciones que impliquen pedir palabras desde
 * cin, puede declarar un vector con un contenido predeterminado en el constructor y que nos
 * permite inicializar la bolsa. Si quiere, puede usar:
 *       "caballero",  "Dulcinea",   "historia",   "escudero",
 *       "rocinante",  "adelante",   "gobernador", "andantes",
 *       "voluntad",   "capitulo",   "menester",   "doncella",
 *       "caballeria", "castillo",   "Fernando",   "finalmente",
 *       "aventura",   "hermosura",  "palabras",   "gobierno",
 *       "intencion",  "cardenio",   "pensamientos", "Luscinda",
 *       "lagrimas",   "aposento",   "aventuras",  "quisiera",
 *       "libertad",   "desgracia",  "entendimiento", "pensamiento",
 *       "licencia",   "Mercedes",   "semejantes", "silencio",
 *       "valeroso",   "doncellas",  "labrador",   "caballerias",
 *       "cristiano",  "cristianos", "discreto",   "hicieron",
 *       "llegaron",   "quisiere",   "espaldas",   "muestras",
 *       "escuderos",  "discurso",   "grandeza",   "altisidora",
 *       "princesa",   "haciendo",   "renegado",   "provecho",
 *       "quedaron",   "resolucion", "presente",   "encantadores",
 *       "enamorado",  "valiente",   "encantado",  "molino",
 *       "licenciado", "necesidad",  "responder",  "discrecion",
 *       "ejercicio",  "hacienda",   "posadero",   "Rocinante",
 *       "presencia",  "historias",  "presentes",  "verdadero"
 *
 * Observe que una vez que tenga las dos clases, puede declarar una bolsa de palabras y después
 * inicializar un objeto de la clase Ahorcado con una palabra aleatoria, ya que la palabra se pide
 * al objeto "Bolsa".
 *
 * Para programar el juego, puede definir la clase "Ahorcado" e inicializar el objeto con una palabra
 * fija y conocida (por ejemplo, en el constructor). Una vez que ya lo ha depurado y obtenido
 * una solución que funciona, puede añadir la clase bolsa y crear un programa que juega varias
 * partidas del ahorcado.
```

```
   87:    *
   88:    * En concreto, el programa pedirá cuántas palabras quiere adivinar y repetirá el juevo con un
   89:    * bucle que permita al usuario jugar varias partidas. Note que declarará una Bolsa al principio del
   90:    * main y el bucle que repite las partidas pedirá a dicha bolsa una nueva palabra para cada nueva parti
da.
   91:    *
   92:    */
```

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

//Mediana movil

#include <iostream>
using namespace std;

int main(){
  const float ceroKelvin = -273,15;
        const int TAMANIO = 1e6;
        double temperaturas[TAMANIO];
        double ordenados[TAMANIO];
        int posicion = 0, N = 3;
   double mediana;

        cout << "Introduzca las temperaturas: \n";
        cin >> temperaturas[posicion];


   //Lectura de Datos

        while(temperaturas[posicion] >= ceroKelvin){
                posicion++;
                cin >> temperaturas[posicion];
        }
   //Computo

   double a_insertar;
   int k;

 /*//Cálculo de la mediana
   Primero ordenamos las temperaturas y posteriormente hacemos su mediana*/

        for(int iter = 0; iter < posicion - (N-1); iter++){
      //asignamos los N temperaturas al vector ordenador
      for(int i=iter;i < N+iter; i++)
         ordenados[i] = temperaturas[i];
      for (int j = iter; j < N+iter; j++){
         a_insertar = ordenados[j];

         for (k = j; k > iter && a_insertar < ordenados[k-1]; k--)    // Ordenación de menor a mayor
            ordenados[k] = ordenados[k-1];

         ordenados[k] = a_insertar;
      }
   /*//Comprobar ordenacion
   cout << "El vector ordenado es: ";
   for (int j = iter; j < N+iter; j++)
      cout << ordenados[j] << " ";
   */
                mediana = 0;
                if(N % 2 != 0)
                        mediana = ordenados[iter+(N/2)];
                else
                        mediana = (ordenados[iter+(N/2)] + ordenados[iter+(N/2)-1]) / 2;

                cout << " " << mediana;
        }

        cout << " " << temperaturas[posicion];
}

```

```
/**
 * @file mediana_movil.cpp
 * @brief Calcula la mediana movil de una secuencia de temperaturas
 *
 * @author Fulanito...
 * @date Diciembre-2020
 *
 * Escriba un programa que procesa una secuencia de valores de temperatura hasta que
 * se introduce una temperatura menor que el cero absoluto  (-273.15 grados).
 *
 * Como resultado, escribirá una secuencia de datos que corresponde a la mediana móvil
 * con tamaño N. Cada valor de la secuencia de esta mediana móvil corresponde a:
 *
 *    - La mediana de los primeros N datos desde el 0 al N-1,
 *    - la mediana de los N siguientes desde el 1 al N,
 *    - la mediana de los N siguientes desde el 2 al N+1,
 *    - etc.
 *
 * Por tanto, si hay D datos, la secuencia resultado tendrá D-(N-1) valores. Esta
 * secuencia, además, estará también terminada con un valor centinela.
 *
 * El problema se puede resolver cargando toda la secuencia de datos y luego
 * calculando la mediana móvil para cada N datos, aunque también se podría limitar el
 * tamaño de la memoria ocupada evitando tener toda la secuencia, pues sólo es
 * necesario almacenar los últimos N valores.
 *
 * Nota: En el problema, puede suponer que N es fijo y tiene un valor
 * predeterminado. Así, evita tener que introducirlo; tanto la entrada como la
 * salida serán una simple secuencia.
 *
 * Un ejemplo de ejecución, con N valiendo 5, es:
 *     1 2 3 4 5 6 7 8 9 -300
 *     3 4 5 6 7 -300
 * donde la primera línea es la entrada y la segunda la salida.
 *
 * Otra ejemplo, ahora con N valiendo 5, es:
 *     -0.4 -0.5 -0.9 -0.6 -0.7 -0.1 0.9 0.7 0.2 1.2 1.5 -300
 *     -0.6 -0.6 -0.6 -0.1 0.2 0.7 0.9 -300
 *
 * El mismo ejemplo, ahora con N valiendo 3, es:
 *     -0.4 -0.5 -0.9 -0.6 -0.7 -0.1 0.9 0.7 0.2 1.2 1.5 -300
 *     -0.5 -0.6 -0.7 -0.6 -0.1 0.7 0.7 0.7 1.2 -300
 */
```