```
1: /*
2:   Autores: Don Oreo y el compa mamado
3:   Fecha: Enero-2020
4:  */
5:
6: //Juego del ahorcado
7:
8: #include<iostream>
9: #include<string>
10: #include <stdlib.h>
11: #include <time.h>
12:
13: using namespace std;
14:
15: void Separador(){
16:    cout << "\n";
17:    for(int i = 0; i < 25; i++)
18:       cout << "*";
19: }
20:
21: //Crea un minijuego donde hay que introducir letras para acertar una palabra aleatoria
22:
23: class Ahorcado{
24:
25: /*
26:   El formato del juego es:
27:
28:    _ _ _ _ _ _ //Cada raya representa una letra de la palabra a acertar
29:
30:    Nº de intentos:_
31: */
32:
33: private:
34:         string inicial, palabra, car_introducidos;
35:    int intentos = 5, tamanio;
36:
37: public:
38:    //Para quitar los "warnings". Esto no es estrictamente necesario hacerlo
39:    Ahorcado()
40:    :inicial(""),palabra(""),car_introducidos(""),tamanio(0)
41:    {
42:    }
43:
44:    //Toma la palabra a acertar de la clase bolsa
45:         void SeleccionarPalabra(string candidato){
46:         palabra = candidato;
47:         tamanio = palabra.size();
48:    }
49:
50:    //Inicia la partida
51:    string Inicio(){
52:       /*Inicializa todos las variables que se usaran en la partida
53:         Crea la plantilla inicial con los guiones necesarios
54:       */
55:       intentos = 5;
56:       inicial = "";
57:       car_introducidos = "";
58:       for(int i = 1; i <= tamanio ; i++)
59:          inicial.push_back('_');
60:
61:         return inicial;
62:    }
63:
64:    /*Metodos que devuelven los valores de los miembros de la clase*/
65:
66:    string Inicial(){
67:       return inicial;
68:    }
69:
70:    string Palabra(){
71:       return palabra;
72:    }
73:
74:    string Car_Introducidos(){
75:       return car_introducidos;
76:    }
77:
78:    int Intentos(){
79:       return intentos;
80:    }
81:
82:    /*Metodos que imprimen el estado actual del juego*/
83:
84:    void ImprimeInicial(int num_partida){
85:                  cout << "\nPartida " << num_partida << endl;
86:
```

```
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:    */
```