

```

1: #include <iostream>
2: #include <cmath>
3: #include <iomanip>
4: using namespace std;
5:
6: int main()
7: {
8:     cout << "Este programa calcula el alcance de un proyectil: ";
9:
10:    const long double PI = 3.1415;
11:    const long double g = 9.806;
12:
13:    double vel_lanz, angulo;
14:    double tiempo_vuelo, h_max, alcance;
15:
16:    cout << "\nIntroduzca velocidad de lanzamiento(m/s): ";
17:    cin >> vel_lanz;
18:    cout << "\nIntroduzca angulo de elevacion en grados (0-90): ";
19:    cin >> angulo;
20:    const double angulo_deg = angulo*PI/180;           //La funcion sin está en radianes, por lo qu
e hay pasarla a grados.
21:    tiempo_vuelo = (2*vel_lanz*sin(angulo_deg))/g;
22:
23:    h_max = ((vel_lanz*sin(angulo_deg))*(vel_lanz*sin(angulo_deg)))/(2*g);
24:
25:    alcance = (vel_lanz*vel_lanz*sin(2*angulo_deg))/g;
26:
27:    cout << "Tiempo de vuelo:" << tiempo_vuelo << " s" << endl
28:    << "Altura maxima:" << h_max << " metros" << endl
29:    << "Alcance:" << alcance << " metros" << endl;
30:
31: }
32:
33:

```

```

1: ///////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreó
7: //
8: ///////////////////////////////////////////////////////////////////
9:
10: #include <iostream>
11: #include <cmath>
12: #include <iomanip>
13: using namespace std;
14:
15: int main()
16: {
17:     const long double PI = 3.1415;
18:     const long double g = 9.806;
19:
20:     int n, x0, y0, vel, elev, rot;           //valores de entrada
21:     double tiempo_vuelo, h_max, alcance, xf, yf; //valores de calculo y salida
22:
23:     cout << "Este programa calcula el alcance de un proyectil: " << endl;
24:
25:     //Entrada de datos...
26:     cout << "Introduce las coordenadas del canon(x0,y0): ";
27:     cin >> x0 >> y0;
28:     cout << "Introduce la velocidad de lanzamiento: ";
29:     cin >> vel;                               //Expresado en m/s
30:     if(vel<0){
31:         cout << "Error: La elevacion debe ser positiva";
32:         exit(-1);
33:     }
34:     cout << "Introduce la elevacion del canon: ";
35:     cin >> elev;                               //Expresado en grados
36:     if(elev>=90){
37:         cout << "Error: La elevacion debe estar entre 0-90 grados";
38:         exit(-1);
39:     }
40:     cout << "Introduce la rotacion del canon: ";
41:     cin >> rot;                               //Angulo 0 (en grados) es igual al Norte(Eje
y)
42:     if(rot>180 || rot<=-180){
43:         n=rot/180;                             //truncamiento de decimales por ser variable
entera
44:         rot = rot - 360*n;
45:         cout << "El angulo de rotacion rectificado es: " << rot << endl;
46:     }
47:
48:     const double elev_deg =elev*PI/180;         //La funcion sin está en radianes, por lo qu
e hay pasarla a grados.
49:     const double rot_deg = rot*PI/180;         //***
50:
51:     //Cálculos...
52:     tiempo_vuelo = (2*vel*sin(elev_deg))/g;
53:     h_max = ((vel*sin(elev_deg))*(vel*sin(elev_deg)))/(2*g);
54:     alcance = (vel*vel*sin(2*elev_deg))/g;       //Se repite vel a proposito para no tener qu
e usar pow(más tiempo de calculo)
55:     xf = x0 + alcance*sin(rot_deg);
56:     yf = y0 + alcance*cos(rot_deg);
57:
58:     //Salida de datos...
59:
60:     cout << "-----Resultado de Disparo-----"
61:         << "\nTiempo de vuelo:" << tiempo_vuelo << " s" << endl
62:         << "Altura maxima:" << h_max << " metros" << endl
63:         << "Alcance:" << alcance << " metros" << endl
64:         << "Impacto:" << xf << ", " << yf << " metros" << endl;
65:
66:     //-----Exito de Disparo-----
-----
67:
68:     //Entrada de datos...
69:
70:     cout << "-----Comprobacion de Objetivo-----";
71:
72:     double x, y, d, d0, d1;
73:
74:     cout << "\nIntroduzca la posicion del objetivo(x,y): ";
75:     cin >> x >> y;
76:     cout << "Introduzca distancias de impacto(dos valores): ";
77:     cin >> d0 >> d1;
78:     if(d0<0 || d1<0){
79:         cout << "Error: Las distancias de impacto no deben ser negativas";
80:         exit(-1);
81:     }

```

```
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: }
```

```

1: ///////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreó
7: //
8: ///////////////////////////////////////////////////////////////////
9:
10: //Introduce la frecuencia y la temperatura e imprime las temperaturas en función de su frecuencia
11:
12: #include <iostream>
13: #include <cmath>
14:
15: using namespace std;
16:
17: int main() {
18:     const int CENTINELA=0;
19:     const int TERMINADOR=-300;
20:     double frecuencia, temperatura;
21:     bool error;
22:
23:     cin >> frecuencia;
24:     cin >> temperatura;
25:
26:     while(frecuencia!=CENTINELA && error==false){
27:
28:         for(int i=0; i<frecuencia; i++)
29:             cout << temperatura << " ";
30:
31:         cin >> frecuencia;
32:
33:         if(frecuencia!=trunc(frecuencia)){
34:             cout << "Error inesperado. Frecuencia decimal." << endl;
35:             error=true;
36:         }
37:         else if(frecuencia<0){
38:             cout << "Error inesperado. Frecuencia sin sentido." << endl;
39:             error=true;
40:         }
41:         else if(frecuencia!=CENTINELA)
42:             cin >> temperatura;
43:     }
44:
45:     cout << TERMINADOR << " ";
46: }
47:

```

```

1: //////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreó
7: //
8: //////////////////////////////////////
9:
10: //Introduce las temperaturas e imprime su frecuencia
11:
12: #include <iostream>
13: #include <cmath>
14:
15: using namespace std;
16:
17: int main(){
18:     const double TERMINADOR = -273.15;
19:     const int CENTINELA = 0;
20:     double temperatura;
21:     double frecuencia = 1;           //Al menos se van a repetir 1 vez
22:     double copia;
23:
24:     cin >> temperatura;
25:     copia = temperatura;
26:
27:     while(temperatura > TERMINADOR){
28:         copia = temperatura;
29:         cin >> temperatura;
30:
31:         while(temperatura == copia){
32:             frecuencia++;
33:             cin >> temperatura;
34:         }
35:         if(temperatura != copia){
36:             cout << frecuencia << " " << copia << " ";
37:             frecuencia=1;
38:             if(temperatura < TERMINADOR)
39:                 cout << " " << CENTINELA << endl;
40:         }
41:     }
42:
43:     return 0;
44:
45: }
46:

```

```

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:     N° de intentos:_
31:     */
32:
33: private:
34:     string inicial, palabra, car_introducidos;
35:     int intentos = 5, tamano;
36:
37: public:
38:     //Para quitar los "warnings". Esto no es estrictamente necesario hacerlo
39:     Ahorcado()
40:     :inicial(""),palabra(""),car_introducidos(""),tamano(0)
41:     {
42:     }
43:
44:     //Toma la palabra a acertar de la clase bolsa
45:     void SeleccionarPalabra(string candidato){
46:         palabra = candidato;
47:         tamano = 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 <= tamano ; 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 < tamano; 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 < tamano && 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 < tamano && 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.Tamano() - 1) == 0 && i != 0){
302:             //Si bolsa es recorrida completamente tienes dos opciones:
303:             //Seguir jugando barajando la bolsa otra vez
304:             //Salir del juego
305:
306:             cout << "\n¿Quieres seguir jugando? Se volverá a barajar la bolsa con las mismas palabras\n"
<< "Escribe Y/y para seguir, N/n para salir." << endl;
307:
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: }
```

```
1: /**
2:  * @file ahorcado.cpp
3:  * @brief Programa para jugar al ahorcado en consola
4:  *
5:  * @author Fulanito...
6:  * @date Enero-2020
7:  *
8:  * El programa implementa una versión básica del juego del ahorcado como
9:  * ejercicio de uso de clases simples.
10:  *
11:  * El objetivo que se pretende es programar el juego en base a una clase "Ahorcado"
12:  * que resuelva las operaciones que se tienen que realizar. El programa principal
13:  * deberá declarar un objeto de esta clase y llamar a los distintos métodos hasta
14:  * el final de la partida.
15:  *
16:  * Para hacerse una idea de qué componentes formarán parte de la clase:
17:  *
18:  * - Un objeto de esta clase, deberá tener información sobre:
19:  *   - La frase o palabra que hay que acertar.
20:  *   - La plantilla con el estado actual de la partida (la palabra con guiones).
21:  *   - El número de intentos que quedan.
22:  *   - Las letras que ya se han dicho hasta el momento.
23:  *
24:  * - Para que la clase controle el proceso de la partida, deberá tener operaciones para:
25:  *   - Consultar el estado actual de la plantilla (la cadena de letras/guiones actual)
26:  *   - Consultar el número de intentos que quedan.
27:  *   - Consultar la cadena de letras que ya se han usado.
28:  *   - Saber si la partida se ha terminado.
29:  *   - Saber si una letra se puede decir o no.
30:  *   - Saber si se ha ganado la partida.
31:  *   - Procesar una nueva letra seleccionada por el jugador, teniendo en cuenta que si
32:  *     se da en mayúscula también debe funcionar. La función devolverá si se ha procesado
33:  *     correctamente, es decir, si la letra tenía sentido y no se había usado antes. Por
34:  *     ejemplo, si le da un carácter que no es una letra no se puede procesar.
35:  *
36:  * Para hacer el programa más interesante, el juego debería "inventarse" una palabra. Para
37:  * resolver esto, creamos una clase con la responsabilidad de seleccionar una palabra
38:  * aleatoria. El diseño que se propone es crear una nueva clase "Bolsa" que nos hace de
39:  * generador aleatorio.
40:  *
41:  * Para hacerse una idea de qué componenetes formarán parte de la clase, tenga en cuenta
42:  * que deberá tener múltiples palabras y nos debería permitir "sacar palabras" en un orden
43:  * arbitrario. Para ello, puede considerar
44:  *   - Deberá contener un vector privado con las palabras que hay en la bolsa.
45:  *   - El constructor debería cargar ese vector privado con múltiples palabras en un
46:  *     orden aleatorio.
47:  *   - Debería tener un método para seleccionar una nueva palabra.
48:  *
49:  * Ya que es una bolsa, podemos realizar el siguiente comportamiento:
50:  *   - Cuando se declara un objeto de la bolsa, se cargan las palabras y se barajan.
51:  *   - Se puede pedir la siguiente palabra, dando lugar a una secuencia de palabras que
52:  *     surgen con un orden aleatorio según hayan quedado ordenadas al construir la bolsa.
53:  *   - Si se llegan a pedir todas las palabras, pedir la siguiente palabra implicará volver
54:  *     a barajar la bolsa y comenzar con la primera de ellas.
55:  *
56:  * Para simplificar el problema sin entrar en soluciones que impliquen pedir palabras desde
57:  * cin, puede declarar un vector con un contenido predeterminado en el constructor y que nos
58:  * permite inicializar la bolsa. Si quiere, puede usar:
59:  *   "caballero", "Dulcinea", "historia", "escudero",
60:  *   "rocinante", "adelante", "gobernador", "andantes",
61:  *   "voluntad", "capitulo", "menester", "doncella",
62:  *   "caballeria", "castillo", "Fernando", "finalmente",
63:  *   "aventura", "hermosura", "palabras", "gobierno",
64:  *   "intencion", "cardenio", "pensamientos", "Luscinda",
65:  *   "lagrimas", "aposento", "aventuras", "quisiera",
66:  *   "libertad", "desgracia", "entendimiento", "pensamiento",
67:  *   "licencia", "Mercedes", "semejantes", "silencio",
68:  *   "valeroso", "doncellas", "labrador", "caballerias",
69:  *   "cristiano", "cristianos", "discreto", "hicieron",
70:  *   "llegaron", "quisiere", "espaldas", "muestras",
71:  *   "escuderos", "discurso", "grandeza", "altisidora",
72:  *   "princesa", "haciendo", "renegado", "provecho",
73:  *   "quedaran", "resolucion", "presente", "encantadores",
74:  *   "enamorado", "valiente", "encantado", "molino",
75:  *   "licenciado", "necesidad", "responder", "discrecion",
76:  *   "ejercicio", "hacienda", "posadero", "Rocinante",
77:  *   "presencia", "historias", "presentes", "verdadero"
78:  *
79:  * Observe que una vez que tenga las dos clases, puede declarar una bolsa de palabras y después
80:  * inicializar un objeto de la clase Ahorcado con una palabra aleatoria, ya que la palabra se pide
81:  * al objeto "Bolsa".
82:  *
83:  * Para programar el juego, puede definir la clase "Ahorcado" e inicializar el objeto con una palabra
84:  * fija y conocida (por ejemplo, en el constructor). Una vez que ya lo ha depurado y obtenido
85:  * una solución que funciona, puede añadir la clase bolsa y crear un programa que juega varias
86:  * partidas del ahorcado.
```

```
87:  *
88:  * En concreto, el programa pedirá cuántas palabras quiere adivinar y repetirá el juego 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:  */
```

```

1: //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2: //
3: // Fundamentos de Programación
4: // ETS Informática y Telecomunicaciones
5: // Universidad de Granada
6: // // Don Oreó y el Compa Mamado
7: //
8: //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
9:
10: //Mediana movil
11:
12: #include <iostream>
13: using namespace std;
14:
15: int main() {
16:     const float ceroKelvin = -273,15;
17:     const int TAMANIO = 1e6;
18:     double temperaturas[TAMANIO];
19:     double ordenados[TAMANIO];
20:     int posicion = 0, N = 3;
21:     double mediana;
22:
23:     cout << "Introduzca las temperaturas: \n";
24:     cin >> temperaturas[posicion];
25:
26:
27:     //Lectura de Datos
28:
29:     while(temperaturas[posicion] >= ceroKelvin){
30:         posicion++;
31:         cin >> temperaturas[posicion];
32:     }
33:     //Computo
34:
35:     double a_insertar;
36:     int k;
37:
38:     /*//Cálculo de la mediana
39:     Primero ordenamos las temperaturas y posteriormente hacemos su mediana*/
40:
41:     for(int iter = 0; iter < posicion - (N-1); iter++){
42:         //asignamos los N temperaturas al vector ordenador
43:         for(int i=iter; i < N+iter; i++)
44:             ordenados[i] = temperaturas[i];
45:         for (int j = iter; j < N+iter; j++){
46:             a_insertar = ordenados[j];
47:
48:             for (k = j; k > iter && a_insertar < ordenados[k-1]; k--) // Ordenación de menor a mayor
49:                 ordenados[k] = ordenados[k-1];
50:
51:             ordenados[k] = a_insertar;
52:         }
53:         /*//Comprobar ordenacion
54:         cout << "El vector ordenado es: ";
55:         for (int j = iter; j < N+iter; j++)
56:             cout << ordenados[j] << " ";
57:         */
58:         mediana = 0;
59:         if(N % 2 != 0)
60:             mediana = ordenados[iter+(N/2)];
61:         else
62:             mediana = (ordenados[iter+(N/2)] + ordenados[iter+(N/2)-1]) / 2;
63:
64:         cout << " " << mediana;
65:     }
66:
67:     cout << " " << temperaturas[posicion];
68: }
69:

```

```
1: /**
2:  * @file mediana_movil.cpp
3:  * @brief Calcula la mediana movil de una secuencia de temperaturas
4:  *
5:  * @author Fulanito...
6:  * @date Diciembre-2020
7:  *
8:  * Escriba un programa que procesa una secuencia de valores de temperatura hasta que
9:  * se introduce una temperatura menor que el cero absoluto (-273.15 grados).
10: *
11: * Como resultado, escribirá una secuencia de datos que corresponde a la mediana móvil
12: * con tamaño N. Cada valor de la secuencia de esta mediana móvil corresponde a:
13: *
14: *   - La mediana de los primeros N datos desde el 0 al N-1,
15: *   - la mediana de los N siguientes desde el 1 al N,
16: *   - la mediana de los N siguientes desde el 2 al N+1,
17: *   - etc.
18: *
19: * Por tanto, si hay D datos, la secuencia resultado tendrá D-(N-1) valores. Esta
20: * secuencia, además, estará también terminada con un valor centinela.
21: *
22: * El problema se puede resolver cargando toda la secuencia de datos y luego
23: * calculando la mediana móvil para cada N datos, aunque también se podría limitar el
24: * tamaño de la memoria ocupada evitando tener toda la secuencia, pues sólo es
25: * necesario almacenar los últimos N valores.
26: *
27: * Nota: En el problema, puede suponer que N es fijo y tiene un valor
28: * predeterminado. Así, evita tener que introducirlo; tanto la entrada como la
29: * salida serán una simple secuencia.
30: *
31: * Un ejemplo de ejecución, con N valiendo 5, es:
32: *   1 2 3 4 5 6 7 8 9 -300
33: *   3 4 5 6 7 -300
34: * donde la primera línea es la entrada y la segunda la salida.
35: *
36: * Otra ejemplo, ahora con N valiendo 5, es:
37: *   -0.4 -0.5 -0.9 -0.6 -0.7 -0.1 0.9 0.7 0.2 1.2 1.5 -300
38: *   -0.6 -0.6 -0.6 -0.1 0.2 0.7 0.9 -300
39: *
40: * El mismo ejemplo, ahora con N valiendo 3, es:
41: *   -0.4 -0.5 -0.9 -0.6 -0.7 -0.1 0.9 0.7 0.2 1.2 1.5 -300
42: *   -0.5 -0.6 -0.7 -0.6 -0.1 0.7 0.7 0.7 1.2 -300
43: */
```