LENGUAJES DE PROGRAMACIÓN

Solución al Trabajo Práctico - Convocatoria ordinaria de 2025

Ejercicio 1

El vector posición de una partícula puntual que se mueve en el espacio se define de la manera siguiente:

$$\vec{r} = a \cdot \cos(w \cdot t) \cdot \vec{u}_x + b \cdot \operatorname{sen}(w \cdot t) \cdot \vec{u}_y + c \cdot t \cdot \vec{u}_z$$

donde $\vec{u}_x$, $\vec{u}_y$, $\vec{u}_z$ son los vectores unitarios de los ejes de coordenadas cartesianos X, Y, Z; t es una variable de tipo real que representa el tiempo; w, a, b, c son constantes conocidas de tipo real.

La longitud L de la trayectoria de la partícula entre los instantes t_A y t_B, satisfaciéndose $t_A < t_B$, puede calcularse de manera aproximada de la forma descrita a continuación.

1. Sea N un número entero mayor que cero.
2. Sean
$$t_i = t_A + i \cdot \frac{t_B - t_A}{N} \qquad \text{para} \quad i = 0, \ldots, N$$
instantes de tiempo en el intervalo $[t_A, t_B]$.
3. Sea $\vec{r}_i$ la posición de la partícula en el instante t_i.
4. Sea L_i la longitud del segmento recto que une la posición de la partícula en el instante t_i con la posición de la partícula en el instante t_{i+1}.

5. El valor aproximado de la longitud L se calcula de la forma siguiente:

$$L = \sum_{i=0}^{i=N-1} L_i$$

Escriba un programa en C++ que realice las acciones siguientes:

1. Declarar las constantes globales w, a, b, c asignándoles los valores siguientes: $w = 10$, $a = 1.4$, $b = 2$, $c = 0.3$.

2. Mostrar un mensaje en la consola solicitando al usuario que introduzca por consola el valor de N y los valores de los instantes t_A y t_B. Leer los valores introducidos por consola por el usuario.

3. Comprobar que se satisface $0 \leq t_A < t_B$ y $N > 0$. En caso contrario, mostrar un mensaje en la consola indicándolo y terminar.

4. Calcular y mostrar en la consola la longitud aproximada de la trayectoria de la partícula entre los instantes t_A y t_B, calculada como se ha indicado anteriormente, y escribirla en la consola en formato científico con cinco dígitos decimales.

5. Terminar.

Muestre en el informe el resultado de ejecutar su programa para los casos de prueba siguientes:

Caso de prueba 1:	$N = 100$,	$t_A = 0.72$,	$t_B = 1.48$
Caso de prueba 2:	$N = 1000$,	$t_A = 0.72$,	$t_B = 1.48$
Caso de prueba 3:	$N = 10000$,	$t_A = 0.72$,	$t_B = 1.48$
Caso de prueba Error 1:	$N = -1$,	$t_A = 0$,	$t_B = 1$
Caso de prueba Error 2:	$N = 10$,	$t_A = -1$,	$t_B = 1$
Caso de prueba Error 3:	$N = 10$,	$t_A = 2$,	$t_B = 1$

Solución al Ejercicio 1

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>

const double w = 10, a = 1.4, b = 2, c = 0.3;

int main() {
   // Entrada por consola
   int N;
   double t_A, t_B;
   std::cout << "Introduzca N: ";
   std::cin >> N;
   std::cout << "Introduzca t_A: ";
   std::cin >> t_A;
   std::cout << "Introduzca t_B: ";
   std::cin >> t_B;
   // Comprobacion de las entradas
   if ( !(t_A >= 0 && t_A < t_B && N > 0) ) {
      std::cerr << "Entrada no valida" << std::endl;
      return 0;
   }
   const double dt  = (t_B - t_A) / N;
   const double dz2 = std::pow(c*dt,2);
   // Longitud aproximada de la trayectoria
   double L = 0;
   double rx = a*std::cos(w*t_A);
   double ry = b*std::sin(w*t_A);
   for (int i=1; i <= N; i++) {
      double wt_1 = w*(t_A + i*dt);
      double rx_1 = a*std::cos(wt_1);
      double ry_1 = b*std::sin(wt_1);
      L += std::sqrt(std::pow(rx_1-rx,2) + std::pow(ry_1-ry,2) + dz2);
      rx = rx_1;
      ry = ry_1;
   }
   // Salida por consola
   std::cout << "Longitud aproximada: "
             << std::scientific << std::setprecision(5)
             << L << std::endl;
   return 0;
}
```

Ejercicio 2

Se desea calcular el importe de los intereses que deben abonarse en una cuenta bancaria cuyo funcionamiento es el siguiente: al finalizar cada año se abona en concepto de intereses el 2 % del saldo medio existente en la cuenta durante dicho año.

En un fichero de texto están registrados los movimientos en la cuenta durante el año. El formato es el siguiente. En cada fila del fichero está registrado un movimiento. En la primera columna del fichero se escribe el día en que se ha producido el movimiento, en formato dd/mm/aaaa, en la segunda columna se escribe la hora a la cual se ha producido el movimiento, en formato HH:MM:SS, y en la tercera columna se indica el saldo existente en la cuenta una vez realizado el movimiento. Una de las filas del fichero indica el saldo existente a comienzo del año, esto es, el día 1 de enero a las 00:00:00 horas. Veamos un ejemplo, que corresponde a los movimientos de una cuenta durante el año 2024. Como puede observarse, los movimientos no se encuentran ordenados.

```
25/03/2024  12:14:02    7500.00
12/01/2024  13:11:01    9400.00
01/01/2024  00:00:00   12400.00
25/03/2024  22:59:02    7450.00
14/10/2024  07:11:03   10300.00
```

En el ejemplo anterior, el saldo al comienzo del año era de 12400 Euros. El día 12 de enero se realizó un movimiento, pasando el saldo de la cuenta a ser de 9400 Euros. El día 25 de marzo se realizaron dos movimientos: uno a las 12:14:02 y otro a las 22:59:02. El día 14 de octubre se realizó otro movimiento en la cuenta, tras el cual el saldo pasó a ser de 10300 Euros, manteniéndose la cuenta con ese saldo hasta finalizar el año 2024.

El saldo medio del año se calcula sumando el saldo en la cuenta al finalizar cada uno de los días del año y dividiendo por el número total de días del año (365 en año no bisiesto y 366 en año bisiesto). Obsérvese que se supone que el saldo de la cuenta cada día es igual al saldo existente en la cuenta al finalizar el día, es decir, a las 23:59:59 horas. En el ejemplo anterior, el día 25 de marzo se producen dos movimientos. Dado que al finalizar el día 25 de marzo el saldo de la cuenta era de 7450 Euros, se supone que ese fue el saldo de ese día.

En el ejemplo anterior, dado que el año 2024 fue bisiesto (febrero tuvo 29 días), el saldo promedio es:

$$S = \frac{12400 \cdot 11 + 9400 \cdot 73 + 7450 \cdot 203 + 10300 \cdot 79}{366} = 8602.87 \text{ Euros}$$

Deberán abonarse $0.02 \cdot 8602.87 = 172.06$ Euros en concepto de intereses.

Escriba un programa en C++ que lea un fichero de texto llamado *movimientos.txt* en el cual están escritos los movimientos de una cuenta a lo largo de un año, y escriba en la consola el saldo medio anual y el importe que debe abonarse de intereses. Estas cantidades deben escribirse en formato fijo, con dos dígitos decimales.

Al escribir el programa puede suponer que el fichero *movimientos.txt* no contiene errores de formato. Puede suponer también que no contendrá varios movimientos realizados exactamente el mismo día y a la misma hora, minuto y segundo.

Muestre en el informe el resultado obtenido al ejecutar su programa cuando el contenido del fichero *movimientos.txt* es el siguiente:

```
25/03/2024  12:14:02    7500.00
12/01/2024  13:11:01    9400.00
01/01/2024  00:00:00   12400.12
25/03/2024  22:59:02    7450.00
14/10/2024  07:11:03   10300.14
19/03/2024  20:20:13    7537.92
25/03/2024  02:59:02   17450.00
25/03/2024  03:59:02   27450.10
25/03/2024  04:59:02   37450.45
20/05/2024  15:11:12   11034.65
12/07/2024  11:21:13    7648.27
11/11/2024  10:00:03    5637.12
29/12/2024  11:21:08    7986.36
01/01/2024  10:10:00    6400.26
08/09/2024  17:12:06   10761.89
12/05/2024  16:16:20   11987.42
13/05/2024  16:27:56     945.32
20/05/2024  11:11:12   12034.65
12/01/2024  09:11:27    9000.00
12/07/2024  13:25:23    7000.27
29/12/2024  16:22:03    8986.36
```

Solución al Ejercicio 2

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <iomanip>

const char nombreFich[] = "movimientos.txt";

struct Movimiento {
   std::string hora;  // Formato HH:MM:SS
   double saldo;
};

int getNumDiaDesdeComienzoAgno(std::string fecha, bool &bisiesto) {
// fecha en formato dd/mm/aaaa
// Devuelve el numero de dia desde comienzo del agno
   const int DIAS[] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273,
    304, 334};
   int dia = 10*(fecha[0]-'0')+fecha[1]-'0';
   int mes = 10*(fecha[3]-'0')+fecha[4]-'0';
   int agno = 1000*(fecha[6]-'0') + 100*(fecha[7]-'0')
              + 10*(fecha[8]-'0') + fecha[9]-'0';
   bisiesto = (agno%4==0 && agno%100!=0) || agno%400==0;
   if (bisiesto && mes>2)
      return dia + DIAS[mes-1] + 1;
   else
      return dia + DIAS[mes-1];
}


int main() {
   // Apertura del fichero
   std::ifstream inFich(nombreFich, std::ios::in);
   if (!inFich) {
      std::cerr << "Error al abrir el fichero " << nombreFich;
      return 0;
   }
   // Lectura del fichero
   std::map<int,Movimiento> mMovs;
   std::string fecha;
   Movimiento mov;
   bool bisiesto;
   while ( inFich >> fecha >> mov.hora >> mov.saldo ) {
      int dia = getNumDiaDesdeComienzoAgno(fecha, bisiesto);
      std::map<int,Movimiento>::iterator p = mMovs.find(dia);
      if (p == mMovs.end() || p->second.hora.compare(mov.hora)<0)
         mMovs[dia] = mov;
   }
   // Cerrar el fichero
   inFich.close();
```

```
50    int diasAgno = 365;
51    if (bisiesto) diasAgno = 366;
52    std::map<int,Movimiento>::iterator p = mMovs.begin();
53    double saldoMedio = 0;
54    while ( p != mMovs.end() ) {
55       int desde = p->first;
56       double saldo = p->second.saldo;
57       p++;
58       if ( p == mMovs.end() )
59          saldoMedio += saldo * (diasAgno - desde + 1);
60       else
61          saldoMedio += saldo * (p->first - desde);
62    }
63    saldoMedio /= diasAgno;
64    std::cout << std::fixed << std::setprecision(2)
65          << "Saldo medio anual: " << saldoMedio
66          << " Euros \nImporte de los intereses: "
67          << 0.02 * saldoMedio << " Euros" << std::endl;
68    return 0;
69 }
```

Ejercicio 3

Consideremos el siguiente sistema de dos ecuaciones diferenciales ordinarias

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= \mu \cdot \left(1 - x^2\right) \cdot y - x \end{aligned}$$

donde t representa el tiempo y μ es un parámetro real de valor conocido. En el instante inicial t_0, las variables x e y tienen valores conocidos: $x(t_0) = x_0$, $y(t_0) = y_0$.

Se desea calcular la evolución de las variables x e y entre los instantes t_0 y T, con $t_0 < T$. Para ello, se aplica el método de integración de Euler implícito, con un tamaño del paso de integración igual a h. Se define el entero N de la forma siguiente:

$$N = \left\lceil \frac{T - t_0}{h} \right\rceil$$

donde $\lceil \cdot \rceil$ representa la *función techo*, la cual, aplicada a un número real, devuelve el menor número entero mayor o igual que el real.

El procedimiento iterativo de cálculo es el siguiente:

$$\begin{aligned} \frac{x_n - x_{n-1}}{h} &= y_n \\ \frac{y_n - y_{n-1}}{h} &= \mu \cdot \left(1 - x_n^2\right) \cdot y_n - x_n \end{aligned}$$

para $n = 1, \ldots, N$, donde x_n e y_n representan respectivamente el valor de las variables x e y en el instante $t_n = t_0 + n \cdot h$.

Obsérvese que en el paso n son conocidos x_{n-1} e y_{n-1}, y deben calcularse x_n e y_n resolviendo el sistema de dos ecuaciones con dos incógnitas siguiente (se han señalado en color azul las incógnitas a calcular):

$$x_n - h \cdot y_n = x_{n-1} \tag{1.1}$$

$$h \cdot x_n + (1 - h \cdot \mu) \cdot y_n + h \cdot \mu \cdot {x_n}^2 \cdot y_n = y_{n-1} \tag{1.2}$$

Despejando x_n de la Ec. (1.1) y sustituyendo en la Ec. (1.2) obtenemos:

$$a \cdot {y_n}^3 + b \cdot {y_n}^2 + c \cdot y_n + d = 0 \tag{1.3}$$

donde los coeficientes a, b, c, d se definen de la forma siguiente:

$$a = \mu \cdot h^3 \tag{1.4}$$
$$b = 2 \cdot \mu \cdot h^2 \cdot x_{n-1} \tag{1.5}$$
$$c = 1 + h^2 + h \cdot \mu \cdot \left(x_{n-1}^2 - 1\right) \tag{1.6}$$
$$d = h \cdot x_{n-1} - y_{n-1} \tag{1.7}$$

El cálculo de y_n de la Ec (1.3) puede hacerse aplicado el método de Newton:

$$y_{n,k+1} = y_{n,k} - \frac{a \cdot y_{n,k}^3 + b \cdot y_{n,k}^2 + c \cdot y_{n,k} + d}{3 \cdot a \cdot y_{n,k}^2 + 2 \cdot b \cdot y_{n,k} + c} \tag{1.8}$$

siendo $k = 0, 1, \ldots$ el índice de iteración del método de Newton. Como valor inicial de la iteración, escoja: $y_{n,0} = y_{n-1}$. La condición de finalización de la iteración del método de Newton es que el número de iteraciones alcance el valor $k = 100$ o que se satisfaga

$$\left|a \cdot y_{n,k}^3 + b \cdot y_{n,k}^2 + c \cdot y_{n,k} + d\right| < TOL$$

Escriba un programa en C++ que resuelva numéricamente el sistema de dos ecuaciones diferenciales, empleando el procedimiento descrito. El programa debe declarar las siguientes constantes globales: $t_0 = 0$, $T = 20$, $h = 10^{-6}$, $\mu = 1.5$, $TOL = 10^{-6}$. El usuario debe introducir por consola las condiciones iniciales: x_0, y_0.

El programa debe escribir en un fichero de texto llamado *resSimulacion.txt* los valores de las variables t, x e y en los instantes siguientes: t_0, $t_0 + H$, $t_0 + 2 \cdot H$, $\ldots$, con $H = 10000 \cdot h$.

El formato del fichero debe ser el siguiente: en la primera columna se escribe el valor del tiempo, en la segunda columna el valor de x en ese instante y en la tercera columna el valor de y en ese instante. Todo ello en formato fijo, con 4 dígitos decimales. Las columnas deben estar separadas por un espacio en blanco.

Muestre en el informe las 10 primeras filas y las 10 últimas filas del fichero generado por su programa cuando es ejecutado para las condiciones iniciales siguientes: $x_0 = 2$, $y_0 = 0$.

Solución al Ejercicio 3

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <cmath>

const char nombreFich[] = "resSimulacion.txt";
const double mu  = 1.5;
const double TOL = 1e-6;
const double t0  = 0;
const double T   = 20;
const double h   = 1e-6;
const double H   = 0.1;

int main() {
   // Entrada por consola
   double x, y;
   std::cout << "Valor inicial de x: ";
   std::cin >> x;
   std::cout << "Valor inicial de y: ";
   std::cin >> y;
   // Apertura del fichero para escritura
   std::ofstream file_out(nombreFich,
                          std::ios::out | std::ios::trunc);
   if (!file_out) {
      std::cerr << "Error al abrir el fichero " << nombreFich;
      return 0;
   }
   file_out << std::fixed << std::setprecision(4)
            << t0 << " " << x << " " << y << std::endl;
   // Integracion
   const int    N       = std::ceil( (T-t0)/h );
   const double a       = mu*std::pow(h,3);
   const double coef_3a = 3*a;
   const double coef_b  = 2*mu*std::pow(h,2);
   const double coef_c  = h*mu;
   const double indep_c = 1 + std::pow(h,2) - coef_c;
   int i_log = 0;
   for (double n=1; n <= N; n++) {
      double t = t0 + n*h;
      i_log++;
      // Iteracion metodo de Newton
      double b = coef_b*x;
      double c = coef_c*std::pow(x,2) + indep_c;
      double d = h*x - y;
      for (int k=0; k <= 100; k++) {
         double y2 = y*y;
         double y3 = y2*y;
         double num = a*y3 + b*y2 + c*y + d;
         y -= num/(coef_3a*y2 + 2*b*y + c);
         if ( std::abs(num) < TOL )
            break;
      }
```

```
54        x += h*y;
55        if ( i_log == 10000 ) {
56           file_out << std::fixed << std::setprecision(4)
57                    << t << " " << x << " " << y << std::endl;
58           i_log = 0;
59        }
60     }
61     file_out.close();
62     return 0;
63  }
```

Ejercicio 4

En un fichero de texto llamado *pisos.txt* se encuentra almacenada información acerca de pisos ofertados en alquiler en una ciudad. La información acerca de un piso está descrita mediante una secuencia de expresiones del tipo `IDENT=N` separadas entre sí por uno o más espacios en blanco o saltos de línea, donde `IDENT` es uno de los identificadores mostrados en la tabla siguiente y `N` es un número entero. La información correspondiente a un piso finaliza con un punto y coma (;) precedido y sucedido de uno o varios espacios en blanco y/o saltos de línea.

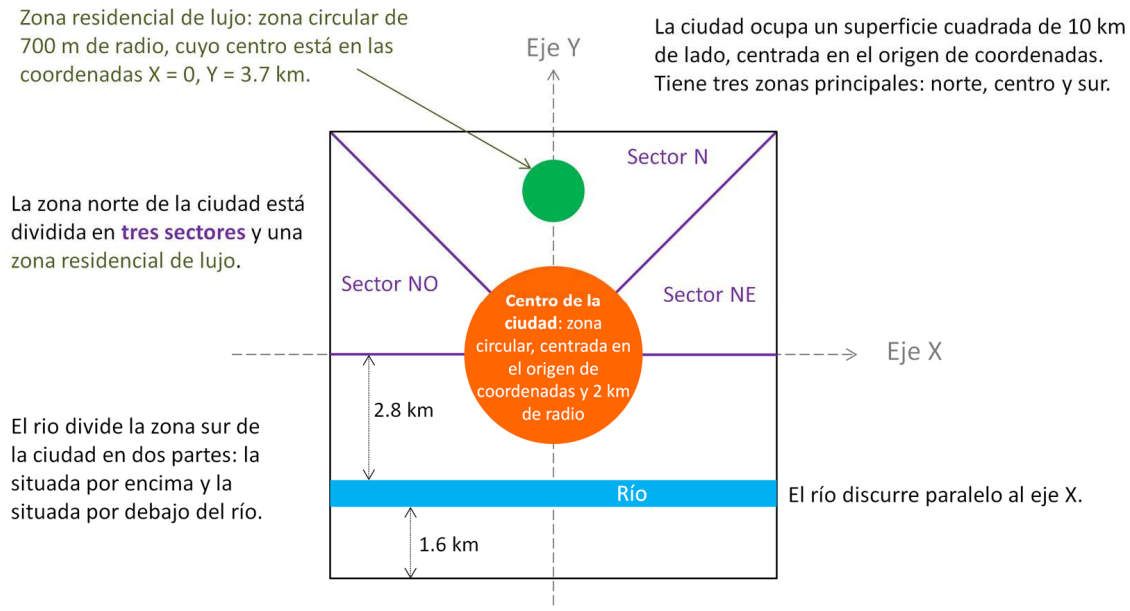
`IDENT`	**Significado**
`PLAN`	Número de la planta donde está el piso.
`EDIF`	Número de plantas del edificio.
`DORM`	Número de dormitorios que tiene el piso.
`BAGN`	Número de baños.
`SUPT`	Superficie total del piso (m^2).
`SUPU`	Superficie útil del piso (m^2).
`POSX`	Coordenada X (m) de la ubicación del piso en la ciudad.
`POSY`	Coordenada Y (m) de la ubicación del piso en la ciudad.
`PREC`	Importe del alquiler mensual.

Por ejemplo, las dos siguientes líneas del fichero

```
PLAN=2 PREC=1000 POSX=2300 POSY=300 EDIF=14
DORM=3 BAGN=1 SUPT=130 SUPU=70 ;
```

describen un piso ubicado en una segunda planta de un edificio de catorce plantas, que tiene tres dormitorios, un baño, una superficie total de 130 m^2 y una superficie útil de 70 m^2, se encuentra en la coordenadas (X=2300 m, Y=300 m) del plano de la ciudad, y tiene un precio mensual de alquiler de 1000 Euros. Obsérvese que las expresiones pueden estar escritas en cualquier orden.

En la figura de la página siguiente están representadas las distintas zonas de la ciudad, desde el punto de vista del mercado del alquiler. La ciudad ocupa una superficie cuadrada de 10 km de lado. Para describir la ubicación de los pisos en la ciudad, se emplea un sistema de coordenadas cartesiano, con los ejes paralelos a los lados del cuadrado y con el origen de coordenadas en el centro del cuadrado.



La zona denominada *Centro de la Ciudad* es un círculo de radio 2 km cuyo centro coincide con el centro de la ciudad. La zona de la ciudad situada por encima del eje X, excluyendo el *Centro de la Ciudad*, se denomina *Zona Norte* y está dividida en tres sectores denominados *Sector NO* (noroeste), *Sector N* (norte) y *Sector NE* (noreste), y una *Zona Residencial de Lujo*, que ocupa una zona circular de 700 m de radio centrada en (X=0, Y=3.7 km). Los dos segmentos de recta diagonales que delimitan los sectores intersectan en el origen de coordenadas. La zona de la ciudad situada por debajo del eje X, excluyendo el *Centro de la Ciudad*, se denomina *Zona Sur* y se compone de dos partes: la situada por encima del río y la situada por debajo del río.

El criterio para determinar a qué zona corresponden los pisos que se encuentran sobre la frontera entre zonas es el siguiente. Si el piso se encuentra sobre la frontera de la zona centro o residencial, pertenece a la zona centro o residencial respectivamente. Si no se satisface lo anterior y el piso se encuentra en la frontera de los Sectores NO o NE, pertenece a los Sectores NO o NE respectivamente.

Escriba un programa en C++ que solicite al usuario que introduzca por consola la descripción del tipo de piso que busca, especificando para ello todo lo siguiente:

- Número mínimo de dormitorios y número mínimo de baños.
- Número mínimo de metros cuadrados útiles.
- Importe máximo que está dispuesto a pagar mensualmente por el alquiler.

– Zona de la ciudad donde desea que se encuentre el piso: (0) Centro; (1) Sector NO; (2) Sector N; (3) Sector NE; (4) Residencial de Lujo; (5) Sur encima del río; o (6) Sur debajo del río.

El programa debe leer el fichero *pisos.txt* y mostrar en la consola la descripción de los pisos que cumplen simultáneamente todas esas condiciones, ordenados de menor a mayor importe mensual del alquiler.

Puede suponer que el fichero *pisos.txt* no tiene errores de formato y que las descripciones de los pisos están completas (en cada piso se asigna valor a todos los identificadores). A continuación se muestra el contenido del fichero *pisos.txt* (continúa en la página siguiente).

```
PLAN=2 PREC=1300 POSX=2000 POSY=0 EDIF=14
DORM=3 BAGN=1 SUPT=130 SUPU=70 ;
PLAN=4 PREC=1100 POSX=-2000 POSY=0 EDIF=6
DORM=3 BAGN=1 SUPT=120 SUPU=90
;
PLAN=2 EDIF=14 POSY=-2000 POSX=0 PREC=1200 SUPT=90 SUPU=65
 DORM=2 BAGN=1 ;
PLAN=8 EDIF=10 POSY=500 POSX=500 PREC=2000 SUPT=180 SUPU=140
 DORM=4 BAGN=2 ;
PLAN=1 PREC=1300 POSX=-550 POSY=250 EDIF=4
DORM=3 BAGN=1 SUPT=120 SUPU=85 ;
PLAN=3 PREC=2500 POSX=1000 POSY=-1000 EDIF=4 DORM=4 BAGN=2 SUPT=160 SUPU=120 ;
PLAN=1 EDIF=3 POSY=-300 POSX=-350 PREC=800 SUPT=60 SUPU=35
BAGN=1 DORM=0 ;
PLAN=1 EDIF=3 POSY=50 POSX=-50 PREC=2800 SUPT=220 SUPU=160 DORM=4 BAGN=2 ;
POSY=0 POSX=-3000 PREC=1800
PLAN=1 EDIF=3 SUPT=220 SUPU=160 DORM=4 BAGN=2 ;
POSY=2500 POSX=-2500 PREC=1400
PLAN=2 EDIF=4 SUPT=160 SUPU=110 DORM=3 BAGN=2 ;
POSY=3000 POSX=-3000 PREC=1400
PLAN=3 EDIF=4 SUPT=140 SUPU=105 DORM=3 BAGN=2 ;
POSY=400 POSX=-4000 PREC=1550
PLAN=6 EDIF=8 SUPT=140 SUPU=120 DORM=4 BAGN=2 ;
POSY=3000 POSX=-3000 PREC=800
PLAN=6 EDIF=8 SUPT=70 SUPU=60 DORM=1 BAGN=1 ;
PLAN=1 EDIF=3 POSX=-500 POSY=2550 PREC=600 SUPT=50 SUPU=30
BAGN=1 DORM=0 ;
PLAN=2 EDIF=4 POSX=500 POSY=2000 PREC=700 SUPT=80 SUPU=60
BAGN=1 DORM=2 ;
POSX=0 POSY=4500 PREC=1200
```

```
PLAN=3 EDIF=10 SUPT=160 SUPU=100 DORM=3 BAGN=2 ;
POSX=0 POSY=3750 PREC=2200
PLAN=1 EDIF=3 SUPT=160 SUPU=100 DORM=3 BAGN=2 ;
POSX=0 POSY=3000 PREC=1800
PLAN=2 EDIF=4 SUPT=180 SUPU=130 DORM=4 BAGN=2 ;
POSX=500 POSY=3700 PREC=3000
PLAN=1 EDIF=1 SUPT=400 SUPU=300 DORM=6 BAGN=3 ;
PLAN=1 EDIF=3 POSY=0 POSX=2050 PREC=1200 SUPT=120 SUPU=90 DORM=3 BAGN=2 ;
PLAN=2 EDIF=5 POSY=2350 POSX=2900 PREC=1000 SUPT=100 SUPU=80 DORM=3 BAGN=2 ;
PLAN=4 EDIF=10 POSY=700 POSX=4000 PREC=900 SUPT=100 SUPU=80 DORM=3 BAGN=1 ;
PLAN=6 EDIF=12 POSY=3000 POSX=3000 PREC=800 SUPT=60 SUPU=50 DORM=1 BAGN=1 ;
PLAN=3 EDIF=4 POSY=3550 POSX=4900 PREC=600 SUPT=50 SUPU=35 DORM=0 BAGN=1 ;
PLAN=6 EDIF=6 POSX=-2500 POSY=-2550 PREC=600 SUPT=50 SUPU=30
BAGN=1 DORM=0 ;
PLAN=4 EDIF=6 POSX=-500 POSY=-2550 PREC=650 SUPT=50 SUPU=30
BAGN=1 DORM=0 ;
PLAN=2 EDIF=6 POSX=600 POSY=-2550 PREC=680 SUPT=50 SUPU=35
BAGN=1 DORM=0 ;
PLAN=2 EDIF=4 POSX=-500 POSY=-2600 PREC=690 SUPT=55 SUPU=30
BAGN=1 DORM=0 ;
PLAN=1 EDIF=4 POSX=0 POSY=-2550 PREC=650 SUPT=55 SUPU=37
BAGN=1 DORM=0 ;
PLAN=4 EDIF=8 POSX=-2500 POSY=-2800 PREC=690 SUPT=60 SUPU=40
BAGN=1 DORM=0 ;
PLAN=6 EDIF=6 POSX=2500 POSY=-2500 PREC=1000 SUPT=100 SUPU=80
BAGN=1 DORM=3 ;
PLAN=6 EDIF=6 POSX=500 POSY=-2500 PREC=1200 SUPT=120 SUPU=90
BAGN=2 DORM=3 ;
PLAN=6 EDIF=6 POSX=-2500 POSY=-3550 PREC=450 SUPT=50 SUPU=30
BAGN=1 DORM=0 ;
PLAN=4 EDIF=6 POSX=-500 POSY=-3750 PREC=550 SUPT=50 SUPU=30
BAGN=1 DORM=0 ;
PLAN=2 EDIF=6 POSX=600 POSY=-4550 PREC=580 SUPT=50 SUPU=35
BAGN=1 DORM=1 ;
PLAN=2 EDIF=4 POSX=-500 POSY=-3600 PREC=490 SUPT=55 SUPU=30
BAGN=1 DORM=0 ;
PLAN=1 EDIF=4 POSX=0 POSY=-3550 PREC=550 SUPT=55 SUPU=37
BAGN=1 DORM=0 ;
PLAN=4 EDIF=8 POSX=-2500 POSY=-3800 PREC=600 SUPT=60 SUPU=40
BAGN=1 DORM=1 ;
PLAN=3 EDIF=6 POSX=2500 POSY=-4600 PREC=900 SUPT=100 SUPU=80
BAGN=2 DORM=3 ;
PLAN=6 EDIF=6 POSX=500 POSY=-4900 PREC=1000 SUPT=120 SUPU=90
BAGN=2 DORM=4 ;
```

Muestre en el informe el resultado de ejecutar su programa para los tres casos de prueba indicados a continuación.

Caso de prueba 1:

```
Numero minimo de dormitorios: 2
Numero minimo de bagnos: 1
Numero minimo de m2 utiles: 80
Importe maximo alquiler: 1000
Zonas
(0) Centro
(1) Sector NO
(2) Sector N
(3) Sector NE
(4) Residencial Lujo
(5) Sur encima rio
(6) Sur debajo rio
Seleccione zona: 6
```

Caso de prueba 2:

```
Numero minimo de dormitorios: 3
Numero minimo de bagnos: 1
Numero minimo de m2 utiles: 80
Importe maximo alquiler: 1600
Zonas
(0) Centro
(1) Sector NO
(2) Sector N
(3) Sector NE
(4) Residencial Lujo
(5) Sur encima rio
(6) Sur debajo rio
Seleccione zona: 0
```

Caso de prueba 3:

```
Numero minimo de dormitorios: 2
Numero minimo de bagnos: 1
Numero minimo de m2 utiles: 90
Importe maximo alquiler: 1500
Zonas
(0) Centro
(1) Sector NO
(2) Sector N
(3) Sector NE
(4) Residencial Lujo
(5) Sur encima rio
(6) Sur debajo rio
Seleccione zona: 2
```

Solución al Ejercicio 4

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#include <vector>
#include <list>

const char nombreFich[] = "pisos.txt";

enum Zona { ZONA_CENTRO, ZONA_NO, ZONA_N, ZONA_NE, ZONA_LUJO,
            ZONA_SUR_ENCIMA_RIO, ZONA_SUR_DEBAJO_RIO, NUM_ZONAS};

enum Ident { PLAN, EDIF, DORM, BAGN, SUPT, SUPU,
             POSX, POSY, PREC, NUM_IDENTS };
const std::vector<std::string> IDENTS {"PLAN", "EDIF",
   "DORM", "BAGN", "SUPT", "SUPU", "POSX", "POSY", "PREC"};


Zona getZona(int x_m, int y_m) {
// Entrada: coordenadas X, Y del piso expresadas en metros
// Salida:
// (0) Centro; (1) Sector NO; (2) Sector N; (3) Sector NE;
// (4) Residencial de lujo; (5) Sur encima rio; (6) Sur debajo rio.
   const int RadioCen_m = 2000; // Radio centro ciudad
   const int YRes_m     = 3700; // Coord. Y centro zona residencial
   const int RadioRes_m = 700;  // Radio zona recidencial lujo
   if ( x_m*x_m + y_m*y_m <= RadioCen_m*RadioCen_m )
      return ZONA_CENTRO;
   if ( y_m >= 0 ) {
      if ( x_m*x_m + std::pow(y_m-YRes_m, 2) <= RadioRes_m*RadioRes_m )
         return ZONA_LUJO;
      if ( y_m <= -x_m )
         return ZONA_NO;
      if ( y_m <= x_m )
         return ZONA_NE;
      return ZONA_N;
   }
   if ( y_m >= -2800 )
      return ZONA_SUR_ENCIMA_RIO;
   return ZONA_SUR_DEBAJO_RIO;
}


int main() {
   // Criterio de busqueda
   int minDor, minBagn, minSupUtil, maxAlq, selZona;
   std::cout << "Numero minimo de dormitorios: ";
   std::cin >> minDor;
   std::cout << "Numero minimo de bagnos: ";
   std::cin >> minBagn;
   std::cout << "Numero minimo de m2 utiles: ";
```

```
52    std::cin >> minSupUtil;
53    std::cout << "Importe maximo alquiler: ";
54    std::cin >> maxAlq;
55    for (;;) {
56       std::cout << "Zonas\n"
57                 << "(0) Centro\n"           << "(1) Sector NO\n"
58                 << "(2) Sector N\n"         << "(3) Sector NE\n"
59                 << "(4) Residencial lujo\n" << "(5) Sur encima rio\n"
60                 << "(6) Sur debajo rio\n"
61                 << "Seleccione zona: ";
62       std::cin >> selZona;
63       if ( selZona >= 0 && selZona < NUM_ZONAS )
64          break;
65    }
66    // Apertura del fichero
67    std::ifstream inFich(nombreFich, std::ios::in);
68    if (!inFich) {
69       std::cerr << "Error al abrir el fichero " << nombreFich;
70       return 0;
71    }
72    // Lectura del fichero
73    std::string expr;
74    std::vector<int> piso(NUM_IDENTS);
75    std::list< std::vector<int> > lPisos;
76    while ( inFich >> expr ) {
77       if ( expr == ";" ) {
78          bool cond = piso[DORM] >= minDor     &&
79                      piso[BAGN] >= minBagn    &&
80                      piso[SUPU] >= minSupUtil &&
81                      piso[PREC] <= maxAlq     &&
82                      getZona(piso[POSX], piso[POSY]) == selZona;
83          if ( cond ) {
84             if ( lPisos.size() == 0 ) {
85                lPisos.push_back(piso);
86             } else {   // Inserta en orden creciente de precio
87                std::list< std::vector<int> >::iterator p =
88                   lPisos.begin();
89                bool insertado = false;
90                while ( p != lPisos.end() ) {
91                   if ( piso[PREC] < (*p)[PREC] ) {
92                      lPisos.insert(p,piso);
93                      insertado = true;
94                      break;
95                   }
96                   p++;
97                }
98                if ( !insertado )
99                   lPisos.push_back(piso);
100            }
101         }
102      } else {
103         std::string ident = expr.substr(0,4);
104         for ( unsigned i=0; i<NUM_IDENTS; i++) {
105            if ( ident == IDENTS[i] ) {
```

```
106                piso[i] = std::stoi(expr.substr(5));
107                break;
108             }
109          }
110       }
111    }
112    inFich.close();
113    // Listado por consola
114    if ( lPisos.size() == 0 ) {
115       std::cout << "Ningun piso cumple los criterios\n";
116       return 0;
117    }
118    std::list< std::vector<int> >::iterator p = lPisos.begin();
119    while ( p != lPisos.end() ) {
120       for (int i=0; i<NUM_IDENTS; i++)
121          std::cout << IDENTS[i] << "=" << (*p)[i] << " ";
122       std::cout << "\n";
123       p++;
124    }
125    return 0;
126 }
```