LENGUAJES DE PROGRAMACIÓN

Solución al Trabajo Práctico - Junio de 2022

EJERCICIO 1

En el interior de una región rectangular, cuyos lados miden L_x y L_y respectivamente, deben ubicarse dos rectángulos satisfaciéndose que:

- Los lados de los rectángulos son paralelos a los lados de la región.
- Los rectángulos están completamente contenidos en la región, pudiendo solapar entre sí.
- Un rectángulo es de color rojo y el otro de color azul.

Para describir la posición de los rectángulos, establecemos un sistema de coordenadas cartesiano X-Y tal que las direcciones X e Y son paralelas a los lados L_x y L_y, y el origen de coordenadas es el vértice inferior izquierdo de la región.

Escriba un programa en C++ que compruebe si dos rectángulos, introducidos por consola por el usuario, satisfacen las condiciones anteriores. El programa debe realizar las acciones siguientes:

1. Declarar dos constantes globales de tipo **double** llamadas `Lx` y `Ly`, y asignarles valor 12.5 y 8 respectivamente.
2. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola, para cada uno de los dos rectángulos, su color (escribiendo la palabra *azul* o la palabra *rojo*), las coordenadas X e Y de su vértice inferior izquierdo, y las longitudes de sus lados.
3. Leer los datos introducidos por consola y comprobar si cumplen las especificaciones dadas en el enunciado. Es decir, que cada rectángulo es azul o

rojo, que son de diferente color, que sus lados tienen una longitud mayor que cero y que ambos están completamente contenidos en la región.

4. Escribir en la consola un mensaje indicando si los rectángulos cumplen las especificaciones.
5. Terminar.

Muestre en el informe las pruebas que ha realizado para comprobar que su programa tiene la funcionalidad pedida.

Solución al Ejercicio 1

```cpp
#include <iostream>
#include <string>

const double Lx = 12.5, Ly = 8;

struct Rectan {
  std::string color;
  double      vertice_X, vertice_Y;
  double      lado_X, lado_Y;
};

int main() {
  // Entrada de datos por consola
  std::cout << "Introduzca color, coords vertice y longitud lados\n";
  std::cout << "-- Del rectangulo 1:" << std::endl;
  Rectan rec1;
  std::cin >> rec1.color  >> rec1.vertice_X >> rec1.vertice_Y
           >> rec1.lado_X >> rec1.lado_Y;
  std::cout << "-- Del rectangulo 2:" << std::endl;
  Rectan rec2;
  std::cin >> rec2.color  >> rec2.vertice_X >> rec2.vertice_Y
           >> rec2.lado_X >> rec2.lado_Y;
  // Comprobación
  bool color = ( rec1.color == "azul" || rec1.color == "rojo") &&
               ( rec2.color == "azul" || rec2.color == "rojo") &&
               rec1.color != rec2.color;
  bool posR1 = rec1.vertice_X >= 0                &&
               rec1.vertice_X + rec1.lado_X <= Lx &&
               rec1.vertice_Y >= 0                &&
               rec1.vertice_Y + rec1.lado_Y <= Ly;
  bool posR2 = rec2.vertice_X >= 0                &&
               rec2.vertice_X + rec2.lado_X <= Lx &&
               rec2.vertice_Y >= 0                &&
               rec2.vertice_Y + rec2.lado_Y <= Ly;
  bool longLados = rec1.lado_X > 0 && rec1.lado_Y > 0 &&
                   rec2.lado_X > 0 && rec2.lado_Y > 0;
  if ( !color || !posR1 || !posR2 || !longLados )
    std::cout << "No se satisfacen las condiciones" << std::endl;
  else
    std::cout << "Satisfacen las condiciones" << std::endl;
  return 0;
}
```

EJERCICIO 2

El polinomio $f(x)$ puede expresarse de la forma:

$$f(x) = x^n + a_1 \cdot x^{n-1} + \cdots + a_{n-1} \cdot x + a_n$$

o equivalentemente

$$f(x) = (x - \alpha_1) \cdot (x - \alpha_2) \cdots (x - \alpha_n)$$

donde n es el grado del polinomio; a_1, a_2, ..., a_n son los coeficientes; y α_1, α_2, ..., α_n son las raíces. En este ejercicio supondremos que todas las raíces del polinomio $f(x)$ son números reales.

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

1. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola el grado del polinomio. Leer el valor introducido por consola, almacenándolo en una variable de tipo entero llamada `n`.
2. Si $n < 1$, mostrar un mensaje en la consola indicándolo y terminar.
3. Escribir un mensaje en la consola solicitando al usuario que introduzca las n raíces del polinomio. Leer los valores introducidos por consola, almacenándolos en un vector de **double** llamado `alpha`.
4. Calcular los coeficientes del polinomio (a_1, a_2, ..., a_n) y escribirlos en la consola, en formato científico, con 8 dígitos detrás del punto decimal.
5. Terminar.

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

Caso 1: $n = 1$, $\alpha_1 = 3$
Caso 2: $n = 3$, $\alpha_1 = 2$, $\alpha_2 = -2$, $\alpha_3 = 1.2$
Caso 3: $n = 6$, $\alpha_1 = \alpha_2 = 1.1$, $\alpha_3 = 3.2$, $\alpha_4 = -11.4$, $\alpha_5 = 0$, $\alpha_6 = -0.1$

Solución al Ejercicio 2

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

int main() {
  // Entrada de datos por consola
  std::cout << "Grado del polinomio: ";
  int n;
  std::cin >> n;
  if ( n < 1 ) {
    std::cout << "Valor no valido";
    return 0;
  }
  std::cout << "Introduzca las raices del polinomio:" << std::endl;
  std::vector<double> alpha(n,0);
  for (int i=0; i<n; i++)
    std::cin >> alpha[i];
  // Calculo de los coeficientes
  std::vector<double> a(n+1,0);
  a[0] = -alpha[0];
  a[1] = 1;
  for (int r=1; r<n; r++) {
    for (int j=r+1; j>0; j--)
      a[j] = a[j-1] - alpha[r] * a[j];
    a[0] = - alpha[r] * a[0];
  }
  // Salida por consola
  std::cout << "Coeficientes del polinomio:\n";
  for (int i=0; i<n; i++)
    std::cout << std::scientific << std::setprecision(8)
      << "a(" << n-i << ") = " << a[i] << "\n";
  return 0;
}
```

EJERCICIO 3

Se desea calcular, mediante el método de los mínimos cuadrados, el polinomio de grado dos

$$y = b_0 + b_1 \cdot x + b_2 \cdot x^2$$

que se ajusta a un conjunto de datos $(x_1, y_1), \ldots, (x_n, y_n)$ que se encuentran almacenados en un fichero de texto. Las parejas de valores (x_i, y_i) son números reales.

Los coeficientes del polinomio (b_0, b_1, b_2) se calculan resolviendo el sistema lineal de tres ecuaciones con tres incógnitas siguiente.

$$\begin{aligned}
b_0 \cdot n + b_1 \cdot \sum_{j=1}^{n} x_j + b_2 \cdot \sum_{j=1}^{n} x_j^2 &= \sum_{j=1}^{n} y_j \\
b_0 \cdot \sum_{j=1}^{n} x_j + b_1 \cdot \sum_{j=1}^{n} x_j^2 + b_2 \cdot \sum_{j=1}^{n} x_j^3 &= \sum_{j=1}^{n} x_j \cdot y_j \\
b_0 \cdot \sum_{j=1}^{n} x_j^2 + b_1 \cdot \sum_{j=1}^{n} x_j^3 + b_2 \cdot \sum_{j=1}^{n} x_j^4 &= \sum_{j=1}^{n} x_j^2 \cdot y_j
\end{aligned}$$

El fichero con los datos $(x_1, y_1), \ldots, (x_n, y_n)$ tiene n filas y 2 columnas. La fila i-ésima contiene x_i en la primera columna e y_i en la segunda.

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

1. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola el nombre del fichero de texto donde se encuentran los datos. Leer dicho nombre de la consola.
2. Abrir el fichero para lectura. Si se produce error, terminar.
3. Leer el fichero, almacenando los valores en sendos vectores de tipo **double** llamados `vX` y `vY`.
4. Resolver el sistema de ecuaciones. Para ello, calcular la inversa de la matriz de coeficientes del sistema de ecuaciones. Si la matriz es singular, terminar.
5. Escribir en la consola el polinomio ajustado, con los valores numéricos en formato científico, con 4 dígitos de precisión.
6. Terminar.

Muestre en el informe el resultado obtenido al ejecutar su programa para cada uno de los dos ficheros de datos siguientes.

Fichero datos1.txt

```
0  5
2  4
4  1
6  6
8  7
```

Fichero datos2.txt

```
-10 235.6419
-8  157.9218
-6   96.4157
-4   50.2922
-2   20.4595
0     6.1557
2     7.5357
4    26.3491
6    60.4340
8   110.1787
10  176.2577
```

Solución al Ejercicio 3

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


int main() {
  // Entrada por consola del nombre del fichero
  std::cout << "Introduzca el nombre del fichero:";
  std::string nombreFich;
  std::cin >> nombreFich;
  // Apertura del fichero para lectura
  std::ifstream inFich(nombreFich.c_str(), std::ios::in);
  if (!inFich) {
    std::cerr << "Error al abrir el fichero" << std::endl;
    return 0;
  }
```

```
  // Lectura del fichero almacenando los valores en vectores vX, vY
  std::vector<double> vX, vY;
  double datoX, datoY;
  while (inFich >> datoX >> datoY) {
    vX.push_back(datoX);
    vY.push_back(datoY);
  }
  inFich.close();
  // Coeficientes del sistema de ecuaciones
  int n = vX.size();
  double m[5] = { (double)n, 0, 0, 0, 0 };
  double c[3] = { 0,0,0 };
  for (int i=0; i<n; i++) {
    double vX2 = vX[i] * vX[i];
    double vX3 = vX2   * vX[i];
    m[1] += vX[i];
    m[2] += vX2;
    m[3] += vX3;
    m[4] += vX3   * vX[i];;
    c[0] += vY[i];
    c[1] += vX[i] * vY[i];
    c[2] += vX2   * vY[i];
  }
  // Solución del sistema de ecuaciones
  double detA = m[0]*m[2]*m[4] + m[1]*m[3]*m[2] + m[2]*m[1]*m[3]
      - m[2]*m[2]*m[2] - m[1]*m[1]*m[4] - m[0]*m[3]*m[3];
  if ( !detA ) {
    std::cout << "El determinante vale cero";
    return 0;
  }
  double inv[3][3] = {
        { (  m[2]*m[4] - m[3]*m[3] ) / detA,
          (- m[1]*m[4] + m[3]*m[2] ) / detA,
          (  m[1]*m[3] - m[2]*m[2] ) / detA
        },
        { (- m[1]*m[4] + m[2]*m[3] ) / detA,
          (  m[0]*m[4] - m[2]*m[2] ) / detA,
          (- m[0]*m[3] + m[1]*m[2] ) / detA
        },
        { (  m[1]*m[3] - m[2]*m[2] ) / detA,
          (- m[0]*m[3] + m[2]*m[1] ) / detA,
          (  m[0]*m[2] - m[1]*m[1] ) / detA
        }
      };
  for (int i=0; i<3; i++) {
    double b = 0;
    for (int j=0; j<3; j++)
      b += inv[i][j] * c[j];
    std::cout << std::scientific << std::setprecision(4)
      << "b[" << i << "] = " << b << "\n";
  }
  return 0;
}
```

EJERCICIO 4

Escriba un programa en C++ que traduzca la descripción de una ruta, dada en un formato que denominaremos *formato nativo*, a una descripción más fácilmente comprensible por el ser humano y escriba esta última descripción en la consola.

Las instrucciones en *formato nativo* se encuentran almacenadas en un fichero de texto llamado *ruta.txt*. Estas instrucciones son sentencias de los dos tipos siguientes:

1. Sentencias mediante las cuales se indica que debe transitarse por una determinada vía hasta un cierto punto kilométrico de la ruta:

   `VIA_<IDENT_VÍA> <KM_RUTA_ABANDONA_VIA>`

2. Sentencias que indican cómo debe realizarse el tránsito de una vía a otra:

`SD`	Tome la salida derecha
`SI`	Tome la salida izquierda
`R_<nSalida>`	Abandone la rotonda por la `<nSalida>` salida

La primera y última sentencias del fichero *ruta.txt* son del Tipo 1. Entre dos sentencias del Tipo 1 siempre hay una sentencia del Tipo 2.

Puede suponer, al realizar el programa, que el formato del fichero *ruta.txt* es correcto y que la información es correcta.

Mediante los siguientes dos ejemplos se explica cómo el programa debe obtener su salida a partir del fichero *ruta.txt*.

Muestre en la memoria los resultados que ha obtenido al ejecutar su programa con cada uno de los ficheros *ruta.txt* de estos dos ejemplos.

Ejemplo 1

Contenido del fichero *ruta.txt*

```
VIA_BA20 3.9
R_segunda
VIA_A5 142
SD
VIA_NVa 146
```

Salida del programa

```
Comenzamos
Siga por BA20 durante 3.9 km
En la rotonda, salga por la segunda salida direccion A5
Siga por A5 durante 138.1 km
Abandone A5 por la salida derecha direccion NVa
Siga por NVa durante 4 km
Ha llegado a su destino
```

A continuación se explica cómo el programa ha obtenido la salida.

La salida del programa debe comenzar con la línea:
`Comenzamos`
y finalizar con la línea:
`Ha llegado a su destino`.

La primera línea del fichero *ruta.txt* indica que debe transitarse por la vía BA20 desde el kilómetro cero de la ruta hasta el kilómetro 3.9 de la ruta, por tanto el programa escribe:
`Siga por BA20 durante 3.9 km`

La primera y tercera líneas del fichero *ruta.txt* indican que desde el kilómetro 3.9 de la ruta (en que se abandona BA20) hasta el kilómetro 142 de la ruta (en que se abandona A5) se transita por la vía A5. Así pues, se recorren $142 - 3.9 = 138.1$ km por la vía A5. La segunda línea del fichero indica que el cambio de vía se realiza saliendo por la segunda salida de la rotonda. El programa escribe:
`En la rotonda, salga por la segunda salida direccion A5`
`Siga por A5 durante 138.1 km`

En el kilómetro 142 de la ruta se debe abandonar la vía A5, por la salida de la derecha, siguiendo por la vía NVa. Se recorren 4 km por la vía NVa: desde el punto kilométrico 142 de la ruta (en que se abandona la vía A5) hasta el punto kilométrico 146 de la ruta, en el cual ésta acaba.

Ejemplo 2

Contenido del fichero *ruta.txt*

```
VIA_N634     2.5
R_segunda
VIA_A8     130
SD
VIA_A6     168
R_segunda
VIA_N634   200
R_segunda
VIA_A54    210
R_tercera
VIA_SC20   212
SD
VIA_AP9    274
SD
VIA_C550   274.1
SI
VIA_PO531   275
```

Salida del programa

```
Comenzamos
Siga por N634 durante 2.5 km
En la rotonda, salga por la segunda salida direccion A8
Siga por A8 durante 127.5 km
Abandone A8 por la salida derecha direccion A6
Siga por A6 durante 38 km
En la rotonda, salga por la segunda salida direccion N634
Siga por N634 durante 32 km
En la rotonda, salga por la segunda salida direccion A54
Siga por A54 durante 10 km
En la rotonda, salga por la tercera salida direccion SC20
Siga por SC20 durante 2 km
Abandone SC20 por la salida derecha direccion AP9
Siga por AP9 durante 62 km
Abandone AP9 por la salida derecha direccion C550
Siga por C550 durante 0.1 km
Abandone C550 por salida izquierda direccion PO531
Siga por PO531 durante 0.9 km
Ha llegado a su destino
```

Solución al Ejercicio 4

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>

const std::string nombreFich = "ruta.txt";

struct Tramo {
  std::string via;
  double      punto_km_abandona;
};

int main() {
  // Apertura del fichero para lectura
  std::ifstream inFich(nombreFich.c_str(), std::ios::in);
  if (!inFich) {
    std::cerr << "Error al abrir el fichero" << std::endl;
    return 0;
  }
  // Lectura del fichero
  std::vector<Tramo> vTramo;
  std::vector<std::string> vTransito;
  std::string id1;
  while (inFich >> id1) {
    if ( id1.substr(0, 4) == "VIA_" ) {  // Sentencia Tipo 1
      double ptoKm;
      inFich >> ptoKm;
      Tramo tr = { id1.substr(4, id1.size()-4), ptoKm };
      vTramo.push_back(tr);
    } else {   // Sentencia Tipo 2
      vTransito.push_back(id1);
    }
  }
  inFich.close();
  int numTramos = vTramo.size();
  if ( !numTramos ) {
    std::cout << "El numero de tramos es cero";
    return 0;
  }
  // Salida por consola
  std::stringstream ss;
  ss << "Comenzamos\n";
  double ptoPrevio = 0;
  for (int i=0; i<numTramos-1; i++) {
    ss << "Siga por " << vTramo[i].via << " durante " <<
        vTramo[i].punto_km_abandona - ptoPrevio << " km\n";
    ptoPrevio = vTramo[i].punto_km_abandona;
    if ( vTransito[i] == "SD" ) {
      ss << "Abandone " << vTramo[i].via << " por la salida derecha";
    } else if ( vTransito[i].substr(0, 2) == "R_" ) {
```

```
52        ss << "En la rotonda, salga por la "
53            << vTransito[i].substr(2, vTransito[i].length()-2)
54            << " salida";
55      } else {
56        ss << "Abandone " << vTramo[i].via << " por la salida izquierda";
57      }
58      ss << " direccion " << vTramo[i+1].via << "\n";
59    }
60    ss << "Siga por " << vTramo[numTramos-1].via << " durante "
61            << vTramo[numTramos-1].punto_km_abandona - ptoPrevio
62            << " km\n";
63    ss << "Ha llegado a su destino";
64    std::cout << ss.str() << std::endl;
65    return 0;
66  }
```