# Examen escrito de Programación 1

Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas

6 de septiembre de 2016

- Se debe disponer sobre la mesa en lugar visible un **documento de identificación** provisto de fotografía.
- Se debe escribir **nombre y dos apellidos** en cada una de las hojas de papel que haya sobre la mesa.
- Se debe comenzar a resolver cada uno de los problemas del examen **en una hoja diferente** para facilitar su corrección por profesores diferentes.
- El tiempo total previsto para realizar el examen es de **tres horas**.
- No está permitido utilizar dispositivos electrónicos de ningún tipo, ni consultar libros ni apuntes, excepto los dos documentos facilitados por los profesores de la asignatura: *Guía de sintaxis ANSI/ISO estándar C++* y *Resumen de recursos predefinidos en C++ que son utilizados en la asignatura.*

## Problema 1.º (1 punto)

Especifica y escribe la cabecera y el cuerpo de una función denominada `sumarCiertosMultiplos(nombreFichero, a, b)` que, dado el nombre de un fichero binario, `nombreFichero`, que contiene una secuencia de datos enteros de tipo **`int`**, y dos enteros positivos `a` y `b`, devuelve la suma de los datos del fichero que son múltiplos de `a` pero no de `b`.

# Problema 2.º (2,5 puntos)

Se dispone de un conjunto de ficheros de texto con información meteorológica relativa a rachas de viento observadas en el territorio nacional o en sus cercanías a lo largo del tiempo y registrados a través de sistemas de observación terrestres. Una racha de viento en una localización y hora determinadas se define como la velocidad máxima del viento en esa localización, en los 60 minutos anteriores a la hora indicada, expresada en km/h. Sobre cada racha de viento, se recopila además otro tipo de información.

Los ficheros de texto mencionados se componen de una línea de cabecera, que contiene los nombres de los campos que componen la información sobre rachas de viento que se ofrece, a la que le siguen líneas con información sobre cada racha de viento a razón de una por línea. Sobre cada racha de viento se ofrece la siguiente información y este orden: identificador único de la racha de viento, fecha, hora, latitud y longitud en las que se observó, velocidad del viento, dirección y descripción textual de su localización. La estructura de estos ficheros expresada en notación Backus-Naur es la siguiente:

```
<fichero_rachas> ::= <cabecera> fin_de_línea { <racha> fin_de_línea }
<racha> ::= <identificador> <sep> <fecha> <sep> <hora> <sep> <latitud>
            <sep> <longitud> <sep> <velocidad> <sep> <dirección> <sep>
            <localización>
<cabecera> ::= "Identificador        Fecha           Hora ... Localización"
<identificador> ::= literal_cadena
<fecha> ::= literal_cadena
<hora> ::= literal_cadena
<latitud> ::= literal_real
<longitud> ::= literal_real
<velocidad> ::= literal_real
<dirección> ::= literal_cadena
<localizacion> ::= literal_cadena
<sep> ::= "␣" {"␣"}
```

La longitud máxima de cualquier cadena de caracteres es de 15 caracteres, a excepción de la que describe la `dirección`, que no excede de 3 caracteres, y de la `localización`, que no excede de 140 caracteres.

A continuación, se muestra parte del contenido de un fichero de texto de rachas de viento, a modo de ejemplo.

```
Identificador       Fecha      Hora    Latitud  Longitud  Velocidad  Dirección  Localización
geoss1359568    02/01/2016  01:55:10    39,0315   -2,634       63,98      SE     Sa Pobla (Islas Baleares)
geoss1359598    02/01/2016  18:51:31    37,3524   -2,2295      67,57      S      Valmadrid (Zaragoza)
geoss1359724    04/01/2016  20:22:53    37,5972   -1,5951      37,83     SSO     Lekeitio (Vizcaya)
aemet16ntxs     07/01/2016  03:44:43    42,635    -9,0346      60,91     NNE     Palacios de la Sierra (Burgos)
aemet16nxkt     07/01/2016  22:05:25    42,6534   -8,9592      57,57     SSE     Zaragoza, Valdespartera (Zaragoza)
aemet16ociv     09/01/2016  01:37:31    37,8829   -5,035       48,91      SE     Valdepeñas (Ciudad Real)
```

De entre estos datos, en este problema y los dos siguientes nos van a interesar exclusivamente los datos relativos al identificador, latitud, longitud, velocidad y dirección de cada racha de viento. Se encuentra disponible un módulo de biblioteca denominado `racha-viento` cuyo fichero de interfaz (`racha-viento.h`) se muestra a continuación y que puede y debe ser utilizado en la resolución de este problema.

```
#ifndef RACHA_VIENTO_H
#define RACHA_VIENTO_H

const int MAX_LONG_IDENTIFICADOR = 16;
const int MAX_LONG_DIRECCION = 4;

/*
 * Almacena parte de la información asociada a una racha de viento:
 * identificador, latitud, longitud, dirección y velocidad.
 */
struct RachaViento {
    char identificador[MAX_LONG_IDENTIFICADOR];
    double latitud;
    double longitud;
    char direccion[MAX_LONG_DIRECCION];
    double velocidad;
};

/*
 * Pre:  -90 <= latitud <= 90; -180 <= longitud <= 180; velocidad > 0.0;
 *       longitud de identificador < MAX_LONG_IDENTIFICADOR y
 *       longitud de direccion < MAX_LONG_DIRECCION.
 * Post: Ha devuelto un registro de tipo «RachaViento» cuyos campos tomado
 *       los valores correspondientes de los parámetros de mismo nombre.
 */
RachaViento definirRachaViento(const char identificador[],
        const double latitud, const double longitud,
        const char direccion[], const double velocidad);

/*
 * Pre:  La cadena de caracteres a la que hace referencia el parámetro
 *       «identificador» tiene una capacidad suficiente como para almacenar
 *       el identificador de la racha de viento «racha».
 * Post: Ha copiado el identificador de la racha de viento «racha» en la
 *       cadena a la que hace referencia el parámetro «identificador».
 */
void identificador(const RachaViento racha, char identificador[]);
```

```
/*
 * Pre:  ---
 * Post: Ha devuelto la latitud de la racha de viento «racha».
 */
double latitud(const RachaViento racha);

/*
 * Pre:  ---
 * Post: Ha devuelto la longitud de la racha de viento «racha».
 */
double longitud(const RachaViento racha);

/*
 * Pre:  La cadena de caracteres a la que hace referencia el parámetro
 *       «direccion» tiene una capacidad suficiente como para almacenar
 *       el valor de la dirección de la racha de viento «racha».
 * Post: Ha copiado la dirección de la racha de viento «racha» en la cadena de
 *       caracteres a la que hace referencia el parámetro «direccion».
 */
void direccion(const RachaViento racha, char direccion[]);

/*
 * Pre:  ---
 * Post: Ha devuelto la velocidad que registró la racha de viento «racha».
 */
double velocidad(const RachaViento racha);
#endif
```

Se pide el código de la siguiente función:

```
/*
 * Pre:  «nombreFichero» representa el nombre de un fichero de texto existente
 *       que almacena información sobre rachas de viento de acuerdo con la
 *       sintaxis presentada en el enunciado; la tabla «rachas» tiene
 *       capacidad suficiente como para almacenar todos las rachas de viento
 *       contenidas en el fichero de nombre «nombreFichero».
 * Post: Si se ha podido abrir sin problemas el fichero cuyo nombre es
 *       «nombreFichero», ha copiado la información sobre rachas de viento
 *       contenida en el fichero en las primeras «numRachas» componentes de
 *       la tabla «rachas» y ha devuelto true. En caso contrario, ha
 *       devuelto false, sin escribir ningún mensaje de error en ninguna parte.
 */
bool leerFichero(const char nombreFichero[],
                 RachaViento rachas[], int& numRachas);
```

# Problema 3.º (2,5 puntos)

Escribir el código de la siguiente función:

```
/*
 * Pre:  La tabla «rachasFiltradas» tiene capacidad suficiente como para
 *       albergar el número de rachas de viento que hay que filtrar de acuerdo
 *       con la postcondición; «numRachasTotales» >= 0; -90 <= latitud <= 90
 *       y -180 <= longitud <= 180.
 * Post: Ha copiado en las primeras «numRachasFiltradas» componentes de la
 *       tabla «rachasFiltradas» todos aquellos rachas de viento almacenados
 *       en las primeras «numRachasTotales» componentes de la tabla
 *       «rachasTotales» que se produjeron dentro del área de la superficie
 *       terrestre definida por el rectángulo de vértices
 *       «latitud» ± UMBRAL_LATITUD y «longitud» ± UMBRAL_LONGITUD.
 */
void filtrar(const RachaViento rachasTotales[], const int numRachasTotales,
             RachaViento rachasFiltradas[], int& numRachasFiltradas,
             double latitud, double longitud)
```

Los identificadores `UMBRAL_LATITUD` y `UMBRAL_LONGITUD` corresponden con dos constantes reales ya definidas de forma global en el ámbito de la función `filtrar` y cuyo valor concreto no es relevante para la resolución del ejercicio. El tipo `RachaViento` es el definido en el problema 2.º.

# Problema 4.º (4 puntos)

Se debe escribir un programa que solicite al usuario el nombre de un fichero de texto con información sobre rachas de viento que siga la sintaxis explicada en el problema 2.º. Si el fichero se puede abrir sin problemas, solicita unos valores de latitud $lat$ y longitud $long$ y finalmente muestra un listado de las 10 rachas de viento de mayor velocidad (ordenadas de mayor a menor) que se encuentran en el área de la superficie terrestre cuya latitud sea $lat \pm UMBRAL\_LATITUD$ y cuya longitud sea $long \pm UMBRAL\_LONGITUD$.

En el caso de que el fichero no pudiera abrirse, el programa se limitará a mostrar un mensaje de error en la pantalla.

Se muestran a continuación dos ejemplos de ejecución del programa solicitado:

```
Escriba el nombre de un fichero de rachas de viento: rachas-2016.txt

Escriba un valor de latitud: 41.65
Escriba un valor de longitud: -0.8833

RACHAS DE VIENTO MÁS INTENSAS CERCANAS A LAS COORDENADAS DADAS
============================================================

   Identificador   Latitud  Longitud Dirección Velocidad
    geoss1359908   35.6004   -3.8056       NNE     122.0
    geoss1359598   37.3524   -2.2295         S      67.6
    geoss1359568   39.0315   -2.6340        SE      64.0
     aemet16ntxs   42.6350   -9.0346       NNE      60.9
     aemet16nxkt   42.6534   -8.9592       SSE      57.6
    geoss1359805   39.0377   -2.6411       ENE      54.2
     aemet16ociv   37.8829   -5.0350        SE      48.9
    geoss1359831   38.5305   -0.8772         O      40.5
    geoss1359724   37.5972   -1.5951       SSO      37.8
    geoss1359878   37.7351   -1.6331       ENE      36.8
```

```
Escriba el nombre de un fichero de rachas de viento: rachas-2017.txt

Error al abrir el fichero "rachas-2017.txt".
```

En la solución presentada se exigirá, además de su corrección, las siguientes propiedades:

1. Independencia del código de la forma en que haya sido definido el tipo `RachaViento`. Puede y debe utilizarse el módulo `rachas-viento` cuya interfaz se ha proporcionado en el problema 2.º.
2. Diseño descendente.
3. Legibilidad del código.
4. Minimización del esfuerzo de desarrollo por haber hecho uso adecuado del código solicitado en los problemas 2.º y 3.º (funciones `leerFichero` y `filtrar`, respectivamente).

# Solución al problema 1.º

```
#include <iostream>
#include <fstream>
using namespace std;

/*
 * Pre:  «nombreFichero» representa el nombre de un fichero binario existente
 *       que almacena una secuencia de enteros y «a» y «b» son positivos.
 * Post: Ha devuelto el valor de la suma de los datos almacenados en el fichero
 *       de nombre «nombreFichero» que son múltiplos de «a» pero no de «b».
 */
int sumarCiertosMultiplos(const char nombreFichero[],
                          const int a, const int b) {
    int suma = 0;
    ifstream f;
    f.open(nombreFichero, ios::binary);
    if (f.is_open()) {
        int datoLeido;
        f.read(reinterpret_cast<char*>(&datoLeido), sizeof(datoLeido));
        while (!f.eof()) {
            if (datoLeido % a == 0 && datoLeido % b != 0) {
                suma += datoLeido;
            }
            f.read(reinterpret_cast<char*>(&datoLeido), sizeof(datoLeido));
        }
        f.close();
    }
    else {
        cerr << "Error al abrir el fichero \"" << nombreFichero << "\"" << endl;
    }
    return suma;
}
```

# Solución a los problemas 2.º, 3.º y 4.º

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cmath>
#include "racha-viento.h"

using namespace std;

const int MAX_LONG_NOMBRE_FICHERO = 120;
const int MAX_RACHAS = 1200;
const int NUMERO_RESULTADOS_ORDENADOS = 10;
const int LONG_CABECERA = 150;
const int ANCHO_VISUALIZACION = 10;
const int MAX_LONG_GENERAL = 100;
const double UMBRAL_LATITUD = 10.0;
const double UMBRAL_LONGITUD = 10.0;

/*
 * Pre:  «nombreFichero» representa el nombre de un fichero de texto existente
 *       que almacena información sobre rachas de viento de acuerdo con la
 *       sintaxis presentada en el enunciado; la tabla «rachas» tiene
 *       capacidad suficiente como para almacenar todos las rachas de viento
 *       contenidas en el fichero de nombre «nombreFichero».
 * Post: Si se ha podido abrir sin problemas el fichero cuyo nombre es
 *       «nombreFichero», ha copiado la información sobre rachas de viento
 *       contenida en el fichero en las primeras «numRachas» componentes de
 *       la tabla «rachas» y ha devuelto true. En caso contrario, ha
 *       devuelto false, sin escribir ningún mensaje de error en ninguna parte.
 */
bool leerFichero(const char nombreFichero[],
                 RachaViento rachas[], int& numRachas) {
    ifstream f;
    f.open(nombreFichero);
    if (f.is_open()) {
        char cabecera[LONG_CABECERA];
        f.getline(cabecera, LONG_CABECERA);

        char idRacha[MAX_LONG_IDENTIFICADOR],
             fecha[MAX_LONG_GENERAL], hora [MAX_LONG_GENERAL],
             direccion[MAX_LONG_DIRECCION], resto[MAX_LONG_GENERAL];
        double latitud, longitud, velocidad;
```

```
        numRachas = 0;
        f >> idRacha >> fecha >> hora >> latitud >> longitud >> velocidad
          >> direccion;
        f.getline(resto, MAX_LONG_GENERAL);

        while (!f.eof()) {
            numRachas++;
            rachas[numRachas] = definirRachaViento(idRacha, latitud,
                    longitud, direccion, velocidad);
            f >> idRacha >> fecha >> hora >> latitud >> longitud >> velocidad
              >> direccion >> resto;
            f.getline(resto, MAX_LONG_GENERAL);
        }
        f.close();
        return true;
    }
    else {
        return false;
    }
}

/*
 * Pre:  ---
 * Post: Ha devuelto true si y solo si la racha de viento «racha» se produjo
 *       dentro del área de la superficie terrestre definida por el rectángulo
 *       de vértices «latitudInteres» +/- UMBRAL_LATITUD y
 *       «longitudInteres» +/- UMBRAL_LONGITUD.
 */
bool estaCerca(const RachaViento racha,
               const double latitudInteres, const double longitudInteres) {
    return abs(latitud(racha) - latitudInteres) <= UMBRAL_LATITUD
            && abs(longitud(racha) - longitudInteres) <= UMBRAL_LONGITUD;
}

/*
 * Pre:  La tabla «rachasFiltradas» tiene capacidad suficiente como para
 *       albergar el número de rachas de viento que hay que filtrar de acuerdo
 *       con la postcondición; «numRachasTotales» >= 0; -90 <= latitud <= 90
 *       y -180 <= longitud <= 180.
 * Post: Ha copiado en las primeras «numRachasFiltradas» componentes de la
 *       tabla «rachasFiltradas» todos aquellos rachas de viento almacenados
 *       en las primeras «numRachasTotales» componentes de la tabla
 *       «rachasTotales» que se produjeron dentro del área de la superficie
```

```
 *       terrestre definida por el rectángulo de vértices
 *       «latitud» +/- UMBRAL_LATITUD y «longitud» +/- UMBRAL_LONGITUD.
 */
void filtrar(const RachaViento rachasTotales[], const int numRachasTotales,
             RachaViento rachasFiltradas[], int& numRachasFiltradas,
             double latitud, double longitud) {
    numRachasFiltradas = 0;
    for (int i = 0; i < numRachasTotales; i++) {
        if (estaCerca(rachasTotales[i], latitud, longitud)) {
            rachasFiltradas[numRachasFiltradas] = rachasTotales[i];
            numRachasFiltradas++;
        }
    }
}

/*
 * Pre:  racha1 = R1 y racha2 = R2.
 * Post: racha1 = R2 y racha2 = R1.
 */
void intercambia(RachaViento& racha1, RachaViento& racha2) {
    RachaViento auxiliar = racha1;
    racha1 = racha2;
    racha2 = auxiliar;
}

/*
 * Pre:  «numRachas» > 0 y «numResultadosOrdenados» >= 0.
 * Post: Las primeras «numRachas» componentes de la tabla «rachasViento» son una
 *       permutación de los datos que contenían inicialmente y las primeras
 *       «numResultadosOrdenados» contienen las rachas de viento de mayores
 *       velocidades, ordenadas de mayor a menor.
 */
void ordenarPorVelocidad(RachaViento rachasViento[], const int numRachas,
                         const int numResultadosOrdenados) {
    int limiteOrdenacion = min(numRachas, numResultadosOrdenados);
    for (int i = 0; i < limiteOrdenacion; i ++) {
        int iMayorVelocidad = i;
        for (int j = i+1; j < numResultadosOrdenados; j++) {
            if (velocidad(rachasViento[j])
                    > velocidad(rachasViento[iMayorVelocidad])) {
                iMayorVelocidad = j;
            }
        }
        intercambia(rachasViento[i], rachasViento[iMayorVelocidad]);
```

```
    }
}

/*
 * Pre:  «numRachas» > 0 y «numResultadosOrdenados» >= 0.
 * Post: Ha escrito en la pantalla una cabecera y los datos de las primeras
 *       «numResultadosOrdenados» componentes de la tabla «rachas» (o solo las
 *       «numRachas» componentes de la tabla, si este último número es menor),
 *       de acuerdo con el formato establecido en el enunciado.
 */
void mostrar(const RachaViento rachas[], const int numRachas,
             const int numResultadosOrdenados) {

    cout << endl;
    cout << "RACHAS␣DE␣VIENTO␣MÁS␣INTENSAS␣CERCANAS␣A␣LAS␣COORDENADAS␣DADAS"
         << endl;
    cout << "================================================================"
         << endl;
    cout << endl;

    cout << setw(MAX_LONG_IDENTIFICADOR) << "Identificador"
         << setw(ANCHO_VISUALIZACION) << "Latitud"
         << setw(ANCHO_VISUALIZACION) << "Longitud"
         << setw(ANCHO_VISUALIZACION) << "Dirección"
         << setw(ANCHO_VISUALIZACION) << "Velocidad"
         << endl;
    cout << fixed;
    int limite = min(numRachas, numResultadosOrdenados);
    for (int i = 0; i < limite; i++) {
        char idRacha[MAX_LONG_IDENTIFICADOR];
        char direccionRacha[MAX_LONG_DIRECCION];
        identificador(rachas[i], idRacha);
        direccion(rachas[i], direccionRacha);
        cout << setw(MAX_LONG_IDENTIFICADOR) << idRacha
             << setprecision(4)
             << setw(ANCHO_VISUALIZACION) << latitud(rachas[i])
             << setw(ANCHO_VISUALIZACION) << longitud(rachas[i])
             << setw(ANCHO_VISUALIZACION) << direccionRacha
             << setprecision(1)
             << setw(ANCHO_VISUALIZACION) << velocidad(rachas[i])
             << endl;
    }
}
```

```
/*
 * Pre:  ---
 * Post: Este programa ha solicitado al usuario el nombre de un fichero de texto
 *       con información sobre rachas de viento que sigue la sintaxis explicada
 *       en el enunciado del problema 2.º. Si el fichero se ha podido abrir sin
 *       problemas, ha solicitado unos valores de latitud «lat» y longitud
 *       «long» y finalmente ha mostrado el listado de las 10 rachas de viento
 *       de mayor velocidad (ordenadas de mayor a menor) que se encuentran en el
 *       area de la superficie terrestre cuya latitud sea
 *       «lat» +/- UMBRAL_LATITUD$ y cuya longitud sea
 *       «long» +/- UMBRAL_LONGITUD.
 *       En el caso de que el fichero no haya podido abrirse, el programa se ha
 *       limitado a mostrar un mensaje de error en la pantalla.
 */
int main() {
    cout << "Escriba␣el␣nombre␣de␣un␣fichero␣de␣rachas␣de␣viento:␣" << flush;

    char nombreFichero[MAX_LONG_NOMBRE_FICHERO];
    cin >> nombreFichero;

    RachaViento rachasLeidas[MAX_RACHAS];
    int numRachasLeidas;

    if (leerFichero(nombreFichero, rachasLeidas, numRachasLeidas)) {
        double latitud, longitud;
        cout << endl;
        cout << "Escriba␣un␣valor␣de␣latitud:␣" << flush;
        cin >> latitud;

        cout << "Escriba␣un␣valor␣de␣longitud:␣" << flush;
        cin >> longitud;

        RachaViento rachasViento[MAX_RACHAS];
        int numRachasViento;

        filtrar(rachasLeidas, numRachasLeidas,
                rachasViento, numRachasViento,
                latitud, longitud);
        ordenarPorVelocidad(rachasViento, numRachasViento,
                            NUMERO_RESULTADOS_ORDENADOS);
        mostrar(rachasViento, numRachasViento, NUMERO_RESULTADOS_ORDENADOS);
        return 0;
    }
    else {
```

```
        cout << endl;
        cerr << "Error␣al␣abrir␣el␣fichero␣\"" << nombreFichero << "\".";
        return 1;
    }
}
```