LENGUAJES DE PROGRAMACIÓN

Solución al examen de Septiembre 2018

PREGUNTA 1 (3 puntos)

Indique la veracidad o falsedad de cada una de las afirmaciones siguientes, explicando detalladamente en cada caso el motivo de su respuesta.

A. (0.5 puntos) En C++, los datos Booleanos son escritos en el flujo de salida por defecto como *true* y *false*.

B. (0.5 puntos) Una variable local no puede ocultar a una variable global del programa.

C. (0.5 puntos) La variable `var` declarada en la sentencia

```
std::list<int>::iterator var;
```

es una lista doblemente enlazada.

D. (0.5 puntos) En C++, los operadores aritméticos tienen mayor precedencia que los relacionales.

E. (0.5 puntos) Las funciones no pueden formar parte de expresiones.

F. (0.5 puntos) En C y C++, todos los parámetros de las funciones se pasan por referencia.

Solución a la Pregunta 1

A	Falso	Véase la página 98 del texto base.
B	Falso	Véase la página 115 del texto base.
C	Falso	Véase la página 458 del texto base.
D	Verdadero	Véase la página 176 del texto base.
E	Falso	Véase la página 347 del texto base.
F	Falso	Véase la página 349 del texto base.

PREGUNTA 2 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>

int x1 = 10;

int main () {
   int x1, x2;
   int xv[3] = {1, 2, 3};
   int *p1, *p2;
   p1 = &xv[0];
   x1 = xv[2];
   { // Inicio bloque 1
     std::cout << x1  << std::endl;
     std::cout << *p1 << std::endl;
     int x1 = 4;
     std::cout << ::x1    << " " << x1 << std::endl;
     std::cout << *(p1++) << std::endl;
     { // Inicio bloque 2
       x2 = x1;
       p2 = &x2;
       std::cout << x2      << std::endl;
       std::cout << *(++p1) << std::endl;
       std::cout << *p2     << std::endl;
     }  // Fin bloque 2
   } // Fin bloque 1
   p1  = p2;
   *p1 = -2;
   std::cout << *p1 << " " << *p2 << std::endl;
   return 0;
}
```

Solución a la Pregunta 2

```
3
1
10 4
1
4
3
4
-2 -2
```

PREGUNTA 3 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <queue>
#include <string>

int main() {
    std::queue<int> var;
    for (int i=0; i<3; i++) {
       var.push(i*i);
    }
    std::cout << var.front() << " ";
    std::cout << var.back()  << " ";
    std::cout << var.size()  << std::endl;
    while (!var.empty()) {
       std::cout << var.front() << " ";
       var.pop();
    }
    std::cout << var.size() << std::endl;
    return 0;
}
```

Solución a la Pregunta 3

```
0 4 3
0 1 4 0
```

PREGUNTA 4 (2.5 puntos)

El encargado de un casino desea evaluar si es necesario cambiar una de sus ruletas. La ruleta se considera que funciona adecuadamente si la probabilidad de que la pelota caiga en cada uno de los 36 números muestra una distribución aleatoria uniforme. Para realizar esta tarea, se quiere aplicar la *prueba χ^2 de Pearson* a las jugadas producidas con la ruleta durante el último año. Cuanto menor sea el valor χ^2 de Pearson, más se acerca la ruleta al comportamiento deseado.

El valor χ^2 de Pearson se calcula de la forma siguiente:

$$\chi^2 = \sum_{i=1}^{N} \frac{(\text{valor observado}_i - \text{valor teórico})^2}{\text{valor teórico}} \tag{1.1}$$

donde:

- N es el número de casos.
- valor observado$_i$ es el número de experimentos de la muestra a estudiar para el caso i.
- valor teórico corresponde con el valor que tendría que tener la muestra en el caso de que haya una distribución uniforme. Se calcula como el cociente entre el número total de experimentos y el número de casos.

Por ejemplo, consideremos que una muestra total de 100 experimentos de una ruleta tiene la siguiente distribución para cada caso:

Número 1: 2
Número 2: 3
Número 3: 2
...
Número 36: 1

El valor de χ^2 de Pearson se calcula en este ejemplo de la forma siguiente:

$$\begin{aligned}\chi^2 &= \sum_{i=1}^{N} \frac{(\text{valor observado}_i - 100/36)^2}{100/36} \\ &= \frac{(2-2.7777)^2}{2.7777} + \frac{(3-2.7777)^2}{2.7777} + \frac{(2-2.7777)^2}{2.7777} + \cdots + \frac{(1-2.7777)^2}{2.7777}\end{aligned}$$

En un fichero de texto llamado *ruleta.txt* se ha almacenado en cada fila el número de veces que la pelota ha caído en cada uno de los 36 posible casos (números).

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

1. Abrir el fichero de texto llamado *ruleta.txt* para lectura. Si la operación no se realiza adecuadamente, mostrar en la consola un mensaje de error y terminar.
2. Almacenar los datos leídos del fichero en una lista de números enteros llamada *experimentos*.
3. Cerrar el fichero.
4. A partir de los valores almacenados en la lista, calcular el número de experimentos total que se han producido en la ruleta en el último año, y que se calcula como la suma de los experimentos de cada uno de los 36 casos. Almacenar el resultado en una variable entera llamada *numTotal* y mostrar en la consola su valor.
5. Calcular el valor teórico de la fórmula (1.1) del enunciado, empleando el valor obtenido en el apartado anterior. Mostrar el valor teórico en la consola, en formato fijo, con tres dígitos detrás del punto decimal.
6. Calcular el valor de χ^2 de Pearson según la fórmula (1.1) y mostrarlo en la consola, en formato científico, con 4 dígitos detrás del punto decimal.

Solución a la Pregunta 4

```
#include <iostream>
#include <fstream>
#include <list>
#include <cmath>
#include <iomanip>

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

int main() {
    // Apertura del fichero para lectura
        std::ifstream file_in(nombreFich.c_str(),std::ios::in);
    if (!file_in) {
        std::cout << "No se puede abrir el fichero"
            << std::endl;
        return 0;
    }
    // Lectura de los datos y carga en la lista
    std::list<int> experimentos;
    while (!file_in.eof()) {
        int d;
        file_in >> d;
        // Si fin de fichero, entonces salir del bucle while
        if (file_in.eof()) break;
        experimentos.push_back(d);
    }
    // Cerrar el fichero
    file_in.close();
    // Número total de experimentos
    int numTotal = 0;
    std::list<int>::iterator p = experimentos.begin();
    while ( p != experimentos.end() ) {
        numTotal += *p;
        p++;
    }
    std::cout << "Num. experimentos: " << numTotal << std::endl;
    // Valor teórico
    double valorTeorico = (double)numTotal / experimentos.size();
    std::cout << "Valor teorico: "
          << std::fixed << std::setprecision(3)
          << valorTeorico << std::endl;
    // Valor chi-cuadrado
    p = experimentos.begin();
    double chiCuadrado = 0;
    while ( p != experimentos.end() ) {
        chiCuadrado += std::pow(*p - valorTeorico,2);
        p++;
    }
    chiCuadrado /= valorTeorico;
    std::cout << "Chi-cuadrado = "
          << std::scientific << std::setprecision(4)
          << chiCuadrado << std::endl;
    return 0;
}
```

PREGUNTA 5 (2.5 puntos)

El *método de integración de Runge-Kutta de orden 2* permite resolver numéricamente problemas de valores iniciales descritos mediante ecuaciones diferenciales ordinarias.

Consideremos el problema siguiente:

$$\begin{aligned} \frac{dy}{dt} &= 0.5 \cdot e^{-(t-\mu_1)^2} + e^{-(t-\mu_2)^2} + \lambda \cdot y, \qquad y \in \mathbb{R}, \qquad t > 0 \\ y(t_0) &= y_0 \end{aligned}$$

donde:

t representa el tiempo.

μ_1, μ_2, λ tienen los siguientes valores constantes conocidos:

$$\mu_1 = 7 \qquad \mu_2 = 4 \qquad \lambda = -2$$

La condición inicial (t_0, y_0) es conocida.

El método de Runge-Kutta de orden 2 permite calcular el valor de la variable y en los instantes de tiempo

$$t_n = t_0 + n \cdot h \qquad \text{para} \qquad n = 1, 2, \ldots, N$$

siendo h un valor constante conocido, denominado *tamaño del paso en el tiempo del método de integración*.

Sea y_n el valor de la variable y calculado en el instante de tiempo t_n.

El método de Runge-Kutta de orden 2 consiste en la aplicación del esquema iterativo siguiente:

$$\begin{aligned} a_n &= f(t_n, y_n) \\ b_n &= f(t_n + \tfrac{1}{2} \cdot h, y_n + \tfrac{1}{2} \cdot h \cdot a_n) \\ y_{n+1} &= y_n + h \cdot b_n \end{aligned} \tag{1.2}$$

para $n = 0, 1, \ldots, N$

donde, en el problema que nos ocupa,

$$f(t, y) = 0.5 \cdot e^{-(t-\mu_1)^2} + e^{-(t-\mu_2)^2} + \lambda \cdot y \tag{1.3}$$

Escriba el código C++ indicado a continuación:

5.a) (0.5 puntos) Una función en C++ que implemente la función definida en (1.3). A continuación se muestra el prototipo de la función:

```
double f(double t, double y)
          throw (std::invalid_argument);
```

La función tiene dos argumentos reales (t, y).

Si el valor de t es menor que cero, la función lanza una excepción. En caso contrario, devuelve el valor $f(t, y)$ calculado como se indica en (1.3).

En el propio cuerpo de la función, declare μ_1, μ_2, λ como constantes llamadas `mu_1`, `mu_2` y `lambda`, asignándoles respectivamente los valores 7, 4, -2.

5.b) (2 puntos) Escriba un programa en C++ que realice las acciones siguientes:

1. Declarar las tres constantes globales indicadas a continuación, asignándoles los correspondientes valores.

Nombre	**Valor**
h	0.01
N	1200
I	100

2. Mediante la escritura de un mensaje en la consola, solicitar al usuario que éste introduzca por consola el valor de t_0 y el valor de y_0. Leer los valores, almacenándolos en sendas variables.
3. Aplicar el método de Runge-Kutta de orden 2 para obtener los valores y_1, y_2, $\ldots$, y_N. Invocar para ello la función definida al contestar a la Pregunta 5.a. Almacenar los valores y_0, y_1, $\ldots$, y_N en un vector de elementos reales llamado `yVals`.
4. Escribir en la consola el valor de los componentes del vector `yVals` cuyo número de componente sea cero o múltiplo entero de I, así como el correspondiente valor del tiempo para cada uno de los componentes anteriormente indicados.
5. Terminar.

Solución a la Pregunta 5

```
#include <iostream>
#include <vector>
#include <cmath>
#include <stdexcept>

double f(double t, double y)
        throw (std::invalid_argument);

const double h = 0.01;
const int N = 1200;
const int I = 100;

int main() {
    // Entrada por consola
    double t0;
    std::cout << "Introduzca t0:" << std::endl;
    std::cin >> t0;
    double y0;
    std::cout << "Introduzca y0:" << std::endl;
    std::cin >> y0;
    // Valor inicial
    std::vector<double> yVals;
    yVals.push_back(y0);
    // R-K orden 2
    for (unsigned n=0; n<N; n++) {
        double tn = t0 + n*h;
        try {
            double an = f(tn, yVals[n]);
            double bn = f(tn+0.5*h, yVals[n]+0.5*h*an);
            yVals.push_back(yVals[n]+h*bn);
        } catch (std::invalid_argument exc) {
            std::cout << exc.what() << std::endl;
            return 0;
        }
    }
    // Salida por consola
    unsigned int i = 0;
    while ( i < yVals.size() ) {
        std::cout << "t = " << t0 + i*h
              << "\t y = " << yVals[i] << std::endl;
        i += I;
    }
    return 0;
}
```

```
double f(double t, double y)
            throw (std::invalid_argument){
    if ( t < 0 )
        throw std::invalid_argument("t menor que cero");
    const double mu_1 = 7;
    const double mu_2 = 4;
    const double lambda = -2;
    return    0.5*std::exp(-std::pow(t-mu_1,2))
        + std::exp(-std::pow(t-mu_2,2))
        + lambda * y;

}
```