LENGUAJES DE PROGRAMACIÓN

Solución al examen de Septiembre 2014

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) Las palabras binarias almacenadas en memoria que son leídas por la unidad aritmética de la máquina de von Neumann son interpretadas por dicha unidad como instrucciones.

B. (0.5 puntos) La parte del programa en que una variable es visible es siempre igual al ámbito de dicha variable.

C. (0.5 puntos)

C. (0.5 puntos) Una variable del tipo `Estrella`, declarado en la siguiente sentencia C++

```
union Estrella { int i; char c; };
```

contiene en cada instante simultáneamente un valor de tipo **int** y otro de tipo **char**.

D. (0.5 puntos) En notación prefija, el producto de `+ab` y `c` se escribe `+*bca`.

E. (0.5 puntos) En el cuerpo de una función en C++ pueden escribirse una o varias sentencias **return**.

F. (0.5 puntos) El algoritmo de ordenación por mezcla sigue el algoritmo de la fuerza bruta.

Solución a la Pregunta 1

A	Falsa	Véase la página 37 del texto base.
B	Falsa	Véase la página 115 del texto base.
C	Falsa	Véase la página 120 del texto base.
D	Falsa	Véase la página 172 del texto base.
E	Verdadera	Véase la página 379 del texto base.
F	Falsa	Véase la página 512 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 main () {
   int* p;
   int a[2];
   int j;
   a[0] = 0;
   a[1] = 1;
   j = a[0];
   p = &a[0];
   std::cout << p[0] << std::endl;
   std::cout << p[1] << std::endl;
   *p = 10;
   p++;
   j  = *p;
   std::cout << a[0] << std::endl;
   std::cout << j    << std::endl;
   return 0;
 }
```

Solución a la Pregunta 2

```
0
1
10
1
```

PREGUNTA 3 (1 punto)

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

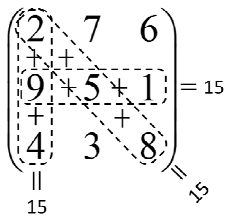
```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

int main() {
   std::vector<int> v;
   for (int i=0; i<5; i++)
      v.push_back(i%2);

   for (unsigned int i=0; i<v.size()-1; i++)
      std::cout << v[i] << ",";
   std::cout << v[v.size()-1] << std::endl;

   std::vector<int>::iterator pBegin = v.begin()+2;
   std::vector<int>::iterator pEnd   = v.begin()+4;
   std::cout << (*pBegin) << ", " << (*pEnd) << std::endl;

   reverse(pBegin,pEnd);
   for (unsigned int i=0; i<v.size()-1; i++)
      std::cout << v[i] << ",";
   std::cout << v[v.size()-1] << std::endl;

   std::cout << (*pBegin) << ", " << (*pEnd) << std::endl;
   return 0;
}
```

Solución a la Pregunta 3

```
0,1,0,1,0
0, 0
0,1,1,0,0
1, 0
```

PREGUNTA 4 (2.5 puntos)

Un cuadrado mágico consiste en una matriz cuadrada $N \times N$ de números enteros que tiene la propiedad siguiente: la suma de cada una de las columnas, filas y diagonales principales da un mismo resultado, al cual se denomina "constante mágica". En la figura mostrada más abajo se muestra un cuadrado mágico de dimensión $N = 3$, con una constante mágica 15.



Se propone la realización de un programa que permita comprobar si una matriz cuadrada de dimensión menor o igual que 5 es un cuadrado mágico. Escriba un programa en C++ que realice las acciones siguientes:

1. Escribir un mensaje en la consola en el cual solicite al usuario que éste introduzca por consola la dimensión (N) de la matriz cuadrada que se desea comprobar.
2. Comprobar que el valor entero introducido por el usuario es mayor que uno, y menor o igual que cinco. Si no lo es, mostrar un mensaje indicándolo y terminar.
3. Solicitar al usuario que introduzca por la consola los elementos de la matriz y almacenarlos en un array de dimensión 2 llamado `M`.
4. Calcular la suma de los elementos de la primera fila de la matriz. Almacenar dicha suma en una variable entera llamada `numMagico`.
5. Ir comprobando si la suma de cada una de las filas, cada una de las columnas y cada una de las diagonales principales es igual a la variable `numMagico`. Si una de las sumas no es igual a `numMagico`, mostrar en la consola un mensaje indicando que la matriz no es un cuadrado mágico y terminar.
6. Mostrar en la consola un mensaje indicando que la matriz es un cuadrado mágico.
7. Terminar.

Solución a la Pregunta 4

```
#include <iostream>

const int Nmax = 5;   // Maxima dimension matriz

int main () {
    // Dimension de la matriz
    std::cout << "Dimension de la matriz. N = ";
    int N;   std::cin >> N;
    if ( N <= 1 ||N > Nmax ) {
         std::cout << "Dimension no valida"<< std::endl;
         return 0;
    }
    // Declaración de la matriz y entrada elementos
    int M[Nmax][Nmax];
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; j++) {
            std::cout << "M["<< i+1 << ","<< j+1 << "] = ";
            std::cin  >> M[i][j];
        }
    }
    // Suma elementos primera fila
    int numMagico = 0;
    for (int j=0; j<N; j++) {
        numMagico += M[0][j];
    }
    // Comprobar filas, columnas y diagonales
    int sumaDiag1 = 0, sumaDiag2 = 0;
    for (int i=0; i<N; i++) {
        int sumaFila = 0, sumaCol = 0;
        for (int j=0; j<N; j++) {
            sumaFila += M[i][j];
            sumaCol  += M[j][i];
        }
        if ( sumaFila!=numMagico ||sumaCol!=numMagico ) {
            std::cout << "No es cuadrado magico"<< std::endl;
            return 0;
        }
        sumaDiag1 += M[i][i];
        sumaDiag2 += M[N-1-i][N-1-i];
    }
    if ( sumaDiag1!=numMagico ||sumaDiag2!=numMagico ) {
        std::cout << "No es cuadrado magico"<< std::endl;
        return 0;
    }
    std::cout << "Es cuadrado magico"<< std::endl;
    return 0;
}
```

PREGUNTA 5 (2.5 puntos)

Se propone la realización de un programa que resuelva la ecuación diferencial ordinaria

$$\frac{dx}{dt} = \cos(x \cdot t) \cdot t - \sin(t) \tag{1.1}$$

empleando el método de integración de Runge-Kutta de 4^0 orden y que escriba el resultado en un fichero de texto. La función de paso del método de Runge-Kutta de 4^0 orden es la siguiente:

$$\begin{aligned} k_1 &= f(t_i, x_i) && (1.2)\\ k_2 &= f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} \cdot k_1\right) && (1.3)\\ k_3 &= f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} \cdot k_2\right) && (1.4)\\ k_4 &= f\left(t_i + h, x_i + h \cdot k_3\right) && (1.5)\\ x_{i+1} &= x_i + \frac{h}{6} \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4) && (1.6) \end{aligned}$$

El valor de x en el instante t_i se denomina x_i. Conocidos t_i y x_i, y teniendo en cuenta que en este problema

$$f(t, x) = \cos(x \cdot t) \cdot t - \sin(t) \tag{1.7}$$

de las Ecs. (1.2)–(1.5) se calculan k_1, k_2, k_3 y k_4. A continuación, de la Ec. (1.6) se calcula x_{i+1}, que es como se denomina al valor de x en el instante $t_{i+1} = t_i + h$.

El parámetro h, que es el tamaño del paso de integración, puede calcularse de la Ec. (1.8),

$$h = \frac{t_1 - t_0}{n_{\text{Pasos}}} \tag{1.8}$$

a partir de las constantes siguientes:

Tipo	Nombre	Significado	Valor
double	t_0	Valor inicial de t	0
double	t_1	Valor final de t	10.0
double	x_0	Valor de x en el instante t_0	0.01
int	n_{Pasos}	Número de pasos de integración	2000

Escriba el código C++ indicado a continuación.

5.a (0.5 puntos) Escriba una función en C++ que implemente la función $f(t, x)$ descrita en la Ec. (1.7).

5.b (1 punto) Escriba una función en C++ cuyo prototipo sea:

```
void escribeFich(std::string nombreFich,
                 double t0, double h,
                 std::vector<double> &x)
                    throw (std::invalid_argument);
```

y que realice las acciones siguientes:

1. Abrir para escritura el fichero de texto cuyo nombre está almacenado en la variable `nombreFich`. Si no es posible abrir el fichero, la función debe lanzar una excepción.
2. Escribir en el fichero dos columnas de números. En la segunda columna debe escribir el contenido del vector `x`. La primera columna debe tener el mismo número de elementos que la segunda. El elemento `i`-ésimo de la primera columna es el resultado de evaluar `t0+(i-1)*h`.

   Todos los números deben escribirse en formato fijo. Los de la primera columna con 3 dígitos decimales y los de la segunda columna con 10 dígitos decimales. Las dos columnas deben separarse mediante tabulador.
3. Cerrar el fichero.
4. Devolver el control.

5.c (1 punto) Escriba el programa principal, que debe realizar las acciones siguientes:

1. Declarar una variable de tipo **double** llamada `h` y asignarle el valor del tamaño del paso calculado de la Ec. (1.8).

2. Declarar un vector de tipo **double** llamado `x` y almacenar en él los valores $x_0, x_1, \ldots, x_{n_{\text{Pasos}}}$ obtenidos de aplicar el método de Runge-Kutta de 4^0 orden. Los valores de la función f deben obtenerse invocando la función definida en el Apartado 5.a.
3. Escribir las parejas

$$t_i \;\; x_i \qquad\qquad \text{para } i = 0, \ldots, n_{\text{Pasos}}$$

   en un fichero de texto llamado *resultado.txt*, invocando para ello la función definida en el Apartado 5.b. Si la función lanza una excepción, debe capturarse la excepción mostrándose un mensaje de aviso en la consola en el que se indique que no ha sido posible abrir el fichero.
4. Terminar.

Solución a la Pregunta 5

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>
#include <string>
#include <stdexcept>
#include <vector>

const std::string nombreFich = "resultado.txt";
const double t0      = 0;
const double t1      = 10;
const double x0      = 0.01;
const int    nPasos = 2000;

double f(double t, double x) {
    return std::cos(x*t)*t - std::sin(t);
}
```

```
void escribeFich(std::string nombreFich,
                 double t0, double h,
                 std::vector<double> &x)
                     throw (std::invalid_argument) {
    // Apertura del fichero para escritura
    std::ofstream outFich(nombreFich.c_str(),
                          std::ios::out |std::ios::trunc);
    if (!outFich)
        throw std::invalid_argument("Error abriendo fichero");
    // Escritura de datos
    for (int i=0; i<x.size(); i++)
       outFich << std::fixed << std::setprecision(3) <<
                  t0+i*h << "\t"<<
                  std::fixed << std::setprecision(10) <<
                  x[i] << std::endl;
    // Cerra fichero
    outFich.close();
    return;
}

int main() {
    double h = (t1 - t0) / nPasos;
    std::vector<double> x(nPasos+1,0);
    // RK-4
    double t = t0;
    x[0] = x0;
    for (int i=0; i<nPasos; i++) {
        double k1 = f(t,      x[i]);
        double k2 = f(t+h/2, x[i]+h/2*k1);
        double k3 = f(t+h/2, x[i]+h/2*k2);
        double k4 = f(t+h,    x[i]+h*k3);
        x[i+1] = x[i] + h/6 *( k1 + 2*k2 + 2*k3 + k4 );
        t += h;
    }
    // Escritura en fichero
    try {
        escribeFich(nombreFich, t0, h, x);
    } catch (std::invalid_argument exc) {
         std::cout << exc.what() << std::endl;
    }
    return 0;
}
```