LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2016, Primera Semana

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 la máquina de von Neumann, cuando la palabra binaria se representa como dato, ésta representa un número entero sin signo codificado en base 2.

B. (0.5 puntos) La principal desventaja de la interpretación pura es el tiempo de ejecución del código fuente.

C. (0.5 puntos) La variable del bucle de Pascal es visible dentro del cuerpo del bucle y se permite modificar su valor mediante asignaciones dentro del cuerpo del bucle.

D. (0.5 puntos) La sentencia:

```
x = ++i;
```

es equivalente a las dos sentencias siguientes:

```
x = i;
i = i+1;
```

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

F. (0.5 puntos) El algoritmo de la STL `count(p1, p2, val)` devuelve el número de elementos de la secuencia delimitada por los iteradores p1 y p2 cuyo valor es menor o igual que val.

Solución a la Pregunta 1

A	Falsa	Véase la página 37 del texto base.
B	Verdadera	Véase la página 68 del texto base.
C	Falsa	Véase la página 272 del texto base.
D	Falsa	Véase la página 174 del texto base.
E	Falsa	Véase la página 347 del texto base.
F	Falsa	Véase la página 528 del texto base.

PREGUNTA 2 (1 punto)

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

```
#include <iostream>
#include <iomanip>

double x = 9.3421;

int main () {
   double x = 1.123813;
   double *p1;
   double *p2;
   double a[3] = {4.2322, 3.32213, 6.99983};

   std::cout << std::scientific << std::setprecision(2)
             << a[1] << std::endl;
   std::cout << std::scientific << std::setprecision(2)
             << ::x << std::endl;

   p1  = &a[1];
   p2  = &x;
   *p1 = *p2;

   std::cout << a[1] << std::endl;
   std::cout << *p1  << std::endl;
   std::cout << *p2  << std::endl;

   p2 = p1++;

   std::cout << a[1] << std::endl;
   std::cout << *p1  << std::endl;
   std::cout << *p2  << std::endl;

   return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
3.32e+000
9.34e+000
1.12e+000
1.12e+000
1.12e+000
1.12e+000
7.00e+000
1.12e+000
```

PREGUNTA 3 (1 punto)

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

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

int main() {
   std::vector<std::string> SS;
   std::stack<std::string>  pila;

   SS.push_back("A");
   SS.push_back("B");
   SS.push_back("D");

   for (unsigned int ii = 0; ii < SS.size(); ii++) {
      std::cout << SS[ii] << std::endl;
      pila.push(SS[ii]);
   }

   std::vector<std::string>::iterator cii;
   for (cii = SS.begin(); cii != SS.end(); cii++)
      std::cout << *cii << std::endl;

   while( !pila.empty() ) {
      std::cout << pila.top() << std::endl;
      pila.pop();
   }

   return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
A
B
D
A
B
D
D
B
A
```

PREGUNTA 4 (5 puntos)

En un proyecto cuyo objetivo es la detección de asteroides peligrosos para la tierra, se encarga la realización de un programa en C++ que lea de un fichero la velocidad y posición de los asteroides, ordene los asteroides en función de su peligrosidad y muestre un mensaje en la consola indicando cuáles de ellos tienen una peligrosidad superior a un cierto valor especificado por el usuario.

Escriba el código C++ indicado a continuación.

4.1 (0.2 puntos) Escriba la declaración de un tipo estructura (*struct*) llamado `Asteroide` que tenga los miembros siguientes:

- Un *string* llamado `nombre`.
- Un array de 3 componentes **double** llamado `posicion`.
- Un array de 3 componentes **double** llamado `velocidad`.
- Un **double** llamado `c`.

4.2 (0.3 puntos) Escriba la definición de una función llamada `moduloVec` que acepte como argumento un array $\mathbf{x}$ de N componentes de tipo **double** y devuelva el módulo del vector, $|\mathbf{x}|$, calculado como la raíz cuadrada de la suma de los cuadrados de los componentes:

$$|\mathbf{x}| = \sqrt{\sum_{i=0}^{N-1} {x_i}^2}$$

donde $\mathbf{x} = \{x_0, \ldots, x_{N-1}\}$.

La función debe tener el prototipo siguiente:

```
double moduloVec(const double x[], int N);
```

4.3 (0.5 puntos) Escriba la definición de una función llamada `coefC` que acepte como parámetros dos arrays **r** y **v** de tres componentes de tipo **double**, y devuelva el valor real c calculado de la forma siguiente:

$$c = \sqrt{\left(\frac{v_0}{|\mathbf{v}|} + \frac{r_0}{|\mathbf{r}|}\right)^2 + \left(\frac{v_1}{|\mathbf{v}|} + \frac{r_1}{|\mathbf{r}|}\right)^2 + \left(\frac{v_2}{|\mathbf{v}|} + \frac{r_2}{|\mathbf{r}|}\right)^2}$$

donde $\mathbf{r} = \{r_0, r_1, r_2\}$ y $\mathbf{v} = \{v_0, v_1, v_2\}$. El módulo de los arrays **r** y **v** debe calcularse invocando la función programada en el Apartado 4.2.

Simplemente a modo de aclaración, indicar que si el vector **r** es la posición del asteroide y **v** es su velocidad, medidas suponiendo que la tierra se encuentra en reposo en la posición $\{0, 0, 0\}$, entonces el valor de c es una medida del riesgo que supone el asteroide para la tierra.

4.4 (2 puntos) Escriba la definición de una función que acepte como argumento un *string* y devuelva una lista de elementos del tipo estructura llamado `Asteroide`, que fue declarado en el Apartado 4.1. El prototipo de la función es el siguiente:

```
std::list<Asteroide> LeeDatos(std::string nombreFich)
                     throw (std::invalid_argument);
```

La función debe abrir para lectura el fichero de texto cuyo nombre viene dado por el valor del parámetro de la función. Si se produce error al abrir el fichero, la función debe lanzar una excepción.

Si no se produce error al abrir el fichero, la función debe leer el fichero palabra a palabra. En el fichero, los datos están escritos en una o más filas. En cada fila están escritos los datos de un asteroide. Cada fila está compuesta por las 7 columnas siguientes:

- La primera columna es un *string*: el nombre del asteroide.
- Las siguientes tres columnas son números reales: las coordenadas (r_0, r_1, r_2) de la posición del asteroide.

– Las últimas tres columnas son números reales: las componentes (v_0, v_1, v_2) de la velocidad del asteroide.

La función debe calcular el valor de c de cada asteroide, invocando para ello la función definida en el Apartado 4.3.

La lista devuelta por la función debe tener un elemento por cada asteroide leído del fichero. El elemento debe almacenar los correspondientes datos de nombre, posición, velocidad y valor de c del asteroide.

Los elementos deben insertarse en la lista de manera que queden ordenados crecientemente de acuerdo a su valor de c. El valor de c de un elemento debe ser menor o igual que el valor de c del elemento siguiente en la lista. La ordenación relativa de los elementos con el mismo valor de c puede ser cualquiera.

4.5 (2 puntos) Escriba el programa principal, el cual debe realizar las acciones siguientes:

1. Solicitar al usuario que introduzca por consola el nombre del fichero donde se encuentran los datos.
2. Leer dicho valor, almacenándolo en una variable de tipo *string* llamada `fichIn`.
3. Invocar la función `LeeDatos` definida en el Apartado 4.4, pasándole como argumento la variable `fichIn`. La lista devuelta por la función debe llamarse `lAst`. Si la función lanza una excepción, mostrar en la consola un mensaje indicando que el nombre del fichero no es válido y terminar.
4. Solicitar al usuario que éste introduzca por consola un número denominado "factor de alarma". Almacenar el valor introducido en una variable **double** llamada a.
5. Mostrar en la consola los componentes de la lista `lAst` cuyo valor de c satisface $c < a$. Los datos deben mostrarse en dos columnas separadas por un tabulador. En la primera columna debe mostrarse el nombre del asteroide. En la segunda columna debe mostrarse el valor de c en formato científico, con 3 dígitos de precisión.
6. Mostrar un mensaje en la consola indicando el número de asteroides peligrosos (los mostrados anteriormente en la consola) y el número total de asteroides analizados (número total de elementos de la lista).
7. Terminar.

Solución a la Pregunta 4

```
#include <iostream>
#include <string>
#include <cmath>
#include <list>
#include <stdexcept>
#include <fstream>
#include <sstream>
#include <iomanip>


struct Asteroide {
    std::string nombre;
    double posicion[3];
    double velocidad[3];
    double c;
};


double moduloVec(const double x[], int N) {
    double modulo2 = 0;
    for ( int i=0; i<N; i++ )
        modulo2 += x[i]*x[i];
    return std::sqrt(modulo2);
}


double coefC(const double r[], const double v[]) {
    double mod_r = moduloVec(r, 3);
    double mod_v = moduloVec(v, 3);
    double c2 = 0;
    for ( unsigned int i = 0; i < 3; i++ ) {
        double comp = v[i]/mod_v + r[i]/mod_r;
        c2 += comp*comp;
    }
    return std::sqrt(c2);
}
```

```
std::list<Asteroide> LeeDatos(std::string nombreFich)
   throw (std::invalid_argument){

      // Apertura del fichero de texto para lectura
      std::ifstream inFich(nombreFich.c_str(), std::ios::in);
      if (!inFich) {
         std::stringstream ss;
         ss << "Error - fichero no encontrado: "
            << nombreFich;
         throw std::invalid_argument (ss.str());
      }
      // Lista de asteroides
      std::list<Asteroide> listaAst;
      // Lectura del fichero e inserción en la lista
      while ( !inFich.eof() ) {
         Asteroide ast;
         inFich >> ast.nombre;
         if (inFich.eof()) break;
         for ( unsigned int i=0; i<3; i++ )
            inFich >> ast.posicion[i];
         for ( unsigned int i=0; i<3; i++ )
            inFich >> ast.velocidad[i];
         ast.c = coefC(ast.posicion, ast.velocidad);
         // Insercion del elemento en la lista
         if ( listaAst.size() == 0 ) {    // La lista está vacía
            listaAst.push_back(ast);
         } else {              // La lista ya tiene algún elemento
            std::list<Asteroide>::iterator p = listaAst.begin();
            bool insertado = false;
            while ( !insertado && p != listaAst.end() ) {
               if ( p->c >= ast.c ) {
                  listaAst.insert(p,ast);
                  insertado = true;
               } else {
                  p++;
               }
            }
            if (!insertado)
               listaAst.push_back(ast);
         }
      }
      inFich.close();
      return listaAst;
}
```

```
int main() {
    // Solicitar al usuario nombre fichero
    std::cout << "Introduzca el nombre del fichero: " << std::endl;
    std::string fichIn;
    std::cin >> fichIn;
    // Obtención lista asteroides
    std::list<Asteroide> lAst;
    try {
        lAst = LeeDatos(fichIn);
    } catch (std::invalid_argument exc) {
        std::cout << exc.what() << std::endl;
        return 0;
    }
    // Solicitar al usuario factor de alarma
    std::cout << "Introduzca el factor de alarma: " << std::endl;
    double a;
    std::cin >> a;
    // Mostrar en la consola elementos con c < a
    std::list<Asteroide>::iterator p = lAst.begin();
    int numAstPeligrosos = 0;
    while ( p != lAst.end() && p->c < a ) {
        std::cout << p->nombre << "\t"
            << std::scientific << std::setprecision(3)
            << p->c << std::endl;
        numAstPeligrosos++;
        p++;
    }
    // Informe
    std::cout << "Peligrosos / Total = "
        << numAstPeligrosos << " / " << lAst.size()
        << std::endl;
    return 0;
}
```