LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2026, Segunda 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) FORTRAN I fue el primer lenguaje diseñado con idea de no estar condicionado al hardware de una máquina específica.

B. (0.5 puntos) En C++ se permite emplear expresiones numéricas enteras como expresiones lógicas.

C. (0.5 puntos) En C y C++ la expresión de control del bucle **for** es opcional.

D. (0.5 puntos) El algoritmo `transform` aplica una función especificada a cada uno de los elementos de la secuencia origen, almacenando el resultado en la misma secuencia origen.

E. (0.5 puntos) Cuando se pasan los parámetros por referencia, los cálculos realizados en la función pueden modificar el valor de los parámetros actuales.

F. (0.5 puntos) El algoritmo de la STL

```
replace_copy(pBegin, pEnd, pResultBegin, val, val1)
```

copia la secuencia origen en la secuencia destino, eliminando de la copia los elementos cuyo valor sea `val` o `val1`.

Solución a la Pregunta 1

A	Falso	Véase la página 35 del texto base.
B	Verdadero	Véase la página 173 del texto base.
C	Verdadero	Véase la página 260 del texto base.
D	Falso	Véase la página 561 del texto base.
E	Verdadero	Véase la página 367 del texto base.
F	Falso	Véase la página 558 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 = 1, y = 2, z = 3;

int main() {
   double* p;
   double a[3] = { 3.2386, 5.4382, 7.45464 };
   p = a;
   x = *(++p);
   std::cout << std::scientific << std::setprecision(2)
             << "a[1] " << a[2] << std::endl;
   std::cout << "*p " << *p << std::endl;
   std::cout << "x "  << x  << std::endl;
   {
      double x = -10;
      double y = 5;
      a[1] = x;
      z += y;
   }
   std::cout << "*(p+1) " << *(p + 1) << std::endl;
   std::cout << std::setprecision(3)
             << "a[1] " << a[1] << std::endl;
   std::cout << "x " << x << std::endl;
   std::cout << "y " << y << std::endl;
   std::cout << "z " << z << std::endl;
   p = p + 1;
   std::cout << "p " << *p << std::endl;
   std::cout << std::setw(4) << "a[0] " << a[0] << std::endl;
   return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
a[1] 7.45e+000
*p 5.44e+000
x 5.44e+000
*(p+1) 7.45e+000
a[1] -1.000e+001
x 5.438e+000
y 2.000e+000
z 8.000e+000
p 7.455e+000
a[0] 3.239e+000
```

PREGUNTA 3 (1 punto)

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

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

int main() {
   std::list<int> listaI;
   for (int i = 1; i < 10; i += 2)
      listaI.push_back(i);
   std::list<int> listaD(listaI.size());
   std::list<int>::iterator pEndD =
      std::transform(listaI.begin(),
                     listaI.end(),
                     listaD.begin(),
                     [](int n) -> int { return n*2; });
   std::cout << "\nlistaD:";
   for (std::list<int>::iterator p = listaD.begin();
        p != pEndD; p++)
      std::cout << " " << (*p);
   std::list<int> listaS(listaD.size());
   std::list<int>::iterator pEndS =
      std::transform(
        listaD.begin(),
        listaD.end(),
        listaS.begin(),
        [](int n) -> int { return n%3; });
   std::cout << "\nlistaS:";
   for (std::list<int>::iterator p = listaS.begin();
        p != pEndS; p++)
      std::cout << " " << (*p);
   std::cout << std::endl;
   return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
listaD: 2 6 10 14 18
listaS: 2 0 1 2 0
```

PREGUNTA 4 (3 puntos)

Se desea estudiar la peligrosidad para la Tierra de un conjunto de asteroides detectados dentro del sistema solar. Se conoce la posición y velocidad de cada uno de estos asteroides respecto de la Tierra, en un sistema de coordenadas cartesiano cuyo origen $(0, 0, 0)$ se fija en el centro de la Tierra. El centro de un asteroide se encuentra en la posición $\vec{r} = (x, y, z)$ km y se mueve respecto a la Tierra con velocidad $\vec{v} = (v_x, v_y, v_z)$ km/s. El asteroide tiene una masa M kg.

La información de los asteroides detectados se encuentra almacenada en un fichero de texto llamado *asteroides.txt*. En cada fila del fichero se encuentra la información que describe un asteroide. Ésta consiste en los datos siguientes, separados por uno o más espacios en blanco: el nombre del asteroide, su masa M, y las coordenadas de su posición (x, y, z) y de su velocidad (v_x, v_y, v_z).

Por ejemplo, la línea del fichero:

```
B1 7.8e10 -52004 12040 6256 2.5 -0.8 -1.1
```

describe un asteroide llamado B1, cuya masa es $M = 7.8\text{e}10$ kg, que se encuentra en la posición $\vec{r} = (-52004, 12040, 6256)$ km y se mueve con velocidad $\vec{v} = (2.5, -0.8, -1.1)$ km/s.

La distancia mínima (d_{min}) a la que un asteroide pasará del centro de la Tierra puede estimarse, a partir de su posición y velocidad, de la forma siguiente:

$$d_{min} = \frac{\sqrt{(y \cdot v_z - z \cdot v_y)^2 + (z \cdot v_x - x \cdot v_z)^2 + (x \cdot v_y - y \cdot v_x)^2}}{\sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}}$$

Hay *riesgo de impacto* del asteroide con la Tierra si se satisface: $d_{min} \leq R_T$, donde $R_T = 6371$ km es el radio de la Tierra.

En caso de que haya riesgo de impacto, se puede calcular el *nivel de peligro*. Éste se clasifica en tres niveles – ALTO, MEDIO o BAJO – en función del valor de la energía cinética del asteroide, E_c, la cual se calcula de la forma siguiente:

$$E_c = \frac{1}{2} \cdot 10^6 \cdot M \cdot \left({v_x}^2 + {v_y}^2 + {v_z}^2\right)$$

donde la energía cinética está expresada en J, la velocidad en km/s y la masa en kg.

La clasificación del nivel de peligro se realiza de la forma indicada a continuación, donde la energía cinética está expresada en J:

$$Nivel = \begin{cases} ALTO & si \quad E_c \geq 10^{20} \\ MEDIO & si \quad 10^{18} \leq E_c < 10^{20} \\ BAJO & si \quad E_c < 10^{18} \end{cases}$$

Escriba el código C++ indicado a continuación.

4.1 (0.5 puntos). Una función en C++ con el prototipo

```
double dMin(const Asteroide &a);
```

que calcule la distancia mínima entre el asteroide pasado como argumento y la Tierra. La estructura `Asteroide` está declarada de la forma siguiente:

```
struct Asteroide {
  std::string nombre;
  double masa;
  double x, y, z;
  double vx, vy, vz;
};
```

4.2 (0.5 puntos). Una función en C++ con el prototipo

```
double energiaCinetica(const Asteroide &a);
```

que devuelva la energía cinética del asteroide pasado como argumento.

4.3 (0.5 puntos). Una función en C++ con el prototipo

```
bool hayRiesgo(const Asteroide &a);
```

que devuelva true si existe riesgo de colisión con la Tierra del asteroide pasado como argumento. Esta función debe invocar la función `dMin`.

El radio de la Tierra, $R_T = 6371$ km, es una constante global del programa.

4.4 (0.5 puntos). Una función en C++ con el prototipo

```
Peligro NivelPeligro(const Asteroide &a);
```

que devuelva el nivel de peligro del asteroide pasado como argumento. El valor devuelto por la función es del tipo siguiente:

```
enum Peligro { BAJO=0, MEDIO, ALTO };
```

4.5 (1 punto). Un programa principal en C++ que, invocando las funciones anteriormente definidas, realice las tareas enumeradas a continuación:

1. Declarar y asignar valor a dos constantes globales: el radio de la Tierra ($R_T = 6371$ km) y el nombre del fichero (*asteroides.txt*).
2. Leer el contenido del fichero *asteroides.txt* y almacenarlo en un vector, llamado `vAst`, de estructuras tipo `Asteroide`. Si se produce error al abrir el fichero para lectura, el programa debe mostrar un mensaje en la consola indicándolo y terminar.
3. Escribir en la consola la información siguiente acerca de cada asteroide: nombre, distancia mínima a la Tierra y si existe riesgo de que impacte con la Tierra. En caso de que exista riesgo de impacto, el programa debe mostrar la energía cinética del asteroide y su nivel de peligro.

   La distancia mínima a la Tierra debe escribirse en formato fijo, con dos dígitos decimales.

   La energía cinética debe escribirse en formato científico, con tres dígitos decimales.
4. Terminar.

Solución a la Pregunta 4

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

const std::string fichero = "asteroides.txt";
const double RT = 6371; // Radio de la tierra [km]

enum Peligro { BAJO=0, MEDIO, ALTO };
const std::string strPeligro[] = {"BAJO", "MEDIO", "ALTO"};

struct Asteroide {
   std::string nombre;
   double masa;
   double x, y, z;
   double vx, vy, vz;
};

double dMin(const Asteroide &a) {
   return std::sqrt( std::pow(a.y*a.vz - a.z*a.vy, 2) +
                     std::pow(a.z*a.vx - a.x*a.vz, 2) +
                     std::pow(a.x*a.vy - a.y*a.vx, 2) ) /
          std::sqrt( std::pow(a.vx, 2) +
                     std::pow(a.vy, 2) +
                     std::pow(a.vz, 2) );
}

double energiaCinetica(const Asteroide &a) {
   return 0.5e6 * a.masa *
              ( std::pow(a.vx, 2) +
                std::pow(a.vy, 2) +
                std::pow(a.vz, 2) );
}

bool hayRiesgo(const Asteroide &a) {
   return dMin(a) <= RT;
}

Peligro NivelPeligro(const Asteroide &a) {
   double Ec = energiaCinetica(a);
   if (Ec >=1e20)
      return ALTO;
   else if (Ec >= 1e18)
      return MEDIO;
   else
      return BAJO;
}

```

```
int main() {
   std::ifstream inFich(fichero.c_str(), std::ios::in);
   if (!inFich) {
      std::cerr << "Error al abrir el fichero\n";
      return 0;
   }
   std::vector<Asteroide> vAst;
   Asteroide ast;
   while (inFich >> ast.nombre >> ast.masa >>
                    ast.x  >> ast.y  >> ast.z >>
                    ast.vx >> ast.vy >> ast.vz) {
      vAst.push_back(ast);
   }
   for (unsigned i=0; i<vAst.size(); i++) {
      std::cout << "Nombre: " << vAst[i].nombre
                << "\nDistancia minima [km]: "
                << std::fixed << std::setprecision(2)
                << dMin(vAst[i])
                << "\nExiste riesgo de impacto: " << std::boolalpha
                << hayRiesgo(vAst[i]) << std::endl;
      if (hayRiesgo(vAst[i]))
         std::cout << "Energia cinetica [J]: "
                   << std::scientific << std::setprecision(3)
                   << energiaCinetica(vAst[i])
                   << "\nNivel de peligro: "
                   << strPeligro[NivelPeligro(vAst[i])]
                   << std::endl;
   }
   return 0;
}
```

PREGUNTA 5 (2 puntos)

Se desea traducir mensajes a secuencias de bits. Los mensajes están construidos empleando únicamente ocho símbolos: `A`, `B`, `C`, `D`, `E`, `F`, `G` y `H`. Para ello, se emplea la tabla de codificación siguiente:

`A` 0	`C` 1010	`E` 1100	`G` 1110
`B` 100	`D` 1011	`F` 1101	`H` 1111

Mediante este código, el mensaje

```
BACADAEAFABBAAAGAH
```

se representaría mediante la siguiente secuencia de bits:

```
100010100101101100011010100100000111001111
```

Escriba un programa en C++ que solicite al usuario que introduzca por consola un mensaje empleando el código de ocho símbolos. El programa debe comprobar si es un mensaje válido.

Si la cadena de caracteres introducida por el usuario no es un mensaje válido, el programa debe indicarlo a través de la consola y terminar.

Si se trata de un mensaje válido, el programa debe almacenar la secuencia de símbolos del mensaje en una lista de **char** y mediante el algoritmo `transform` traducir ésta a otra lista con los correspondientes códigos binarios. Finalmente, el programa debe escribir en la consola el contenido de esta segunda lista, mostrando así la traducción a código binario del mensaje introducido por el usuario.

Solución a la Pregunta 5

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <list>
#include <algorithm>

bool mensaje2lista(const std::string mensaje,
                   std::list<char> &lSimbolosMensaje) {
   for (unsigned int i = 0; i < mensaje.length(); i++)
      if (mensaje[i] >= 'A' && mensaje[i] <= 'H')
         lSimbolosMensaje.push_back(mensaje[i]);
      else
         return false;
   return true;
}

std::string codifica(char c) {
   switch (c) {
   case 'A':
      return "0";
   case 'B':
      return "100";
   case 'C':
      return "1010";
   case 'D':
      return "1011";
   case 'E':
      return "1100";
   case 'F':
      return "1101";
   case 'G':
      return "1110";
   case 'H':
      return "1111";
   default:
      return " ";
   }
}

std::string lstString2string(std::list<std::string> &lst) {
   std::stringstream ss;
   std::list<std::string>::iterator p = lst.begin();
   while (p != lst.end()) {
      ss << *p;
      p++;
   }
   return ss.str();
}

```

```
50 int main() {
51    // Entrada por consola
52    std::cout << "Introduzca el mensaje: ";
53    std::string mensaje;
54    std::cin >> mensaje;
55    // Comprueba cadena y almacena simbolos en listaOrigen
56    std::list<char> listaOrigen;
57    if (!mensaje2lista(mensaje, listaOrigen)) {
58       std::cerr << "Cadena no valida" << std::endl;
59       return 0;
60    }
61    // Transforma lista de simbolos a lista de palabras binarias
62    std::list<std::string> listaDestino(listaOrigen.size());
63    std::transform(listaOrigen.begin(), listaOrigen.end(),
64                   listaDestino.begin(), codifica);
65    // Escribe en la consola el mensaje como secuencia de bits
66    std::cout << lstString2string(listaDestino) << std::endl;
67    return 0;
68 }
```