# LENGUAJES DE PROGRAMACIÓN

## Solución al examen de Junio 2016, 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) En la máquina de von Neumann, cuando la palabra binaria se representa como instrucción, representa una instrucción codificada mediante 40 bits.

**B.** (0.5 puntos) La expresión en notación infija

```
c*(a+b)
```

se escribe en notación prefija como

```
*c+ab
```

**C.** (0.5 puntos) Si la variable `v` es declarada de la forma siguiente:

```
std::vector<double> v(4,0);
```

entonces la sentencia

```
v.clear();
```

es equivalente a la sentencia

```
v.erase(v.begin(),v.end());
```

**D.** (0.5 puntos) El último elemento que se ha añadido a una cola es el primer elemento en ser extraído de dicha cola.

**E.** (0.5 puntos) La función sobre el flujo de entrada

```
std::cin.clear()
```

devuelve el último carácter del flujo de entrada.

**F.** (0.5 puntos) El ámbito de una variable estática en C++ está limitado al bloque de código en que se ha declarado.

**Solución a la Pregunta 1**

| | | |
|---|---|---|
| **A** | Falsa | Véase la página 37 del texto base. |
| **B** | Verdadera | Véase la página 172 del texto base. |
| **C** | Verdadera | Véase la página 228 del texto base. |
| **D** | Falsa | Véase la página 432 del texto base. |
| **E** | Falsa | Véase la página 312 del texto base. |
| **F** | Verdadera | Véase la página 378 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 funcion(int n, int m) {
   if ( n == 0 ) {
      std::cout << m+1 << std::endl;
      return m+1;
   } else if ( m == 0 ) {
      std::cout << "funcion(" << n-1 << ",1)" << std::endl;
      return funcion(n-1,1);
   } else {
      std::cout << "funcion(" << n-1
                << " funcion(" << n << "," << m-1 << "))"
                << std::endl;
      return funcion(n-1, funcion(n,m-1));
   }
   return 0;
}

int main() {
   funcion(1,2);
   return 0;
}
```

**Solución a la Pregunta 2**

```
funcion(0 funcion(1,1))
funcion(0 funcion(1,0))
funcion(0,1)
2
3
4
```

**PREGUNTA 3** (1 punto)

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

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

std::string listaInt2string(std::list<int> &lst) {
   std::stringstream ss;
   std::list<int>::iterator p = lst.begin();
   while ( p != lst.end() ) {
      ss << *p << " ";
      p++;
   }
   return ss.str();
}

int main() {
   std::list<int>::iterator p, p1, p2;
   std::list<int> lista1;

   for (int i=0; i<10; i++)
      lista1.push_back(i);
   std::cout << listaInt2string(lista1) << std::endl;

   p1 = lista1.begin();
   for (int i = 0; i < 6; i++)
      p1++;
   p2 = p1;   p2--;   p2--;
   std::cout <<  *p1 << std::endl;
   std::cout <<  *p2 << std::endl;

   p = lista1.erase(p2,p1);
   std::cout << listaInt2string(lista1) << std::endl;
   std::cout << *p                      << std::endl;

   p1 = lista1.erase(p);
   std::cout << listaInt2string(lista1) << std::endl;
   std::cout << *p1                     << std::endl;
   return 0;
}
```

**Solución a la Pregunta 3**

```
0 1 2 3 4 5 6 7 8 9
6
4
0 1 2 3 6 7 8 9
6
0 1 2 3 7 8 9
7
```

**PREGUNTA 4** (5 puntos)

Se propone realizar un programa en C++ que lea de fichero dos imágenes binarias, y calcule la superposición y el contraste de las dos imágenes, mostrando el resultado en la consola.

Una imagen se representa mediante una matriz binaria de $3 \times 3$ elementos. Un pixel negro está representado por el valor 1 y uno blanco por el valor 0. Las operaciones de superposición y de contraste de dos imágenes se realizan de la forma siguiente:

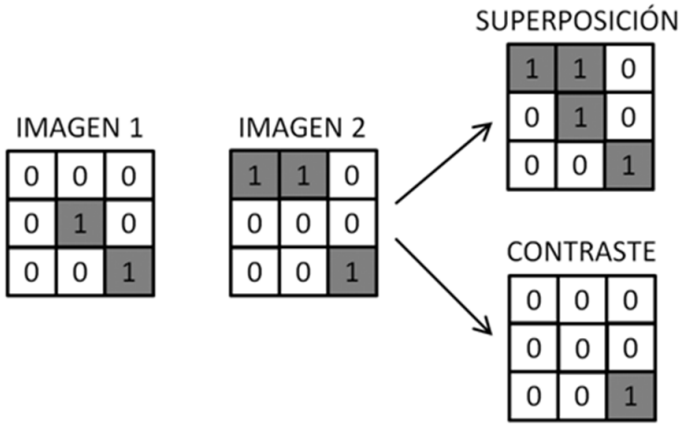
- La imagen $S$ resultante de la operación de *superposición* de dos imágenes $I_1$ e $I_2$ se obtiene realizando la operación OR elemento a elemento:

$$S[i,j] = I_1[i,j] \quad \text{OR} \quad I_2[i,j] \qquad \text{con } i,j : 0,1,2$$

- La imagen $C$ resultante de la operación de *contraste* de dos imágenes $I_1$ e $I_2$ se obtiene realizando la operación AND elemento a elemento:

$$C[i,j] = I_1[i,j] \quad \text{AND} \quad I_2[i,j] \qquad \text{con } i,j : 0,1,2$$

A continuación se muestra un ejemplo:




Escriba el siguiente código C++:

**4.1** (1.5 puntos) Escriba la definición de una función que acepte como argumento un *string* y devuelva un vector de elementos booleanos. El prototipo de la función es el siguiente:

```
std::vector<bool> 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 carácter a carácter. Si el fichero no contiene exactamente 9 caracteres, o si alguno de los caracteres es diferente de '0' y de '1', entonces la función debe lanzar una excepción.

La información de los nueve caracteres leídos $c_0, \ldots, c_8$ debe almacenarse en un vector de **bool** llamado `imagen`.

- Si el carácter $c_i$ vale '0', el componente $i$ del vector deberá valer *false*.
- Si el carácter $c_i$ vale '1', el componente $i$ del vector deberá valer *true*.

Una vez finalizada la lectura de fichero, la función debe devolver dicho vector.

**4.2** (0.5 puntos) Escriba la definición de una función llamada `procImag`, que realice la operación de superposición y de contraste sobre las dos imágenes pasadas como parámetros. Un pixel negro está representado por el valor *true* y uno blanco por el valor *false*.

Los parámetros son pasados por referencia. El resultado de realizar la operación de superposición debe asignarse al primer parámetro. El resultado de la operación de contraste debe asignarse al segundo parámetro.

El prototipo de la función debe ser el siguiente:

```
void procImag(  std::vector<bool> &b1,
                std::vector<bool> &b2 );
```

**4.3** (1 punto) Defina una función cuyo prototipo sea

```
std::string imag2string(std::vector<bool> *imag);
```

y que devuelva un *string* construido de la forma indicada a continuación.

Inicialmente el *string* está vacío. Se van leyendo los consecutivos componentes del vector:

- Si el componente vale *true*, se añade el carácter 'X' al final del *string*.
- Si el componente vale *false*, se añade el carácter 'O' al final del *string*.

**4.4** (2 puntos) Escriba un programa en C++ que realice las acciones siguientes:

1. Mostrar un mensaje por consola solicitando al usuario que introduzca el nombre de los dos ficheros de texto donde están almacenadas las imágenes.
2. Guardar los valores introducidos por el usuario en dos variables *string* llamadas `fichImag1` y `fichImag2`.
3. Invocar dos veces la función `LeeDatos` definida en el Apartado 4.1, pasando como argumento en el primer caso `fichImag1` y en el segundo `fichImag2`.

   Si se captura una excepción, mostrar un mensaje de error y terminar.

   El valor devuelto en la primera llamada a la función debe almacenarse en un vector de **bool** llamado `imag1`.

   El valor devuelto en la segunda llamada debe almacenarse en otro vector de **bool** llamado `imag2`.
4. Empleando la función definida en el Apartado 4.2, realizar la operación de superposición y de contraste de `imag1` e `imag2`. El resultado de la superposición debe quedar almacenado en `imag1` y el resultado del contraste en `imag2`.
5. Invocando la función `imag2string` definida en el Apartado 4.3, convertir en un *string* cada uno de los dos vectores calculados en el paso anterior. El *string* obtenido de la imagen superpuesta deberá llamarse `sI1` y el obtenido de la imagen contraste deberá llamarse `sI2`.
6. Escriba en la consola los dos *strings* obtenidos en el paso anterior, escribiendo un salto de línea cada tres caracteres del *string*. Así, por ejemplo, el resultado de escribir en la consola el *string*:

   `"XXOOXOOOX"`

   deberá ser:

   ```
   XXO
   OXO
   OOX
   ```

7. Terminar.

**Solución a la Pregunta 4**

```
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <stdexcept>
#include <fstream>

std::vector<bool> 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());
      }
      // Lectura del fichero
      char c;
      std::vector<bool> imagen;
      while ( !inFich.eof() ) {
         inFich >> std::noskipws >> c;
         if ( inFich.eof() ) break;
         // El fichero contiene más de 9 caracteres
         if ( imagen.size() == 9 ) {
            std::stringstream ss;
            ss << "Error en fichero " << nombreFich
               << "\nEl fichero contiene mas de 9 caracteres";
            throw std::invalid_argument (ss.str());
         }
         // El carácter debe ser '0' o '1'
         if ( c != '0' && c != '1' ) {
            std::stringstream ss;
            ss << "Error en fichero " << nombreFich << "\n";
            ss << "El caracter numero "
               << imagen.size() << " no es valido: "
               << c;
            throw std::invalid_argument (ss.str());
         }
         // Añade el bit al vector
         if ( c == '0' ) {
            imagen.push_back(false);
         } else {
            imagen.push_back(true);
         }
      }
      inFich.close();
      // Comprobar si hay menos de 9 caracteres
      if ( imagen.size() < 9 ) {
         std::stringstream ss;
         ss << "Error en fichero " << nombreFich
            << "\nEl fichero contiene unicamente "
            << imagen.size() << " caracteres";
         throw std::invalid_argument (ss.str());
      }
      return imagen;
}
```

```
void procImag(std::vector<bool> &b1, std::vector<bool> &b2) {
    for ( unsigned int i=0; i<b1.size() && i<b2.size(); i++ ) {
        bool aux = b1[i];
        b1[i] = b1[i] || b2[i];
        b2[i] = aux && b2[i];
    }
    return;
}

std::string imag2string(std::vector<bool>*imag) {
    std::string s;
    for ( unsigned int i=0; i<imag->size(); i++ ) {
        if ( (*imag)[i] ) {
            s += 'X';
        } else {
            s += 'O';
        }
    }
    return s;
}

int main() {
    // Solicitar al usuario nombre fichero
    std::cout << "Nombres de los dos ficheros: " << std::endl;
    std::string fichImag1, fichImag2;
    std::cin >> fichImag1 >> fichImag2;
    // Lectura de los ficheros
    std::vector<bool> imag1, imag2;
    try {
        imag1 = LeeDatos(fichImag1);
        imag2 = LeeDatos(fichImag2);
    } catch (std::invalid_argument exc) {
        std::cout << exc.what() << std::endl;
        return 0;
    }
    // Procesado de las imágenes
    procImag(imag1, imag2);
    // Escritura en la consola
    std::string sI1 = imag2string(&imag1);
    std::cout << "Imagen superpuesta: " << std::endl;
    for (unsigned int i=0; i<sI1.size(); i++) {
        std::cout << sI1[i];
        if ( !((i+1) %3) )
            std::cout << std::endl;
    }
    std::cout << "Imagen contraste: " << std::endl;
    std::string sI2 = imag2string(&imag2);
    for (unsigned int i=0; i<sI2.size(); i++) {
        std::cout << sI2[i];
        if ( !((i+1) %3) )
            std::cout << std::endl;
    }
    return 0;
}
```