LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2013, 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 ALGOL 58 los identificadores han de tener una longitud máxima de 6 caracteres.

B. (0.5 puntos) El rango de valores que puede tomar cada tipo de dato básico en el lenguaje C++ viene determinado por las reglas del lenguaje.

C. (0.5 puntos) El lenguaje Java no presenta el problema de las variables dinámicas perdidas.

D. (0.5 puntos) En el método de la burbuja el número de intercambios depende de la secuencia de entrada.

E. (0.5 puntos) La recursividad de cola es un tipo de recursividad lineal.

F. (0.5 puntos) La forma más común de implementar una lista es mediante el uso de arrays.

Solución a la Pregunta 1

A	Falsa	Véase la página 47 del texto base.
B	Falsa	Véase la página 137 del texto base.
C	Verdadera	Véase la página 127 del texto base.
D	Verdadera	Véase la página 510 del texto base.
E	Verdadera	Véase la página 355 del texto base.
F	Falsa	Véase la página 429 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[] = {1, 2, 3, 4};
   p = &a[1];
   std::cout << p[1] <<std::endl;
   std::cout << p[2] <<std::endl;
   p[0] = p[0]+1;
   std::cout<<a[0]<<std::endl;
   std::cout<<a[1]<<std::endl;
   return 0;
}
```

Solución a la Pregunta 2

```
3
4
1
3
```

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

Solución a la Pregunta 3

```
0
0
1
0
2
0
3
0 1 2
```

PREGUNTA 4 (2.5 puntos)

En las telecomunicaciones, el bit de paridad es un dígito binario que se añade a una cadena binaria para detectar posibles errores. El bit de paridad es 1 si la cadena tiene un número par de bits 1. Si la cadena tiene un número impar de bits 1, entonces el bit de paridad es 0.

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

1. Solicitar la entrada por consola de una cadena binaria compuesta por 8 dígitos binarios. Si alguno de los valores introducidos por el usuario no es un dígito binario, mostrar el correspondiente mensaje de error y terminar.
2. Almacenar los 8 dígitos introducidos por consola en los 8 primeros componentes de un array de 9 componentes **int**.
3. Calcular el bit de paridad de la cadena.
4. Asignar el valor del bit de paridad al último componente de array.
5. Mostrar en la consola el contenido del array. Es decir, la cadena de 8 dígitos binarios y el bit de paridad.

Solución a la Pregunta 4

```
#include <iostream>

const int longVector = 9;

int main() {
      int v[longVector];
      // Entrada de la cadena binaria por consola
      for (int i=0; i<longVector-1; i++) {
            std::cout << "Introduzca un digito binario: ";
            std::cin >> v[i];
            if (!std::cin || (v[i]!=1 && v[i]!=0) ) {
                  std::cout << "Dato erroneo"<< std::endl;
                  return 0;
            }
      }
      // Cálculo de bit de paridad
      bool par = true;
      for (int i=0; i<longVector-1; i++)
            if (v[i]) par = !par;
      if (par) {
            v[longVector-1] = 1;
      } else {
            v[longVector-1] = 0;
      }
      // Mostrar en la consola el contenido del vector
      for (int i=0; i<longVector; i++)
            std::cout << v[i];
      std::cout << std::endl;
      return 0;
}
```

PREGUNTA 5 (2.5 puntos)

Escriba en C++ un programa que realice las acciones descritas a continuación.

1. *Apertura para lectura de un fichero de texto*.

   El nombre del fichero de texto en el cual se encuentran los datos debe almacenarse en una constante global del tipo `std::string`, llamada *nombreFich*, cuyo valor debe ser `"datos.txt"`. El programa debe abrir el fichero de texto para lectura. Si se produce error, debe mostrar un mensaje en la consola indicándolo y terminar.

2. *Lectura del fichero de texto*.

   El fichero de texto contiene datos, que son números reales. El programa debe leerlos y almacenarlos en un vector llamado `datos`.

3. *Comprobar que* $N > 0$.

   El número N de datos leídos del fichero debe ser mayor que cero. En caso contrario, el programa debe mostrar un mensaje en la consola y terminar.

4. *Cálculo del número de datos que pertenecen al intervalo* $[0, 1]$.

   El programa debe calcular el número de datos que pertenecen al intervalo $[0, 1]$. Para ello, debe aplicar el algoritmo `count_if` al vector `datos`.

5. *Mostrar en la consola el resultado*.

   El programa debe mostrar en la consola el número calculado en el paso anterior.

6. *Terminar*.

Solución a la Pregunta 5

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

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

bool datoEnInterv(double x) {
    return x >= 0 && x <= 1;
}

int main() {
    std::ifstream file_in(nombreFich.c_str(),std::ios::in);
    if (!file_in) {
        std::cout << "No se puede abrir el fichero"<< std::endl;
        return 1;
    }
    // Lectura del fichero
    std::vector<double> datos;
    while (!file_in.eof()) {
        double d;
        file_in >> d;
        if (!file_in.eof())
            datos.push_back(d);
    }
    // Comprobación de que N>0
    int N = datos.size();
    if (N<1) {
        std::cout << "ERROR: Datos insuficientes"<< std::endl;
        return 0;
    }
    // Aplicación del algoritmo
    int nDatosEnInterv = count_if(datos.begin(),datos.end(),datoEnInterv);
    // Mostrar el resultado en la consola
    std::cout << "Datos en [0,1]: "<< nDatosEnInterv << std::endl;
    return 0;
}
```