LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2012, 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) La máquina de von Neumann tiene dos memorias: una para almacenar datos y otra para almacenar instrucciones.

B. (0.5 puntos) En FORTRAN I no se declara el tipo de las variables.

C. (0.5 puntos) Las sentencias de control del flujo del programa en Ada finalizan con una palabra reservada, que depende del tipo de sentencia.

D. (0.5 puntos) Algunos lenguajes de programación no permiten que la variable del bucle sea modificada dentro del cuerpo del bucle.

E. (0.5 puntos) En un bucle lógico postcondición se ejecuta siempre el cuerpo del bucle al menos una vez.

F. (0.5 puntos) La inserción de nuevos elementos en una cola se produce en el principio de la misma.

Solución a la Pregunta 1

A	Falsa	Véase la página 35 del texto base.
B	Verdadera	Véase la página 43 del texto base.
C	Verdadera	Véase la página 264 del texto base.
D	Verdadera	Véase la página 272 del texto base.
E	Verdadera	Véase la página 277 del texto base.
F	Falsa	Véase la página 432 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 a = 0;
int b = 2;
int i = 1;

int main()
{
    int b = 4;
    for (int i = 1; i<4; i++) {
        a++;
    }
    std::cout << i << std::endl;
    std::cout << a << std::endl;
    std::cout << b << " " << ::b << std::endl;
    {
        int a = 5;
        std::cout << a << " " << ::a << std::endl;
    }
    std::cout << a << std::endl;
    return 0;
}
```

Solución a la Pregunta 2

```
1
3
4 2
5 3
3
```

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>

void imprimevector(const std::vector<int> &v) {
    for (unsigned int i=0; i<v.size()-1; i++)
        std::cout << v[i] << ",";
    std::cout << v[v.size()-1] << std::endl;
    return;
}

int main()
{
    std::vector<int> v;
    for (int i=0; i<10; i++)
        v.push_back(i);
    v.pop_back();
    imprimevector(v);

    std::vector<int>::iterator pBegin = v.begin()+1;
    std::vector<int>::iterator pEnd   = v.begin()+4;
    std::cout <<    "(*pBegin): " << (*pBegin) <<
                    "\t(*pEnd): " << (*pEnd)   << std::endl;

    reverse(pBegin,pEnd);
    imprimevector(v);
    std::cout <<    "(*pBegin): " << (*pBegin) <<
                    "\t(*pEnd): " << (*pEnd)   << std::endl;
    return 0;
}
```

Solución a la Pregunta 3

```
0,1,2,3,4,5,6,7,8
(*pBegin): 1 (*pEnd): 4
0,3,2,1,4,5,6,7,8
(*pBegin): 3 (*pEnd): 4
```

PREGUNTA 4 (2.5 puntos)

Escriba en C++ el código siguiente:

A. (0.5 puntos) La definición de una función que calcule el valor medio de los elementos de la lista que se pasa como argumento. Si el número de elementos de la lista es cero, la función lanza una excepción. La declaración de la función es:

```
double mediaValores(std::list<double> &v)
   throw (std::domain_error);
```

B. (2 puntos) Escriba un programa que calcule las medias móviles, con una ventana de 5 valores, de los valores numéricos almacenados en un fichero de texto. El nombre del fichero, así como el número de valores de la ventana, deben ser constantes globales. Se indica a continuación el nombre, tipo y valor de las mismas.

Nombre	Tipo	Valor
`NUM_DATOS_VENTANA`	**int**	5
`NOMBRE_FICH`	`std::string`	`"datos.txt"`

El programa debe realizar las acciones siguientes:

1. Abrir para lectura el fichero de texto. Si se produce error, mostrar un mensaje indicándolo y terminar.
2. Leer los primeros `NUM_DATOS_VENTANA` datos almacenados en el fichero, almacenándolos en una lista de **double**. Si el número de datos que contiene el fichero es menor que `NUM_DATOS_VENTANA`, mostrar un mensaje en la consola indicándolo y terminar.
3. Mostrar en la consola el valor medio de los elementos de la lista, invocando para ello la función programada en la Parte A del ejercicio.
4. A continuación, el programa debe continuar leyendo uno a uno los valores almacenados en el fichero, añadiendo en cada caso el valor leído del fichero al final de la lista, extrayendo el primer valor de la lista, calculando a continuación el valor medio de los elementos de la lista y mostrando dicho valor en la consola. Dado que cada vez que se añade un elemento a la lista se extrae otro, el número de elementos

almacenados en la lista es siempre `NUM_DATOS_VENTANA`. El programa finaliza cuando se ha leído el último número almacenado en el fichero y se ha mostrado en la consola la correspondiente media móvil.

Al invocar la función `mediaValores`, debe escribirse el código necesario para capturar la excepción que puede lanzar la función. Si se captura una excepción lanzada por la función, debe mostrarse el correspondiente mensaje en la consola y terminar.

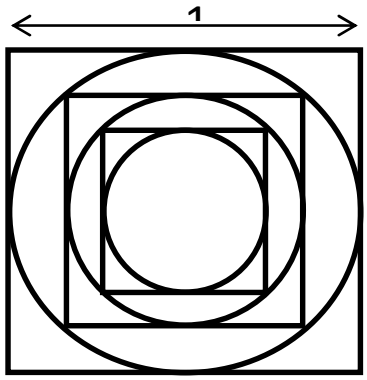
Solución a la Pregunta 4

```
#include <iostream>
#include <string>
#include <fstream>
#include <list>
#include <stdexcept>

const std::string NOMBRE_FICH = "datos.txt";
const int NUM_DATOS_VENTANA = 5;

double mediaValores(std::list<double> &v)
     throw (std::domain_error);

int main() {
     std::ifstream fich(NOMBRE_FICH.c_str(),std::ios::in);
     if (!fich) {
          std::cerr << "No es posible abrir el fichero "  <<
               NOMBRE_FICH << std::endl;
          return -1;
     }
     std::list<double> ventana;
     double datoLeido;
     while (fich >> datoLeido) {
          ventana.push_back(datoLeido);
          if (ventana.size()>NUM_DATOS_VENTANA)
               ventana.pop_front();
          if (ventana.size()==NUM_DATOS_VENTANA) {
               try {
                    std::cout << "Media: "<< mediaValores(ventana)
                                                << std::endl;
               } catch (std::domain_error exc) {
                    std::cerr << exc.what() << std::endl;
                    return -1;
               }
          }
     }
     if (ventana.size()<NUM_DATOS_VENTANA)
          std::cout << "El fichero contiene menos de "<<
               NUM_DATOS_VENTANA << " valores"<< std::endl;
     fich.close();
     return 0;
}

double mediaValores(std::list<double> &v) throw (std::domain_error)
{
     if (!v.size())
          throw std::domain_error("ERROR: Lista con cero elementos");
     double suma = 0;
     for (std::list<double>::iterator p=v.begin(); p!=v.end(); p++)
          suma += (*p);
     return suma/v.size();
}
```

PREGUNTA 5 (2.5 puntos)

Se desea calcular la secuencia de valores de las áreas de los círculos inscritos en cuadrados y de los cuadrados inscritos en dichos círculos.

Se parte de un cuadrado de lado unidad. El círculo inscrito en este cuadrado tiene diámetro unidad. El siguiente elemento de la secuencia corresponde al cuadrado inscrito en dicho círculo y así sucesivamente. En la figura se representan gráficamente los primeros seis elementos geométricos de dicha secuencia.



Escriba en C++ el código siguiente:

A. (0.25 puntos) Una función que calcule el radio del círculo inscrito en un cuadrado, conocido el lado del cuadrado.

```
double radioCirculoInscrito(double ladoCuadrado);
```

B. (0.25 puntos) Una función que calcule el área de un círculo a partir de su radio.

```
double areaCirculo(double radioCirculo);
```

C. (0.25 puntos) Una función que calcule el lado del cuadrado inscrito en un círculo, conocido el radio del círculo.

```
double ladoCuadradoInscrito(double radioCirculo);
```

D. (0.25 puntos) Una función que calcule el área de un cuadrado a partir de su lado.

```
double areaCuadrado(double ladoCuadrado);
```

E. (1.5 puntos) Un programa que, invocando las funciones anteriores, muestre en la consola el área de los 10 primeros elementos geométricos de la serie (5 cuadrados y 5 círculos). El elemento inicial es un cuadrado de lado unidad.

Solución a la Pregunta 5

```
#include <iostream>
#include <cmath>

const double PI = 3.14159265;
const int      ITERACIONES = 10;
const double LADO_INICIAL_CUADRADO = 1;

double areaCuadrado(double ladoCuadrado);
double areaCirculo(double radioCirculo);
double ladoCuadradoInscrito(double radioCirculo);
double radioCirculoInscrito(double ladoCuadrado);

int main() {
     int numTerminosSerie = 1;

     double ladoCuadrado = LADO_INICIAL_CUADRADO;
     std::cout << "Cuadrado:\t"<<
          areaCuadrado(ladoCuadrado) << std::endl;
     bool esCuadrado = false;    // true: cuadrado; false: círculo

     double radioCirculo = 0;
     while ( numTerminosSerie < ITERACIONES ) {
          if (esCuadrado) {
               ladoCuadrado = ladoCuadradoInscrito(radioCirculo);
               std::cout << "Cuadrado:\t"<<
                    areaCuadrado(ladoCuadrado) << std::endl;
          } else {
               radioCirculo = radioCirculoInscrito(ladoCuadrado);
               std::cout << "Circulo:\t"<<
                    areaCirculo(radioCirculo) << std::endl;
          }
          esCuadrado = !esCuadrado;
          numTerminosSerie++;
     }
     return 0;
}

double areaCuadrado(double ladoCuadrado) {
     return ladoCuadrado*ladoCuadrado;
}

double areaCirculo(double radioCirculo) {
     return PI*radioCirculo*radioCirculo;
}

double ladoCuadradoInscrito(double radioCirculo) {
     return std::sqrt(2*radioCirculo*radioCirculo);
}

double radioCirculoInscrito(double ladoCuadrado) {
     return ladoCuadrado/2;
}
```