LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2012, 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) Java fue el primer lenguaje en introducir los conceptos de puntero y de gestión dinámica de la memoria.

B. (0.5 puntos) Una estructura o record es una colección de una o más variables, que forzosamente deben ser del mismo tipo, agrupadas bajo un nombre común.

C. (0.5 puntos) Las variables en memoria dinámica dejan de existir cuando la ejecución del programa abandona el bloque en que han sido declaradas.

D. (0.5 puntos) En el lenguaje C las cadenas de caracteres se almacenan en arrays unidimensionales de caracteres.

E. (0.5 puntos) En C y C++ los parámetros de tipo array se pasan por valor a las funciones.

F. (0.5 puntos) En C++ las variables estáticas declaradas en el cuerpo de una función tienen el mismo ámbito que las variables globales.

Solución a la Pregunta 1

A	Falsa	Véase la página 51 del texto base.
B	Falsa	Véase la página 119 del texto base.
C	Falsa	Véase la página 114 del texto base.
D	Verdadera	Véase la página 123 del texto base.
E	Falsa	Véase la página 372 del texto base.
F	Falsa	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 main()
{
    int *p1, *p2;
    int k = 20;
    p1 = &k;
    std::cout <<  "*p1: " << *p1 << std::endl;
    *p1 = 5;
    std::cout <<  "*p1: " << *p1 << std::endl;
    std::cout << "k: " << k << std::endl;
    p2 = p1;
    *p2 = 3;
    std::cout << "*p1: " << *p1 << std::endl;
    std::cout << "*p2: " << *p2 << std::endl;
    p1 = new int;
    std::cout << "*p2: " << *p2 << std::endl;
    *p1 = 8;
    std::cout << "*p1: " << *p1 << std::endl;
    std::cout << "*p2: " << *p2 << std::endl;
    return 0;
}
```

Solución a la Pregunta 2

```
*p1: 20
*p1: 5
k: 5
*p1: 3
*p2: 3
*p2: 3
*p1: 8
*p2: 3
```

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>

int main()
{
    std::list<int> lista1;
    for (int i=0;i<10;i++)
        lista1.push_back(i);
    std::list<int>::iterator p1 = lista1.begin();
    p1++;
    p1++;
    std::cout << *p1 << std::endl;

    std::list<int>::iterator p2 = lista1.end();
    p2--;
    std::cout << *p2 << std::endl;

    std::list<int> lista2(p1,p2);
    lista2.pop_front();
    lista2.pop_front();
    std::cout << *(lista2.begin()) << std::endl;
    return 0;
}
```

Solución a la Pregunta 3

```
2
9
4
```

PREGUNTA 4 (3 puntos)

El *método de la bisección* es un algoritmo de búsqueda de raíces para ecuaciones de una variable. El método se basa en el teorema del valor intermedio, que establece que toda función continua $f(x)$ en un intervalo cerrado $[a, b]$ toma todos los valores que se hallan entre $f(a)$ y $f(b)$. Esto es, que cualquier valor entre $f(a)$ y $f(b)$ es la imagen de al menos un valor en el intervalo $[a, b]$. Si $f(a)$ y $f(b)$ tienen signos opuestos, entonces el valor cero es un valor intermedio entre $f(a)$ y $f(b)$, por lo que con certeza existe un p en $[a, b]$ que cumple $f(p) = 0$. De esta forma, se asegura la existencia de al menos una solución de la ecuación $f(x) = 0$.

Escriba un programa en C++ que aplique el método de la bisección a la función siguiente:

```
double f(double x);
```

El programa debe realizar las acciones siguientes:

1. Solicitar que se introduzcan por consola los extremos a y b del intervalo inicial de búsqueda $[a, b]$. Leer de la consola los valores, almacenándolos en variables de tipo **double**. Comprobar que se satisface $a < b$. Si no se satisface esta condición, mostrar un mensaje de error y terminar.

2. Comprobar que $f(a) \cdot f(b) < 0$. Si no se verifica esta condición, mostrar un mensaje indicándolo y terminar.

3. Solicitar que se introduzca por consola el número máximo de iteraciones. Almacenar el valor introducido por consola en la variable N, de tipo **int**.

4. Declarar la variable entera i, que almacena el número de iteraciones del algoritmo, e inicializarla a cero ($i = 0$).

5. Calcular el punto medio m del intervalo de búsqueda $[a, b]$. Evaluar $f(m)$. TOL es una constante global de tipo **double**, con valor 10^{-6}.

   - Si $|f(m)| \leq$ TOL, entonces se ha encontrado una raíz. Mostrar en consola el valor de la raíz m, del residuo $f(m)$ y terminar.
   - Si $|f(m)| >$ TOL, decidir si $f(m)$ tiene signo opuesto a $f(a)$ o a $f(b)$. Redefinir el intervalo de búsqueda como $[a, m]$ o $[m, b]$, según se haya determinado en cuál de estos intervalos ocurre un cambio de signo. Es decir, en el primer caso se asigna a b el valor m y en el segundo caso se asigna a a el valor m.

6. Si el número de iteraciones (i) ha alcanzado el valor máximo (N), mostrar en consola el intervalo actual de búsqueda, que es la acotación obtenida para la raíz, y terminar. En caso contrario, incrementar i en uno y saltar al Paso 5.

Solución a la Pregunta 4

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

const double TOL = 1e-6;

double f(double x) {
     return x-1;
}

int main() {
     std::cout << "Intervalo de busqueda [a,b]\n\ta: ";
     double a,b;
     std::cin >> a;
     std::cout << "\tb: ";
     std::cin >> b;
     if (a >= b) {
          std::cout << "ERROR: b debe ser mayor que a"<< std::endl;
          return -1;
     }
     if (f(a)*f(b) >= 0) {
          std::cout << "ERROR: no se satisface f(a)*f(b)<0"<< std::endl;
          return -1;
     }
     int N;
     std::cout << "Num. iteraciones: ";
     std::cin >> N;

     for (int i=0; i<N; i++) {
          double m = (a+b)/2;
          if (fabs(f(m)) <= TOL) {
               std::cout << "Raiz: "<< m <<
                    "\nResiduo: "<< f(m) << std::endl;
               return 0;
          }
          if (f(a)*f(m)<0) {
               b = m;
          } else {
               a = m;
          }
     }
     std::cout << "Raiz acotada al intervalo: ("<<
          a << ", "<< b << ")"<< std::endl;
     return 0;
}
```

PREGUNTA 5 (2 puntos)

Los sistemas L son sistemas recurrentes que se emplean, por ejemplo, para representar el crecimiento de las plantas. Un sistema sencillo tiene las reglas siguientes:

$$\begin{aligned} n_i &= n_{i-2} + n_{i-1} \qquad \text{para } i : 3, 4, \ldots \\ n_1 &= cadena1 \\ n_2 &= cadena2 \end{aligned}$$

donde el operador + debe interpretarse como concatenación. Por ejemplo, si

$$\begin{aligned} cadena1 &= A \\ cadena2 &= B \end{aligned}$$

se obtiene la secuencia:

$$\begin{aligned} n_1 &= A \\ n_2 &= B \\ n_3 &= AB \\ n_4 &= BAB \\ n_5 &= ABBAB \\ &\ldots \end{aligned}$$

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

1. Solicite al usuario la entrada por consola de:
   - El número de elementos N de la secuencia, donde N debe ser mayor que 2.
   - El valor de las cadenas de caracteres $cadena1$ y $cadena2$. Las cadenas deben almacenarse en variables del tipo `std::string`.

2. Genere la secuencia $n_1, \ldots, n_N$, almacenándola en una lista cuyos elementos sean del tipo `std::string`. Cada elemento de la lista debe almacenar un elemento de la secuencia.

3. Muestre en la consola el último elemento de la lista.

Solución a la Pregunta 5

```
//Generación de una cadena de una sistema de tipo L
#include <iostream>
#include <list>
#include <string>

int main() {
    int N;
    do {
        std::cout << "Numero de iteraciones (>2): ";
        std::cin >> N;
    } while (N<=2);
    std::string cadena1,cadena2;
    std::cout << "\ncadena1: ";
    std::cin >> cadena1;
    std::cout << "\ncadena2: ";
    std::cin >> cadena2;

    std::list <std::string> secuencia;
    secuencia.push_back(cadena1);
    secuencia.push_back(cadena2);
    for (int i=1; i<N-1; i++) {
        std::list <std::string>::iterator p = secuencia.end();
        std::string ultimoElem = *(--p);
        std::string penultimoElem = *(--p);
        secuencia.push_back(penultimoElem+ultimoElem);
    }
    // Muestra el último elemento de la lista
    std::cout << *(--secuencia.end()) << std::endl;
    return 0;
}
```