LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2023, 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) En el lenguaje Python los bloques de código se delimitan por medio de llaves.

B. (0.5 puntos) Las variables en memoria dinámica existen hasta que el programa termina o hasta que son eliminadas.

C. (0.5 puntos) Sea un vector `v` que es declarado en C++ de la manera siguiente:

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

La función `v.end()` devuelve un iterador que apunta al último componente de `v`.

D. (0.5 puntos) En C++ no es posible definir un tipo de estructura en la cual uno de sus miembros sea una estructura de ese mismo tipo.

E. (0.5 puntos) En la cola, la inserción de nuevos elementos se produce en el principio de la lista.

F. (0.5 puntos) El tipo `std::map` de la STL es una implementación del tipo abstracto de datos mapa. Cada elemento almacenado en el mapa contiene dos campos: clave y valor, no pudiendo haber en el mapa dos elementos con la misma clave.

Solución a la Pregunta 1

A	Falso	Véase la página 55 del texto base.
B	Verdadero	Véase la página 118 del texto base.
C	Falso	Véase la página 223 del texto base.
D	Verdadero	Véase la página 236 del texto base.
E	Falso	Véase la página 466 del texto base.
F	Verdadero	Véase la página 504 del texto base.

PREGUNTA 2 (1 punto)

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

```
#include <iostream>

struct nodo {
    int d;
    struct nodo* x;
};

struct nodo* funcCon();

int main() {
    struct nodo* nodo0 = funcCon();
    while (nodo0 != nullptr) {
        std::cout << nodo0->d << std::endl;
        nodo0 = nodo0->x;
    }
    return 0;
}

struct nodo* funcCon() {
    struct nodo* n1 = new nodo;
    struct nodo* n2 = new nodo;
    struct nodo* n3 = new nodo;
    struct nodo* n4 = new nodo;
    struct nodo* n5 = new nodo;
    struct nodo* n6 = new nodo;
    struct nodo* n7 = new nodo;
    n1->d = 10; n1->x = n2;
    n2->d = -4; n2->x = n4;
    n3->d = 2;  n3->x = n4;
    n4->d = 8;  n4->x = n5;
    n5->d = 3;  n5->x = n6;
    n6->d = 5;  n6->x = n7;
    n7->d = -9; n7->x = nullptr;
    return n1;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
10
-4
8
3
5
-9
```

PREGUNTA 3 (1 punto)

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

```
#include <iostream>
#include <string>

enum MiTipo1 { error1a, error1b, error1c };
enum MiTipo2 { error2a, error2b };

int main() {
  for (int i = 1; i < 9; i++)
    try {
      switch (i) {
      case 1: { int n = -8;                  throw n; }
      case 2: { char c = 'b';                throw c; }
      case 3: { std::string s = "Excepcion"; throw s; }
      case 4: { bool a = false;              throw a; }
      case 5: { MiTipo1 mte1 = error1c;      throw mte1; }
      case 6: { MiTipo2 mte2 = error2a;      throw mte2; }
      case 7: { double x = 2.64;             throw x; }
      default: std::cout << "Error\n";
      }
    }
    catch (char exc) {
      std::cout << "Excepcion char: " << exc << std::endl;
    }
    catch (int exc) {
      std::cout << "Excepcion int: " << exc << std::endl;
    }
    catch (double exc) {
      std::cout << "Excepcion double: " << exc << std::endl;
    }
    catch (bool exc) {
      std::cout << "Excepcion bool: " << std::boolalpha
                << exc << std::endl;
    }
    catch (std::string exc) {
      std::cout << "Excepcion string: " << exc << std::endl;
    }
    catch (MiTipo1 exc) {
      std::cout << "Excepcion MiTipo1: " << exc << std::endl;
    }
    catch (...) { std::cout << "Otra" << std::endl; }
    return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
Excepcion int: -8
Excepcion char: b
Excepcion string: Excepcion
Excepcion bool: false
Excepcion MiTipo1: 2
Otra
Excepcion double: 2.64
Error
```

PREGUNTA 4 (2.5 puntos)

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

1. Declarar una constante global llamada `E` y asignarle el valor 0.001.
2. Mediante la escritura de mensajes en la consola, solicitar al usuario que introduzca por consola las coordenadas cartesianas (x, y) de tres puntos en el plano. Leer las coordenadas de los tres puntos, almacenándolas en variables del tipo estructura `Punto` llamadas `p1`, `p2` y `p3`. Dicha estructura es declarada de la forma siguiente:
   ```
   struct Punto {
       double x, y;
   };
   ```
3. Calcular la longitud de los tres lados del triángulo cuyos vértices son los tres puntos introducidos por el usuario. Almacenar estas longitudes en un array de tres elementos **double** llamado `lados`.

   Para ello, programe usted una función en C++ tal que dados dos puntos devuelva la distancia entre ellos e invóquela desde el programa. El prototipo de la función es:
   ```
   double distancia(Punto p1, Punto p2);
   ```
4. Empleando el array `lados`, comprobar si alguna de las longitudes es menor que la tolerancia E (la constante global del programa). En caso afirmativo, mostrar un mensaje en la consola indicando que dos puntos son casi iguales y terminar.
5. Calcular si el triángulo es aproximadamente equilátero y escribir un mensaje en la consola indicándolo. El triángulo es aproximadamente equilátero si la longitud de sus lados difiere en un valor menor a la tolerancia E.
6. Calcular el área del triángulo cuyos vértices son los tres puntos introducidos por el usuario y escribir el área en la consola en formato fijo con 3 dígitos de precisión. Debe emplearse la fórmula de Herón, que permite calcular el área (A) de un triángulo a partir de la longitud de sus lados (a, b, c) y de su semiperímetro (s):
   $$\begin{aligned} s &= \frac{a+b+c}{2} \\ A &= \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)} \end{aligned}$$
7. Terminar.

Solución a la Pregunta 4

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>

const double E = 0.001;

struct Punto {
   double x, y;
};

double distancia(Punto p1, Punto p2) {
   return std::sqrt(
         std::pow(p1.x - p2.x, 2) +
         std::pow(p1.y - p2.y, 2) );
}

int main() {
   // Entrada por consola
   Punto p1, p2, p3;
   std::cout << "Coordenadas punto 1: ";
   std::cin >> p1.x >> p1.y;
   std::cout << "Coordenadas punto 2: ";
   std::cin >> p2.x >> p2.y;
   std::cout << "Coordenadas punto 3: ";
   std::cin >> p3.x >> p3.y;
   // Longitud de los lados
   double lados[] = {
                  distancia(p1,p2),
                  distancia(p1,p3),
                  distancia(p2,p3) };
   // Puntos casi iguales
   if ( lados[0] < E || lados[1] < E || lados[2] < E ) {
      std::cout << "Puntos casi iguales";
      return 0;
   }
   // Casi equilatero
   if (  std::abs(lados[0] - lados[1]) < E &&
         std::abs(lados[0] - lados[2]) < E &&
         std::abs(lados[1] - lados[2]) < E )
      std::cout << "Aproximadamente equilatero\n";
   else
      std::cout << "No aproximadamente equilatero\n";
   // Area del triangulo
   double s = 0.5*(lados[0]+lados[1]+lados[2]);
   double A = std::sqrt(s*(s-lados[0])*(s-lados[1])*(s-lados[2]));
   std::cout << std::fixed << std::setprecision(3)
             << A << std::endl;
   return 0;
}
```

PREGUNTA 5 (2.5 puntos)

El método de la secante es un algoritmo para resolver $f(x) = 0$. Partiendo de dos valores iniciales x_0 y x_1, el método de la secante obtiene x_2, x_3, ... aplicando la ecuación siguiente:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \cdot f(x_k) \qquad \text{para } k = 1, 2, \ldots$$

Aplicando el método de la secante, se desea resolver la ecuación $f(x) = 0$ para

$$f(x) = a \cdot x^n - b \cdot \cos(x) + c$$

donde a, b y c representan números reales, y n representa un número entero mayor que cero. Los valores de a, b, c y n están almacenados en un fichero de texto llamado *config.txt*. El formato de dicho fichero es el siguiente. Cada valor se especifica en una línea. En la primera columna se indica el nombre (es decir, a, b, c o n) y en la segunda el valor. Las cuatro líneas no siguen ningún orden determinado. Por ejemplo, el contenido del fichero *config.txt* podría ser el siguiente:

```
b   2.4
a  -1.35
c   6.2
n   2
```

5.1 (1 punto) Escriba una función en C++ con el prototipo siguiente:

```
Config leeFichero(std::string nombreFichero)
                    throw (std::invalid_argument);
```

donde `Config` es una estructura definida de la manera siguiente:

```
struct Config {
    double a, b, c;
    unsigned int n;
};
```

La función debe leer el fichero de texto, cuyo nombre es pasado como parámetro, devolviendo en los correspondientes campos de la estructura los

valores de a, b, c y n almacenados en el fichero. El formato del fichero es el indicado anteriormente para el fichero *config.txt*.

Si se produce error al abrir el fichero para lectura, la función debe lanzar una excepción.

Al programar la función, puede asumir que el fichero no tiene errores de formato.

5.2 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
double f(Config &params, double *x);
```

que devuelva el valor resultante de evaluar la función $f(x)$ definida mediante la ecuación:

$$f(x) = a \cdot x^n - b \cdot \cos(x) + c$$

donde los valores de a, b, c y n son especificados mediante el primer parámetro pasado a la función, y el valor de x es especificado mediante el segundo parámetro.

5.3 (1 punto) Escriba un programa principal en C++ que realice las acciones siguientes:

1. Solicitar por consola al usuario que éste introduzca por consola los valores iniciales x_0 y x_1, y un número entero N que es el número máximo de iteraciones del método.
2. Invocar la función `leeFichero`, a fin de leer los valores de a, b, c y n del fichero *config.txt*, almacenando el valor devuelto en una variable llamada `params`. Si la función lanza una excepción, mostrar un mensaje en la consola indicándolo y terminar.
3. El programa debe ir calculando las soluciones aproximadas x_2, x_3, $\ldots$, aplicando el método de la secante, para lo cual debe invocar repetidamente la función `f`.

   El programa debe ir escribiendo en la consola los valores de k, x_k y $f(x_k)$, formando una tabla con tres columnas separadas por un tabulador, donde cada fila corresponde con un valor de k. Los valores de x_k y $f(x_k)$ deben mostrarse en formato científico, con 10 dígitos de precisión.

   La condición de finalización del método de la secante es que se satisfaga cualquiera de las tres condiciones siguientes:

a) Que el número de iteraciones alcance un determinado valor máximo N.

b) Que se satisfaga $f(x_{k+1}) = f(x_k)$.

c) Que se satisfaga $\left|\frac{x_{k+1}-x_k}{x_{k+1}}\right| < \varepsilon$, donde ε es una constante global del programa cuyo valor es $\varepsilon = 10^{-10}$.

4. Cuando se satisfaga la condición de finalización, terminar.

Solución a la Pregunta 5

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

const std::string nombreFich = "config.txt";
const double eps = 1e-10;

struct Config {
   double a, b, c;
   unsigned int n;
};

Config leeFichero(std::string nombreFichero)
                     throw (std::invalid_argument) {
   std::ifstream inFich(nombreFichero.c_str(), std::ios::in);
   if (!inFich)
      throw std::invalid_argument("Error al abrir el fichero");
   Config params;
   char param;
   double val;
   while (inFich >> param >> val)
      switch (param) {
      case 'a':
         params.a = val;
         break;
      case 'b':
         params.b = val;
         break;
      case 'c':
         params.c = val;
         break;
```

```
35        case 'n':
36           params.n = val;
37           break;
38        }
39     inFich.close();
40     return params;
41  }
42
43  double f(Config &params, double *x) {
44     return   params.a * std::pow(*x,params.n)
45            - params.b * std::cos(*x)
46            + params.c;
47  }
48
49  int main() {
50     // Entrada por consola
51     double x0, x1;
52     int N;
53     std::cout << "x0: ";   std::cin >> x0;
54     std::cout << "x1: ";   std::cin >> x1;
55     std::cout << "N: ";    std::cin >> N;
56     // Lectura del fichero
57     Config params;
58     try {
59        params = leeFichero(nombreFich);
60     } catch (std::invalid_argument &exc) {
61        std::cerr << exc.what() << std::endl;
62        return 0;
63     }
64     // Metodo de la secante
65     double f0 = f(params,&x0);
66     for (int k = 1; k <= N; k++) {
67        double f1 = f(params,&x1);
68        if ( f1 == f0 ) {
69           std::cout << "Se satisface f1 = f0" << std::endl;
70           return 0;
71        }
72        double dif = (x1-x0)*f1/(f1-f0);
73        x0 = x1;
74        f0 = f1;
75        x1 -= dif;
76        // Salida por consola
77        std::cout << k << "\t" <<
78           std::scientific << std::setprecision(10) <<
79              x1 << "\t" << f1 << "\t" << std::endl;
80        if ( x1 != 0 && std::abs(dif/x1) < eps ) {
81           std::cout << "Se satisface abs(dif/x1)<eps" << std::endl;
82           return 0;
83        }
84     }
85     std::cout << "Alcanzado limite de iteraciones" << std::endl;
86     return 0;
87  }
```