LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2019, 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 FORTRAN I una variable cuyo nombre sea ANUM es de tipo entero.

B. (0.5 puntos) La parte del programa en que una variable es visible puede ser mayor que el ámbito de dicha variable.

C. (0.5 puntos) El tipo unión `uni` es declarado en C++ de la manera siguiente:

```
union uni {int i; char nombre;};
```

Una variable de este tipo contiene en cada instante un valor de tipo **int** y además un valor de tipo **char**.

D. (0.5 puntos) La excepción de C++ `std:bad_alloc` se puede producir al emplear el operador `new`.

E. (0.5 puntos) El tipo de acceso a los elementos del tipo de dato `std::list` es secuencial y bidireccional.

F. (0.5 puntos) El algoritmo de ordenación por mezcla sigue el paradigma de la fuerza bruta.

Solución a la Pregunta 1

A	Falso	Véase la página 43 del texto base.
B	Falso	Véase la página 115 del texto base.
C	Falso	Véase la página 120 del texto base.
D	Verdadero	Véase la página 305 del texto base.
E	Verdadero	Véase la página 455 del texto base.
F	Falso	Véase la página 512 del texto base.

PREGUNTA 2 (1 punto)

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

```
#include <iostream>
#include <iomanip>

int main () {
   double *p1, *p2, *p3;
   double a[3];
   a[0] = 0.041398;
   a[1] = 3.27382;
   a[3] = 2.43984;
   p1 = a;
   p2 = p1+1;
   p3 = &a[1];
   *p3= 1.383;
   std::cout << std::scientific << std::setprecision(3)
             << *p1             << std::endl;
   std::cout << std::setprecision(2) << p1[1] << std::endl;
   std::cout << std::setprecision(2) << *p2   << std::endl;
   std::cout << std::setprecision(2) << *p3   << std::endl;
   p3 = &a[2];
   p1 = p1 + 2;
   *p3 = 10.994;
   std::cout << a[2]    << std::endl;
   std::cout << p1[0]   << std::endl;
   std::cout << *(p1-1) << std::endl;
   std::cout << *p1     << std::endl;
   return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
4.140e-002
1.38e+000
1.38e+000
1.38e+000
1.10e+001
1.10e+001
1.38e+000
1.10e+001
```

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 <algorithm>

std::string L2S(std::list<int> &lst) {
   std::stringstream ss;
   if (lst.empty()) {
      ss << "(lista vacia)";
   } else {
      std::list<int>::iterator p = lst.begin();
      while ( p != lst.end() ) {
         ss << *p << " ";
         p++;
      }
   }
    return ss.str();
}

int main() {
   std::list<int> lista1, lista2;
   for (int i=0; i<6; ++i) {
      lista1.push_back(i*2);
      lista2.push_front(i+5);
   }
   std::cout << "lista1 -1 "   << L2S(lista1) <<
                "\nlista2 -1 " << L2S(lista2) << std::endl;
   std::list<int>::iterator p =
                std::find(lista2.begin(),lista2.end(), 6);
   lista2.splice(p, lista1);
   std::cout << "lista1 -2 "   << L2S(lista1) <<
                "\nlista2 -2 " << L2S(lista2) << std::endl;
   lista2.splice(lista2.end(), lista2, lista2.begin());
   std::cout << "lista1 -3 "   << L2S(lista1) <<
                "\nlista2 -3 " << L2S(lista2) << std::endl;
   lista2.sort();
   lista1 = lista2;
   std::cout << "lista1 -4 "   << L2S(lista1) <<
                "\nlista2 -4 " << L2S(lista2) << std::endl;
   lista1.merge(lista2);
   std::cout << "lista1 -5 "   << L2S(lista1) <<
                "\nlista2 -5 " << L2S(lista2) << std::endl;
   return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
lista1 -1 0 2 4 6 8 10
lista2 -1 10 9 8 7 6 5
lista1 -2 (lista vacia)
lista2 -2 10 9 8 7 0 2 4 6 8 10 6 5
lista1 -3 (lista vacia)
lista2 -3 9 8 7 0 2 4 6 8 10 6 5 10
lista1 -4 0 2 4 5 6 6 7 8 8 9 10 10
lista2 -4 0 2 4 5 6 6 7 8 8 9 10 10
lista1 -5 0 0 2 2 4 4 5 5 6 6 6 6 7 7 8 8 8 8 9 9 10 10 10 10
lista2 -5 (lista vacia)
```

PREGUNTA 4 (2 puntos)

Se desea traducir mensajes a secuencias de bits. Los mensajes están construidos empleando únicamente ocho símbolos: `A`, `B`, `C`, `D`, `E`, `F`, `G` y `H`. Para ello, se emplea la tabla de codificación siguiente:

`A` 0	`C` 1010	`E` 1100	`G` 1110
`B` 100	`D` 1011	`F` 1101	`H` 1111

Mediante este código, el mensaje

BACADAEAFABBAAAGAH

se representaría mediante la siguiente secuencia de bits:

1000101001011011000110101001000000111001111

Escriba un programa en C++ que:

1. Solicite al usuario que éste introduzca por consola una cadena de caracteres.
2. Realice una de las dos acciones siguientes:
   - Si se trata de un mensaje, el programa debe traducirlo a su correspondiente codificación binaria y mostrar ésta por consola.

     Para ello, el programa debe invocar una función que acepte como argumento un símbolo y devuelva su código equivalente. Si el símbolo pasado como argumento no es ninguno de los ocho posibles símbolos, la función debe lanzar una excepción.

     Programe dicha función, teniendo en cuenta que su cabecera debe ser:

     ```
     std::string codifica(char c)
                    throw (std::invalid_argument);
     ```

   - Si la cadena de caracteres introducida por el usuario no es un mensaje válido, el programa debe notificárselo por consola al usuario, indicando además en qué posición de la cadena de caracteres se encuentra el primer carácter no válido y cuál es.
3. Termine.

Solución a la Pregunta 4

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

std::string codifica(char c)
          throw (std::invalid_argument);

int main() {
    // Entrada por consola
    std::cout << "Introduzca una cadena de caracteres:"
       << std::endl;
    std::string cadenaI;
    std::cin >> cadenaI;
    // Comprueba si la cadena está vacía
    if ( cadenaI.length() == 0 ) {
       std::cout << "Error: cadena vacia" << std::endl;
       return 0;
    }
    // Codifica el mensaje
    std::stringstream ss;
    for (int i=0; i<cadenaI.length(); i++) {
       try {
          ss << codifica(cadenaI[i]);
       } catch ( std::invalid_argument exc ) {
          std::cout << exc.what()
                    << " en posicion " << i+1 << ": "
                    << cadenaI[i] << std::endl;
          return 0;
       }
    }
    std::cout << ss.str() << std::endl;
    return 0;
}

std::string codifica(char c) throw (std::invalid_argument) {
    switch (c) {
       case 'A':
          return "0";
       case 'B':
          return "100";
       case 'C':
          return "1010";
       case 'D':
          return "1011";
       case 'E':
          return "1100";
       case 'F':
          return "1101";
       case 'G':
          return "1110";
       case 'H':
          return "1111";
       default:
          throw std::invalid_argument ("Caracter no valido");
    }
}
```

PREGUNTA 5 (3 puntos)

El *método Radix* de ordenación de números emplea varias veces el método de ordenación del casillero y ha sido empleado para la ordenación de cartas postales. El método consiste en ordenar los números en función de cada dígito, comenzando por el dígito menos significativo y finalizando por el más significativo.

A continuación se muestra un ejemplo. Deseamos ordenar el siguiente conjunto de números enteros de 2 dígitos comprendidos entre 0 y 99:

84, 12, 45, 01, 05, 17

En primer lugar, se aplica la ordenación de casillero por el dígito menos significativo.

Casillero 0:
Casillero 1: 01
Casillero 2: 12
Casillero 3:
Casillero 4: 84
Casillero 5: 45 05
Casillero 6:
Casillero 7: 17
Casillero 8:
Casillero 9:

Se obtiene la secuencia siguiente:

01, 12, 84, 45, 05, 17

A continuación, se aplica a esta secuencia la ordenación del casillero para el segundo dígito.

Casillero 0: 01 05
Casillero 1: 12 17
Casillero 2:
Casillero 3:
Casillero 4: 45
Casillero 5:
Casillero 6:
Casillero 7:
Casillero 8: 84
Casillero 9:

La secuencia obtenida está ordenada crecientemente:

01, 05, 12, 17, 45, 84

Programe el código C++ indicado a continuación.

5.1 (1 punto) Una función llamada `casillero`, que tenga como argumentos un vector de números enteros y un número entero. El prototipo se muestra a continuación.

```
std::vector<int> casillero(std::vector<int> c, int d);
```

La función debe realizar la ordenación del vector de números enteros `c`, mediante el método del casillero anteriormente descrito, para el dígito `d`. Suponga que el vector contiene números enteros positivos, incluyendo el cero. La función devuelve un vector con los números ordenados para el dígito `d`.

5.2 (2 puntos) Escriba un programa en C++ que realice las acciones siguientes.

1. Mediante un mensaje escrito en la consola, solicite al usuario que éste introduzca por consola el número de dígitos que tendrán los números a ordenar. En el ejemplo anterior, el usuario introduciría el número 2, ya que los números a ordenar tienen dos dígitos. El programa debe almacenar este número en una variable llamada `nd`.
2. Mediante un mensaje escrito en la consola, solicite al usuario que éste introduzca por consola cuántos números desea ordenar. En el ejemplo anterior, el usuario introduciría el número 6, ya que debían ordenarse seis números. El programa debe almacenar este número en una variable llamada `na`.
3. Mediante un mensaje escrito en la consola, solicite al usuario que éste introduzca por consola los números a ordenar. El programa debe almacenar estos números en un vector llamado `vNum`.
4. Realizar llamadas a la función `casillero`, a fin de ordenar los números mediante el *método Radix*.
5. Mostrar los números ordenados en la consola.
6. Terminar.

Ayuda. El dígito D de un número n se puede calcular como:

$$D = \text{floor}\left(\frac{n}{10^{d-1}}\right) - 10 \cdot \text{floor}\left(\frac{n}{10^{d}}\right)$$

siendo d la posición del dígito a calcular. La función `floor()` devuelve la parte entera de un número real.

Solución a la Pregunta 5

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

std::vector<int> casillero(std::vector<int> c, int d) {
     std::vector<int> v;
     for(int nCasillero=0; nCasillero<10; nCasillero++) {
          for (unsigned int k=0; k<c.size(); k++){
               int D = floor(c[k]/pow(10,d-1))-10*floor(c[k]/pow(10,d));
               if (D == nCasillero)
                    v.push_back(c[k]);
          }
     }
     return v;
}


int main() {
    // ------------------
    // Entrada por consola
    // ------------------
    int nd;
     std::cout << "Numero de digitos:" << std::endl;
     std::cin>> nd;
     int na;
     std::cout << "Numero de valores a ordenar:" << std::endl;
     std::cin>> na;
     std::cout << "Valores a ordenar: " << std::endl;
     std::vector<int> vNum;
     for (int i = 0; i < na; i++) {
        int dat;
          std::cin >> dat;
          vNum.push_back(dat);
     }
     // ----------
     // Ordenación
     // ----------
     for (int d = 1; d <= nd; d++)
        vNum = casillero(vNum, d);
     // ------------------
     // Salida por consola
     // ------------------
     std::cout << "Resultado:" << std::endl;
     for (int j = 0; j < na; j++)
          std::cout << vNum[j] << std::endl;
     return 0;
}
```