LENGUAJES DE PROGRAMACIÓN

Solución al examen de Junio 2025, 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 FORTRAN, la sentencia `implicit none` obliga al programador a declarar todos los nombres de variable.

B. (0.5 puntos) En un lenguaje de programación, cuando se pasa un parámetro por valor a un subprograma el valor original del parámetro puede ser modificado.

C. (0.5 puntos) En el lenguaje C++, el rango de valores que puede tomar cada tipo de dato básico está fijado por las reglas del lenguaje.

D. (0.5 puntos) El algoritmo de la STL `count(p1, p2, val)` devuelve el número de elementos de la secuencia delimitada por los iteradores en el rango [`p1`,`p2`] cuyo valor es igual que `val`.

E. (0.5 puntos) C++ no permite inicializar una lista al declararla.

F. (0.5 puntos) El lenguaje C no facilita la gestión de las excepciones.

Solución a la Pregunta 1

A	Verdadero	Véase la página 39 del texto base.
B	Falso	Véase la página 41 del texto base.
C	Falso	Véase la página 132 del texto base.
D	Falso	Véase la página 556 del texto base.
E	Falso	Véase la página 493 del texto base.
F	Verdadero	Véase la página 266 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>

double x, y;

int main() {
   x = 1.2135787;
   y = 5.912352;
   double* p;
   double a[2] = { 2.32, 6.55 };
   {
      double x = y;
      std::cout << std::scientific << std::setprecision(4)
                << x << std::endl;
      std::cout << ::x << std::endl;
      p = a;
   }
   p[0] = p[0] + 1.0;
   std::cout << a[0] << std::endl;
   p = p + 1;
   std::cout << p[0] << std::endl;
   std::cout << a[0] << std::endl;
   return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
5.9124e+000
1.2136e+000
3.3200e+000
6.5500e+000
3.3200e+000
```

PREGUNTA 3 (1 punto)

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

```
#include <iostream>

int main() {

   struct nodo {
      int num1;
      struct nodo* nI;
      struct nodo* nD;
   };

   int a[] = { 1, 2, 3, 4, 5 };
   int* p = a;
   struct nodo* nodo1;
   struct nodo* no1 = new nodo;
   struct nodo* no2 = new nodo;
   struct nodo* no3 = new nodo;
   struct nodo* no4 = new nodo;
   no1->num1 = *(p + 1);
   no2->num1 = *(p + 2);
   no3->num1 = *p;
   no4->num1 = a[4];
   no3->nI = no1;
   no3->nD = no2;
   no1->nI = no4;
   no1->nD = NULL;
   no4->nD = NULL;
   no4->nI = no2;
   no4->nD = NULL;
   no2->nI = NULL;
   no2->nD = NULL;
   nodo1 = no3;
   while (nodo1 != nullptr) {
      std::cout << nodo1->num1 << " " ;
      (nodo1->nI == nullptr) ?
         nodo1 = nodo1->nD : nodo1 = nodo1->nI;
   }
   return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
1 2 5 3
```

PREGUNTA 4 (2 puntos)

En un fichero de texto llamado *temperatura.txt* están almacenadas las medidas de la temperatura realizadas a lo largo de un día. Cada fila del fichero describe una medida: en la primera columna está escrito el instante de tiempo en que se ha realizado la medida y en la segunda columna está escrito el valor medido de la temperatura expresado en grados centígrados. Ambos valores son de tipo real.

En la primera columna del fichero, el tiempo está expresado en minutos. El intervalo de tiempo $[0, 60)$ minutos corresponde a la primera hora del día, el intervalo $[60, 120)$ minutos a la segunda y así sucesivamente. Dado que el día tiene 24 horas, la última hora del día corresponde al intervalo $[1380, 1440)$ minutos.

A continuación se muestra un ejemplo de las primeras líneas del fichero, donde puede observarse que las medidas no se encuentran ordenadas cronológicamente.

```
 123.12    3.7
  20.74    0.3
1406.02   -6.8
 370.80    8.9
 900.03   11.6
```

Escriba un programa en C++ que lea el contenido del fichero *temperatura.txt* y escriba en la consola, para cada una de las 24 horas del día:

- El número de medidas realizadas durante la hora.
- Si el número de medidas es mayor que cero, la temperatura media durante dicha hora.

El programa debe escribir en la consola las temperaturas medias en formato fijo, con dos dígitos decimales.

Al escribir el programa puede suponer que el fichero no tiene errores de formato y que no contiene medidas repetidas.

Solución a la Pregunta 4

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

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

int main() {
   std::ifstream inFich(nombreFich, std::ios::in);
   if (!inFich) {
      std::cerr << "Error al abrir el fichero\n";
      return 0;
   }
   std::vector<int> numMedidas(24,0);
   std::vector<double> sumaTemps(24,0);
   double tiempo, temperatura;
   while (inFich >> tiempo >> temperatura) {
      int hora = tiempo/60;
      numMedidas[hora]++;
      sumaTemps[hora] += temperatura;
   }
   inFich.close();
   std::cout << "Hora\tMedidas\tTemp";
   for (unsigned int i=0; i<24; i++) {
      std::cout << "\n" << (i+1) << "\t"  << numMedidas[i] << "\t";
      if (numMedidas[i])
         std::cout << std::fixed << std::setprecision(2)
                   << sumaTemps[i] / numMedidas[i];
   }
   return 0;
}
```

PREGUNTA 5 (3 puntos)

La estructura en C++ mostrada a continuación describe un evento, el cual se especifica mediante dos datos: un número entero denominado identificador (`ident`) que indica de qué tipo de evento se trata y un *string* que describe el instante de tiempo en que está planificado el evento (`time`). Dicho instante está descrito en formato HH:MM, donde HH representa la hora y MM el minuto.

```
struct Evento {
   int         ident;
   std::string time;
};
```

5.1 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
bool esTimeValido(std::string time);
```

que devuelva el valor *true* si el string pasado como argumento representa una hora válida en el formato HH:MM. Para ello, el string debe tener longitud 5, los dos dígitos HH deben ser 00, 01, ... 22 o 23, los dos dígitos MM deben ser 00, 01, ... 58 o 59, y entre los dígitos HH y MM debe encontrarse el carácter `':'`.

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

```
bool leeEv(std::list<Evento> &lEvs);
```

que primeramente muestre un mensaje en la consola solicitando al usuario que introduzca el identificador y el instante de un evento, y lea los datos introducidos por el usuario.

A continuación, empleando la función definida en el Apartado 5.1, debe comprobar si el instante introducido por el usuario es una hora válida en el formato HH:MM.

- En caso negativo, la función devuelve el valor *false*.
- En caso afirmativo, la función añade el nuevo evento al final de la lista pasada como argumento y devuelve el valor *true*.

5.3 (0.75 puntos) Escriba una función en C++ con el prototipo siguiente:

```
std::string evInminente(std::list<Evento> &lEvs,
                        int ident);
```

que devuelva el instante de tiempo del evento de la lista pasada como primer argumento cuyo identificador coincida con el segundo argumento de la función y cuyo instante de tiempo sea más temprano. La función debe además eliminar dicho evento de la lista.

Si no existe ningún elemento en la lista con ese identificador, la función debe devolver el string `"--:--"`.

5.4 (0.75 puntos) Escriba una función en C++ con el prototipo siguiente:

```
std::vector<std::string> extraeEvs(
                      std::list<Evento> &lEvs,
                      int ident,
                      std::string time_desde,
                      std::string time_hasta);
```

que devuelva un vector en el que se hayan copiado los instantes de los eventos de la lista pasada como primer argumento que cumplan simultáneamente todas estas condiciones:

- Su identificador coincide con el segundo argumento de la función.
- Su instante de tiempo está comprendido entre los instantes pasados como tercer y cuarto argumentos de la función, los cuales están en el formato HH:MM. Es decir, el instante del evento es igual o posterior a `time_desde`, y además es anterior o igual a `time_hasta`.

El vector devuelto debe estar ordenado cronológicamente.

Si varios elementos del vector tienen el mismo valor, deben eliminarse todos ellos excepto uno, de manera que el vector no contenga elementos repetidos.

5.5 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
bool escribeFichEvs(std::list<Evento> &lEvs,
                    std::string nombreFich);
```

que escriba (borrando el contenido previo) en un fichero de texto, cuyo nombre es pasado como segundo argumento a la función, los eventos de la lista pasada como primer argumento.

Los eventos deben escribirse en el fichero en el mismo orden en que aparecen en la lista. Cada evento debe escribirse en una línea del fichero: en la primera columna el identificador y en la segunda el instante de tiempo en el cual está planificado el evento. La función devuelve el valor *true* si la escritura del fichero se realiza correctamente y *false* en caso contrario (si se produce error al abrir el fichero para escritura).

Solución a la Pregunta 5

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

struct Evento {
   int ident;
   std::string time;
};


bool esTimeValido(std::string time) {
   int HH[2] = { time[0] - '0', time[1] - '0' };
   int MM[2] = { time[3] - '0', time[4] - '0' };
   return  time.size() == 5          &&
           HH[0] >= 0 && HH[0] <= 2 &&
           HH[1] >= 0 && HH[1] <= 9 &&
           10*HH[0] + HH[1] <= 23   &&
           time[2] == ':'           &&
           MM[0] >= 0 && MM[0] <= 5 &&
           MM[1] >= 0 && MM[1] <= 9 &&
           10*MM[0] + MM[1] <= 59;
}


bool leeEv(std::list<Evento> &lEvs) {
   Evento ev;
   std::cout << "Identificador del evento: ";
   std::cin  >> ev.ident;
   std::cout << "Instante del evento: ";
   std::cin  >> ev.time;
   if (esTimeValido(ev.time)) {
      lEvs.push_back(ev);
      return true;
   }
   return false;
}


std::string evInminente(std::list<Evento> &lEvs, int ident) {
   bool noEvento = true;
   std::string timeEv;
   std::list<Evento>::iterator p = lEvs.begin();
   while ( p != lEvs.end() ) {
      if (p->ident == ident && (noEvento || p->time < timeEv) ) {
         noEvento = false;
         timeEv = p->time;
      }
      p++;
```

```
52    }
53    if (noEvento)
54       return "--:--";
55    p = lEvs.begin();
56    while ( p != lEvs.end() ) {
57       if ( p->ident == ident && p->time == timeEv )
58          p = lEvs.erase(p);
59       else
60          p++;
61    }
62    return timeEv;
63 }
64
65
66 std::vector<std::string> extraeEvs(std::list<Evento> &lEvs,
67                                    int ident,
68                                    std::string time_desde,
69                                    std::string time_hasta) {
70    std::list<std::string> lTime;
71    std::list<Evento>::iterator p = lEvs.begin();
72    while (p != lEvs.end()) {
73       if (p->ident == ident     &&
74           p->time >= time_desde && p->time <= time_hasta)
75          lTime.push_back(p->time);
76       p++;
77    }
78    lTime.sort();
79    std::vector<std::string> vTime;
80    std::list<std::string>::iterator p1 = lTime.begin();
81    while (p1 != lTime.end()) {
82       if (vTime.empty() || (*p1) != vTime[vTime.size()-1])
83          vTime.push_back(*p1);
84       p1++;
85    }
86    return vTime;
87 }
88
89
90 bool escribeFichEvs(std::list<Evento> &lEvs,
91                     std::string nombreFich) {
92    std::ofstream outFich(nombreFich.c_str(),
93                          std::ios::out | std::ios::trunc);
94    if (!outFich)
95       return false;
96    std::list<Evento>::iterator p = lEvs.begin();
97    while ( p != lEvs.end() ) {
98       outFich << p->ident << " " << p->time << "\n";
99       p++;
100   }
101   outFich.close();
102   return true;
103 }
```