LENGUAJES DE PROGRAMACIÓN

Solución al examen de Septiembre 2023

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) El lenguaje JavaScript se puede emplear para crear documentos HTML dinámicos.

B. (0.5 puntos) El tipo de datos de una variable únicamente condiciona los valores que puede tomar la variable.

C. (0.5 puntos) En C, C++ y Java el límite inferior de todos los rangos de los índices del array es siempre 0.

D. (0.5 puntos) En C++, la sentencia

```
x = ++i;
```

es equivalente a ejecutar primero la sentencia

```
x = i;
```

y después la sentencia

```
i = i + 1;
```

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

F. (0.5 puntos) En el método de la burbuja, el número de intercambios es independiente de la secuencia de entrada.

Solución a la Pregunta 1

A	Verdadero	Véase la página 54 del texto base.
B	Falso	Véase la página 104 del texto base.
C	Verdadero	Véase la página 113 del texto base.
D	Falso	Véase la página 168 del texto base.
E	Verdadero	Véase la página 314 del texto base.
F	Falso	Véase la página 538 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 = 0.18765434;
   y = 9.03467684;
   double* p;
   double a[2] = { 2.989, 6.55 };
   {
      double x = y;
      std::cout << std::scientific << std::setprecision(4)
                << x << std::endl;
      std::cout << ::x << std::endl;
      p = a;
   }
   p = p + 1;
   std::cout << a[0] << std::endl;
   std::cout << *p   << std::endl;
   *p = 7.77777;
   std::cout << a[1] << std::endl;
   return 0;
}
```

Solución a la Pregunta 2

La salida por consola producida al ejecutar el programa es:

```
9.0347e+000
1.8765e-001
2.9890e+000
6.5500e+000
7.7778e+000
```

PREGUNTA 3 (1 punto)

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

```
#include <iostream>
#include <vector>
#include <algorithm>

int main() {
   std::vector<double> v1, v2;
   for (double x = 10; x >= 0; x -= 2)
      v1.push_back(x);
   std::vector<double>::iterator p1 =
      std::find_if(v1.begin(), v1.end(),
         [](double x) -> bool
         { return x > 3 && x < 8; });
   while( p1 != v1.end() ){
      std::cout << *p1 << " ";
      p1++;
   }
   std::cout << std::endl;
   for (double x = 0; x <= 10; x += 2)
      v2.push_back(x);
   std::vector<double>::iterator p2 =
      std::find_if(v2.begin(), v2.end(),
         [](double x) -> bool
         { return x > 3 && x < 8; });
   while ( p2 != v2.end() ) {
      std::cout << *p2 << " ";
      p2++;
   }
   std::cout << std::endl;
   return 0;
}
```

Solución a la Pregunta 3

La salida por consola producida al ejecutar el programa es:

```
6 4 2 0
4 6 8 10
```

PREGUNTA 4 (2.5 puntos)

Un procedimiento de codificación de mensajes hace uso de dos ficheros. El primer fichero contiene una única línea de texto, compuesta por letras, números y espacios en blanco. El segundo fichero contiene una columna de números enteros positivos, que indican la separación entre los caracteres del texto del primer fichero que conforman el mensaje, partiendo desde el principio. Veamos un ejemplo.

El contenido del primer fichero es:

```
Otras aves, algunas al borde de la extincion en estos lares
```

El contenido del segundo fichero es:

```
4
4
6
1
23
18
```

Las letras de la línea de texto del primer fichero señaladas por el segundo fichero son las que se muestran subrayadas:

Otras aves, algunas al borde de la extincion en estos lares

Obteniéndose como resultado: `seguir`

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

1. Abrir el primer fichero de texto, llamado *texto.txt*, para lectura. Si se produce error, indicarlo mediante un mensaje escrito en la consola y terminar.
2. Leer la línea de texto del fichero, almacenándola en un string llamado `sTexto`.
3. Escribir el string en la consola.
4. Abrir el segundo fichero de texto, llamado *codigo.txt*, para lectura. Si se produce error, indicarlo mediante un mensaje escrito en la consola y terminar.
5. Almacenar el contenido del fichero en una lista de elementos de tipo entero llamada `lPos`.

6. Realizar la búsqueda de los caracteres del mensaje, empleando el string `sTexto` y la lista `lPos`, y escribir el mensaje en la consola.

7. Terminar.

Solución a la Pregunta 4

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

int main() {
   // Apertura y lectura del primer fichero
   std::ifstream inFich1("texto.txt", std::ios::in);
   if (!inFich1) {
      std::cerr << "Error al abrir el primer fichero";
      return 0;
   }
   std::string sTexto;
   getline(inFich1, sTexto);
   inFich1.close();
   // Escritura en consola del contenido del primer fichero
   std::cout << sTexto << std::endl;
   // Apertura y lectura del segundo fichero
   std::ifstream inFich2("codigo.txt", std::ios::in);
   if (!inFich2) {
      std::cerr << "Error al abrir el segundo fichero";
      return 0;
   }
   std::list<int> lPos;
   int n;
   while ( inFich2 >> n )
      lPos.push_back(n);
   inFich2.close();
   // Obtencion y escritura en consola del mensaje
   std::list<int>::iterator p = lPos.begin();
   int pos = 0;
   while ( p != lPos.end() ) {
      pos += *p;
      std::cout << sTexto[pos];
      p++;
   }
   std::cout << std::endl;
   return 0;
}
```

PREGUNTA 5 (2.5 puntos)

En un fichero de texto llamado *puntos.txt* se encuentran almacenadas las coordenadas cartesianas de parejas de puntos en el plano. El fichero contiene cuatro columnas de números reales. Cada pareja de puntos está definida en una línea del fichero. Las dos primeras columnas contienen las coordenadas X e Y del primer punto de la pareja, que llamaremos $A = (A_x, A_y)$, y las dos últimas columnas las coordenadas X e Y del segundo punto, que llamaremos $B = (B_x, B_y)$.

Se trata de determinar si cada una de estas parejas de puntos define una recta, y en caso afirmativo escribir en la consola la pendiente de dicha recta.

A y B definen una recta siempre que no sean el mismo punto, es decir, que no tengan la misma coordenada X ($A_x = B_x$) y la misma coordenada Y ($A_y = B_y$).

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

```
bool leeFich( std::string nombreFichero,
              std::vector<Punto> &vA,
              std::vector<Punto> &vB  );
```

que realice las acciones siguientes:

1. Eliminar el contenido de los dos vectores `vA` y `vB` pasados como argumento.
2. Abrir para lectura el fichero de texto cuyo nombre es pasado como primer argumento. Si se produce error, terminar, devolviendo el valor *false*.
3. Almacenar en el vector `vA` las coordenadas de los puntos A del fichero y en el vector `vB` las coordenadas de los puntos B.
4. Terminar, devolviendo el valor *true*.

La estructura `Punto` está definida de la forma siguiente:

```
struct Punto {
   double x, y;
};
```

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

```
bool mismoPunto( Punto *pA, Punto &pB );
```

que devuelva *true* si los dos puntos pasados como argumento tienen el mismo valor de la coordenada X y el mismo valor de la coordenada Y, y que devuelva *false* en caso contrario.

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

1. Invocando la función `leeFich`, abrir el fichero de texto llamado *puntos.txt* para lectura, pasando como argumento los vectores `vPuntosA` y `vPuntosB`.

   Si la función devuelve *false*, indicar mediante un mensaje escrito en la consola que se ha producido error al abrir el fichero y terminar.

2. Examinar los vectores `vPuntosA` y `vPuntosB` para determinar en cuántas parejas el punto A y el punto B son el mismo punto, empleando para ello la función `mismoPunto`.

   Escribir en la consola el número de parejas en las cuales A y B son el mismo punto, y el número de parejas en las cuales A y B son puntos diferentes.

3. Para cada una de las parejas de puntos A y B que definen una recta en el plano, calcular y escribir en la consola la pendiente de dicha recta,

$$\frac{B_y - A_y}{B_x - A_x}$$

   en formato científico con 8 dígitos de precisión.

4. Terminar.

Solución a la Pregunta 5

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

struct Punto {
   double x, y;
};

bool leeFich( std::string nombreFichero,
              std::vector<Punto> &vA,
              std::vector<Punto> &vB  ) {
   // Elimina el contenido de los dos vectores argumento
   vA.clear();
   vB.clear();
   // Apertura del fichero
   std::ifstream inFich(nombreFichero, std::ios::in);
   if (!inFich)
      return false;
   Punto pA, pB;
   while ( inFich >> pA.x >> pA.y >> pB.x >> pB.y ) {
      vA.push_back(pA);
      vB.push_back(pB);
   }
   inFich.close();
   return true;
}

bool mismoPunto( Punto *pA, Punto &pB ) {
   return pA->x == pB.x && pA->y == pB.y;
}

int main() {
   std::vector<Punto> vPuntosA, vPuntosB;
   if ( !leeFich("puntos.txt", vPuntosA, vPuntosB) ) {
      std::cerr << "Error en la apertura del fichero";
      return 0;
   }
   int numMismoPunto = 0;
   for (unsigned int i=0; i<vPuntosA.size(); i++)
      if ( mismoPunto(&(vPuntosA[i]), vPuntosB[i]) )
         numMismoPunto++;
   std::cout << "Parejas de puntos iguales: "
             << numMismoPunto
             << "\nParejas de puntos diferentes: "
             << vPuntosA.size() - numMismoPunto
             << std::endl;
   for (unsigned int i=0; i<vPuntosA.size(); i++)
      if ( !mismoPunto(&(vPuntosA[i]), vPuntosB[i]) ) {
         double pendiente = ( vPuntosB[i].y - vPuntosA[i].y ) /
                            ( vPuntosB[i].x - vPuntosA[i].x );
         std::cout << std::scientific << std::setprecision(8)
                   << pendiente << std::endl;
      }
   return 0;
}
```