LENGUAJES DE PROGRAMACIÓN

Examen de convocatoria Junio 2020, en el Aula virtual de Examen UNED (AvEx)

INSTRUCCIONES

- El examen debe realizarse de manera individual.
- No está permitido el uso de ningún material.
- El examen se compone de dos preguntas de tipo test y cuatro preguntas de desarrollo.
- Cada pregunta tipo test bien contestada se valorará con un punto. Cada pregunta tipo test mal contestada restará 0.25 puntos. Cada pregunta tipo test no contestada se calificará con cero puntos.
- Cada pregunta de desarrollo se valorará sobre dos puntos.
- Para aprobar el examen debe obtener una puntuación igual a superior a 5 puntos.
- Si le surgen dudas sobre la funcionalidad del código pedido en algún ejercicio, puede tomar las decisiones que usted considere oportunas, siempre que no estén en contradicción con las especificaciones dadas en el enunciado del ejercicio.
- Dispone de 1 hora para realizar el examen. Recomendación: dedique aproximadamente 4 minutos a contestar cada pregunta de tipo test y 12 minutos a contestar cada pregunta de desarrollo.

Aclaración: la aplicación web UNEDenlínea construye el examen de cada alumno seleccionando aleatoriamente una pregunta de cada uno de los bloques.

Test Bloque 1

Test 1.1

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
int n = 10, *p1, *p2;
p1 = &n;
std::cout << *p1 << " " << n << " ";
*p1 = 5;
std::cout << *p1 << " " << n << " ";
p2 = p1;
*p2 = 3;
std::cout << *p1 << " " << *p2 << std::endl;
```

Respuesta

A `0 10 5 5 3 0`

B `0 10 5 5 5 3`

C `10 10 5 5 3 3` ✓

D Ninguna de las anteriores

Test 1.2

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
int A[4] = {10, 20, 30, 40};
int *p1, *p2;
p1 = A;
std::cout << *p1 << " ";
p2 = p1++;
std::cout << *p1 << " " << *p2 << " ";
p2 = p1;
*p2 = A[2];
std::cout << *p1 << " " << *p2 << " ";
```

Respuesta

A `10 20 10 30 30` ✓

B `10 10 11 30 30`

C `10 11 11 10 30`

D Ninguna de las anteriores

Test 1.3

A continuación se muestra un programa en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
#include <iostream>

int y = 3;

int main() {
    int y = 10, z = 0, *pA, *pB;
    pA = &y;
    std::cout << y << " " << *pA << " ";
    {
       int y = 2;
       pB = &y;
       ::y = 5;
       std::cout << *pA << " " << *pB << " ";
       z = y;
       std::cout << z  << " " << ::y << " ";
    }
    std::cout << y << std::endl;
    return 0;
}
```

Respuesta

A `3 3 3 2 2 5 3`

B `10 10 10 2 2 5 10` ✓

C `10 10 10 10 10 10 10`

D Ninguna de las anteriores

Test 1.4

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
int* p;
int V[2] = { 0, 1 };
int m = V[0];
p = &V[0];
std::cout << p[0] << " " << p[1] << " ";
*p = 10;
p++;
m = *p;
std::cout << V[0] << " " << m << std::endl;
```

Respuesta

A `0 0 10 1`

B `0 0 0 11`

C `0 1 10 1` ✓

D Ninguna de las anteriores

Test Bloque 2

Test 2.1

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
std::list<int> La;
for (int i=0; i<10; i++)
   La.push_back(i);
std::list<int>::iterator pA = La.begin();
pA++;
pA++;
std::cout << *pA << " ";
std::list<int>::iterator pB = La.end();
pB--;
std::cout << *pB << " ";
std::list<int> Lb(pA,pB);
Lb.pop_front();
Lb.pop_front();
std::cout << *(Lb.begin()) << std::endl;
```

Respuesta

A	`2 9 4`	✓
B	`3 8 4`	
C	`3 9 3`	
D	Ninguna de las anteriores	

Test 2.2

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
std::vector<double> A(4,5);
std::vector<double>::iterator p;
p = A.begin()+1;
*p = 1;
*(p+1) = 2;
A.at(0) = 10;
*(p+2) = -1*(*p);
A.at(1) = 4;
std::cout << *p     << " "
          << *(p+1) << " "
          << *(p+2) << std::endl;
```

Respuesta

A `10 -4 4`

B `4 4 -10`

C `4 2 -1` ✓

D Ninguna de las anteriores

Test 2.3

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
std::queue<int> Q;
for (int i=0; i<4; i++) {
   Q.push(i);
   int &p = Q.back();
   std::cout << p << " ";
}
Q.push(100);
std::cout << Q.size() << " ";
while (!Q.empty()) {
   int &p = Q.front();
   std::cout << p << " ";
   Q.pop();
}
std::cout << std::endl;
```

Respuesta

A `0 1 2 3 4 0 1 2 3 100`

B `0 1 2 3 5 0 1 2 3 100` ✓

C `0 1 2 3 100 1 2 3 100 100`

D Ninguna de las anteriores

Test 2.4

A continuación se muestra un fragmento de código de un programa principal en C++ que no contiene errores. Señale la salida por consola producida al ejecutarlo.

```
std::map<std::string, int> M;
M["D"] = 40; M["A"] = 10;
M["C"] = 30; M["B"] = 20;
std::map<std::string, int>::iterator p = M.find("B");
while ( p!=M.end() ) {
  std::cout << (*p).first << " " << (*p).second << " ";
  p++;
}
M.erase("B");
p = M.begin();
while(p != M.end()){
   std::cout << (*p).first << " " << (*p).second << " ";
   p++;
}
std::cout << std::endl;
```

Respuesta

A `B 20 D 40 A 10 C 30`

B `B 20 A 10 C 30 D 40`

C `B 20 C 30 D 40 A 10 C 30 D 40` ✓

D Ninguna de las anteriores

Bloque Desarrollo 1

Pregunta 1.1

Escriba un programa en C++ que calcule y muestre en la consola el resultado de la serie

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + \frac{(-1)^n \cdot x^{2 \cdot n+1}}{(2 \cdot n+1)!}$$

para cada uno de los siguientes valores de x:

$$x_i = \frac{\pi \cdot i}{100} \quad \text{con } i = 0, 1, 2, \ldots, 100$$

donde π es el número pi. Debe declararse n como una constante global y asignarle el valor 15.

Respuesta

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

const int n = 15;
const double pi = std::acos(-1);

int main() {
   for (int i=0; i<=100; i++) {
      double x = pi*i/100;
      double serie = x;
      double terminoAnterior = x;
      for (int k=1; k<=n; k++) {
         double nuevoTermino = -terminoAnterior*x*x/((2*k+1)*2*k);
         serie += nuevoTermino;
         terminoAnterior = nuevoTermino;
      }
      std::cout << "x = " << x << "\tSerie = " << serie << std::endl;
   }
   return 0;
}
```

Pregunta 1.2

Escriba un programa en C++ que calcule y muestre en la consola el resultado de la serie

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n)!}$$

para cada uno de los siguientes valores de *x*:

$$x_i = \frac{\pi \cdot i}{100} \quad \text{con } i = 0, 1, 2, \ldots, 100$$

donde π es el número pi. Debe declararse *n* como una constante global y asignarle el valor 15.

Respuesta

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

const int n = 15;
const double pi = std::acos(-1);

int main() {
   for (int i=0; i<=100; i++) {
      double x = pi*i/100;
      double serie = 1;
      double terminoAnterior = 1;
      for (int k=1; k<=n; k++) {
         double nuevoTermino = -terminoAnterior*x*x/(2*k*(2*k-1));
         serie += nuevoTermino;
         terminoAnterior = nuevoTermino;
      }
      std::cout << "x = " << x << "\tSerie = " << serie << std::endl;
   }
   return 0;
}
```

Pregunta 1.3

Escriba un programa en C++ que calcule y muestre en la consola en formato científico, con 10 dígitos de precisión, los sucesivos valores $x_1, x_2, \ldots, x_N$ calculados mediante la fórmula

$$x_{k+1} = x_k - \frac{x_k^2 - 2}{2 \cdot x_k} \quad \text{con} \;\; k = 0, 1, 2, \ldots, N-1$$

donde x_0 es un valor introducido por el usuario a través de la consola. Debe declararse N como una constante global y asignarle el valor 10.

Si para algún valor de k se satisface $x_k = 0$, el programa debe interrumpir el cálculo y terminar.

Respuesta

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

const int N = 10;

int main() {
    std::cout << "Introduzca x_0 ";
    double x_k;
    std::cin >> x_k;
    for (int k = 0; k < N; k++) {
        if ( x_k == 0 ) {
            std::cout << "Denominador cero" << std::endl;
            return 0;
        }
        x_k = x_k - (x_k*x_k - 2) / (2*x_k);
        std::cout << std::scientific  << std::setprecision(10)
                  << k << "\t" << x_k << std::endl;
    }
    return 0;
}
```

Pregunta 1.4

Escriba un programa en C++ que calcule y muestre en la consola en formato científico, con 10 dígitos de precisión, los sucesivos valores x_2, x_3, ..., x_N calculados mediante la fórmula

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

donde x_0 y x_1 son dos valores introducidos por el usuario a través de la consola, y donde la función $f(x)$ es:

$$f(x) = x - \cos(x)$$

Debe declararse N como una constante global y asignarle el valor 10.

Si para algún valor de k se satisface $f(x_k) - f(x_{k-1}) = 0$, el programa debe interrumpir el cálculo y terminar.

Respuesta

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

const int N = 10;

double f(double x) {
    return x - std::cos(x);
}

int main() {
    double x_km1, x_k;
    std::cout << "x0: ";
    std::cin >> x_km1;
    std::cout << "x1: ";
    std::cin >> x_k;
    for (int k = 1; k < N; k++) {
        double dif_f = f(x_k) - f(x_km1);
        if ( dif_f == 0 ) {
            std::cout << "Denominador cero" << std::endl;
            return 0;
        }
        double x_kM1 = x_k - (x_k - x_km1)*f(x_k)/dif_f;
        std::cout << std::scientific << std::setprecision(10)
            << "k = " << k << "\tx_{k+1} = " << x_kM1 << std::endl;
        x_km1 = x_k;
        x_k = x_kM1;
    }
    return 0;
}
```

Bloque Desarrollo 2

Pregunta 2.1

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

1. Declarar un array bidimensional de **double** llamado `M`, con tamaño 4×4.
2. Solicitar al usuario que introduzca por consola 16 valores reales y almacenar los valores en el array. Los primeros cuatro valores deben almacenarse en la fila cero del array, los cuatro siguientes en la fila uno y así sucesivamente.
3. Calcular la media aritmética de los 16 valores.
4. Escribir en la consola la media y los valores del array que sean mayores que la media.
5. Terminar.

Respuesta

```
#include <iostream>

int main() {
    double M[4][4];
    std::cout << "Introduzca datos: ";
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            std::cin >> M[i][j];
    double media = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            media += M[i][j];
    media /= 16;
    std::cout << "Media = " << media << std::endl;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            if (M[i][j] > media)
                std::cout << M[i][j] << " ";
    std::cout << std::endl;
    return 0;
}
```

Pregunta 2.2

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

1. Declarar un array bidimensional de **double** llamado `ABC`, con tamaño 3×2. En dicho array se almacenarán las coordenadas cartesianas X e Y de tres puntos en el plano, que introducirá el usuario. La coordenada X se almacena en la columna cero y la coordenada Y en la columna uno.

2. Solicitar al usuario que introduzca por consola seis valores reales y almacenar los valores en el array. Los primeros dos valores deben almacenarse en la fila cero del array, los dos siguientes en la fila uno y así sucesivamente.

3. Calcular si los tres puntos se encuentran sobre la misma recta. Tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ están alineados si se satisface:

$$\frac{x_2 - x_1}{x_3 - x_2} = \frac{y_2 - y_1}{y_3 - y_2}$$

4. Escribir un mensaje en la consola indicando si los tres puntos se encuentran sobre la misma recta o no.

5. Terminar.

Respuesta

```
#include <iostream>

int main() {
    double ABC[3][2];
    std::cout << "Coordenadas del punto 1:\n";
    std::cin >> ABC[0][0] >> ABC[0][1];
    std::cout << "Coordenadas del punto 2:\n";
    std::cin >> ABC[1][0] >> ABC[1][1];
    std::cout << "Coordenadas del punto 3:\n";
    std::cin >> ABC[2][0] >> ABC[2][1];
    if ( (ABC[1][0]-ABC[0][0])*(ABC[2][1]-ABC[1][1]) -
            (ABC[2][0]-ABC[1][0])*(ABC[1][1]-ABC[0][1]) ) {
        std::cout << "Puntos no alineados" << std::endl;
    } else {
        std::cout << "Puntos alineados" << std::endl;
    }
    return 0;
}
```

Pregunta 2.3

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

1. Declarar tres arrays unidimensionales de **double** llamados `A`, `B` y `C`, de 2 elementos cada uno. En cada array se almacenarán las coordenadas cartesianas X e Y de un punto en el plano, que introducirá el usuario. La coordenada X se almacena en el primer elemento y la coordenada Y en el segundo.
2. Solicitar al usuario que introduzca por consola las coordenadas de los tres puntos y almacenar los valores en los arrays.
3. Calcular si los tres puntos se encuentran sobre la misma recta. Tres puntos $A(x_1,y_1)$, $B(x_2,y_2)$ y $C(x_3,y_3)$ están alineados si se satisface:

$$\frac{x_2-x_1}{x_3-x_2}=\frac{y_2-y_1}{y_3-y_2}$$

4. Escribir un mensaje en la consola indicando si los tres puntos se encuentran sobre la misma recta o no.
5. Terminar.

Respuesta

```
#include <iostream>

int main() {
    double A[2], B[2], C[2];
    std::cout << "Coordenadas del punto A:\n";
    std::cin >> A[0] >> A[1];
    std::cout << "Coordenadas del punto B:\n";
    std::cin >> B[0] >> B[1];
    std::cout << "Coordenadas del punto C:\n";
    std::cin >> C[0] >> C[1];
    if ( (B[0]-A[0])*(C[1]-B[1]) - (C[0]-B[0])*(B[1]-A[1]) ) {
        std::cout << "Puntos no alineados" << std::endl;
    } else {
        std::cout << "Puntos alineados" << std::endl;
    }
    return 0;
}
```

Pregunta 2.4

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

1. Declarar un array bidimensional de **double** llamado `M`, con tamaño 4×4.
2. Solicitar al usuario que introduzca por consola 16 valores reales y almacenar los valores en el array. Los primeros cuatro valores deben almacenarse en la fila cero del array, los cuatro siguientes en la fila uno y así sucesivamente.
3. Calcular la media aritmética de los 16 valores.
4. Escribir en la consola la media y los valores del array que sean menores que la media.
5. Terminar.

Respuesta

```
#include <iostream>

int main() {
    double M[4][4];
    std::cout << "Introduzca datos: ";
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            std::cin >> M[i][j];
    double media = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            media += M[i][j];
    media /= 16;
    std::cout << "Media = " << media << std::endl;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            if (M[i][j] < media)
                std::cout << M[i][j] << " ";
    std::cout << std::endl;
    return 0;
}
```

Bloque Desarrollo 3

Pregunta 3.1

Escriba una función en C++ que calcule la media aritmética de los elementos de la lista que se pasa como argumento. Si el número de elementos de la lista es cero, la función lanza una excepción. El prototipo de la función es:

```
double mVals(std::list<double> &listaVals)
   throw (std::domain_error);
```

Respuesta

```
#include <iostream>
#include <list>
#include <stdexcept>

double mVals(std::list<double> &listaVals) throw (std::domain_error) {
    if (!listaVals.size())
        throw std::domain_error("ERROR: Lista sin elementos");
    double suma = 0;
    for (std::list<double>::iterator p = listaVals.begin();
            p != listaVals.end(); p++)
        suma += (*p);
    return suma / listaVals.size();
}
```

Pregunta 3.2

Escriba una función en C++ que calcule las medias móviles, con una ventana de 3 valores, de los elementos del vector que se pasa como argumento. El prototipo de la función es:

```
std::vector<double> mm(std::vector<double> &v)
                        throw (std::invalid_argument);
```

Si el vector pasado como argumento tiene N elementos, v_1, v_2, …, v_N, el vector devuelto por la función tendrá $N-2$ elementos, w_1, …, w_{N-2}, calculados de la forma:

$$w_i = \frac{v_i + v_{i+1} + v_{i+2}}{3} \quad \text{para} \quad i = 1, \ldots, N-2$$

Si el número de elementos del vector pasado como argumento es menor que tres, entonces la función debe lanzar una excepción.

Respuesta

```
std::vector<double> mm(std::vector<double> &v) throw (std::invalid_argument) {
    if (v.size()<3)
        throw std::invalid_argument("Menos de 3 elementos");
    std::vector<double> mediaMovil;
    for (int i=0; i<v.size()-2; i++)
        mediaMovil.push_back((v[i]+v[i+1]+v[i+2])/3);
    return mediaMovil;
}
```

Pregunta 3.3

Escriba una función en C++ que calcule el número de elementos del vector pasado como argumento que están fuera del intervalo $[a, b]$. Los extremos a y b del intervalo son argumentos de la función. El prototipo de la función es:

```
int nInterv(std::vector<double> &v,
            double a, double b)
                throw (std::invalid_argument);
```

Si el número de elementos del vector pasado como argumento es cero, entonces la función debe devolver el valor -1. Si no se satisface $b > a$, la función debe lanzar una excepción.

Respuesta

```
#include <iostream>
#include <vector>
#include <stdexcept>

int nInterv(std::vector<double> &v, double a, double b)
                                    throw (std::invalid_argument) {
    if ( !(b > a) )
        throw std::invalid_argument("Intervalo mal definido");
    if (v.size() == 0) return -1;
    int num = 0;
    for (unsigned int i=0; i<v.size(); i++)
        if (v[i]<a || v[i]>b) num++;
    return num;
}
```

Pregunta 3.4

Escriba una función en C++ que calcule el número de elementos del vector pasado como argumento que están dentro del intervalo $[a, b]$. Los extremos a y b del intervalo son argumentos de la función. El prototipo de la función es:

```
int nInterv(std::vector<double> &v,
            double a, double b)
                throw (std::invalid_argument);
```

Si el número de elementos del vector pasado como argumento es cero, entonces la función debe devolver el valor -1. Si no se satisface $b > a$, la función debe lanzar una excepción.

Respuesta

```
#include <iostream>
#include <vector>
#include <stdexcept>

int nInterv(std::vector<double> &v, double a, double b)
                                    throw (std::invalid_argument) {
    if ( !(b > a) )
        throw std::invalid_argument("Intervalo mal definido");
    if (v.size() == 0) return -1;
    int num = 0;
    for (unsigned int i=0; i<v.size(); i++)
        if (v[i]>=a && v[i]<=b) num++;
    return num;
}
```

Bloque Desarrollo 4

Pregunta 4.1

Escriba una función en C++ que convierta a decimal un número hexadecimal sin signo de 4 u 8 dígitos, que es pasado como argumento a la función en forma de string. La función debe primeramente comprobar que el argumento es válido, lanzando una excepción si no lo es. Para que el argumento sea válido debe ser un string de longitud 4 u 8, tal que cada uno de sus elementos pertenezca al conjunto $\{0,1,\ldots,9,A,B,C,D,E,F\}$. Si el argumento es válido, la función debe calcular el valor decimal del número hexadecimal pasado como argumento y devolverlo. El prototipo de la función es:

```
unsigned long int hex2dec(std::string numHex)
                        throw (std::invalid_argument);
```

Respuesta

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

unsigned long int hex2dec(std::string numHex) throw (std::invalid_argument) {
    if (numHex.length() != 4 && numHex.length() != 8 )
        throw std::invalid_argument("Numero de digitos no valido");
    for (unsigned int i=0; i<numHex.size(); i++) {
        bool valido = (numHex[i] >=0 && numHex[i] <= 9 ) ||
                (numHex[i] >= 'A' || numHex[i] <='F');
        if ( !valido ) throw std::invalid_argument("Digito no valido");
    }
    unsigned long int numDec = 0, peso = 1;
    for (int i = numHex.size() - 1; i >= 0; i--) {
        if (numHex[i] >= '0' && numHex[i] <= '9')  {
            numDec += (numHex[i] - '0') * peso;
        } else {
            numDec += (10 + numHex[i] - 'A')  * peso;
        }
        peso *= 16;
    }
    return numDec;
}
```

Pregunta 4.2

Escriba una función en C++ con el prototipo siguiente

```
std::map<std::string,int> cuentaPalabras(std::string frase);
```

El string pasado como argumento contiene una o más palabras escritas en minúscula, separadas por uno o más espacios en blanco. La función devuelve un mapa con las diferentes palabras que hay en la frase pasada como argumento: la clave del mapa es la palabra y el valor el número de veces que aparece la palabra en la frase.

Respuesta

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

std::map<std::string, int> cuentaPalabras(std::string frase) {
    std::map<std::string, int> mPalabras;
    int desde = -1;
    for (unsigned int i = 0; i < frase.size(); i++) {
        if (desde == -1 && frase[i] != ' ') {
            desde = i;
        } else if (desde != -1 && frase[i] == ' ') {
            std::string palabra = frase.substr(desde, i - desde);
            if (mPalabras.find(palabra) == mPalabras.end()) {
                mPalabras[palabra] = 1;
            } else {
                mPalabras.find(palabra)->second++;
            }
            desde = -1;
        }
    }
    if (desde != -1) {
        std::string palabra = frase.substr(desde);
        if (mPalabras.find(palabra) == mPalabras.end()) {
            mPalabras[palabra] = 1;
        } else {
            mPalabras.find(palabra)->second++;
        }
    }
    return mPalabras;
}
```

Pregunta 4.3

Este ejercicio tiene dos partes.

Primeramente, escriba una función en C++ con el prototipo siguiente

```
std::list<int> unsignedInt2Lista(unsigned int num);
```

que devuelva una lista cuyos componentes sean los dígitos del número entero pasado como argumento. Los dígitos deben almacenarse en la lista en el mismo orden en que aparecen en el número pasado como argumento, siendo el primer elemento de la lista el dígito más significativo del número.

A continuación, escriba un programa principal en C++ que realice las acciones siguientes: solicitar al usuario que introduzca un valor entero por consola, leer el valor entero introducido por el usuario, pasárselo como argumento a la función unsignedInt2Lista, escribir en la consola los elementos de la lista devuelta por la función y terminar.

Respuesta

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

std::list<int> unsignedInt2Lista(unsigned int num) {
    std::stringstream ss;
    ss << num;
    std::string sNum = ss.str();
    std::list<int> lnum;
    for (unsigned int i = 0; i < sNum.size(); i++)
        lnum.push_back(sNum[i] - '0');
    return lnum;
}


int main() {
    std::cout << "Introduzca numero" << std::endl;
    unsigned int num;
    std::cin >> num;
    std::list<int> lDigitos = unsignedInt2Lista(num);
    std::list<int>::iterator p = lDigitos.begin();
    while (p != lDigitos.end()) {
        std::cout << *p << " ";
        p++;
    }
    return 0;
}
```

Pregunta 4.4

Este ejercicio tiene dos partes.

Primeramente, programe una función en C++ que, dados dos intervalos cerrados de la recta real $I_1 = [a_1, b_1]$ y $I_2 = [a_2, b_2]$, que son pasados como argumento, devuelva el valor true si no existe solapamiento entre ellos y el valor false si los intervalos solapan. El prototipo de la función es:

```
bool noSolapan(Intervalo I1, Intervalo I2)
                        throw (std::invalid_argument);
```

La estructura

```
struct Intervalo {
    double a, b;
};
```

especifica un intervalo, donde a y b son el extremo inferior y superior del mismo, debiéndose por ello satisfacer que $a < b$. Si para alguno de los dos intervalos pasados como argumento no se satisface esta condición, la función lanza una excepción.

A continuación, programe otra función en C++ que tenga la misma funcionalidad que la anterior, pero cuyo prototipo sea el siguiente:

```
bool noSolapan(Intervalo *I1, Intervalo *I2)
                        throw (std::invalid_argument);
```

Respuesta

```
#include <iostream>
#include <stdexcept>
#include <algorithm>

struct Intervalo {
    double a, b;
};

bool noSolapan(Intervalo I1, Intervalo I2) throw (std::invalid_argument) {
    if (I1.a >= I1.b || I2.a >= I2.b)
        throw std::invalid_argument("Intervalo no valido");
    return !(std::max(I1.a,I2.a) <= std::min(I1.b,I2.b));
}

bool noSolapan(Intervalo *I1, Intervalo *I2) throw (std::invalid_argument) {
    if (I1->a >= I1->b || I2->a >= I2->b)
        throw std::invalid_argument("Intervalo no valido");
    return !(std::max(I1->a,I2->a) <= std::min(I1->b,I2->b));
}
```