LENGUAJES DE PROGRAMACIÓN

Examen de convocatoria Junio 2021, 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 *n1, *n2, *n3;
int a = 2, b = 10, c = 20;
n1 = &a;
n2 = n1;
a = 100;
n1 = new int;
*n1 = b;
n3 = n1;
*n1 = 200;
std::cout << *n1 << " " << *n2 << " " << *n3 << " ";
std::cout << a << " " << b << " " << c << std::endl;
```

Respuesta

A `200 100 200 100 10 20` ✓

B `2 100 0 2 10 20`

C `2 100 200 100 10 20`

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.

```
char *k1, *k2, *k3;
char a = 'a', b = 'b', c = 'c';
k2 = &a;
a = 'z';
b = c;
k3 = &c;
k3 = k2;
k1 = k2;
k3 = new char;
*k3 = 'm';
std::cout << *k1 << " " << *k2 << " " << *k3 << " ";
std::cout << a << " " << b << " " << c << std::endl;
```

Respuesta

A `a a m z c c`

B `z z m z c c` ✓

C `a a m z b c`

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.

```
double *p1, *p2;
double a[] = { -1, -2, -3 };
double b[] = { 0, 10, 20, 30 };
double x = 0.0, y = 2.0, z = 2.0;
p1 = &b[1];
p2 = &a[0];
for (int i = 0; i < 2; i++) {
    x += i * z;
    a[i] = x + b[i];
    std::cout << a[i] << " " << *(p1 + i) << " ";
}
*p2 = *p1;
*(p2 + 1) = y;
std::cout << p2[0] << " " << p2[1] << " " << p2[2] << std::endl;
```

Respuesta

A	`0 10 12 20 10 2 -3`	✓
B	`0 10 0 10 10 2 30`	
C	`0 10 0 10 10 2 -3`	
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.

```
double *p1, *p2;
double a[] = { -10, -20, -30, -40};
double b[] = { 1, 2, 3, 4 };
double x = 0.0, z = 3.0;
p1 = &b[0];
p2 = a;
for (int i = 0; i < 2; i++) {
   x += i * z;
   a[i] = x + b[i];
   std::cout << a[i] << " " << *(p1 + i) << " ";
}
p2 = p1;
p2++;
std::cout << p2[0] << " " << p2[1] << " " << p2[2] << std::endl;
```

Respuesta

A `1 1 5 2 2 3 4` ✓

B `1 1 5 2 5 2 3`

C `1 1 5 2 1 2 3`

D Ninguna de las anteriores

Test 1.5

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.

```
double x = 1.1234, y = 2.2345, *p;
double a[3] = {-1.1, -2.2, -3.3};
p = &x;
x = y;
std::cout << std::scientific << std::setprecision(2) << x << "  ";
std::cout << *p << " ";
p = a;
std::cout << *p << " ";
p++;
*p = y*2;
std::cout << a[0] << " " << a[1] << " " << a[2] << std::endl;
```

Respuesta

A `2.23e+00  2.23e+00 -1.10e+00 -1.10e+00 4.47e+00 -3.30e+00` ✓

B `2.23e+00  1.12e+00 -1.10e+00 -1.10e+00 -2.20e+00 -3.30e+00`

C `2.230  1.120 -1.100 -1.100 -2.200 -3.300`

D Ninguna de las anteriores

Test 1.6

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.

```
double *p1, *p2;
double a[] = { 1, 2, 3, 4 };
double b[] = { -1, -2, -3, -4 };
double x = 10, y = 20;
p1 = &b[1];
*p1 = x;
p2 = &a[1];
a[2] = *p1;
for (int i = 0; i < 3; i++)
    std::cout << a[i] << " " << *(p1 + i) << " ";
*p2 = *(p1+1);
*(++p2) = y;
for (int i = 0; i < 3; i++)
    std::cout << a[i] << " " << *(p1 + i) << "  ";
```

Respuesta

A `1 10 2 -3 10 -4 1 10  -3 -3  20 -4` ✓

B `1 10 2 -3 10 -4 1 10  20 -3  10 -4`

C `1 10 2 -3 10 -4 1 10  2 -3  10 20`

D Ninguna de las anteriores

Test 1.7

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 x1, x2, *p1, *p2;
p1 = &x1;
x1 = -8;
{
    std::cout << x1 << "  " << *p1 << "  ";
    int x1 = 3;
    std::cout <<  *p1 << "  ";
    {
        x2 = x1;
        p2 = &x2;
        std::cout << *p1 << "  " << *p2 << "  ";
    }
}
p1 = p2;
*p1 = -10;
std::cout << *p1 << "  " << *p2 << std::endl;
```

Respuesta

A `-8 -8 -8 -8 3 -10 -10` ✓

B `-8 -8 3 3 3 -10 -10`

C `-8 -8 3 3 3 -10 3`

D Ninguna de las anteriores

Test 1.8

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 *p1, *p2, *p3, k;
int a[] = { 1, 2, 3, 4 };
k = -1;
p1 = a;
p2 = &k;
*(p1++) = *p2;
p2 = (p1 + 1);
std::cout << *p1 << " " << *p2 << " ";
p2 = new int;
*p2 = k;
std::cout << a[0] << " " << a[1] << " " << a[2] << "  ";
p3 = p1;
std::cout << *p1 << " " << *p2 << " " << *p3 << "  ";
```

Respuesta

A `3 2 -1 2 3  3 3 3`

B `2 3 -1 2 3  2 -1 2` ✓

C `-1 1 1 -1 3  -1 -1 -1`

D Ninguna de las anteriores

Test Bloque 2

Test 2.1

La estructura `nodo` se ha declarado de la forma siguiente:

```
struct nodo {
  char a;
  struct nodo* x;
};
```

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.

```
struct nodo *nodo1;
struct nodo *no1 = new nodo;
struct nodo *no2 = new nodo;
struct nodo *no3 = new nodo;
no1->a = 'a';
no2->a = 'b';
no3->a = 'c';
no3->x = no2;
no2->x = no1;
no1->x = NULL;
nodo1 = no3;
while (nodo1) {
  std::cout << nodo1->a << " ";
  nodo1 = nodo1->x;
}
```

Respuesta

A `b c a`

B `a b c`

C `c b a` ✓

D Ninguna de las anteriores

Test 2.2

La estructura `nodo` se ha declarado de la forma siguiente:

```
struct nodo {
    int num1;
    struct nodo* n;
};
```

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[] = { 0, 1, 2, 3 };
int *p = a;
struct nodo *nodo1;
struct nodo *no1 = new nodo;
struct nodo *no2 = new nodo;
struct nodo *no3 = new nodo;
no1->num1 = *(p++);
no2->num1 = *(++p);
no3->num1 = *p;
no3->n = no1;
no1->n = no2;
no2->n = NULL;
nodo1 = no3;
while (nodo1) {
   std::cout << nodo1->num1 << " ";
   nodo1 = nodo1->n;
}
```

Respuesta

A `0 1 2`

B `0 2 0`

C `2 0 2` ✓

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::map<std::string, int> mapa;
mapa["Ariadna"] = 10;
mapa["Amalia"] = 8;
mapa["Julia"] = 9;
mapa["Vicente"] = 6;
mapa["Arnaldo"] = 7;
std::cout << mapa.count("Ar") << " ";
std::cout << mapa.count("Julia") << "  ";
std::map<std::string, int>::iterator p = mapa.find("Julia");
while ( p != mapa.end() ) {
    std::cout << (*p).first << " " << (*p).second << "  ";
    p++;
}
mapa.erase("Vicente");
p = mapa.begin();
while ( p != mapa.end() ) {
    std::cout << (*p).first << " " << (*p).second << "  ";
    p++;
}
```

Respuesta

A `1 1 Vicente 6 Arnaldo 9`

B `0 1 Julia 9 Ariadna 10 Arnaldo 7 Amalia 8 Julia 9 Ariadna 10`

C `0 1  Julia 9  Vicente 6  Amalia 8  Ariadna 10  Arnaldo 7  Julia 9` ✓

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::queue<int> vqueue;
for (int i = 0; i < 3; i++) {
   vqueue.push(i*2);
   int &p = vqueue.back();
   std::cout << p << " ";
}
vqueue.push(-4);
while ( !vqueue.empty() ) {
   int &p = vqueue.front();
   std::cout << p << " ";
   vqueue.pop();
}
std::cout << vqueue.size();
```

Respuesta

A `0 2 4 0 2 4 -4 0` ✓

B `0 2 4 -4 0`

C `0 2 4 0 2 4 3`

D Ninguna de las anteriores

Test 2.5

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<int> vec(3, 1);
vec.push_back(10);
vec.push_back(30);
vec.push_back(20);
vec.push_back(30);
for (unsigned int i = vec.size() - 1; i > 0; i--)
   std::cout << vec.at(i) << " ";
std::cout << count(vec.begin(), vec.end(), 1)      << " ";
std::cout << count(vec.begin(), vec.end() - 1, 30) << std::endl;
```

Respuesta

A `30 20 30 0 1`

B `30 20 30 0 2`

C `30 20 30 10 1 1 3 1` ✓

D Ninguna de las anteriores

Test 2.6

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<std::string> SS;
SS.push_back("judias");
SS.push_back("verdes");
SS.push_back("pintas");
for (unsigned int ii = 1; ii < SS.size(); ii++)
     std::cout << SS[ii] << " ";
std::cout << SS[0][1] << " " << SS[2][3] << std::endl;
```

Respuesta

A `verdes pintas u t` ✓

B `judias verdes pintas j r`

C `judias verdes pintas ju pint`

D Ninguna de las anteriores

Test 2.7

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> lista1(4, 2);
std::list<int>::iterator p = lista1.begin();
p++;
*p = 10;
lista1.pop_front();
lista1.push_back(-3);
lista1.push_front(-4);
p = lista1.end();
p--;
p--;
p = lista1.insert(p, 8);
p++;
lista1.insert(p, 80);
p = lista1.begin();
while ( p != lista1.end() ) {
   std::cout << *p << " ";
   p++;
}
```

Respuesta

A `-4 10 2 8 80 2 -3` ✓

B `-4 8 80 10 -3`

C `-4 2 2 2 8 80 -3`

D Ninguna de las anteriores

Bloque Desarrollo 1

Pregunta 1.1

La matriz de Hilbert **H** es una matriz cuadrada cuyos elementos a_{ij} se calculan de la forma:

$$a_{ij} = \frac{1}{i+j+1}$$

donde i y j son el índice de la fila y la columna respectivamente. La numeración de las filas y columnas comienza por el valor cero. Es decir, si N es el número de filas o columnas de la matriz (la matriz es cuadrada), entonces $i = 0, 1, \ldots, N-1$, $j = 0, 1, \ldots, N-1$.

El determinante de la matriz **H** se puede calcular de forma aproximada de la forma:

$$\det(\mathbf{H}) \simeq 0.645 \cdot N^{-1/4} \cdot (2 \cdot \pi)^N \cdot 4^{-N^2}$$

donde π es el número pi. Escriba un programa en C++ que realice las acciones siguientes:

1. Solicitar al usuario la introducción por consola del tamaño de la matriz, N.
2. Mostrar en la consola los elementos de la matriz **H** en formato fijo con 6 decimales.
3. Mostrar en consola el valor aproximado del determinante de la matriz **H** en formato científico con 10 decimales.
4. Terminar.

Respuesta

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

int main() {
  std::cout << "Introduzca N: " << std::endl;
  int N;
  std::cin >> N;
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++)
      std::cout << std::fixed << std::setprecision(6)
          << 1. / (i + j + 1) << "\t";
    std::cout << std::endl;
  }
  std::cout << "Valor aproximado del determinante: "
      << std::scientific << std::setprecision(10)
      << 0.645 * std::pow(N, -1. / 4) * std::pow(2.0 * std::acos(-1), N)
          * std::pow(4., -N * N) << std::endl;
  return 0;
}
```

Pregunta 1.2

La matriz de Vandermonde, **V**, en su forma cuadrada, está definida mediante los elementos reales x_1, x_2, x_3, …, x_N de la forma siguiente:

$$\mathbf{V} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{N-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^{N-1} \end{pmatrix}$$

El determinante de la matriz **V** puede calcularse de la forma siguiente:

$$\det(\mathbf{V}) = \prod_{i=1}^{N-1} \prod_{j=i+1}^{N} (x_j - x_i)$$

Por ejemplo, para $N = 4$,

$$\det(\mathbf{V}) = (x_2 - x_1) \cdot (x_3 - x_1) \cdot (x_4 - x_1) \cdot (x_3 - x_2) \cdot (x_4 - x_2) \cdot (x_4 - x_3)$$

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

1. Declarar una constante entera global N y asignarle el valor 6.
2. Solicitar al usuario que introduzca por consola los N números reales x_1, …, x_N que definen la matriz de Vandermonde, **V**, de tamaño $N \times N$.
3. Escribir en la consola los elementos de la matriz **V** en formato fijo con 6 decimales.
4. Calcular y escribir en la consola el determinante de la matriz **V** en formato científico con 7 decimales.
5. Terminar.

Respuesta

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

const int N = 6;

int main() {
  std::cout << "Introduzca los " << N << " elementos: ";
  double x[N];
  for (int i = 0; i < N; i++)
    std::cin >> x[i];
  // Escritura en consola de la matriz
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++)
      std::cout << std::fixed << std::setprecision(6)
          << std::pow(x[i], j) << "\t";
    std::cout << std::endl;
  }
  // Cálculo del determinante
  double D_V = 1;
  for (int i = 0; i < N - 1; i++)
    for (int j = i + 1; j < N; j++)
      D_V *= x[j] - x[i];
  // Escritura del determinante en consola
  std::cout << "Determinante: "
      << std::scientific << std::setprecision(7)
      << D_V << std::endl;

  return 0;
}
```

Pregunta 1.3

El “problema del cumpleaños” consiste en calcular la probabilidad de que, en un grupo de N personas, al menos dos de ellas cumplan años el mismo día del año. Suponemos que un año tiene 365 días.

Escriba un programa en C++ que estime dicha probabilidad, generando aleatoriamente M grupos (de N personas cada uno) y contando en cuántos de ellos se produce coincidencia. Si se produce coincidencia en C grupos de un total de M, entonces la probabilidad buscada puede estimarse como C/M.

N y M deben ser constantes globales del programa, a las que debe asignarse los valores $N = 23$ y $M = 100000$.

La forma en que el programa determina si en un grupo se produce coincidencia es la siguiente. El programa genera aleatoriamente el día de nacimiento de cada una de las personas del grupo. Éste día será un número entero uniformemente distribuido entre 1 y 365. El grupo está compuesto por N personas, por lo que el programa genera aleatoriamente N números enteros distribuidos uniformemente entre 1 y 365. Si dos o más personas del grupo han nacido el mismo día, entonces en ese grupo se produce coincidencia. En caso contrario, no se produce coincidencia.

Puede obtenerse una observación de la variable aleatoria discreta distribuida uniformemente entre 1 y 365 ejecutando la expresión siguiente: std::rand()%365 + 1. Para emplear std::rand(), el programa debe incluir la cabecera cstdlib.

Respuesta

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

const int N = 23;   // Número de personas por grupo
const int M = 100000; // Número de grupos

int main() {
  std::srand(time(NULL)); // Semilla del generador
  int C = 0;  // Número de grupos en los que se produce coincidencia
  for (int i = 0; i < M; i++) {  // Grupos
    std::vector<int> G(N, 0);
    bool coincidencia = false;
    for (int r = 0; !coincidencia && r < N; r++) { // Personas
      G[r] = std::rand() % 365 + 1;
      for (int j = 0; !coincidencia && j < r; j++)
        coincidencia = (G[r] == G[j]);
    }
    if (coincidencia)
      C++;
  }
  std::cout << "Probabilidad: " << (double) C / M << std::endl;
  return 0;
}
```

Pregunta 1.4

Se desea estimar la probabilidad de que, en un grupo de N personas, al menos tres de ellas cumplan años el mismo día del año. Suponemos que un año tiene 365 días.

Escriba un programa en C++ que estime dicha probabilidad, generando aleatoriamente M grupos (de N personas cada uno) y contando en cuántos de ellos se produce coincidencia. Si se produce coincidencia en C grupos de un total de M, entonces la probabilidad buscada puede estimarse como C/M.

N y M deben ser constantes globales del programa, a las que debe asignarse los valores $N = 23$ y $M = 100000$.

La forma en que el programa determina si en un grupo se produce coincidencia es la siguiente. El programa genera aleatoriamente el día de nacimiento de cada una de las personas del grupo. Éste día será un número entero uniformemente distribuido entre 1 y 365. El grupo está compuesto por N personas, por lo que el programa genera aleatoriamente N números enteros distribuidos uniformemente entre 1 y 365. Si tres o más personas del grupo han nacido el mismo día, entonces en ese grupo se produce coincidencia. En caso contrario, no se produce coincidencia.

Puede obtenerse una observación de la variable aleatoria discreta distribuida uniformemente entre 1 y 365 ejecutando la expresión siguiente: std::rand()%365 + 1. Para emplear std::rand(), el programa debe incluir la cabecera cstdlib.

Respuesta

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

const int N = 23;   // Número de personas por grupo
const int M = 100000; // Número de grupos

int main() {
  std::srand(time(NULL)); // Semilla del generador
  int C = 0;  // Número de grupos en los que se produce coincidencia
  for (int i = 0; i < M; i++) {  // Grupos
    std::vector<int> G(N, 0);
    int coincidencia = 0;
    for (int r = 0; coincidencia<2 && r < N; r++) { // Personas
      G[r] = std::rand() % 365 + 1;
      for (int j = 0; coincidencia<2 && j < r; j++)
        if (G[r] == G[j]) coincidencia++;
    }
    if (coincidencia==2)
      C++;
  }
  std::cout << "Probabilidad: " << (double) C / M << std::endl;
  return 0;
}
```

Pregunta 1.5

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

$$x_{k+1} = x_k - \frac{x_k^2 \cdot \text{sen}(x_k) - 0.5}{2 \cdot x_k \cdot \text{sen}(x_k) + x_k^2 \cdot \cos(x_k)} \quad \text{con} \;\; k = 0, 1, 2, \ldots, N-1$$

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 15. El nombre del fichero de texto debe almacenarse en una variable global de tipo string llamada *outFich* a la cual debe asignarse el valor "resIterNewton.txt".

Si para algún valor de k se satisface

$$2 \cdot x_k \cdot \text{sen}(x_k) + x_k^2 \cdot \cos(x_k) = 0$$

el programa debe interrumpir el cálculo y terminar.

Respuesta

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

const int N = 15;
std::string outFich = "resIterNewton.txt";

int main() {
  std::cout << "Introduzca x_0 ";
  double x_k;
  std::cin >> x_k;
  // Abre fichero de texto para escritura
  std::ofstream fich(outFich.c_str(), std::ios::out | std::ios::trunc);
  if (!fich) {
    std::cout << "Error al abrir fichero para escritura";
    return 1;
  }
  for (int k = 0; k < N; k++) {
    double denom = 2 * x_k * std::sin(x_k) + x_k * x_k * std::cos(x_k);
    if (!denom) {
      std::cout << "Denominador cero" << std::endl;
      fich.close();
      return 0;
    }
    x_k -= (x_k * x_k * std::sin(x_k) - 0.5) / denom;
    fich << std::scientific << std::setprecision(10) << x_k << std::endl;
  }
  fich.close();
  return 0;
}
```

Pregunta 1.6

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

$$x_{k+1} = x_k - \frac{x_k^2 \cdot \cos(x_k) - 0.3}{2 \cdot x_k \cdot \cos(x_k) - x_k^2 \cdot \text{sen}(x_k)} \quad \text{con } k = 0, 1, 2, \ldots, N-1$$

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 15. El nombre del fichero de texto debe almacenarse en una variable global de tipo string llamada *outFich* a la cual debe asignarse el valor "resIterNewton.txt".

Si para algún valor de k se satisface

$$2 \cdot x_k \cdot \cos(x_k) - x_k^2 \cdot \text{sen}(x_k) = 0$$

el programa debe interrumpir el cálculo y terminar.

Respuesta

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

const int N = 15;
std::string outFich = "resIterNewton.txt";

int main() {
  std::cout << "Introduzca x_0 ";
  double x_k;
  std::cin >> x_k;
  // Abre fichero de texto para escritura
  std::ofstream fich(outFich.c_str(), std::ios::out | std::ios::trunc);
  if (!fich) {
    std::cout << "Error al abrir fichero para escritura";
    return 1;
  }
  for (int k = 0; k < N; k++) {
    double denom = 2 * x_k * std::cos(x_k) - x_k * x_k * std::sin(x_k);
    if (!denom) {
      std::cout << "Denominador cero" << std::endl;
      fich.close();
      return 0;
    }
    x_k -= (x_k * x_k * std::cos(x_k) - 0.3) / denom;
    fich << std::scientific << std::setprecision(10) << x_k << std::endl;
  }
  fich.close();
  return 0;
}
```

Pregunta 1.7

Escriba un programa en C++ que calcule y escriba en un fichero de texto, 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:

$$f(x) = x^2 \cdot \text{sen}(x) - 0.5$$

Debe declararse N como una constante global y asignarle el valor 10. El nombre del fichero de texto debe almacenarse en una variable global de tipo string llamada *outFich* a la cual debe asignarse el valor “resultado.txt”.

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

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

const int N = 10;
std::string outFich = "resultado.txt";

double f(double x) {
  return x * x * std::sin(x) - 0.5;
}

int main() {
  double x_km1, x_k;
  std::cout << "x0: ";
  std::cin >> x_km1;
  std::cout << "x1: ";
  std::cin >> x_k;
  // Abre fichero de texto para escritura
  std::ofstream fich(outFich.c_str(), std::ios::out | std::ios::trunc);
  if (!fich) {
    std::cout << "Error al abrir fichero para escritura";
    return 1;
  }
  for (int k = 1; k < N; k++) {
    double dif_f = f(x_k) - f(x_km1);
    if (!dif_f) {
      std::cout << "Denominador cero" << std::endl;
      fich.close();
      return 0;
    }
    double x_kM1 = x_k - (x_k - x_km1) * f(x_k) / dif_f;
    fich << std::scientific << std::setprecision(10) << x_kM1 << std::endl;
    x_km1 = x_k;
    x_k = x_kM1;
  }
  fich.close();
  return 0;
}
```

Pregunta 1.8

Escriba un programa en C++ que calcule y escriba en un fichero de texto, en formato científico, con 8 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:

$$f(x) = x^2 \cdot \cos(x) - 0.3$$

Debe declararse N como una constante global y asignarle el valor 12. El nombre del fichero de texto debe almacenarse en una variable global de tipo string llamada *outFich* a la cual debe asignarse el valor “resul.txt”.

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

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

const int N = 12;
std::string outFich = "resul.txt";

double f(double x) {
  return x * x * std::cos(x) - 0.3;
}

int main() {
  double x_km1, x_k;
  std::cout << "x0: ";
  std::cin >> x_km1;
  std::cout << "x1: ";
  std::cin >> x_k;
  // Abre fichero de texto para escritura
  std::ofstream fich(outFich.c_str(), std::ios::out | std::ios::trunc);
  if (!fich) {
    std::cout << "Error al abrir fichero para escritura";
    return 1;
  }
  for (int k = 1; k < N; k++) {
    double dif_f = f(x_k) - f(x_km1);
    if (!dif_f) {
      std::cout << "Denominador cero" << std::endl;
      fich.close();
      return 0;
    }
    double x_kM1 = x_k - (x_k - x_km1) * f(x_k) / dif_f;
    fich << std::scientific << std::setprecision(8) << x_kM1 << std::endl;
    x_km1 = x_k;
    x_k = x_kM1;
  }
  fich.close();
  return 0;
}
```

Bloque Desarrollo 2

Pregunta 2.1

Un cierto tipo de conjunto de Julia esta compuesto por la sucesión de valores complejos obtenidos al iterar de acuerdo a la expresión siguiente:

$$z_{n+1} = z_n^2 + c \quad \text{con} \;\; n = 0, 1, \ldots$$

La sucesión queda definida especificando su valor inicial, z_0, y la constante c.

Por ejemplo: $z_0 = 0.5 - 0.03i$, $c = 0.285 + 0.1i$.

Este ejercicio tiene dos partes. Primeramente, escriba una función en C++ que calcule el siguiente valor de la sucesión, z_{n+1}, pasándole como argumentos el valor anterior de la sucesión, z_n, y el valor de c. El prototipo de la función es:

```
std::complex<double> calcJulia( std::complex<double> z,
                                std::complex<double> c );
```

A continuación, escriba un programa en C++ que realice las acciones siguientes:

1. Solicitar al usuario que éste introduzca por consola los valores de los números complejos z_0 y c, que son el valor inicial y el parámetro de la sucesión.
2. Leer dichos valores, calcular, invocando la función anteriormente definida, los 100 primeros números de la sucesión y escribirlos en la consola.
3. Terminar.

Respuesta

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

const int N = 100;

std::complex<double> calcJulia(std::complex<double> z, std::complex<double> c) {
  return std::pow(z, 2.0) + c;
}

int main() {
  // Entrada por consola
  std::cout << "Parte real e imaginaria de z0: ";
  double z0_real, z0_imag;
  std::cin >> z0_real >> z0_imag;
  std::complex<double> z(z0_real, z0_imag);
  std::cout << "Parte real e imaginaria de c: ";
  double c_real, c_imag;
  std::cin >> c_real >> c_imag;
  std::complex<double> c(c_real, c_imag);
  // Escritura en consola
  std::cout << "Valor inicial: (" << z.real() << ", " << z.imag() << ")"
      << std::endl;
  for (int i = 0; i < N; i++) {
    z = calcJulia(z, c);
    std::cout << "(" << z.real() << ", " << z.imag() << ")\n";
  }
  return 0;
}
```

Pregunta 2.2

Los componentes de un cierto conjunto de Julia se calculan mediante la expresión siguiente:

$$z_{n+1} = e^{z_n} - 0.65 \quad \text{con } n = 0, 1, \ldots$$

La sucesión queda definida especificando su valor inicial, z_0. Por ejemplo: $z_0 = -1 + 1i$.

Este ejercicio tiene dos partes. Primeramente, escriba una función en C++ que devuelva el siguiente valor de la sucesión, z_{n+1}, calculado a partir del último valor de la sucesión, z_n, el cual es pasado como argumento a la función. El prototipo de la función es:

```
std::complex<double> calcJuliaExp(std::complex<double> z);
```

A continuación, escriba un programa en C++ que realice las acciones siguientes:

1. Escribir un mensaje en la consola solicitando al usuario que introduzca el valor inicial de la sucesión, z_0.
2. Leer dicho valor.
3. Ir calculando y escribiendo en la consola los 100 primeros términos de la sucesión, invocando para ello la función que acaba de definir. A continuación de calcular cada término z_{n+1}, el programa debe comprobar si se satisface la condición:

   $$|z_{n+1}| < 10^6$$

   donde $|z_{n+1}|$ es el módulo del término z_{n+1}. Si no se satisface, el programa debe interrumpir el cálculo de la sucesión y terminar.
4. Terminar.

Respuesta

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

const int N = 100;
const double zLimite = 1e6;

std::complex<double> calcJuliaExp(std::complex<double> z) {
  return std::exp(z) - 0.65;
}

int main() {
  // Entrada por consola
  std::cout << "Parte real e imaginaria de z0: ";
  double z0_real, z0_imag;
  std::cin >> z0_real >> z0_imag;
  std::complex<double> z(z0_real, z0_imag);
  // Escritura en consola
  std::cout << "Valor inicial: (" << z.real() << ", " << z.imag() << ")"
      << std::endl;
  for (int i = 0; i < N; i++) {
    z = calcJuliaExp(z);
    std::cout << "(" << z.real() << ", " << z.imag() << ")\n";
    if (abs(z) >= zLimite) {
      std::cout << "No se satisface abs(z_" << i + 1 << ") < " << zLimite;
      return 0;
    }
  }
  return 0;
}
```

Pregunta 2.3

Un fichero de texto llamado *circulos.txt* contiene tres columnas de números reales. El fichero contiene un número *M* de filas, a priori desconocido. Cada fila describe un círculo, por lo que en el fichero hay descritos *M* círculos. En las dos primeras columnas están escritas respectivamente las coordenadas cartesianas X e Y del centro del círculo, y en la tercera columna el radio del círculo.

Escriba un programa en C++ que lea el fichero de texto y calcule cuántos de los círculos no solapan con ningún otro círculo. Dos círculos solapan si la distancia entre sus centros es menor o igual a la suma de sus radios.

Podemos suponer que el formato del fichero es correcto, por lo que no se produce ningún error en su lectura, y que no hay círculos repetidos.

Respuesta

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

std::string nombreFich = "circulos.txt";

struct Circulo {
  double x, y, R;
  bool solapado;
};

bool solapan(Circulo E1, Circulo E2) {
  return std::pow(E1.x - E2.x, 2) + std::pow(E1.y - E2.y, 2)
      <= std::pow(E1.R + E2.R, 2);
}

int main() {
  // Lectura del fichero y carga en el vector vCirculos
  std::ifstream fich_in(nombreFich.c_str(), std::ios::in);
  if (!fich_in) {
    std::cout << "Error al abrir el fichero" << std::endl;
    return 0;
  }
  double x, y, r;
  std::vector<Circulo> vCirculos;
  while (fich_in >> x >> y >> r) {
    Circulo circ = { x, y, r, false };
    vCirculos.push_back(circ);
  }
  fich_in.close();
  // Analiza solapamiento
  for (unsigned int i = 0; i < vCirculos.size(); i++) {
    for (unsigned int j = i + 1; j < vCirculos.size(); j++) {
      bool solapa = solapan(vCirculos[i], vCirculos[j]);
      vCirculos[i].solapado = vCirculos[i].solapado || solapa;
      vCirculos[j].solapado = vCirculos[j].solapado || solapa;
    }
  }
  // Recuento de no solapados
  int numeroNoSolapados = 0;
  for (unsigned int i = 0; i < vCirculos.size(); i++) {
    if (!vCirculos[i].solapado)
      numeroNoSolapados++;
  }
  // Escritura en consola
  std::cout << "Numero de circulos no solapados: " << numeroNoSolapados;
  return 0;
}
```

Pregunta 2.4

Un fichero de texto llamado *esferas.txt* contiene cuatro columnas de números reales. El fichero contiene un número *M* de filas, a priori desconocido. Cada fila describe una esfera, por lo que en el fichero hay descritas *M* esferas. En las tres primeras columnas están escritas respectivamente las coordenadas cartesianas X, Y y Z del centro de la esfera, y en la cuarta columna el radio de la esfera.

Escriba un programa en C++ que lea el fichero de texto y escriba en la consola las esferas, una por fila, ordenadas de menor a mayor volumen. La información que debe escribirse de cada esfera es la que aparece en el fichero de texto (las coordenadas cartesianas de su centro y su radio) y también su volumen.

Podemos suponer que el formato del fichero es correcto, por lo que no se produce ningún error en su lectura. El volumen de la esfera es $\frac{4}{3} \cdot \pi \cdot R^3$, siendo R el radio de la esfera.

Respuesta

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

std::string nombreFich = "esferas.txt";

struct Esfera {
  double x, y, z, R, volumen;
};

int main() {
  // Apertura del fichero para lectura
  std::ifstream fich_in(nombreFich.c_str(), std::ios::in);
  if (!fich_in) {
    std::cout << "Error al abrir el fichero" << std::endl;
    return 0;
  }
  // Lectura del fichero e inserción ordenada en la lista
  double x, y, z, r;
  std::list<Esfera> vEsferas;
  while (fich_in >> x >> y >> z >> r) {
    Esfera esf = { x, y, z, r, 4. / 3 * std::acos(-1) * std::pow(r, 3) };
    if (!vEsferas.size()) {
      vEsferas.push_back(esf);
    } else {
      bool insertada = false;
      std::list<Esfera>::iterator p = vEsferas.begin();
      while (!insertada && p != vEsferas.end()) {
        if (p->volumen > esf.volumen) {
          vEsferas.insert(p, esf);
          insertada = true;
        }
        p++;
      }
      if (!insertada)
        vEsferas.push_back(esf);
    }
  }
  fich_in.close();
  // Escritura en consola
  std::list<Esfera>::iterator p = vEsferas.begin();
  while (p != vEsferas.end()) {
    std::cout << p->x << "\t" << p->y << "\t" << p->z << "\t" << p->R
        << "\t" << p->volumen << "\n";
    p++;
  }
  return 0;
}
```

Bloque Desarrollo 3

Pregunta 3.1

Dados ciertos números de teléfono, se desea obtener aquellos que comienzan por 901. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
std::list<std::string> Num901(std::vector<std::string> tels)
                              throw (std::invalid_argument);
```

que primeramente compruebe si el vector pasado como argumento sólo contiene números de teléfono válidos. Cada componente del vector debería ser un número de teléfono válido. Si alguno no lo es, la función debe lanzar una excepción.

Un número de teléfono es válido si tiene exactamente 9 dígitos y no comienza por cero. Un dígito es un entero comprendido entre 0 y 9, ambos inclusive.

Si todos los componentes del vector son números de teléfono válidos, la función debe devolver una lista cuyos componentes sean únicamente los números de teléfono, de los contenidos en el vector pasado como argumento, que comienzan por 901.

Respuesta

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

std::list<std::string> Num901(std::vector<std::string> tels)
    throw (std::invalid_argument) {
  // Comprueba que los números de teléfono sean válidos
  const int numDigitosTf = 9;
  for (unsigned int i = 0; i < tels.size(); i++) {
    if (tels[i].size() != numDigitosTf)
      throw std::invalid_argument("Longitud incorrecta");
    if (tels[i][0] == '0')
      throw std::invalid_argument("Empieza por cero");
    for (unsigned int j = 0; j < numDigitosTf; j++)
      if (tels[i][j] < '0' || tels[i][j] > '9')
        throw std::invalid_argument("Caracter no valido");
  }
  // Añade a la lista los teléfonos que comienzan por 901
  std::list<std::string> tels901;
  for (unsigned int i = 0; i < tels.size(); i++)
    if (tels[i][0] == '9' && tels[i][1] == '0' && tels[i][2] == '1')
      tels901.push_back(tels[i]);
  return tels901;
}
```

Pregunta 3.2

Dados ciertos números de teléfono, se desea obtener aquellos que comienzan por 901. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
void Num901(std::list<std::string> &tels)
                throw (std::invalid_argument);
```

que primeramente compruebe si la lista pasada como argumento sólo contiene números de teléfono válidos. Cada componente de la lista debería ser un número de teléfono válido. Si alguno no lo es, la función debe lanzar una excepción.

Un número de teléfono es válido si tiene exactamente 9 dígitos y no comienza por cero. Un dígito es un entero comprendido entre 0 y 9, ambos inclusive.

Si todos los componentes de la lista son números de teléfono válidos, la función debe eliminar de la lista pasada como argumento todos aquellos números de teléfono que no comiencen por 901.

Respuesta

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

void Num901(std::list<std::string> &tels) throw (std::invalid_argument) {
  // Comprueba que los números de teléfono sean válidos
  const int numDigitosTf = 9;
  std::list<std::string>::iterator p = tels.begin();
  while (p != tels.end()) {
    if (p->size() != numDigitosTf)
      throw std::invalid_argument("Longitud incorrecta");
    if ((*p)[0] == '0')
      throw std::invalid_argument("Empieza por cero");
    for (unsigned int j = 0; j < numDigitosTf; j++)
      if ((*p)[j] < '0' || (*p)[j] > '9')
        throw std::invalid_argument("Caracter no valido");
    p++;
  }
  // Elimina de la lista los teléfonos que no comienzan por 901
  p = tels.begin();
  while (p != tels.end()) {
    bool num901 = (*p)[0] == '9' && (*p)[1] == '0' && (*p)[2] == '1';
    if (!num901)
      p = tels.erase(p);
    else
      p++;
  }
  return;
}
```

Pregunta 3.3

Un número de DNI está compuesto por 8 dígitos seguidos de una letra. Por ejemplo, 12345678Z y 01234567L. Un dígito es un número entero comprendido entre 0 y 9, ambos inclusive. La letra es una letra mayúscula del conjunto: {A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z }. Se han eliminado las letras I y O, por su parecido con los números uno y cero, y también la letra Ñ.

Dados unos ciertos números de DNI, se desea obtener todos aquellos que terminen en una determinada letra, a la que denominaremos letra de coincidencia. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
std::list<std::string> NDNI( std::vector<std::string> DNIs,
                             char L )
          throw (std::invalid_argument);
```

La función debe primeramente comprobar si cada uno de los componentes del vector pasado como argumento está compuesto por 8 dígitos, seguidos de una letra mayúscula del conjunto de posibles letras: {A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z }. También debe comprobar que el segundo argumento de la función, la letra de coincidencia, pertenece al anterior conjunto de posibles letras.

Si se satisface lo anterior, la función debe devolver una lista con los DNIs del vector argumento que finalizan en la letra pasada como segundo argumento de la función. Cada elemento de la lista debe ser un número de DNI.

Si no se satisface, la función debe lanzar una excepción.

Respuesta

```cpp
#include <iostream>
#include <string>
#include <stdexcept>
#include <list>
#include <vector>

const char LETRAS[] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L',
    'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'};

bool letraValida(char L) {
  for (unsigned int i=0; i<sizeof(LETRAS)/sizeof(LETRAS[0]); i++)
    if ( L == LETRAS[i] )
      return true;
  return false;
}

std::list<std::string> NDNI( std::vector<std::string> DNIs, char L )
            throw (std::invalid_argument)  {
  // Comprueba que los números de DNI sean válidos
  const int numDigitosDNI = 8;
  for (unsigned int i=0; i<DNIs.size(); i++) {
    if (DNIs[i].size() != numDigitosDNI+1)
      throw std::invalid_argument("Longitud incorrecta");
    for (unsigned int j = 0; j < numDigitosDNI; j++)
      if (DNIs[i][j] < '0' || DNIs[i][j] > '9')
        throw std::invalid_argument("Digito no valido");
    if ( !letraValida(DNIs[i][numDigitosDNI]) )
      throw std::invalid_argument("Letra no valida en DNI");
    if ( !letraValida(L) )
      throw std::invalid_argument("Letra de coincidencia no valida");
  }
  // Lista con los DNIs que tienen la letra de coincidencia
  std::list<std::string> selecDNIs;
  for (unsigned int i=0; i<DNIs.size(); i++)
    if (DNIs[i][numDigitosDNI] == L)
      selecDNIs.push_back(DNIs[i]);
  return selecDNIs;
}
```

Pregunta 3.4

Un número de DNI está compuesto por 8 dígitos seguidos de una letra. Por ejemplo, 12345678Z y 01234567L. Un dígito es un número entero comprendido entre 0 y 9, ambos inclusive. La letra es una letra mayúscula del conjunto: {A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z }. Se han eliminado las letras I y O, por su parecido con los números uno y cero, y también la letra Ñ.

Dados unos ciertos números de DNI, se desea obtener todos aquellos que terminen en una determinada letra, a la que denominaremos letra de coincidencia. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
void LDNI( std::list<std::string> &DNIs, char L )
          throw (std::invalid_argument);
```

La función debe primeramente comprobar si cada uno de los componentes de la lista pasada como argumento está compuesto por 8 dígitos, seguidos de una letra mayúscula del conjunto de posibles letras: {A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z }. También debe comprobar que el segundo argumento de la función, la letra de coincidencia, pertenece al anterior conjunto de posibles letras.

Si se satisface lo anterior, la función debe eliminar de la lista argumento los DNIs que no finalizan en la letra de coincidencia, esto es, que no finalizan en la letra pasada como segundo argumento de la función.

Si no se satisface, la función debe lanzar una excepción.

Respuesta

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

const char LETRAS[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L',
    'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z' };

bool letraValida(char L) {
  for (unsigned int i = 0; i < sizeof(LETRAS) / sizeof(LETRAS[0]); i++)
    if (L == LETRAS[i])
      return true;
  return false;
}

void LDNI(std::list<std::string> &DNIs, char L) throw (std::invalid_argument) {
  // Comprueba que los números de DNI sean válidos
  const int numDigitosDNI = 8;
  std::list<std::string>::iterator p = DNIs.begin();
  while (p != DNIs.end()) {
    if (p->size() != numDigitosDNI + 1)
      throw std::invalid_argument("Longitud incorrecta");
    for (unsigned int j = 0; j < numDigitosDNI; j++)
      if ((*p)[j] < '0' || (*p)[j] > '9')
        throw std::invalid_argument("Digito no valido");
    if (!letraValida((*p)[numDigitosDNI]))
      throw std::invalid_argument("Letra no valida en DNI");
    if (!letraValida(L))
      throw std::invalid_argument("Letra de coincidencia no valida");
    p++;
  }
  // Lista con los DNIs que tienen la letra de coincidencia
  p = DNIs.begin();
  while (p != DNIs.end()) {
    if ((*p)[numDigitosDNI] != L)
      p = DNIs.erase(p);
    else
      p++;
  }
  return;
}
```

Bloque Desarrollo 4

Pregunta 4.1

Puede obtenerse un valor aproximado de la derivada de la función $f(x)$ en el punto x_0 de la forma:

$$s = \frac{f(x_0 + h/2) - f(x_0 - h/2)}{h}$$

Se desea calcular el valor aproximado de la derivada de la función

$$f(x) = x^4$$

para unos determinados valores de x. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
std::vector<double> aproxDer( std::vector<double> &xVals );
```

El vector pasado como argumento contiene los valores de x en los cuales se desea estimar el valor de la derivada. La función devuelve un vector con el valor estimado de la derivada en dichos valores de x.

El valor de h viene definido por una constante global del programa, a la cual se asigna el valor $h = 10^{-3}$.

Respuesta

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

const double h = 1e-3;

double f(double x) {
  return std::pow(x, 4);
}

std::vector<double> aproxDer(std::vector<double> &xVals) {
  std::vector<double> s(xVals.size(), 0);
  for (unsigned int i = 0; i < xVals.size(); i++)
    s[i] = (f(xVals[i] + h / 2) - f(xVals[i] - h / 2)) / h;
  return s;
}
```

Pregunta 4.2

Puede obtenerse un valor aproximado de la derivada de la función $f(x)$ en el punto x_0 de la forma:

$$s = \frac{f(x_0 + h) - f(x_0)}{h}$$

Se desea calcular el valor aproximado de la derivada de la función

$$f(x) = x^4$$

para unos determinados valores de x. A tal fin, escriba una función en C++ con el prototipo siguiente:

```
std::vector<double> aproxDer( std::list<double> &xVals );
```

La lista pasada como argumento contiene los valores de x en los cuales se desea estimar el valor de la derivada. La función devuelve un vector con el valor estimado de la derivada en dichos valores de x.

El valor de h viene definido por una constante global del programa, a la cual se asigna el valor $h = 10^{-4}$.

Respuesta

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

const double h = 1e-4;

double f(double x) {
  return std::pow(x, 4);
}

std::vector<double> aproxDer(std::list<double> &xVals) {
  std::vector<double> s(xVals.size(), 0);
  std::list<double>::iterator p = xVals.begin();
  for (unsigned int i = 0; i < xVals.size(); i++, p++)
    s[i] = (f(*p + h) - f(*p)) / h;
  return s;
}
```

Pregunta 4.3

Se desea calcular cuáles de un conjunto de valores reales están incluidos en un determinado intervalo cerrado de la recta real. A tal fin, programe una función en C++ con el prototipo siguiente:

```
std::list<bool> Incluidos( Intervalo I1,
                           std::list<double> &xVals );
```

donde la estructura

```
struct Intervalo {
    double inferior, superior;
};
```

especifica un intervalo cerrado de la recta real: $inferior$ y $superior$ son el extremo inferior y superior del intervalo respectivamente. Asumimos que se satisface: $inferior < superior$.

Los componentes de la lista pasada como argumento a la función son los valores reales. La función devuelve otra lista del mismo tamaño que la lista argumento, construida de la forma siguiente: el componente de la lista devuelta vale true si el valor del correspondiente componente de la lista argumento está incluido en el intervalo y false en caso contrario.

Respuesta

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

struct Intervalo {
  double inferior, superior;
};

std::list<bool> Incluidos(Intervalo I1, std::list<double> &xVals) {
  std::list<bool> inl;
  std::list<double>::iterator p = xVals.begin();
  while (p != xVals.end()) {
    inl.push_back(*p >= I1.inferior && *p <= I1.superior);
    p++;
  }
  return inl;
}
```

Pregunta 4.4

Se desea calcular cuáles de un conjunto de valores reales están incluidos en un determinado intervalo cerrado de la recta real. A tal fin, programe una función en C++ con el prototipo siguiente:

```
std::vector<bool> Incluidos( Intervalo I,
                             std::vector<double> &xVals );
```

donde la estructura

```
struct Intervalo {
    double inferior, superior;
};
```

especifica un intervalo cerrado de la recta real: *inferior* y *superior* son el extremo inferior y superior del intervalo respectivamente. Asumimos que se satisface: $inferior < superior$.

Los componentes del vector pasado como argumento a la función son los valores reales. La función devuelve otro vector del mismo tamaño que el vector argumento, construido de la forma siguiente: el componente del vector devuelto vale true si el valor del correspondiente componente del vector argumento está incluido en el intervalo y false en caso contrario.

Respuesta

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

struct Intervalo {
  double inferior, superior;
};

std::vector<bool> Incluidos(Intervalo I, std::vector<double> &xVals) {
  std::vector<bool> inl(xVals.size(), false);
  for (unsigned int i = 0; i < xVals.size(); i++)
    inl[i] = xVals[i] >= I.inferior && xVals[i] <= I.superior;
  return inl;
}
```