Laboratorio 19

Functores

Nombre: Gabriel Fernando Rodriguez Cutimbo

CUI: 20212157

Grupo: B

Repositorio GitHub:

https://github.com/gaco123/EPCC_CCII.git

1. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Utilizando Functores, elabore una clase que calcule la ecuación de regresión lineal simple ($y = a + bc$) de un conjunto de pares de datos (x, y) almacenados en un vector, retorne como parte del resultado los valores de a y b. Apóyese en functores para calcular las diferentes sumatorias que pueden presentarse.

Código:

```
/*
1. Utilizando Functores, elabore una clase que calcule la ecuación de regresión lineal
simple (y = bc + a) de un conjunto de pares de datos (x, y) almacenados en un
vector, retorne como parte del resultado los valores de a y b. Apóyese en functores
para calcular las diferentes sumatorias que pueden presentarse.

    original ->          y = bc + a
    modificado - >       y = mx + b

                         x = c
                         m = pendiente
                         b = punto de intersección con la coordenada y (intercepto)
*/

#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

class reg_simple{
public:
```

```
    pair<float,float> operator()(vector<pair<float,float>>& vecx){
        float sumt_x = 0, sumt_y = 0, sumt_xy = 0, sumt_xx = 0;
        float m, b;
        int n = vecx.size();

        //Sumatoria para cada fila de datos x, y, x*y, x*x
        for(auto i = vecx.begin(); i < vecx.end(); i++){
                pair<float,float> temp = *i;
                sumt_x += temp.first;
                sumt_y += temp.second;
                sumt_xy += temp.first * temp.second;
                sumt_xx += pow(temp.first,2);
        }
        m = (sumt_xy - ((sumt_x * sumt_y) / n)) / (sumt_xx - (pow(sumt_x, 2) / n));
        b = (sumt_y / n) - (m * (sumt_x / n));

        return make_pair(m, b);
    }
};

int main(int argc, char *argv[]) {
    vector<pair<float,float>> vecx = {{5,10},{1,9},{3,4},{2,11},{1,20},{19,14}};
    reg_simple x;
    pair<float,float> res = x(vecx);
    cout << "PROGRAMA PARA CALCULAR LA REGRESION LINEAL SIMPLE DE UNA SERIE DE
PUNTOS\n";
    cout << "Puntos precargados\n";
    for(auto i = vecx.begin(); i < vecx.end(); i++){
        pair<float,float> temp = *i;
        cout << "Punto: (" << temp.first << ", " << temp.second << ")\n";
    }
    cout << endl;

    cout << "Ecuación de la recta resultante obtenida por regresión lineal simple\n";
    cout << "y = " << res.first << "x + " << res.second;

    return 0;
}
```

Funcionamiento:

```
PROGRAMA PARA CALCULAR LA REGRESION LINEAL SIMPLE DE UNA SERIE DE PUNTOS
Puntos precargados
Punto: (5, 10)
Punto: (1, 9)
Punto: (3, 4)
Punto: (2, 11)
Punto: (1, 20)
Punto: (19, 14)

Ecuación de la recta resultante obtenida por regresión lineal simple
y = 0.114879x + 10.7398
```

2. Utilizando Functores, elabore una clase que simule el proceso de la función estándar FIND. Se debe trabajar enviando como parámetros el índice de inicio, el índice final de la búsqueda y el dato a buscar. Retorne todas las ocurrencias iguales dentro del rango indicado (debe devolver un vector con los índices de todas las ocurrencias)

Código:

```
/*
2. Utilizando Functores, elabore una clase que simule el proceso de la función
estándar FIND. Se debe trabajar enviando como parámetros el índice de inicio, el
índice final de la búsqueda y el dato a buscar. Retorne todas las ocurrencias
iguales dentro del rango indicado (debe devolver un vector con los índices de
todas las ocurrencias)
*/
#include <iostream>
#include <vector>
using namespace std;

template <class A, class X>
class myfind{
public:
	vector<int> operator()(A ini, A end, X val){
	    vector<int> pos;
	    int n = 0;

	    for(auto i = ini; i <= end; i++){
			if(*i == val){
				pos.push_back(n);
			}
			n++;
	    }
	    return pos;
	}
};

int main(int argc, char *argv[]){
	//Vector x
	vector<int> x = {1,3,2,1,3,5,3,10,2,5,1};
	cout<<"Vector x\n";
	cout<<"[";
	for(auto i = x.begin(); i < x.end(); i++){
	    if(i <  x.end()-1){
			cout << *i << ", ";
	    }
	    else{
			cout << *i << "]";
	    }
	}
	cout<<endl<<endl;
```

```
    //Functor myfind creado
    myfind<vector<int>::iterator,int> a;

    //Vector con las posiciones del valor 3 encontradas mediante el Functor myfind
    vector<int> posf = a(x.begin(), x.end(), 3);
    cout<<"Vector con las posiciones donde se encuentra el valor 3 en el vector x usando myfind\n";
    cout<<"[";
    for(auto i = posf.begin(); i < posf.end(); i++){
        if(i <  posf.end()-1){
                cout << *i << ", ";
        }
        else{
                cout << *i << "]";
        }
    }


    return 0;
}
```

Funcionamiento:

```
Vector x
[1, 3, 2, 1, 3, 5, 3, 10, 2, 5, 1]

Vector con las posiciones donde se encuentra el valor 3 en el vector x usando myfind
[1, 4, 6]
```

3. Utilizando Functores y el método std::sort, elabore una clase Elementos con dos atributos de números enteros a y b. Genere una lista de 20 elementos de esta clase Elementos con valores aleatorios tanto para *a* y *b*. Mediante el método std::sort ordénelos de la forma en que un Objeto al ser comparado con un segundo se tenga la desigualdad : obj1.a < obj2.b . El método std::sort debe de trabajar con un objeto Functores. De ser necesario, investigue como realizar este procedimiento que dependa de un objeto del tipo Functores.

Código:

```
/*
3. Utilizando Functores y el método std::sort, elabore una clase Elementos con dos
atributos de números enteros a y b. Genere una lista de 20 elementos de esta clase
Elementos con valores aleatorios tanto para a y b. Mediante el método std::sort
ordénelos de la forma en que un Objeto al ser comparado con un segundo se tenga
la desigualdad : obj1.a < obj2.b . El método std::sort debe de trabajar con un
```

```
objeto Functores. De ser necesario, investigue como realizar este procedimiento
que dependa de un objeto del tipo Functores.
*/

#include <iostream> //Agrega la funcioncout
#include <algorithm> //Agrega la funcion sort
#include <random> //Agrega los motores para generar número pseudoaleatorios
#include <ctime> //Posibilita usar el tiempo como semilla para los motores de la libreria random
#include <vector> //Permite usar la clase vector<>
using namespace std;

//Motor para generar número aleatorios entre 1 y 100
mt19937 gen(time(0));
uniform_int_distribution<> rand_digits(1,100);

//Clase Elementos como tal
class Elementos{
public:
	Elementos(){
	    val1 = rand_digits(gen);
	    val2 = rand_digits(gen);
	}
	int val1;
	int val2;
};

/*
Suma los dos valores de una clase Elementos y compara
si es menor o mayor respecto a la suma de los dos valores de otra clase Elementos
*/
class sort_Functor{
public:
	bool operator()(Elementos a, Elementos b){
	    return (a.val1 + a.val2) < (b.val1 + b.val2);
	}
};

//Imprime un vector de la clase Elementos
void imprimir(vector<Elementos> lista){
	cout << "[";
	for(auto i = lista.begin(); i<lista.end(); i++){
	    if(i != lista.end()-1){
	            cout << "(" << i->val1 << ", " << i->val2 << "), ";
	    }
	    else{
	            cout << "(" << i->val1 << ", " << i->val2 << ")" << "]";
	    }
	}
}

int main(int argc, char *argv[]){
	//Creación aleatoria de valores para el vector<Elementos> lista
```

```
    vector<Elementos> lista;
    for(int i=0; i<20; i++){
        Elementos x;
        x.val1 = rand_digits(gen);
        x.val2 = rand_digits(gen);
        lista.push_back(x);
    }

    //Impresión del vector<Elementos> lista
    cout<<"Vector lista\n";
    imprimir(lista);
    cout<<endl<<endl;

    //Ordenamiento usando std::sort y el functor sort_Functor
    sort(lista.begin(), lista.end(), sort_Functor());

    //Impresión del vector<Elementos> lista después de usar la función "std::sort" y como comparador el
functor "sort_Functor"
    cout<<"Vector lista ordenado por el functor sort_Functor\n";
    imprimir(lista);

    return 0;
}
```

Funcionamiento:

```
Vector lista
[(3, 69), (3, 64), (18, 72), (21, 85), (40, 89), (72, 64), (13, 87), (42, 12), (92, 61), (80, 64),
(67, 72), (82, 12), (96, 96), (53, 92), (46, 55), (22, 43), (89, 13), (74, 33), (40, 72), (45, 25)]


Vector lista ordenado por el functor sort_Functor
[(42, 12), (22, 43), (3, 64), (45, 25), (3, 69), (18, 72), (82, 12), (13, 87), (46, 55), (89, 13),
(21, 85), (74, 33), (40, 72), (40, 89), (72, 64), (67, 72), (80, 64), (53, 92), (92, 61), (96, 96)]
```

2. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB19_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB19_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB19_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB19_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.