#include <iostream>

#include <vector>

#include <algorithm>

#include <map>

#include <iomanip>

**using namespace** std;

**class** Frecuencia{

**private**:

**int** valVar;

**int** fAbs;

**int** fAcum;

**int** fRel;

**float** fRelAcum;

**int** tBarra;

**public**:

Frecuencia(**int** v,**int** f){valVar=v;fRel=f;}

**void** setValVar(**int** v){valVar = v;}

**void** setFabs(**int** f){fAbs = f;}

**void** setFacum(**int** f){fAcum = f;}

**void** setFrel(**int** r){fRel = r;}

**void** setFracum(**float** a){fRelAcum = a;}

**void** setTbarra(**int** b){tBarra = b;}

**void** sumarFrel(){fRel++;}

**int** getValvar(){**return** valVar;}

**int** getFabs(){**return** fAbs;}

**int** getFacum(){**return** fAcum;}

**int** getFrel(){**return** fRel;}

**float** getFracum(){**return** fRelAcum;}

**int** getTbarra(){**return** tBarra;}

};

**class** descriptivaContinua{

**private**:

vector<**int**> \*Poblacion;

vector<Frecuencia> \*Frecuencias;

**int** n=0;

**public**:

descriptivaContinua(){

Poblacion = **new** vector<**int**>();

Frecuencias = **new** vector<Frecuencia>();

}

**void** setPoblacion(**int** p){

Poblacion->push\_back(p);

}

**void** setFrecuencias(Frecuencia F){

Frecuencias->push\_back(F);

}

vector<**int**> getPoblacion(){

**return** \*Poblacion;

}

vector<Frecuencia> getFrecuencias(){

**return** \*Frecuencias;

}

**void** iterador(**int** i){

cout<<i<<endl;

}

**void** registrarPoblacion(vector<**int**> datos){

std::map<**int**,**int**> Mapa;

std::vector<**int**>::iterator it;

std::sort(datos.begin(),datos.end());

**for**(**auto** dato: datos){

Mapa[dato] = 0;

n++;

}

**for**(**auto** dato : datos){

Mapa[dato] = Mapa[dato]+1;

}

**int** count=0;

**for**(**auto** it = Mapa.begin();it!=Mapa.end();++it){

Frecuencias->push\_back(Frecuencia(it->first,it->second));

}

cout<<"Frecuencia Absoluta\n";

**for**(**auto** &item : \*Frecuencias){

cout<< item.getValvar()<<"---"<< item.getFrel()<<"\n";

}

cout<<"\n";

cout<<"Frecuencia Relativa\n";

**for**(**auto** &item : \*Frecuencias){

//int frRelat;

cout<< item.getValvar()<<"---"<< ((**float**)item.getFrel()/ n)<<"\n";

}

cout<<"\n";

cout<<"Frecuencia Relativa Acumulada\n";

**float** frRelat=0;

**for**(**auto** &item : \*Frecuencias){

frRelat+=(**float**)item.getFrel()/n;

item.setFracum(frRelat);

cout<< item.getValvar()<<"---"<<item.getFracum()<<"\n";

}

cout<<"\n";

cout<<"Frecuencia Absoluta Acumulada\n";

**int** frAbs=0;

**for**(**auto** &item : \*Frecuencias){

frAbs+=item.getFrel();

item.setFacum(frAbs);

cout<< item.getValvar()<<"---"<<item.getFacum()<<"\n";

}

}

**void** mostrarTablaFrecuencias(){

cout<<" Xi Fi hi Hi"<<endl;

**for** (**auto** item : \*Frecuencias) {

cout<<setw(4);

cout<<item.getValvar()<<" "<<item.getFrel()<<" "<<((**float**)item.getFrel()/ n)<<" "<<item.getFracum()<<endl;

}

}

};

**int** main(){

vector<**int**> data = {1,2,8,5,8,3,8,5,6,10,5,7,9,4,10,2,7,6,5,10};

descriptivaContinua poblacion;

poblacion.registrarPoblacion(data);

poblacion.mostrarTablaFrecuencias();

**return** 0;

}

#include <iostream>

#include <vector>

#include <algorithm>

#include <map>

#include <ctime>

#include <cmath>

**using namespace** std;

**class** Frecuencia{

**private**:

**int** V\_variable;

**int** F\_abs=0;

**int** F\_acum=0;

**float** F\_rel;

**float** F\_r\_acum=0;

**int** t\_barra=0;

**public**:

Frecuencia(**int** v,**int** f){

V\_variable=v;

F\_abs=f;

}

**void** set\_V\_variable(**int** v){V\_variable = v;}

**void** set\_F\_abs(**int** f){F\_abs = f;}

**void** set\_F\_acum(**int** f){F\_acum = f;}

**void** set\_F\_rel(**float** numero){F\_rel = numero;}

**void** set\_F\_r\_acum(**float** a){F\_r\_acum = a;}

**void** set\_t\_barra(**int** b){t\_barra = b;}

**int** get\_V\_variable(){**return** V\_variable;}

**int** get\_F\_abs(){**return** F\_abs;}

**int** get\_F\_acum(){**return** F\_acum;}

**float** get\_F\_rel(){ **return** F\_rel;}

**float** get\_F\_r\_acum(){**return** F\_r\_acum;}

**int** get\_t\_barra(){**return** t\_barra;}

};

**class** DescriptivaContinua{

**private**:

vector<**int**> \*Poblacion;

vector<Frecuencia> \*Frecuencias;

**int** total=0;

**int** n=0;

**public**:

DescriptivaContinua()

{

Poblacion = **new** vector<**int**>();

Frecuencias = **new** vector<Frecuencia>();

}

vector<**int**> get\_Poblacion()

{

**return** \*Poblacion;

}

vector<Frecuencia> get\_Frecuencias()

{

**return** \*Frecuencias;

}

**void** registrarPoblacion(vector<**int**> datos)

{

std::map<**int**,**int**> Mapa;

std::vector<**int**>::iterator it;

std::sort(datos.begin(),datos.end());

**for**(**auto** dato: datos)

{

Mapa[dato] = 0;

}

**for**(**auto** dato : datos)

{

Mapa[dato] = Mapa[dato]+1;

}

**for**(**auto** it = Mapa.begin();it!=Mapa.end();++it)

{

total = total + it->second;

Frecuencias->push\_back(Frecuencia(it->first,it->second));

}

cout<<total<<endl;

}

**void** Set\_relativa()

{

**float** numero;

**for**(**auto** &frecuencia : \*Frecuencias)

{

numero = (((**float**)frecuencia.get\_F\_abs()/(**float**)total))\*100.0;

frecuencia.set\_F\_rel(numero);

}

}

**void** Set\_relativa\_acumulada()

{

**double** calc=0.0;

**for**(**auto** &frecuencia : \*Frecuencias)

{

frecuencia.set\_F\_r\_acum(frecuencia.get\_F\_rel()+calc);

calc = calc + frecuencia.get\_F\_rel();

}

}

**void** Set\_absoluta\_acumulada()

{

**int** calc = 0;

**for**(**auto** &frecuencia : \*Frecuencias)

{

frecuencia.set\_F\_acum(frecuencia.get\_F\_abs()+calc);

calc=calc+frecuencia.get\_F\_abs();

}

}

**void** ImprimirTablaFrecuencia()

{

cout<<"Xi fi hi Hi\n";

**for**(**auto** frec : \*Frecuencias)

{

printf("%d %d %.2f %.2f\n",frec.get\_V\_variable(),frec.get\_F\_abs(),frec.get\_F\_rel(),frec.get\_F\_r\_acum());

}

}

};

**int** main()

{

//data ramdom

srand(time(0));

//--------------------

vector<**int**> data(150);

**int** random;

**for**(**int** i=0;i<100;i++)

{

random = rand() %20;

data.push\_back(random);

}

//--------------------

DescriptivaContinua des;

des.registrarPoblacion(data);

des.Set\_relativa();

des.Set\_relativa\_acumulada();

des.Set\_absoluta\_acumulada();

des.ImprimirTablaFrecuencia();

**return** 0;

}