

**Instituto Politécnico Nacional**

**Escuela Superior de Ingeniería Mecánica y Eléctrica**

**Unidad Zacatenco**

**Departamento de Ingeniería Eléctrica**

**Programación Orientada a Objetos.**

**Profesor: Tenorio Huertas José Javier**

**Alumno: González Gálvez Luis Pablo**

**“Tarea 4”**

**“Herencias”**

**Fecha actual:  de octubre del 2020**

**Fecha de entrega: de octubre del 2020**

Índice

Objetivo 3

Introducción 4

Resume (Preliminar) 6

Procedimiento 8

Problema 1 8

Problema 2 18

Problema 3 25

Problema 4 31

Problema 5

Problema 6

Problema 7

Problema 8

Problema 9

Problema 10

Conclusiones

Bibliografía

Objetivo

El objetivo de esta actividad o tarea es hacer que los alumnos entiendan la importancia de las herencias, de las clases padres a las clases hijas, ya que esto puede ser muy útil para reutilizar código, como es el caso de los atributos y de los métodos, y que dado el uso se puedan emplear varios de estos como es el tipo public, private y protected.

Introducción

Herencia

La Herencia es uno de los conceptos fundamentales de la programación orientada a objetos y que permite la reusabilidad de variables y funcionalidades que se han definido en otras clases. Para hablar de herencia se deben introducir los conceptos de clase base y clase derivada. Se conoce como clase base a una clase que va a heredar sus propiedades (variables) y funcionalidades (métodos) a otras clases; por otro lado, se conoce como clase derivada a una clase que se implementa mediante la reutilización de las propiedades y funcionalidades que se heredan de una (o varias) clase base. La herencia contribuye en cierto grado con la escalabilidad de una aplicación ya que cuando se debe modificar o eliminar una variable o un método heredado en todas las clases derivadas, entonces no es necesario que se haga individualmente en cada clase, sino que se hace directamente en la clase base y las clases derivadas simplemente heredan la actualización de esos miembros.

La herencia en C ++ se expresa en la implementación de una clase mediante el uso del operador dos puntos o de ámbito (:) seguido del tipo de herencia y el nombre de la clase base de la que se busca heredar. Vale anotar que el tipo de herencia por defecto en C ++, cuando no se especifica explícitamente ese campo, es la herencia privada. A continuación, se observa la implementación de herencia en C ++:

#include<iostream>

using namespace std;

class ClaseBase

{

protected:

int unaVar = 0;

public:

void unMetodo(void)

{

unaVar++;

cout<<"unaVar = "<<unaVar<<endl;

}

};

class ClaseDerivada : public ClaseBase

/\* Sintaxis para indicar que ClaseDerivada hereda de ClaseBase \*/

{

/\* Esta clase implementa los miembros de clase que hereda de ClaseBase \*/

};

int main()

{

ClaseDerivada obj1;

obj1.unMetodo(); /\* Acceso a los miembros heredados de ClaseBase \*/

obj1.unMetodo();

return 0;

}

Tipos de herencia

**Herencia pública:** Se refiere a la herencia en la que todos los miembros públicos y protegidos de la clase base conservan esos mismos niveles de acceso respectivamente en las clases derivadas.

class ClaseDerivada : public ClaseBase

**Herencia protegida:** Se refiere a la herencia en la que todos los miembros públicos de la clase base adquieren el nivel de acceso protegido en las clases derivadas, mientras que los miembros protegidos conservan su nivel de acceso. Lo anterior indica que una clase derivada puede luego heredar a otra clase los miembros protegidos que heredó de su clase base.

class ClaseDerivada : protected ClaseBase

**Herencia privada:** Se refiere a la herencia en la que todos los miembros públicos y protegidos de la clase base adquieren el nivel de acceso privado en las clases derivadas. De ahí se desprende que una clase derivada que haya heredado mediante herencia privada no puede heredar a otras clases los miembros que ha heredado de otras clases.

class ClaseDerivada : private ClaseBase

Resume Preliminar

El problema #1: Se usaron cuatro clases, de las cuales una es la clase base y las otras tres son las clases derivadas, la herencia de la clase base es de tipo public, se declararon tres atributos uno de tipo float y dos de tipo string arreglo, y un método para regresar el atributo float.

Dos de las clases derivadas tiene tres métodos, dos de tipo float y uno de tipo void.

La ultima clase derivada tiene un atributo de tipo float, y tiene cinco métodos de tipo void, dos de estos para calcular el valor de los componentes pedidos.

El problema #2: Se utilizaron cuatro clases, una es la clase base y las otras tres son las derivadas, la clase base se hereda como public; se declaran cuatro atributos de tipo float.

En las clases derivadas se crean dos métodos de tipo float, que pide una variable de tipo float, y fuera de las clases se declara una función, que pide como valor a los objetos de cada clase, para poder cumplir con la condición previamente pedida en el ejercicio.

El problema #3: En este problema se crearon cinco clases, una clase base y cuatro clases derivadas; en la clase base se declararon doce atributos, el tipo en el que se heredo la clase base a las derivadas es public.

En las clases derivadas se en cada una se crean 3 métodos de tipo float, además de que se crea fuera de las clases una función de tipo void que pide como valor a los objetos y crea una condición para cumplir con lo pedido en el problema.

El problema #4: Se hace uso de tres clases, una es la clase base y las otras dos las derivadas, en la clase base se declaran 7 atributos y cuatro métodos, tres de tipo float y uno de tipo void.

Las clases derivadas tienen 5 métodos donde se hacen todos los cálculos.

El problema #5:

El problema #6:

El problema #7:

El problema #8:

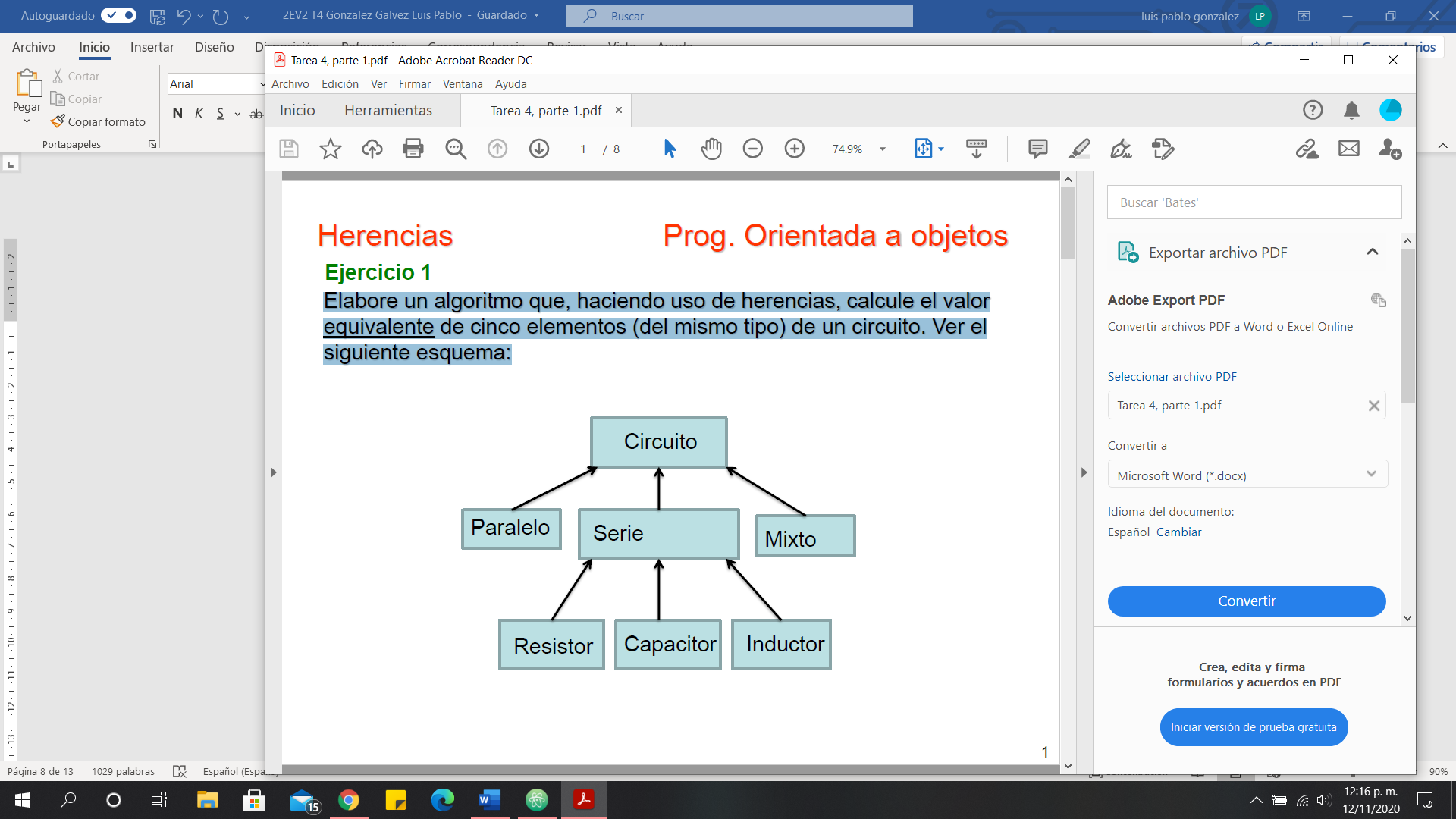
El problema #9:

El problema #10:

Procedimiento

**Problema 1:**

Elabore un algoritmo que, haciendo uso de herencias, calcule el valor equivalente de cinco elementos (del mismo tipo) de un circuito. Ver el siguiente esquema:



Las fórmulas por utilizar son las siguientes:

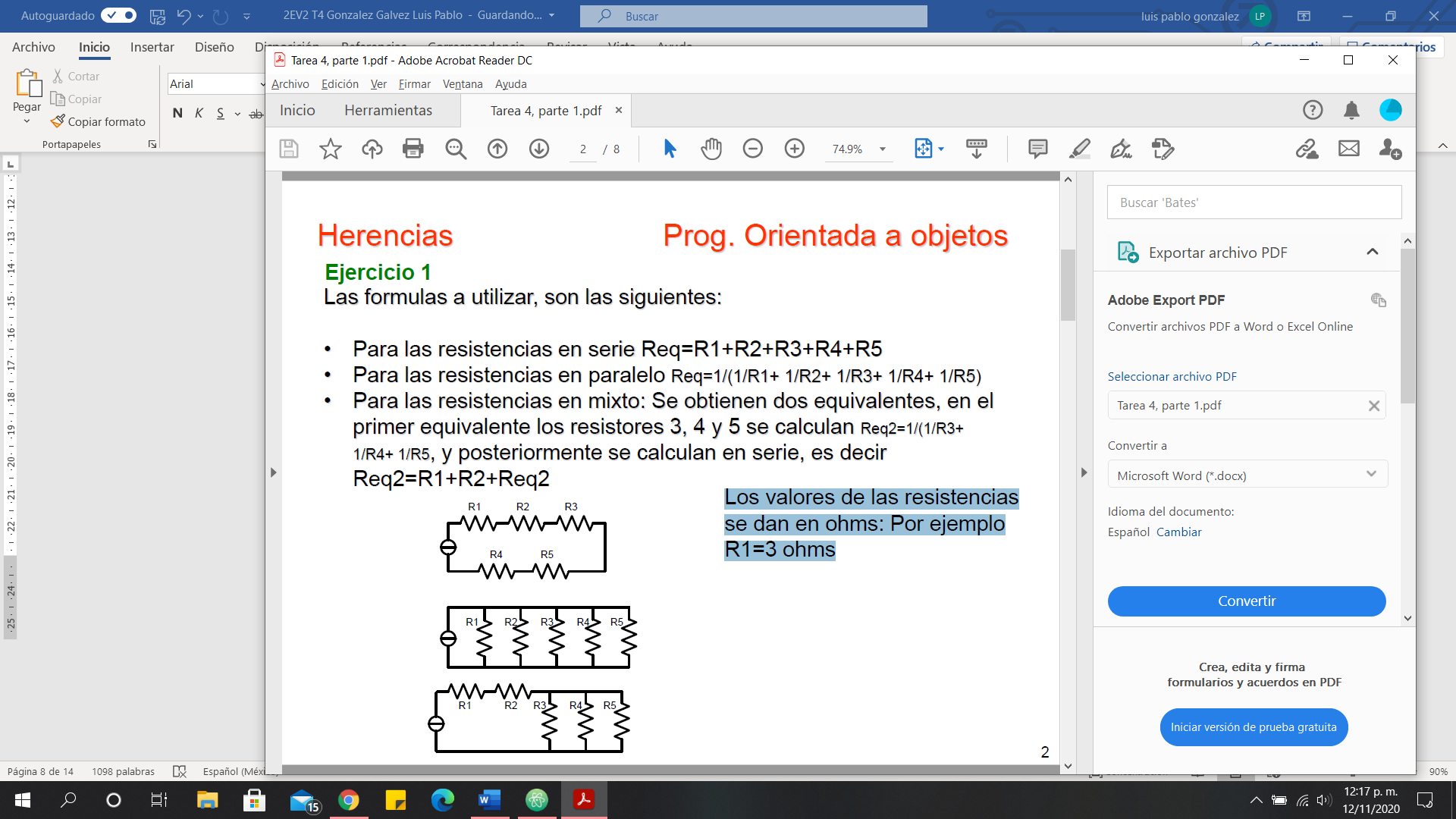
• Para las resistencias en serie Req=R1+R2+R3+R4+R5

• Para las resistencias en paralelo Req=1/(1/R1+ 1/R2+ 1/R3+ 1/R4+ 1/R5)

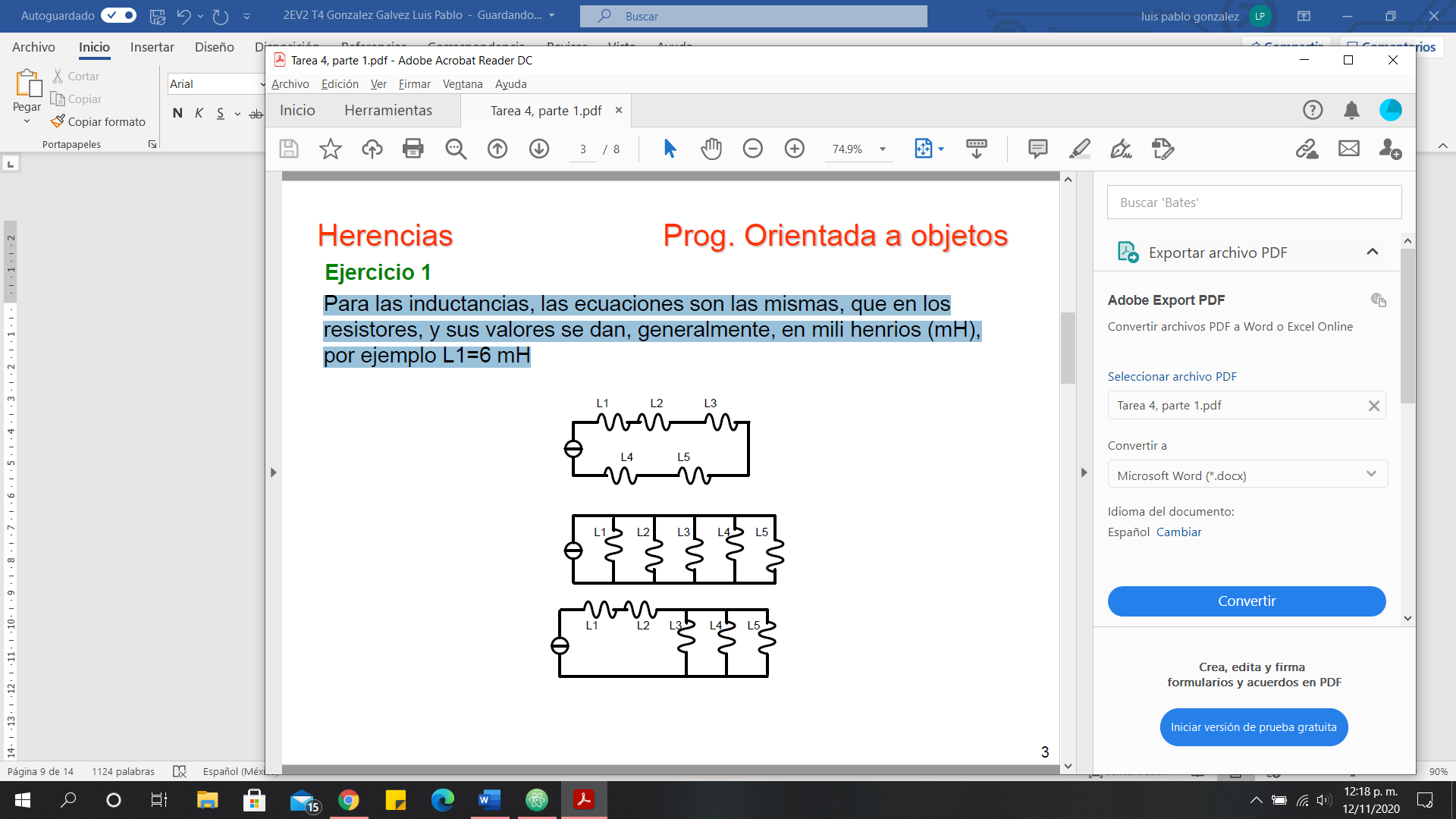
• Para las resistencias en mixto: Se obtienen dos equivalentes, en el primer equivalente los resistores 3, 4 y 5 se calculan Req2=1/(1/R3+1/R4+ 1/R5, y posteriormente se calculan en serie, es decir Req2=R1+R2+Req2.

Nota:

Los valores de las resistencias se dan en ohms: Por ejemplo, R1=3 ohms.



Para las inductancias, las ecuaciones son las mismas, que, en los resistores, y sus valores se dan, generalmente, en mili henrios (mH), por ejemplo L1=6 mH.

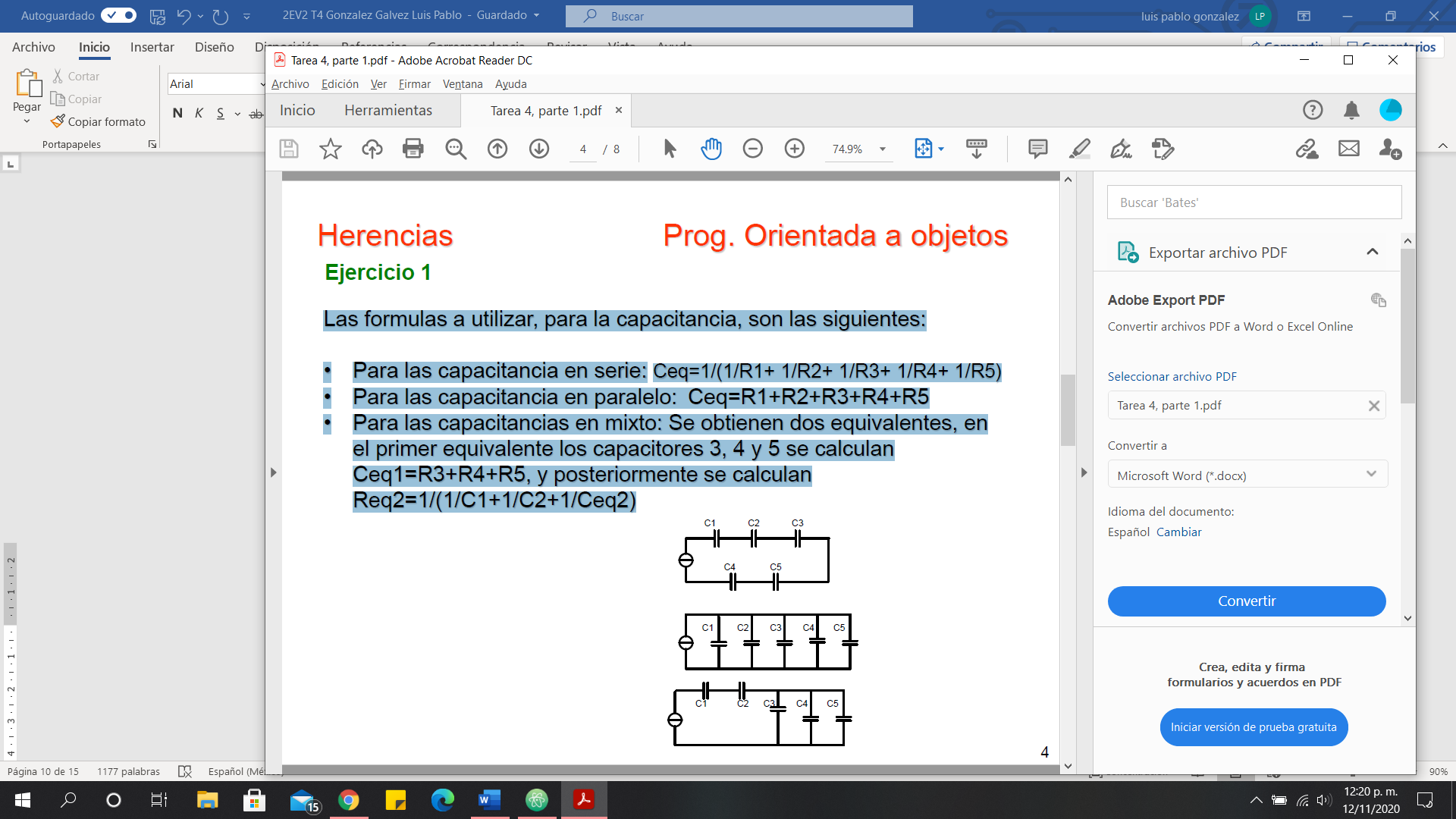


Las fórmulas por utilizar, para la capacitancia, son las siguientes:

• Para las capacitancia en serie: Ceq=1/(1/R1+ 1/R2+ 1/R3+ 1/R4+ 1/R5).

• Para las capacitancias en paralelo: Ceq=R1+R2+R3+R4+R5.

• Para las capacitancias en mixto: Se obtienen dos equivalentes, en el primer equivalente los capacitores 3, 4 y 5 se calculan Ceq1=R3+R4+R5, y posteriormente se calculan Req2=1/(1/C1+1/C2+1/Ceq2).



**Syntaxis:**

////////////////////////

//Librerias

/////////////////////////

#include <iostream>

#include <cstdlib>

#include <cmath>

#include <ctime>

#include <string.h>

using namespace std;

/////////////////////////

//Definiciones

/////////////////////////

#define tam\_max 10

/////////////////////////

//Clases

/////////////////////////

class Circuito{

public:

float equivalencia;

string s[tam\_max]{"resistencia","inductancias"};

string u[tam\_max]{"ohm", "mH", "uf"};

float resultado(){

return equivalencia;

}

};

class Serie:public Circuito{

public:

float suma\_res\_ind(float r1,float r2, float r3,float r4, float r5){

return (r1+r2+r3+r4+r5);

}

float suma\_cap(float r1,float r2, float r3,float r4, float r5){

return (1/(r1+r2+r3+r4+r5));

}

void todo\_serie(){

for(int i=0;i<2;i++){

cout<<"Circuito de "<<s[i]<<" en serie"<<endl;

equivalencia=suma\_res\_ind(1,1,1,1,1);

cout<<"El valor de la equivalencia en serie de "<<s[i]<<" es: "<<resultado()<<u[i]<<endl<<endl;

}

cout<<"Circuito de Capacitores en serie"<<endl;

equivalencia=suma\_cap(1,1,1,1,1);

cout<<"El valor de la equivalencia en serie de capacitores es: "<<resultado()<<u[2]<<endl<<endl;

}

};

class Paralelo:public Circuito{

public:

float suma\_res\_ind(float r1,float r2, float r3,float r4, float r5){

return (1/(r1+r2+r3+r4+r5));

}

float suma\_cap(float r1,float r2, float r3,float r4, float r5){

return (r1+r2+r3+r4+r5);

}

void todo\_paralelo(){

for(int i=0;i<2;i++){

cout<<"Circuito de "<<s[i]<<" en paralelo"<<endl;

equivalencia=suma\_res\_ind(1,1,1,1,1);

cout<<"El valor de la equivalencia en paralelo de "<<s[i]<<" es: "<<resultado()<<u[i]<<endl<<endl;

}

cout<<"Circuito de Capacitores en paralelo"<<endl;

equivalencia=suma\_cap(1,1,1,1,1);

cout<<"El valor de la equivalencia en paralelo de capacitores es: "<<resultado()<<u[2]<<endl<<endl;

}

};

class Mixto:public Circuito{

public:

float r[tam\_max];

void iniciar(int n){

for(int i=0;i<n;i++)

r[i]=1;

}

void imprime(int n){

char c[tam\_max]={'R','L','C'};

for(int j=0;j<3;j++){

for(int i=0; i<n ; i++){

cout<<c[j]<<i+1<<"="<<r[i]<<u[j]<<" ";

}

cout<<endl<<endl;

}

}

void mix\_res\_ind(int l){

float aux\_ser=0;

float aux\_para=0;

int i,n;

char a;

do{

cout<<"Dime que vas a hacer paralelo o serie, s/p"<<endl;

cin>>a;

if(a=='p'){

cout<<"Dime cuantas "<<s[l]<<" estan en paralelo"<<endl;

cin>>n;

for(int j=0;j<n;j++){

cout<<"Dime que "<<s[l]<<endl;

cin>>i;

aux\_para+=1/r[i-1];

}

aux\_para=1/aux\_para;

cout<<"Valor de paralelo: "<<aux\_para<<u[l]<<endl;

}

else{

cout<<"Dime cuantas "<<s[l]<<" estan en serie"<<endl;

cin>>n;

for(int j=0;j<n;j++){

cout<<"Dime que "<<s[l]<<endl;

cin>>i;

aux\_ser+=r[i-1];

}

cout<<"Valor de serie: "<<aux\_ser<<u[l]<<endl;

}

cout<<"Vas a hacer algo mas s/n"<<endl;

cin>>a;

}while(a!='n');

equivalencia=aux\_ser+aux\_para;

cout<<"El valor de la equivalencia en mixto es: "<<resultado()<<u[l]<<endl<<endl;

}

void mix\_cap(){

float aux\_ser=0;

float aux\_para=0;

int i,n;

char a;

do{

cout<<"Dime que vas a hacer paralelo o serie, s/p"<<endl;

cin>>a;

if(a=='p'){

cout<<"Dime cuantas capacitores estan en paralelo"<<endl;

cin>>n;

for(int j=0;j<n;j++){

cout<<"Dime que capacitor"<<endl;

cin>>i;

aux\_para+=r[i-1];

}

cout<<"Valor de paralelo: "<<aux\_para<<u[2]<<endl;

}

else{

cout<<"Dime cuantas capacitores estan en serie"<<endl;

cin>>n;

for(int j=0;j<n;j++){

cout<<"Dime que capacitor"<<endl;

cin>>i;

aux\_ser+=1/r[i-1];

}

aux\_ser=1/aux\_ser;

cout<<"Valor de serie: "<<aux\_ser<<u[2]<<endl;

}

cout<<"Vas a hacer algo mas s/n"<<endl;

cin>>a;

}while(a!='n');

equivalencia=aux\_ser+aux\_para;

cout<<"El valor de la equivalencia en mixto es: "<<resultado()<<u[2]<<endl<<endl;

}

void todo\_mix(){

for(int i=0;i<2;i++){

cout<<"Circuito de "<<s[i]<<" mixto"<<endl;

mix\_res\_ind(i);

}

cout<<"Circuito de Capacitores mixto"<<endl;

mix\_cap();

}

};

/////////////////////////

//Funcion Principal

/////////////////////////

int main(){

Circuito a;

Serie b;

Paralelo c;

Mixto d;

int n;

cout<<"Dime cuantos valores de las Resistencias, Inductancias y Capacitores"<<endl;

cin>>n;

d.iniciar(n);

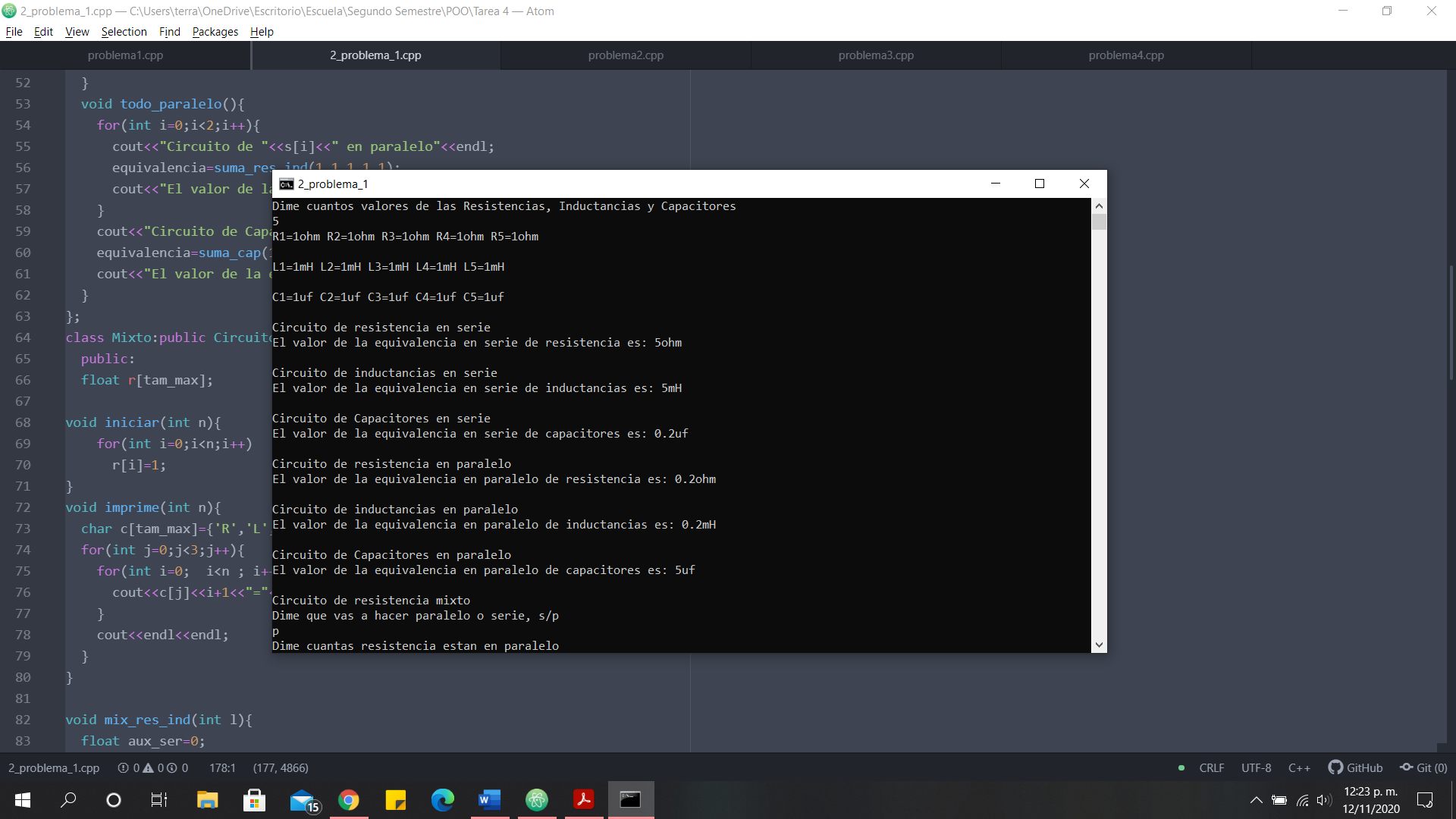
d.imprime(n);

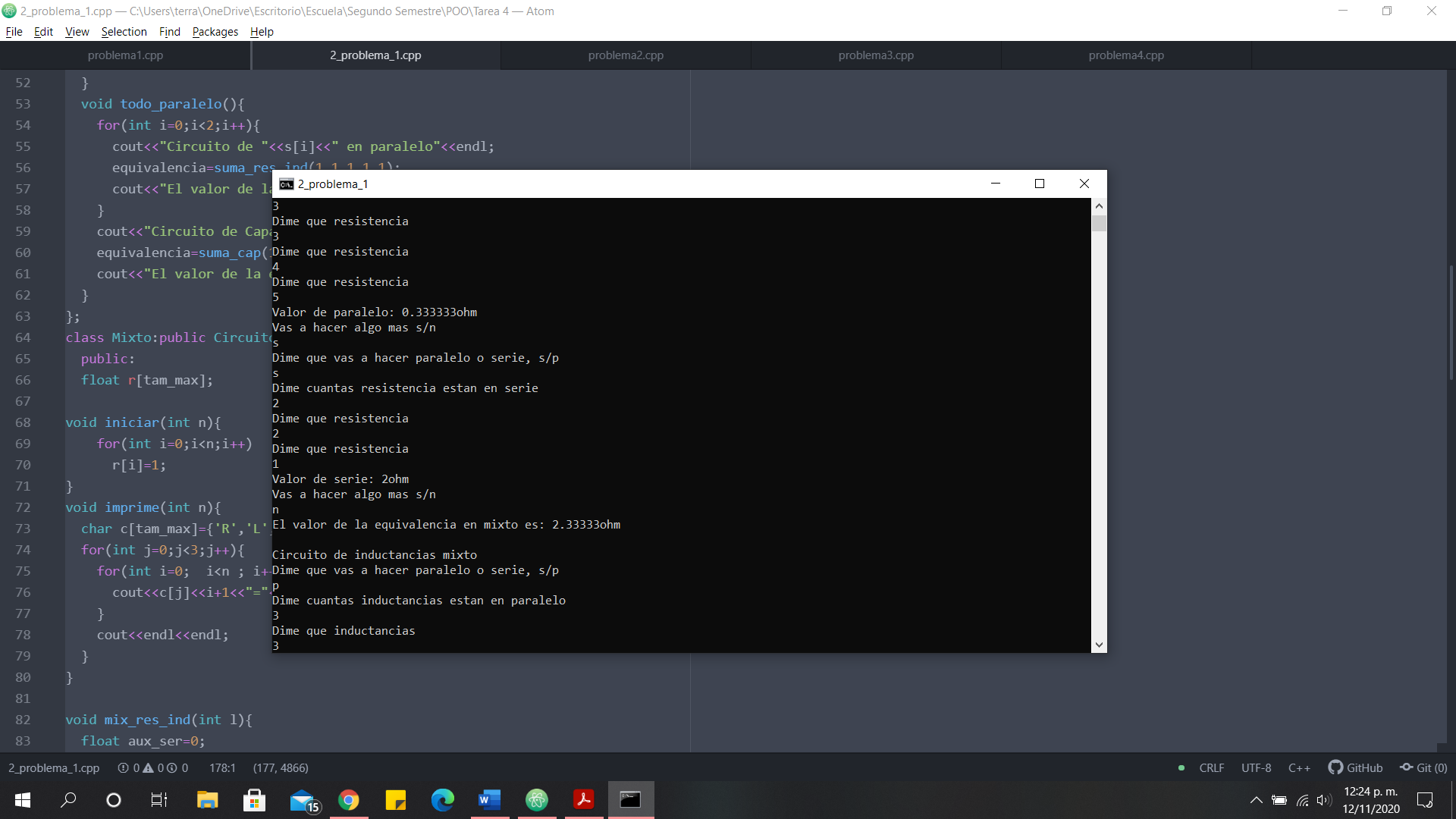
b.todo\_serie();

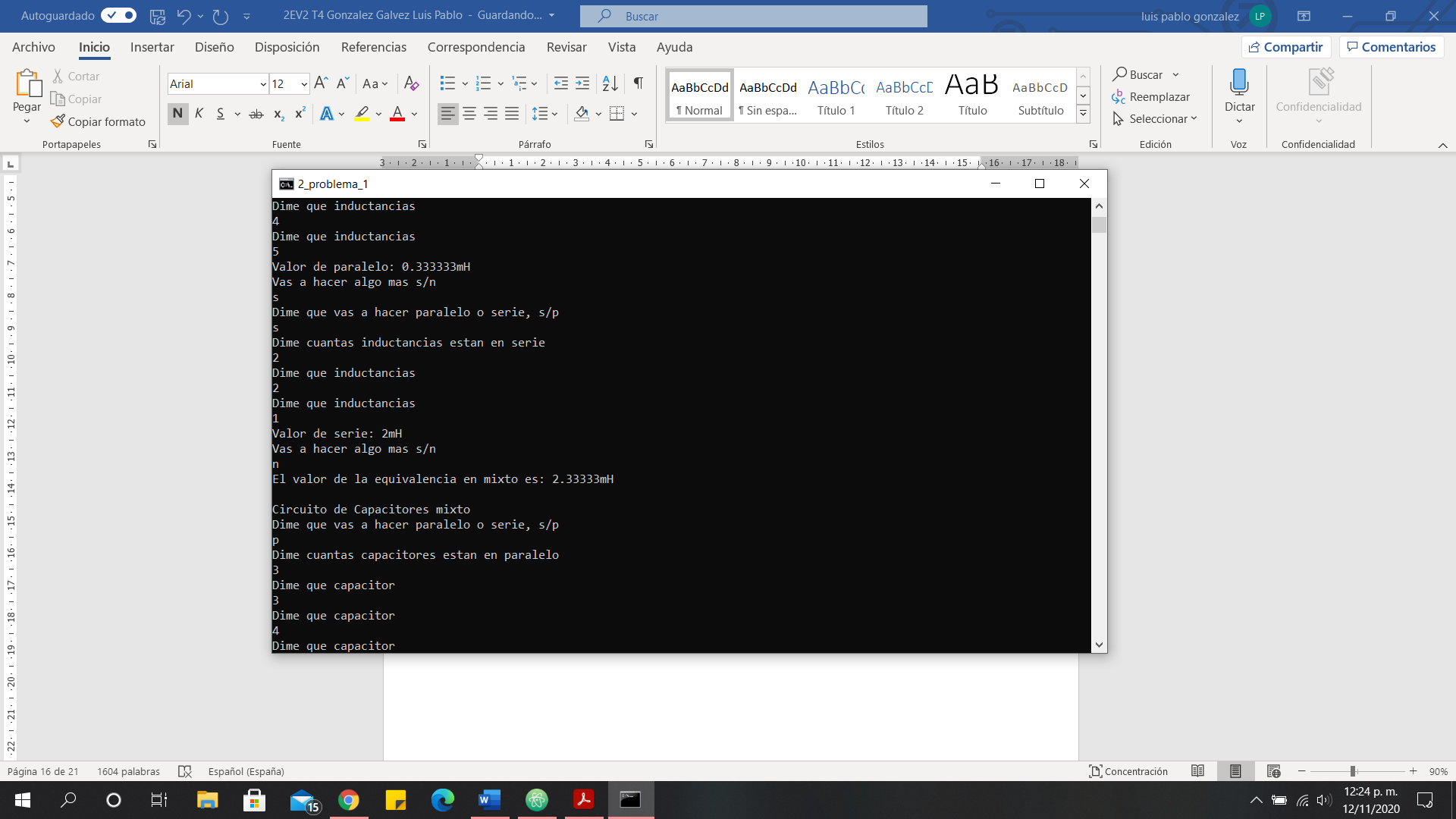
c.todo\_paralelo();

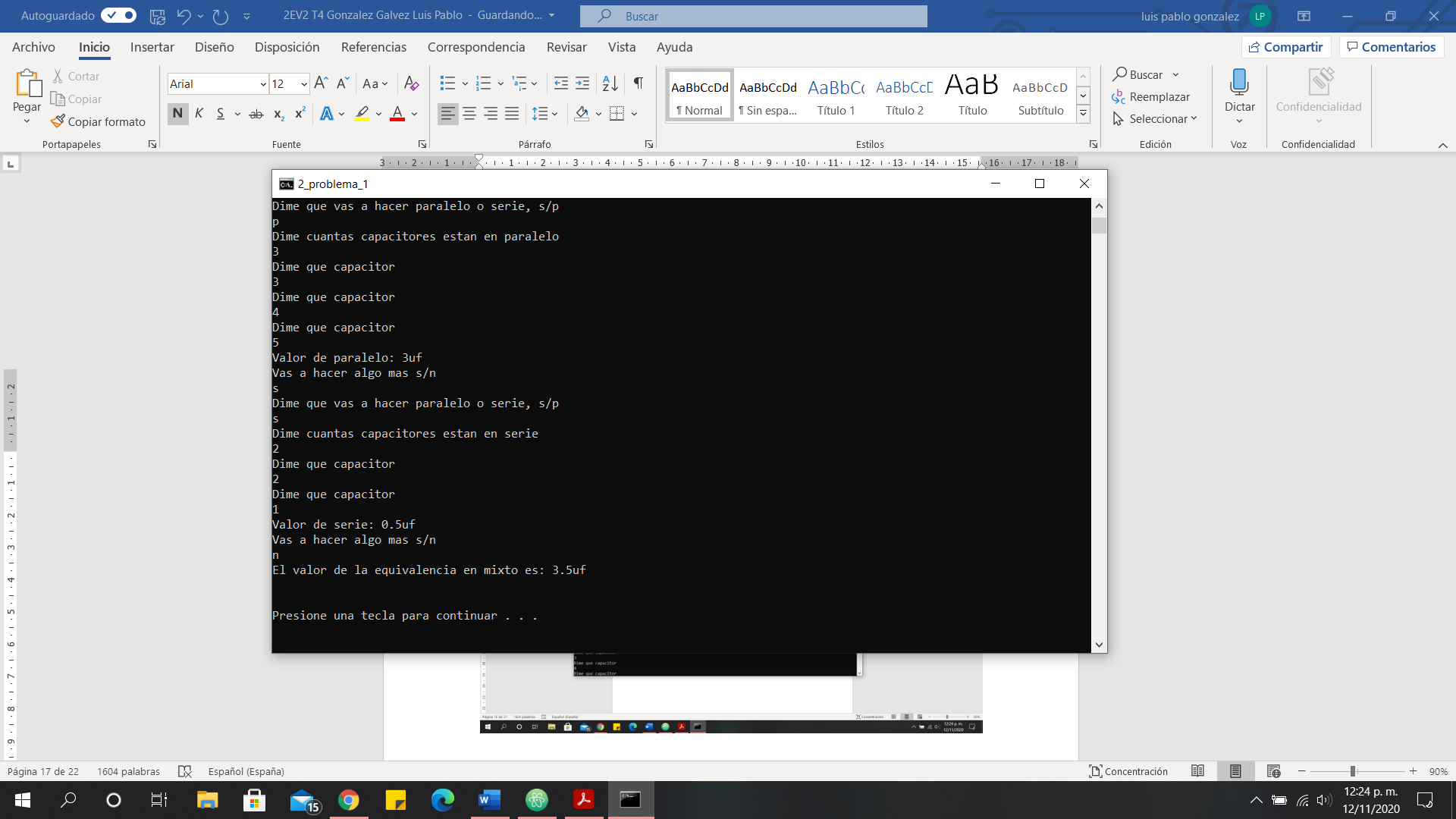
d.todo\_mix();

}



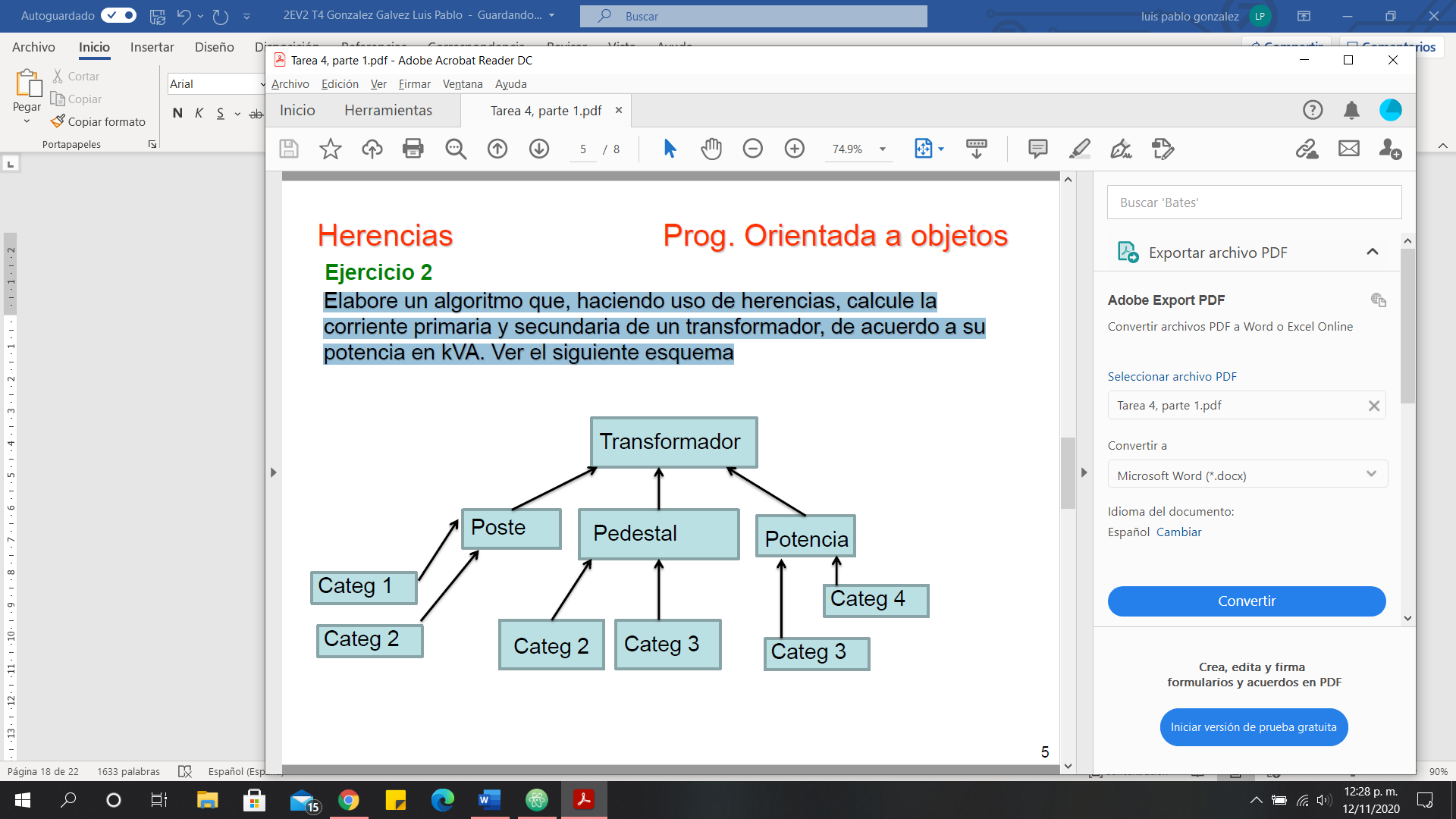




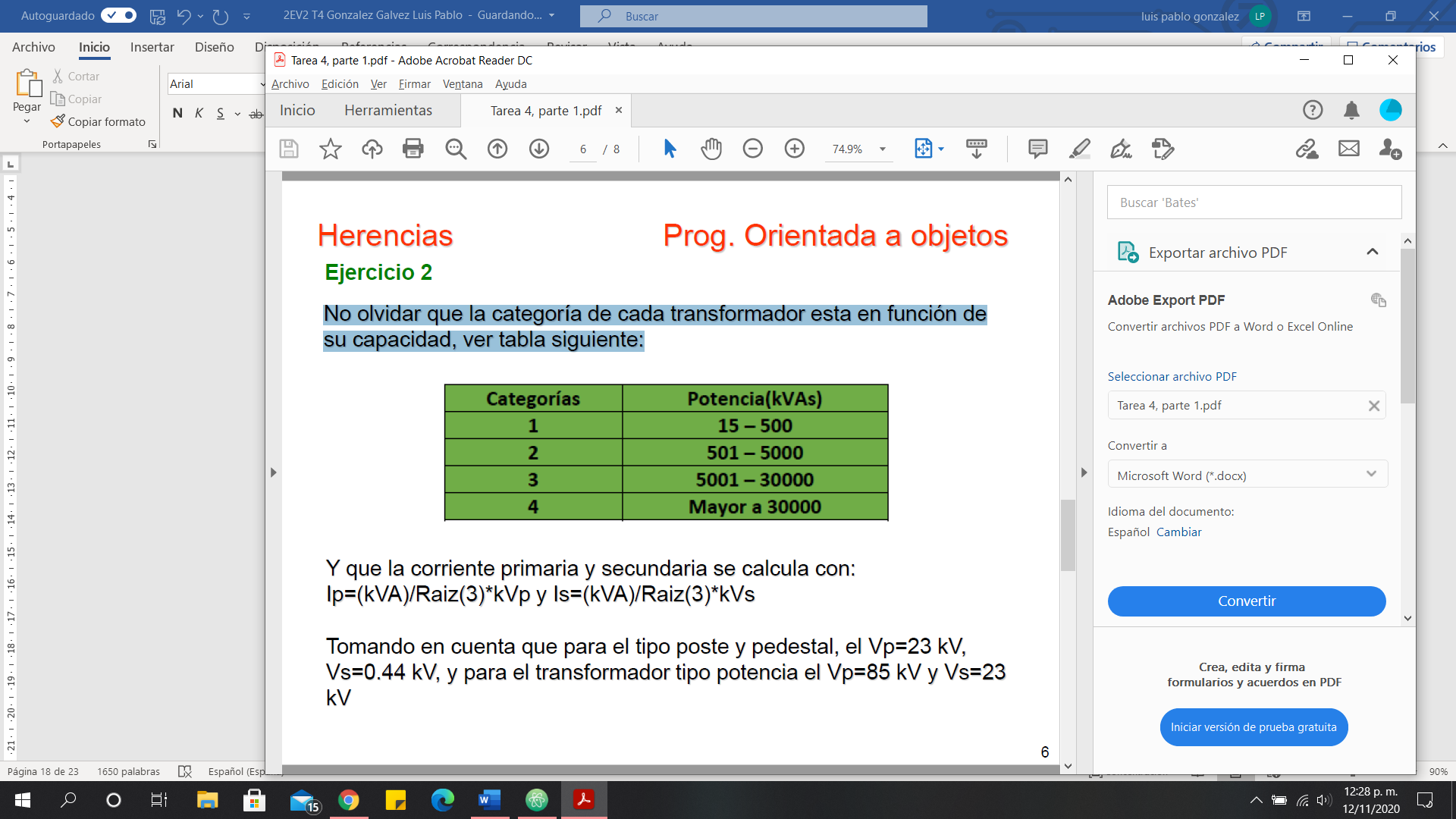


**Problema 2:**

Elabore un algoritmo que, haciendo uso de herencias, calcule la corriente primaria y secundaria de un transformador, de acuerdo a su potencia en kVA . Ver el siguiente esquema:



No olvidar que la categoría de cada transformador esta en función de su capacidad, ver tabla siguiente:



Y que la corriente primaria y secundaria se calcula con:

Ip= kVA /Raiz (3) \* kVp y Is= kVA /Raiz (3) \*kVs

Tomando en cuenta que para el tipo poste y pedestal, el Vp =23 kV,

Vs=0.44 kV, y para el transformador tipo potencia el Vp =85 kV y Vs=23

kV

**Syntaxis:**

////////////////////////

//Librerias

/////////////////////////

#include <iostream>

#include <cstdlib>

#include <cmath>

#include <ctime>

#include <string.h>

using namespace std;

/////////////////////////

//Definiciones

/////////////////////////

#define tam\_max 10

/////////////////////////

//Clases

/////////////////////////

class Transformador{

public:

float vp1=23;

float vp2=85;

float vs1=0.44;

float vs2=23;

};

class Poste:public Transformador{

public:

float co\_pri(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vp1);

return lp;

}

float co\_sec(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vs1);

return lp;

}

};

class Pedestal:public Transformador{

public:

float co\_pri(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vp1);

return lp;

}

float co\_sec(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vs1);

return lp;

}

};

class Potencia:public Transformador{

public:

float co\_pri(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vp2);

return lp;

}

float co\_sec(float kVA){

float lp=0;

lp=kVA/(sqrt(3)\*1000\*vs2);

return lp;

}

};

void todo(Poste b,Pedestal c, Potencia d);

/////////////////////////

//Funcion Principal

/////////////////////////

int main(){

Transformador a;

Poste b;

Pedestal c;

Potencia d;

todo(b,c,d);

}

void todo(Poste b,Pedestal c, Potencia d){

char n;

int f;

char s;

do{

cout<<"Dime el tipo de Transformador al que le quieres calcular"<<endl;

cout<<"su corriente principal y secundaria"<<endl;

cout<<"Para Pedestal/p, Poste/t, Potencia/a"<<endl;

cin>>n;

if(n=='p'||n=='t'){

cout<<"Dime cuantos kVA tiene"<<endl;

cin>>f;

if(f>=15&&f<=500){

cout<<"Categoria 1"<<endl;

cout<<"Corriente principal: "<<b.co\_pri(f)<<endl;

cout<<"Corriente secundaria: "<<b.co\_sec(f)<<endl;

}

else if(f>=501&&f<=5000){

if(n=='p'){

cout<<"Categoria 2"<<endl;

cout<<"Corriente principal: "<<b.co\_pri(f)<<endl;

cout<<"Corriente secuandaria: "<<b.co\_sec(f)<<endl;

}

else{

cout<<"Categoria 2"<<endl;

cout<<"Corriente principal: "<<c.co\_pri(f)<<endl;

cout<<"Corriente secuandaria: "<<c.co\_sec(f)<<endl;

}

}

else if(f>=5001&&f<=30000){

cout<<"Categoria 3"<<endl;

cout<<"Corriente principal: "<<c.co\_pri(f)<<endl;

cout<<"Corriente secuandaria: "<<c.co\_sec(f)<<endl;

}

}

else{

cout<<"Dime cuantos kVA tiene"<<endl;

cin>>f;

if(f>=5001&&f<=30000){

cout<<"Categoria 3"<<endl;

cout<<"Corriente principal: "<<d.co\_pri(f)<<endl;

cout<<"Corriente secuandaria: "<<d.co\_sec(f)<<endl;

}

else{

cout<<"Categoria 4"<<endl;

cout<<"Corriente principal: "<<d.co\_pri(f)<<endl;

cout<<"Corriente secuandaria: "<<d.co\_sec(f)<<endl;

}

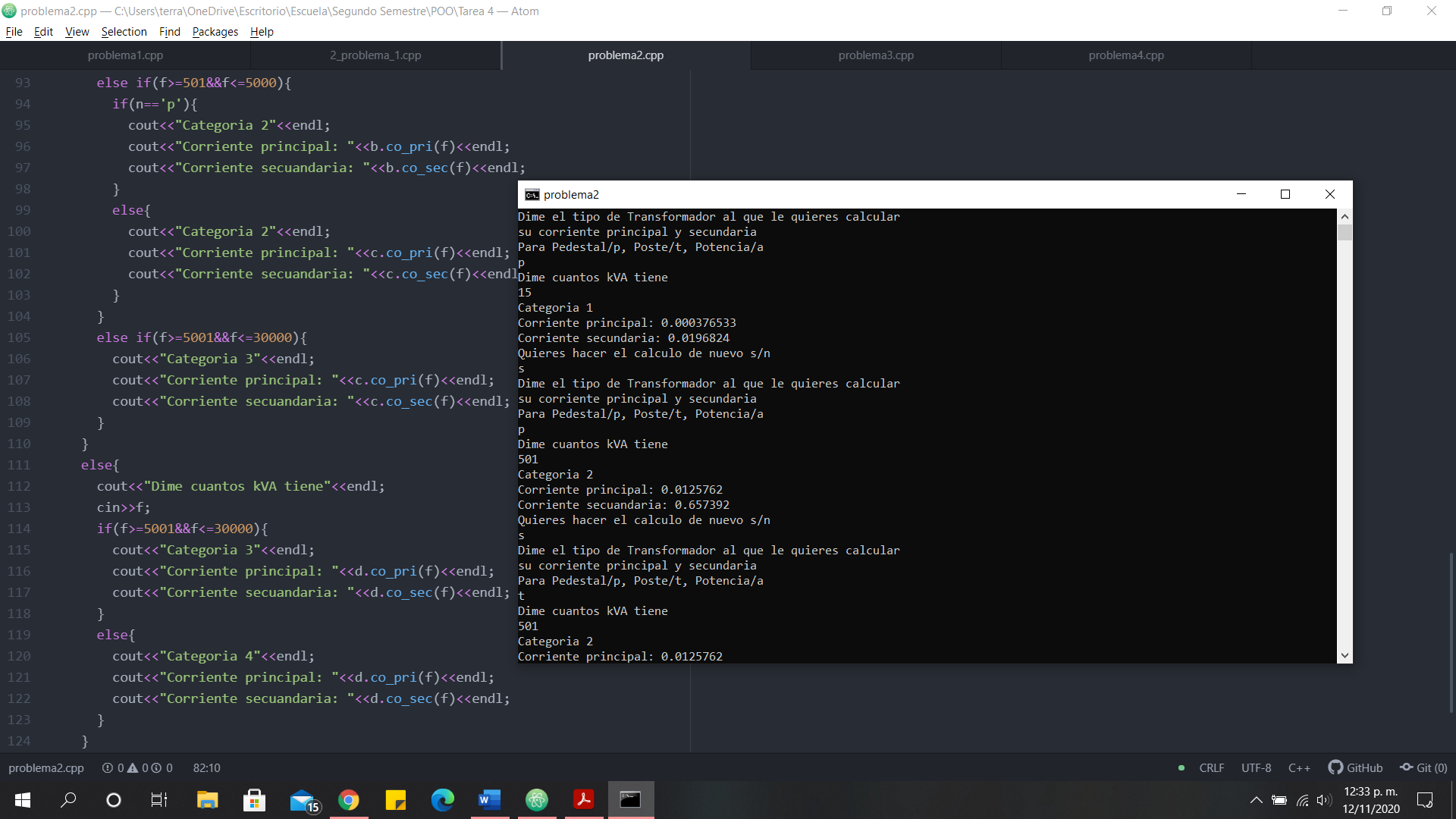
}

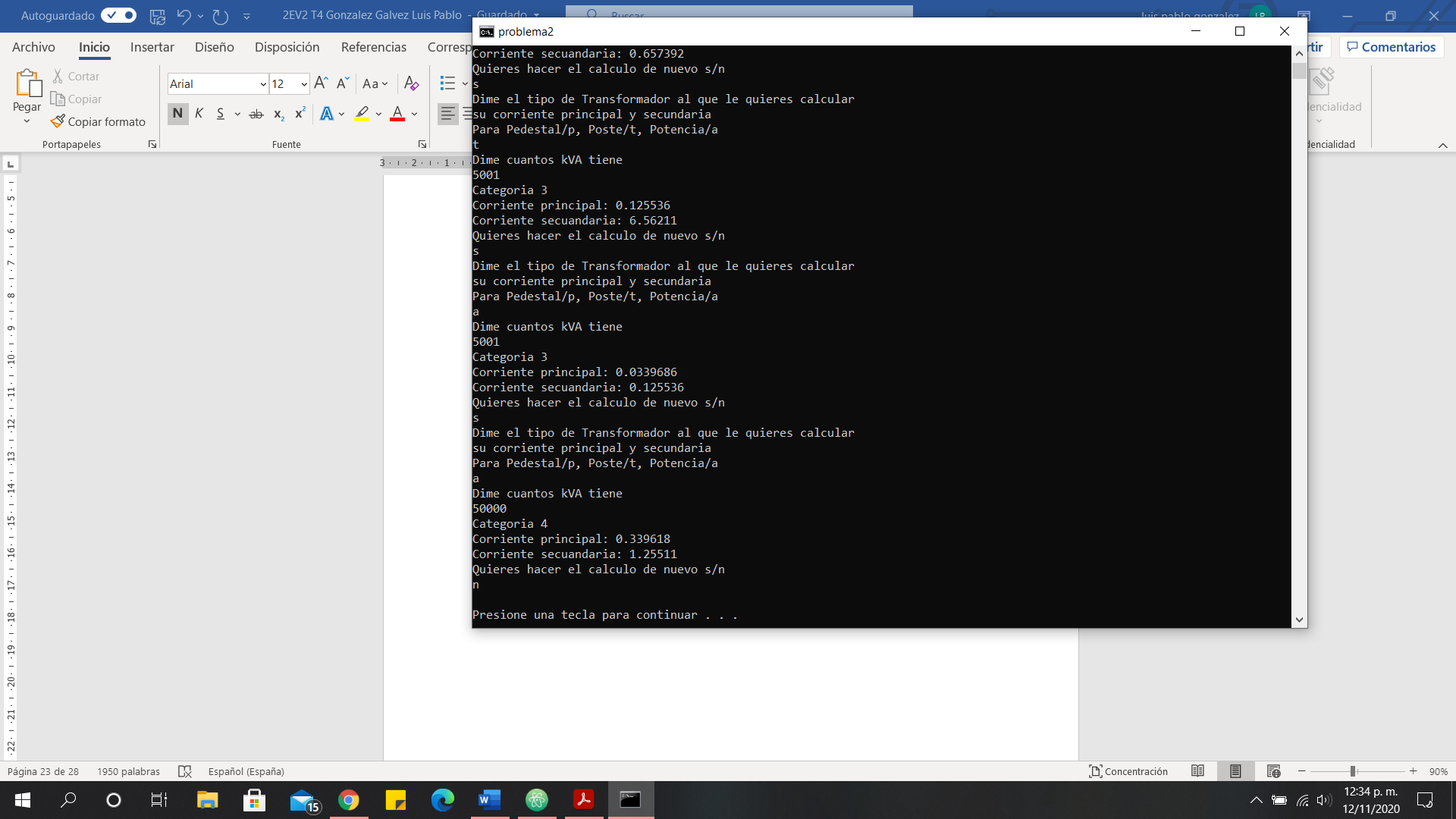
cout<<"Quieres hacer el calculo de nuevo s/n"<<endl;

cin>>s;

}while(s!='n');

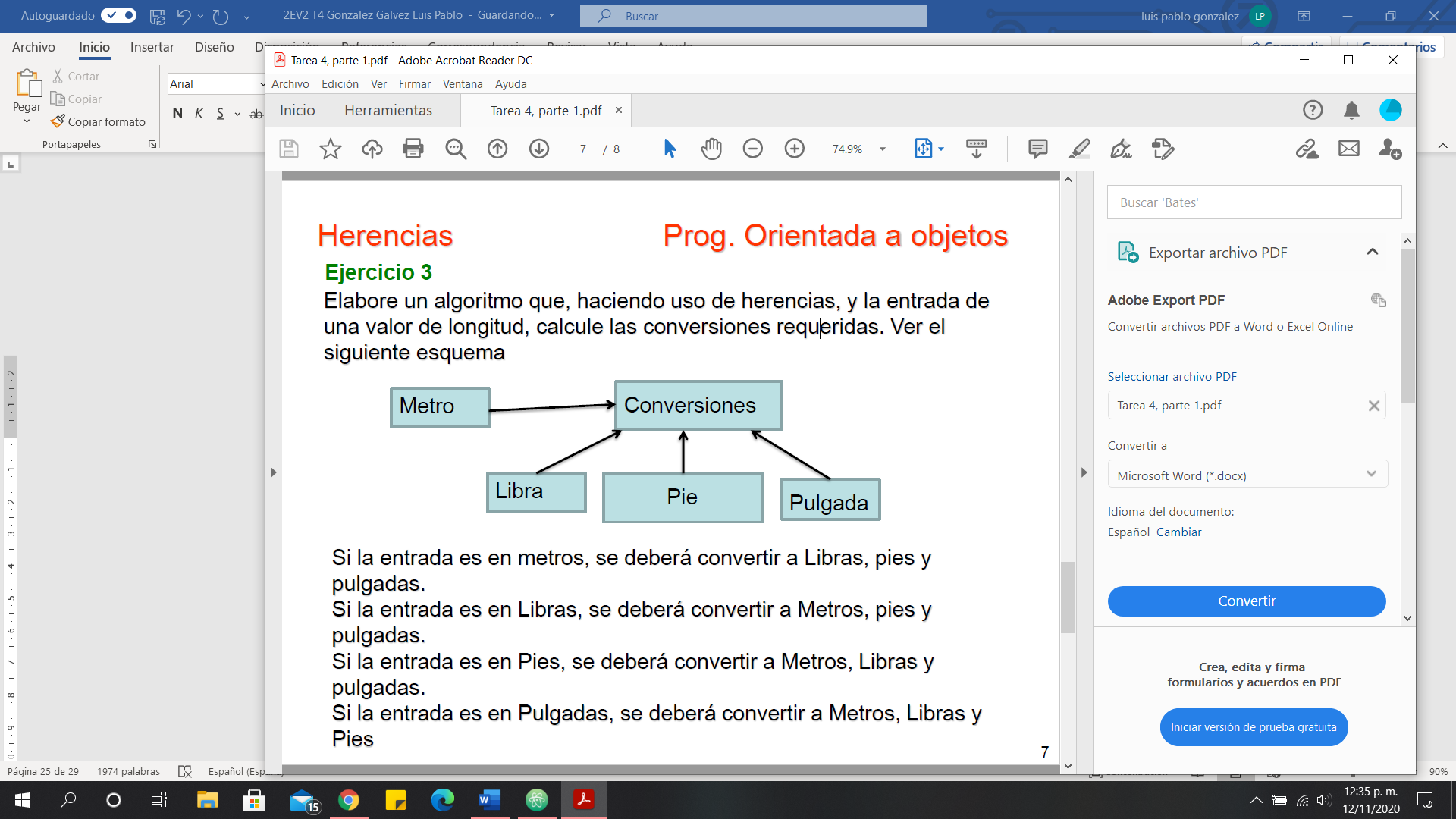
}





**Problema 3:**

Elabore un algoritmo que, haciendo uso de herencias, y la entrada de un valor de longitud, calcule las conversiones requeridas. Ver el siguiente esquema:



Si la entrada es en Metros, se deberá convertir a Libras, Pies y Pulgadas.

Si la entrada es en Libras, se deberá convertir a Metros, Pies y Pulgadas.

Si la entrada es en Pies, se deberá convertir a Metros, Libras y Pulgadas.

Si la entrada es en Pulgadas, se deberá convertir a Metros, Libras y Pies.

**Syntaxis:**

////////////////////////

//Librerias

/////////////////////////

#include <iostream>

#include <cstdlib>

#include <cmath>

#include <ctime>

#include <string.h>

using namespace std;

/////////////////////////

//Definiciones

/////////////////////////

#define tam\_max 10

/////////////////////////

//Clases

/////////////////////////

class Conversion{

public:

float m\_pi=3.28;

float m\_pul=39.37;

float m\_li=0.7376;

float li\_pi=0.083333;

float li\_pul=2.036;

float li\_m=0.01152;

float pi\_m=0.3048;

float pi\_pul=12;

float pi\_li=0.083333;

float pul\_pi=0.0833333;

float pul\_m=0.0254;

float pul\_li=2.036;

};

class Metros:Conversion{

public:

float metros\_pies(float m){

float pies;

pies=m\*m\_pi;

return pies;

}

float metros\_pul(float m){

float pul;

pul=m\*m\_pul;

return pul;

}

float metros\_libras(float m){

float libra;

libra=m\*m\_li;

return libra;

}

};

class Pulgadas:Conversion{

public:

float pul\_met(int pul){

float m;

m=pul\*pul\_m;

return m;

}

float pul\_pie(int pul){

float pie;

pie=pul\*pul\_pi;

return pie;

}

float pul\_libras(int pul){

float li;

li=pul\*pul\_li;

return li;

}

};

class Pies:Conversion{

public:

float pie\_met(float pi){

float m;

m=pi\*pi\_m;

return m;

}

float pie\_pul(float pi){

float pul;

pul=pi\*pi\_pul;

return pul;

}

float pie\_libra(float pi){

float li;

li=pi\*pi\_li;

return li;

}

};

class Libras:Conversion{

public:

float libra\_met(float lib){

float m;

m=lib\*li\_m;

return m;

}

float libra\_pie(float lib){

float pie;

pie=lib\*li\_pi;

return pie;

}

float libra\_pul(float lib){

float pul;

pul=lib\*li\_pul;

return pul;

}

};

void todo(Metros b,Pulgadas c,Pies d,Libras e);

/////////////////////////

//Funcion Principal

/////////////////////////

int main(){

Conversion a;

Metros b;

Pulgadas c;

Pies d;

Libras e;

todo(b,c,d,e);

}

void todo(Metros b,Pulgadas c,Pies d,Libras e){

char a;

float m,lib,pi,pul;

char h;

do{

cout<<"Dime que quieres convertir"<<endl;

cout<<"Metros/m, Pies/t, Pulgadas/p, Libras/l"<<endl;

cin>>a;

if(a=='m'){

cout<<"Dime cuantos metros hay:"<<endl;

cin>>m;

cout<<"Pies: "<<b.metros\_pies(m)<<"ft"<<endl;

cout<<"Pulgadas: "<<b.metros\_pul(m)<<"in"<<endl;

cout<<"Libras: "<<b.metros\_libras(m)<<"lb"<<endl;

}

else if(a=='t'){

cout<<"Dime cuantos pies hay:"<<endl;

cin>>pi;

cout<<"Metros: "<<d.pie\_met(pi)<<"m"<<endl;

cout<<"Pulgadas: "<<d.pie\_pul(pi)<<"in"<<endl;

cout<<"Libras: "<<d.pie\_libra(pi)<<"lb"<<endl;

}

else if(a=='p'){

cout<<"Dime cuantos pulgadas hay:"<<endl;

cin>>pul;

cout<<"Metros: "<<c.pul\_met(pul)<<"m"<<endl;

cout<<"Pies: "<<c.pul\_pie(pul)<<"ft"<<endl;

cout<<"Libras: "<<c.pul\_libras(pul)<<"lb"<<endl;

}

else {

cout<<"Dime cuantos libras hay:"<<endl;

cin>>lib;

cout<<"Metros: "<<e.libra\_met(lib)<<"m"<<endl;

cout<<"Pulgadas: "<<e.libra\_pul(lib)<<"in"<<endl;

cout<<"Pies: "<<e.libra\_pie(lib)<<"ft"<<endl;

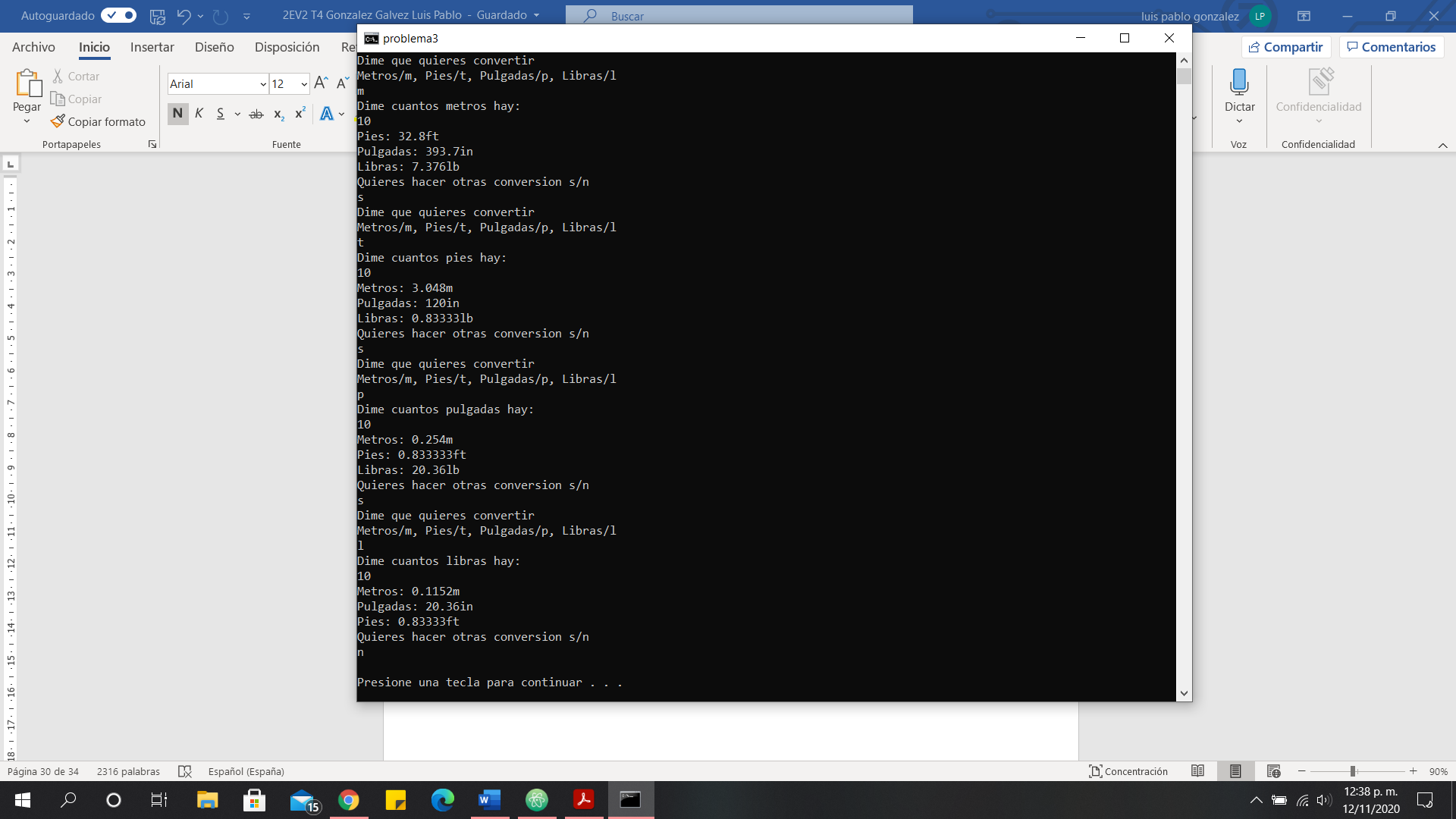
}

cout<<"Quieres hacer otras conversion s/n"<<endl;

cin>>h;

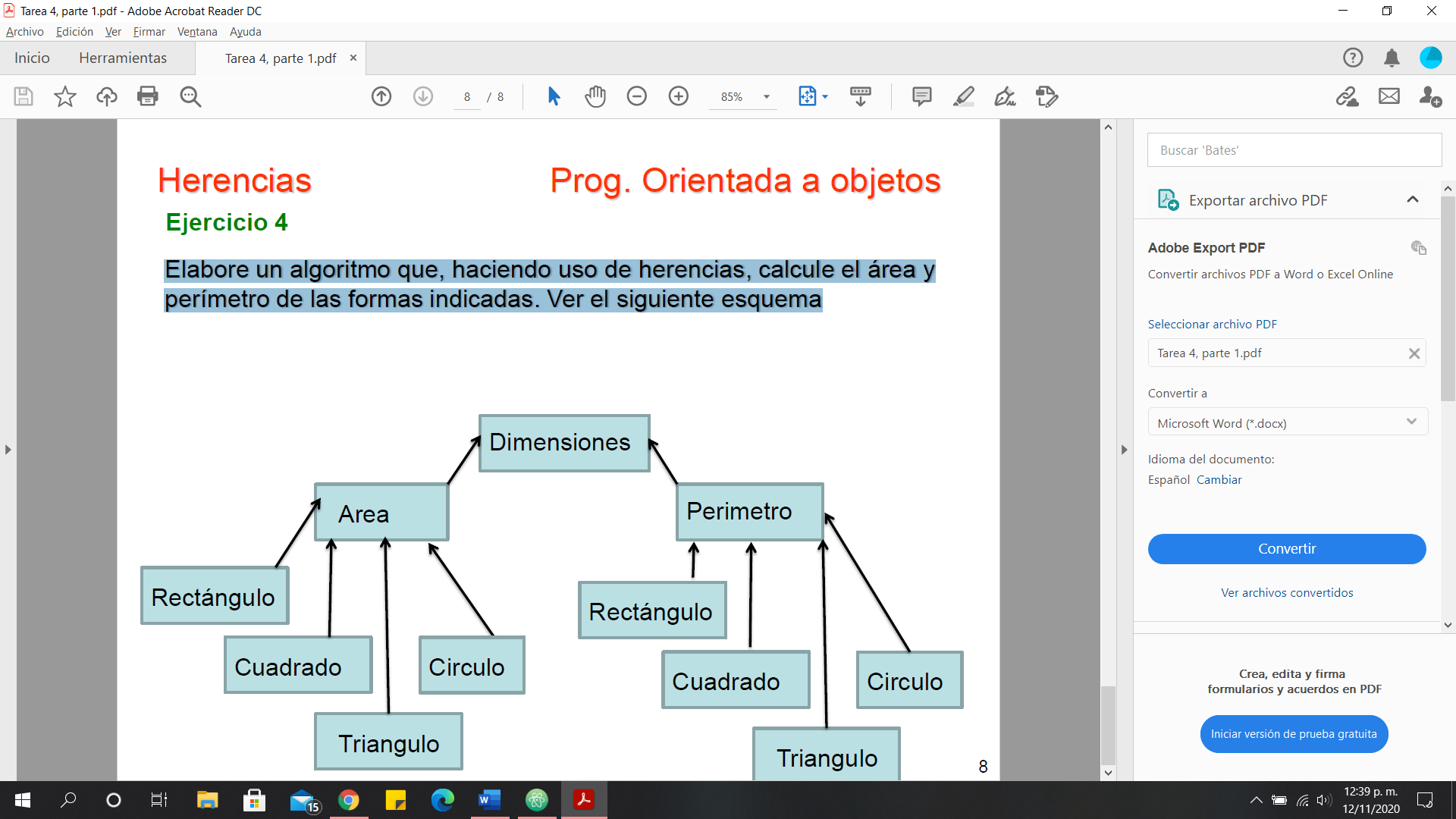
}while(h!='n');

}



**Problema 4:**

Elabore un algoritmo que, haciendo uso de herencias, calcule el área y perímetro de las formas indicadas. Ver el siguiente esquema:



**Syntaxis:**

////////////////////////

//Librerias

/////////////////////////

#include <iostream>

#include <cstdlib>

#include <cmath>

#include <ctime>

#include <string.h>

using namespace std;

/////////////////////////

//Definiciones

/////////////////////////

#define tam\_max 10

/////////////////////////

//Clases

/////////////////////////

class Dimensiones{

public:

float lado=3;

float lado2=2;

float radio=2;

float altura=5;

float pi=3.141592654;

float area;

float perimetro;

float res\_area(){

return area;

}

float res\_peri(){

return perimetro;

}

float valor\_s(){

float s=0;

s=(lado+lado+lado)/2;

return s;

}

void todo\_parametros(){

cout<<"Valor del lado del cuadrado: "<<lado<<endl;

cout<<"Valor del lado del rectangulo: "<<lado<<endl;

cout<<"Valor del otro lado del rectangulo: "<<lado2<<endl;

cout<<"Valor del radio del circulo: "<<radio<<endl;

cout<<"Valor de pi: "<<pi<<endl<<endl;

}

};

class Area:public Dimensiones{

public:

float rectangulo(){

float f=lado\*lado2;

return f;

}

float cuadrado(){

float f=lado\*lado;

return f;

}

float triangulo(){

float s=valor\_s();

float f=sqrt(s\*(s-lado)\*(s-lado)\*(s-lado));

return f;

}

float circulo(){

float f=pi\*radio\*radio;

return f;

}

void todo\_area(){

cout<<"Areas: "<<endl;

area=rectangulo();

cout<<"Rectangulo: "<<res\_area()<<"m^2"<<endl;

area=cuadrado();

cout<<"Cuadrado: "<<res\_area()<<"m^2"<<endl;

area=triangulo();

cout<<"Triangulo: "<<res\_area()<<"m^2"<<endl;

area=circulo();

cout<<"Circulo: "<<res\_area()<<"m^2"<<endl<<endl;

}

};

class Perimetro:public Dimensiones{

public:

float rectangulo(){

float f=lado\*2+lado2\*2;

return f;

}

float cuadrado(){

float f=lado\*4;

return f;

}

float triangulo(){

float f=lado\*3;

return f;

}

float circulo(){

float f=pi\*radio\*2;

return f;

}

void todo\_peri(){

cout<<"Perimetros: "<<endl;

perimetro=rectangulo();

cout<<"Rectangulo: "<<res\_peri()<<"m"<<endl;

perimetro=cuadrado();

cout<<"Cuadrado: "<<res\_peri()<<"m"<<endl;

perimetro=triangulo();

cout<<"Triangulo: "<<res\_peri()<<"m"<<endl;

perimetro=circulo();

cout<<"Circulo: "<<res\_peri()<<"m"<<endl<<endl;

}

};

/////////////////////////

//Funcion Principal

/////////////////////////

int main(){

Dimensiones a;

Area b;

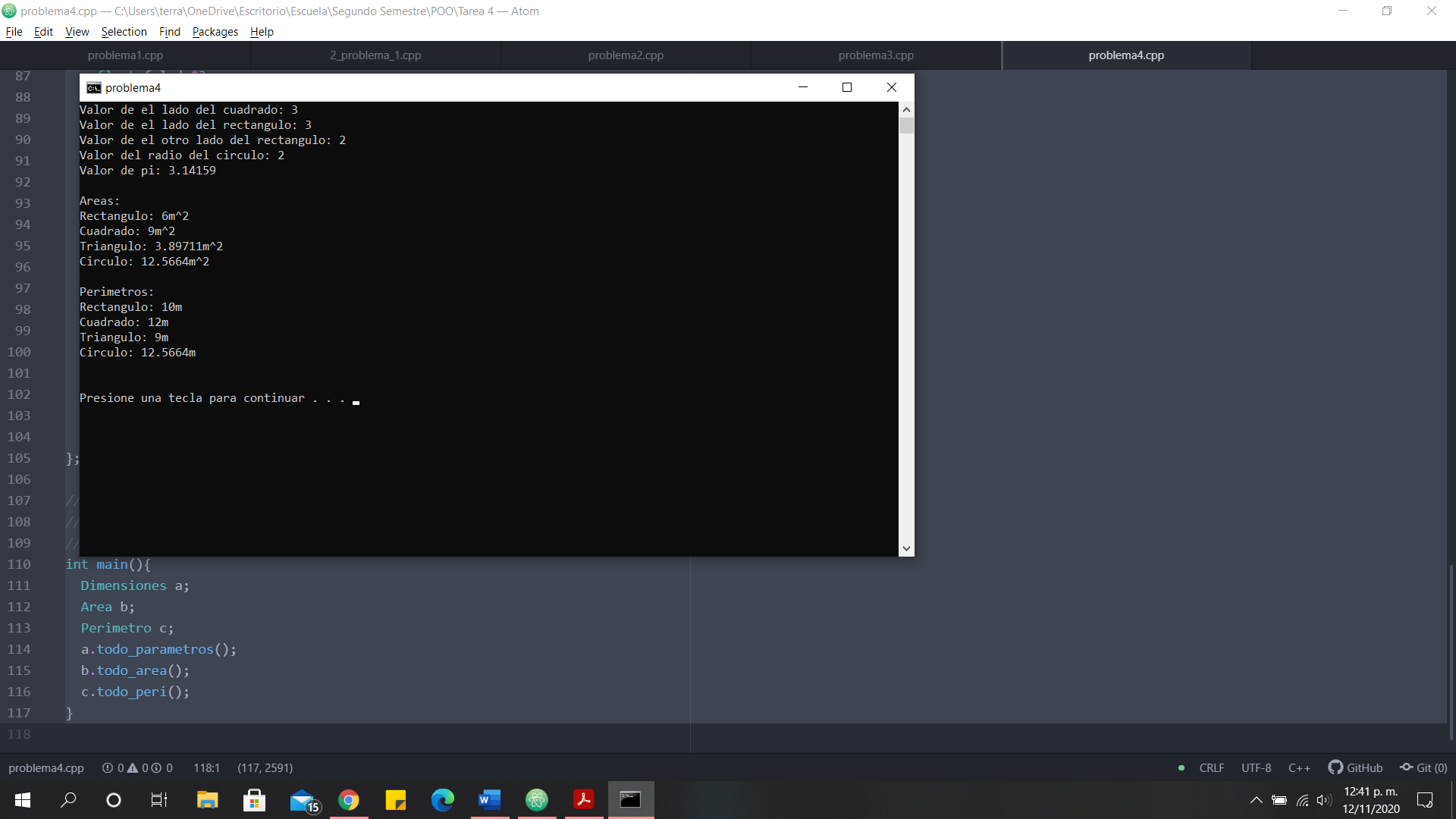
Perimetro c;

a.todo\_parametros();

b.todo\_area();

c.todo\_peri();

}



**Problema 5:**

**Syntaxis:**

**Problema 6:**

**Syntaxis:**

**Problema 7:**

**Syntaxis:**

**Problema 8:**

**Syntaxis:**

**Problema 9:**

**Syntaxis:**

**Problema 10:Syntaxis:**

Conclusiones

Bibliografía:

Concepto de Herencia - Herencia en C++ (Práctica 3). (s. f.). Recuperado 20-11-12, de <https://www.codingame.com/playgrounds/50747/herencia-en-c-practica-3/concepto-de-herencia>