Ejercicios Resueltos – Guía de estudios Nº 7 – POO

Plantillas de función y de clases.

Ejercicio 1
Implemente una función templatizada llamada *Mayor(...)* que reciba dos valores y devuelva el mayor. Compruebe el correcto funcionamiento de la rutina probándola desde un programa cliente con valores de tipo *int* , *float* y *string* .
a. Programe una sobrecarga de la función *Mayor(...)* que reciba un arreglo y el tamaño del mismo. Pruebe la función sobrecargada desde un programa cliente con diversos tipos de datos.
b. Responda: ¿Servirán las funciones anteriores con datos del tipo mostrado en el recuadro? Justifique su respuesta.

```
struct Persona{
string nombre;
string apellido;
 int dni;
};
```

c. Implemente los cambios necesarios, utilizando su propio criterio, para que las funciones desarrolladas funcionen con el tipo de dato del recuadro.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar
//Guia de estudios Nº 7 - Ejercicio Nº 1

#include <iostream>
using namespace std;
struct Persona{//Este es el struct del enunciado
        string nombre;
        string apellido;
        int dni;
};

//Devuelve el Mayor de 2 parámetros
template <typename T>
T Mayor (T a, T b) {
        return (a>b?a:b);//if(a>b)return a else return b;
}
//Envio función de comparación para los tipos definidos por el programador
template <typename T, typename U>
T Mayor (T a, T b, U f) {
        if (f(a,b))
                return a;
        else
                return b;
}

//Item b: Rta. NO porque es un struct y no se puede comparar directamente,
//Item c: especializacion de Clases
//Comparo por el campo apellido entre dos struct
```

```
//Funcion para los struct Persona.
bool MayorApe(Persona A, Persona B){
	return (A.apellido>B.apellido?true:false);
}
//Item a
template<typename T> //T es el tipo generico
T Mayor(T v[], int n) {
	T ma(v[0]);
	for (int i=1; i<n;i++)
		if (v[i]>ma)
			ma=v[i];
	return ma;
}
//Comparo por el campo apellido en un vector de struct
template<typename T, typename U>
T Mayor(T v[],U f, int n) {
	T ma= v[0];
	for (int i=1; i<n;i++)
		if (f(v[i],ma))
			ma=v[i];
	return ma; //Devuelve el struct con el mayor apellido
}

int main(int argc, char *argv[]) {
	//Asigna valores
	int i=15, j=6, k;
	long l=12, m=53, n;
	char a='Q', b='Z', c;
	//llama a la funcion Mayor
	k=Mayor<int>(i,j);
	n=Mayor<long>(l,m);
	c=Mayor<char>(a,b);
	//Muestra resultados
	cout << k << endl;
	cout << n << endl;
	cout << c << endl;
	//Declara e inicializa 2 arreglos de diferentes tipos
	int x[3]= {1,2,3};
	float y[3]= {4.4,51.5,6.6};
	//Muestra por pantalla
	cout<<"Mayor: "<<Mayor(x,3)<<endl;
	cout<<"Mayor: "<<Mayor(y,3)<<endl;
	//Declara e inicializa dos struct tipo persona
	Persona A, B, M;
	A.apellido="Lopez";
	B.apellido="Sanchez";
	//LLama a la funcion Mayor con 2 struct
	M = Mayor(A,B,MayorApe);
	cout<<"Mayor: "<<M.apellido<<endl;
	//Declara e inicializa un arreglo de 3 posiciones, tipo persona
	Persona p[3];
	p[0].apellido="Gutierrez";
	p[1].apellido="Pereira";
	p[2].apellido="Garcia";
	//LLama a la funcion Mayor
```

```
        M = Mayor(p,MayorApe,3);
        cout<<"Mayor: "<<M.apellido<<endl;
        return 0;
}
```

Ejercicio 2

Implemente una función *Clamp(...)* que reciba como parámetros una variable (por referencia) y dos valores indicando los límites superior e inferior para el valor de dicha variable. Si el valor de la variable supera su máximo, este debe ajustarse al máximo valor permitido. De la misma forma si el valor es inferior al mínimo. Pruebe la función templatizada desde un programa cliente.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar
//Guia de estudios Nº 7 - Ejercicio Nº 2

#include <iostream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
//-----------------------------------------------
template <typename T>
T Clamp(T v, T a, T b)//v es un valor, a es el inicio y b es fin del rango
{if (v<a)//Si es menor que el inicio v= a
        v=a;
else
        if (v>b)//Si es mayor que el fin v= b
                v=b;
return v;
}
//-----------------------------------------------
int main(void){
        srand(time(0));//para generar numeros al azar distintos
        //*** con caracteres ***
        char p='a',u='z',d='#';
        cout<<" p='a',u='z',d='#' "<<endl;//muestro en pantalla rango y valor
        d= Clamp(d,p,u);//LLamo a la funcion que verifica que <d> este en rango
        cout<<"d= "<<d<<endl;
        //*** Con entros ***
        int pri=50, ult=100, dato=400;
        cout<<"*****"<<endl<<" pri= 50, ult= 100 "<<endl;//muestro en pantalla rango
        for(int k=0; k<10; k++){//Genero 10 numeros al azar
                dato= rand()%200;
                cout<<"dato= "<<setw(5)<<dato;//Muestro valor original
                dato= Clamp(dato,pri,ult);//LLamo a la funcion con enteros
                cout<<"  -  dato= "<<setw(5)<<dato<<endl;//Muesto valor validado
        }
        return 0;
}
```

Ejercicio 3

Programe una clase templatizada llamada **VectorDinamico** (similar a la de la guía 2). La clase debe poseer:

a. Un constructor que reciba el tamaño inicial del vector, y reserve la memoria necesaria.

b. Un destructor que se encargue de liberar la memoria reservada.

c. Una sobrecarga del operador[] que reciba el número de elemento, devuelva su valor, y permita modificarlo.

d. Modifique o sobrecargue el constructor para que permita generar valores aleatorios con los cuales inicializar las posiciones del arreglo que reserva.

e. Utilice la clase desde un programa cliente creando vectores aleatorios con diversos tipos de datos (int,double,string, etc).

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar
//Guia de estudios Nº 7 - Ejercicio Nº 3
#include <iostream>
#include <cstdlib>
using namespace std;
//Interface ********************************
template<typename T>
class vectorDinamico{
private:
	int tam;
	T *p;
public:
	vectorDinamico(int t);//a)Constructor
	~vectorDinamico();// b)Destructor
	T &operator[](int);//c) Operador []
	void operator+=(T);//d)Operador +=
	int ver_tam(){return tam;};
};
//Implementacion ----------------------------
template <typename T>
vectorDinamico<T>::vectorDinamico(int lar){
	tam= lar;
	p= new T [tam];
	for(int x=0; x < tam; x++)
		*(p+x)= rand()% 27+65;
}
// Item b//Implementacion ----------------------------
template <typename T>
vectorDinamico<T>::~vectorDinamico(){
	delete []p;
}
// Item c//Implementacion ----------------------------
template <typename T>
T & vectorDinamico<T>::operator[](int indice){
	return *(p+indice);
}
// Item d//Implementacion ----------------------------
```

```
template <typename T>
void vectorDinamico<T>::operator+=(T elemento){
	T *aux = new T [tam+1];//Declaro vector auxiliar
	for(int x=0; x < (tam); x++)//Recorro aux asignando los valores de p
		*(aux+x)= *(p+x);
	tam++;//Incremento en 1 el tamaño
	*(aux+tam-1)= elemento;//Asigno en la ultima posicion
	delete []p;//Borro el vector viejo
	p= aux;//Asigno a p el vector auxiliar
}
//*************************************************
int main(int argc, char *argv[]) {
	vectorDinamico<int> v(10);
	int n;
	for(int k= 0; k < v.ver_tam(); k++)
		cout<<v[k]<<", ";
	cout<<endl;
	cout<<"\nIngrese un numero: ";		cin>> n;
	v += n;				//agrego con el += sobrecargado
	cout<<"Arreglo agregando el "<<n<<endl;
	for(int k= 0; k < v.ver_tam(); k++)
		cout<<v[k]<<", ";
	cout<<endl<<endl<<endl;
	//************************************
	vectorDinamico<char> v2(10);
	char n2;
	for(int k= 0; k < v2.ver_tam(); k++)
		cout<<v2[k]<<", ";
	cout<<"\nIngrese un char: ";	cin>> n2;
	v2 += n2;
	cout<<"Arreglo agregando el "<<n2<<endl;
	for(int k= 0; k < v2.ver_tam(); k++)
		cout<<v2[k]<<", ";
	return 0;
}}
```

Ejercicio 4

Desarrolle una clase templatizada llamada **ManejadorArchivo** que posea métodos y atributos para manipular un archivo binario que contenga registros del tipo de dato especificado por el parámetro. La clase debe poseer métodos para:

a. Abrir un archivo binario y cargar los registros en memoria.
b. Obtener el registro en una posición especificada por el usuario.
c. Modificar el registro en una posición determinada.
d. Actualizar la información del archivo con los cambios.
e. Utilice la clase desde un programa cliente para leer los registros escritos en el archivo binario.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar
```

```
//Guia de estudios Nº 7 - Ejercicio Nº 4
#include <iostream>
#include <cstring>
#include <fstream>
#include <vector>
using namespace std;
//-------------------------------------------------------
struct persona{//El codigo en el main esta configurado para este tipo de datos
	char apeynomb[50];
	int diaNac, mesNac, anioNac;
};
//-------------------------------------------------------
//Para usar este registro hay que recodificar el main, la clase funciona igual
struct animal{
	char apeynomb[50];
	int anioNac;
};
//-------------------------------------------------------
template<class T>
class ManejadorArchivo{
	string nombreArchi;//Aqui se guardará el nombre físico del archivo
	vector<T> p;//Este arreglo cargará todos los registros del disco a memoria
public:
	ManejadorArchivo(string nArch){nombreArchi=nArch;}//Constructor
	void carga();//Metodo que permitira la carga de los registros al vector
	T buscar(string cad);//Buscara el nombre (cad) en el arreglo
	T VerReg(int x){return p[x];};//Devuelve registro en posicion x
	void guardar();//Permitira guardar el arreglo en disco;
	bool eliminar(string cad);//Elimina un nombre del arreglo y guarda en disco el vector
midificado
	void agregar(T per);//Agrega un registro en el arreglo y en el archivo en disco.
	int ver_cantidad(){return p.size();}//Devuelve la cantidad de registros o posiciones del
arreglo
	};
//Interface ----------------------------------------------------------------------------------
template<class T>
void ManejadorArchivo<T>::carga(){
	fstream archi_in(nombreArchi,ios::binary|ios::in);//Declaro el objeto archi_in para leer
datos del archivo
	if(!archi_in.is_open()){//Si no se abrio informo del error
		cout<<"Error en apertura de archivo"<<endl;
		//cout<<"Se creara un archivo vacio"<<endl;
		fstream archi(nombreArchi,ios::binary|ios::in|ios::out|ios::trunc);
		archi.clear();//Limpio
		archi.close();//Cierro el archivo
	}else{
		archi_in.seekg(0,ios::end);//Voy con el puntero de lectura al final del archivo
		int total = archi_in.tellg()/sizeof(T);//Calculo el tamaño
		T x;//Declaro el objeto x de tipo T
		archi_in.seekg(0,ios::beg);//Vuelvo el puntero de lectura al principio
		for(int i=0;i<total;i++) {
			archi_in.read((char*)&x, sizeof(T));
```

```
                        //cout<<cantidad<<"- "<<p[cantidad].apeynomb<<endl;//Voy
mostrando lo que leo
                        p.push_back(x);//Agrego el objeto x al final del arreglo p[]
                }//Fin del for de lectura de archivo
                archi_in.close();//Cierro el archivo
        }
}
//-----------------------------------------------------------------
template<class T>
T ManejadorArchivo<T>::buscar(string cad){
        T aux;//Declaro aux de tipo T
        //Es necesario que el tipo de datos T tenga el campo apeynomb
        for(int i=0;i<p.size();i++) {
                if(strcmp(p[i].apeynomb,cad.c_str())==0){//Compara nombres y si lo encuentra
                        aux= p[i];//Devuelvo el struct completo
                        return aux;
                }
        }//Si pasa es porque no lo encontro.
        strcpy(aux.apeynomb,"No Encontrado");//Asigno el mensaje y devuelvo el struct
        return aux;
}
//-----------------------------------------------------------------
template<class T>
void ManejadorArchivo<T>::guardar(){
        //Abro el archivo para sobreescribirlo con el arreglo de struct
        fstream archi_out(nombreArchi,ios::binary|ios::out|ios::trunc);
        if(!archi_out.is_open()){
                cout<<"Error en apertura de archivo"<<endl;
                //getchar();
        }else{
                archi_out.seekg(0,ios::beg);//Vuelvo al principio
                for(int i=0;i<p.size();i++) {
                        archi_out.write((char*)&p[i], sizeof(T));
                }
                archi_out.close();//Cierro el archivo
        }
}
//-----------------------------------------
template<class T>
bool ManejadorArchivo<T>::eliminar(string cad){
        int i=0;
        while(i< p.size()) {
                if(strcmp(p[i].apeynomb,cad.c_str())==0){//Compara nombres
                        break;//Si lo encuentra sale
                }else
                  i++;//Si no lo encuentra, i avanza una posicion
        }//fin del while
        if (i<p.size()){//Si salio del while e i es menor al tamaño del arreglo
                p.erase(p.begin()+i);//quiere decir que lo encontro y lo elimina
                guardar();//Guardo en disco el arreglo modificado
        }else {
                cout<<"NO Encontrado "<<cad.c_str()<<endl;
```

```
	}
	return 0;
}
//-------------------------------------------
template<class T>
void ManejadorArchivo<T>::agregar(T per){
	fstream archi_out(nombreArchi,ios::binary|ios::out|ios::ate);
	if(!archi_out.is_open()){
		cout<<"Error en apertura de archivo"<<endl;
		//getchar();
	}else{
		//archi_out.seekp(0,ios::end);//al final
		archi_out.write((char*)&per, sizeof(T));
		p.push_back(per);
	}
	archi_out.close();
}

int menu(){
	int op;
	cout<<"Manejador de Archivo - Configurado para Personas"<<endl;
	cout<<"1) Salir"<<endl;
	cout<<"2) Ver archivo"<<endl;
	cout<<"3) Buscar Persona"<<endl;
	cout<<"4) Guardar datos en Archivo"<<endl;
	cout<<"5) Eliminar Persona"<<endl;
	cout<<"6) Ver cantidad de personas en el archivo"<<endl;
	cout<<"7) Agregar Persona"<<endl;
	cout<<endl<<"Elija una Opcion: ";cin>>op;
	return op;
}

int main(int argc, char *argv[]) {
	ManejadorArchivo<persona> A("archivo.bin");
	A.carga();
	int opcion=0;
	while(opcion != 1){
		opcion = menu();
		switch (opcion){
		case 1: {A.guardar(); return 0;};
		break;
		case 2:{int c=0;
		for(c=0;c < A.ver_cantidad(); c++){
			cout<<"Nombre : "<<A.VerReg(c).apeynomb<<", Fecha Nac: "

	<<A.VerReg(c).diaNac<<"/"<<A.VerReg(c).mesNac<<"/"<<A.VerReg(c).anioNac<<endl;
		}
		};break;
		case 3:{persona dato,encontro;
		cout<<"Ingrese el nombre y apellido a buscar: "; cin>>dato.apeynomb;
		encontro= A.buscar(dato.apeynomb);
		if(strcmp(encontro.apeynomb, "No Encontrado") == 0){
```

```
                        cout<<"Nombre no encontrado."<<endl;
                }else{
                        cout<<"Nombre: "<<encontro.apeynomb<<endl;
                        cout<<"Fecha de Nacimiento:
"<<encontro.diaNac<<"/"<<encontro.mesNac
                                <<"/"<<encontro.anioNac<<endl;
                };break;

                }//switch
                case 4:{A.guardar();};break;
                case 5:{persona dato,encontro;
                cout<<"Ingrese el nombre y apellido a Eliminar: ";
                cin.ignore();gets(dato.apeynomb);
                encontro= A.buscar(dato.apeynomb);
                if(strcmp(encontro.apeynomb,"No Encontrado")== 0){
                        cout<<"Nombre no encontrado."<<endl;
                }else{
                        cout<<"Nombre: "<<encontro.apeynomb<<endl;
                        cout<<"Fecha de Nacimiento:
"<<encontro.diaNac<<"/"<<encontro.mesNac
                                <<"/"<<encontro.anioNac<<endl;
                        A.eliminar(encontro.apeynomb);
                        cout<<"ELIMINADO"<<endl;
                }};break;
                case 6:{cout<<"Cantidad de personas en al archivo: "<<A.ver_cantidad()<<endl;
                };break;
                case 7:{ persona dato;
                cin.ignore();
                cout<<"Ingrese Apellido y Nombre: "; cin.getline(dato.apeynomb,50);
                cout<<"Ingrese dia de Naciminento: "; cin>>dato.diaNac;
                cout<<"Ingrese mes de Naciminento: "; cin>>dato.mesNac;
                cout<<"Ingrese Anio de Naciminento: "; cin>>dato.anioNac;
                A.agregar(dato);};break;
                }//del switch
        }//del while
        return 0;
}
```

Cuestionario:

1. ¿Qué es la programación genérica? ¿Cómo se implementa en C++?
2. ¿Cuál es la diferencia entre una plantilla de clase y una plantilla de función?
3. ¿A qué se denomina instanciación o especialización de una plantilla?
4. ¿Cómo puede saberse de antemano si la instanciación de una plantilla con un determinado tipo de dato arrojará errores?
5. ¿Por qué la implementación de una clase templatizada no puede separarse en dos archivos .cpp y .h?

Ejercicios adicionales

Ejercicio 1
Diseñe e implemente una clase templatizada llamada Rect que permita representar un rectángulo. La clase debe poseer métodos para obtener el alto y ancho del rectángulo. Implemente los métodos que considere necesarios para la inicialización o carga de datos. Cree un programa cliente que instancie un rectángulo cuyas coordenadas sean enteros y otro cuyas coordenadas sean de tipo double.

```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 7 - Ejercicio Adicional  Nº 1
#include <iostream>
using namespace std;
template<class T>
class rectangulo{
private:
	T base, altura;
public:
	rectangulo(T, T);//Constructor con 2 parametros
	T Area();//Calcula y devuelve el area del rectangulo
	T Perimetro();//Calcula y devuelve el perímetro del rectangulo
	T ver_base();//Devuelve el valor de <base>
	T ver_altura();//Devuelve el valor de <altura>
};
//Implementacion constructor
template<class T>
rectangulo<T>::rectangulo(T b, T a):base(b),altura(a){}
//Implementacion Area
template<class T>
T rectangulo<T>::Area(){
	return base*altura;}
//Implementacion Perimetro
template<class T>
T rectangulo<T>::Perimetro(){
	return 2*base+2*altura;}
//Implementecion ver_base
template<class T>
T rectangulo<T>::ver_base(){
	return base; }
//Implementecion ver_altura
template <class T>
T rectangulo<T>::ver_altura(){
	return altura;
}
//*****************************************************
int main(int argc, char *argv[]) {
	rectangulo<int> *R1= new rectangulo<int>(3,4);
	rectangulo<double> *R2= new rectangulo<double>(3.3,4.4);
	cout<<"R1 Base: "<<R1->ver_base()<<" - Altura: "<<R1->ver_altura()<<endl;
	cout<<"R1 Area:  "<<R1->Area()<<endl;
	cout<<"R1 Perim: "<<R1->Perimetro()<<endl;
```

```
	cout<<"R2 Base: "<<R2->ver_base()<<" - Altura: "<<R2->ver_altura()<<endl;
	cout<<"R2 Area:  "<<R2->Area()<<endl;
	cout<<"R2 Perim: "<<R2->Perimetro()<<endl;
	delete R1;//Un delete por cada new
	delete R2;//Un delete por cada new
	return 0;
}
```

Ejercicio 2

Desarrolle una función Templatizada Intercambia(...) que reciba dos variables por referencia e intercambie sus valores. Pruebe la función desde un programa cliente con al menos dos tipos de dato distintos. Sobrecargue la función para que reciba 2 arreglos dinámicos de igual tamaño y los intercambie.

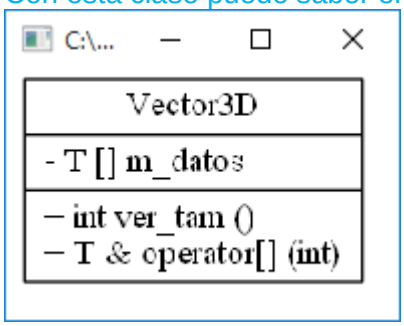
```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 7 - Ejercicio Adicional  Nº 2
#include <iostream>
#include <vector>
using namespace std;
//intercambia 2 parámetros
template <typename T>
void intercambia (T &a, T &b) {
	T aux;
	aux = a;
	a = b;
	b= aux;
}
//intercambia 2 arreglos
template <typename T>
void intercambia (vector<T> &a, vector<T> &b) {
	T aux;
	for(int i=0; i<a.size(); i++){
		aux = a[i];
		a[i] = b[i];
		b[i] = aux;
	}
}

//*********************************
int main(int argc, char *argv[]) {
	int i=15, j=6;
	cout <<"i: "<<i<<" - j: "<<j<< endl;
	intercambia(i,j);
	cout <<"intercambia"<<endl<<"i: "<<i<<" - j: "<<j<< endl;
	char a='Q', b='Z';
	cout <<"a: "<<a<<" - b: "<<b<< endl;
	intercambia(a,b);
	cout <<"intercambia"<<endl<<"a: "<<a<<" - b: "<<b<< endl;
	vector<int> M = {1,2,3,4,5};
	vector<int> N = {6,7,8,9,10};
	for(int i=0; i<(int)M.size(); i++){
		cout<<"M["<<i<<"]= "<<M[i]<<" - N["<<i<<"]= "<<N[i]<<endl;
	}
```

```
        cout<<"Intercambia: "<<endl;
        intercambia(M,N);
        for(int i=0; i<(int)M.size(); i++){
                cout<<"M["<<i<<"]= "<<M[i]<<" - N["<<i<<"]= "<<N[i]<<endl;
        }
        return 0;
}
```

Ejercicio 3

Implemente una clase genérica vector estático similar a la del ejercicio 3, pero utilizando un vector estático, cuya longitud será el argumento de la plantilla. ¿Qué ventajas tendría el uso de dicha clase sobre el uso de un arreglo estático común?

Con esta clase puedo saber el largo de un arreglo estático de cualquier tipo de datos.




```
//© Programación Orientada a Objetos- F.I.C.H. - U.N.L.
//Autor: Prof. Gerardo Sas.
//sitio web: http://e-fich.unl.edu.ar/
//Guia de estudios Nº 7 - Ejercicio Adicional  Nº 3

#include <iostream>
#include <cstdlib>
using namespace std;
//vector estático, cuya longitud será el argumento de la plantilla
template<typename T,int N=3>//Puede inicializarse por defecto
class Vector3D {
        T m_datos[N];
public:
        T & operator[](int i);//Sobrecarga operador []
        int ver_tam();//Devuelve tamaño (size)
};
//Implementacion - observe la sobrecarga del operador []
template<typename T,int N>
T & Vector3D<T,N>::operator[](int i) {
        return m_datos[i];
}
//Metodo que devuelve el tamaño (size) del arreglo
template<typename T,int N>
int Vector3D<T,N>::ver_tam(){
        return sizeof(*this)/sizeof(T);
}
//**********************************
int main(int argc, char *argv[]) {
        Vector3D <int,5> m;
```

```
	Vector3D <string,15> w;
	for(int i= 0; i<m.ver_tam(); i++){
		m[i]= rand()%10;
		cout<<"m["<<i<<"]= "<<m[i]<<endl;
	}
	cout<<"cantidad: "<<m.ver_tam();

	cout<<endl<<"**************"<<endl;
	for(int i= 0; i<w.ver_tam(); i++){
		w[i]= (char)(rand()%16+97);
		cout<<"w["<<i<<"]= "<<w[i]<<endl;
	}
	cout<<"cantidad: "<<w.ver_tam();
	return 0;
	}
```