EJERCICIO DE REPASO DEL TEMA 1

Deduce que métodos y funciones hay que implementar para que se puede ejecutar el siguiente main( ).

```
int main() {
  socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
  socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
  socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
  c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
  c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
  socio c4(c2);                  //crea el socio c4 con los mismos datos que el socio c2
  c2.ver(); c4.ver(); cout << endl;
  if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
    cout << c4 << " == " << c2 << endl;
  c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
  cout << c1 <<", "<< c2 <<", "<< c3 << endl;
  c3.setnombre("Ana");
  c2 = c2;
  c2 = c1;
  c1.setnombre("Luis");
  socio T[3];
  T[0]=c1; T[1]=c2; T[2]=c3;
  for(int i=0; i<3; i++) {
     cout << "T[" << i << "] objeto "
          << (long)&T[i] <<":" << T[i] << endl;
  }
  system("PAUSE"); return EXIT_SUCCESS;
}
```

Implementa los métodos necesarios y realiza el ejercicio considerando que:

a) para guardar el nombre usamos un array estático `char nombre[30];`

b) para guardar el nombre usamos un array dinámico `char *nombre;`
   i. Elimina el constructor por defecto y vea lo que ocurre
   ii. Elimina el constructor de copia y vea lo que ocurre
   iii. Elimina el operator= y vea lo que ocurre
   iv. Elimina el destructor y vea lo que ocurre

c) Tanto para el apartado a) como para el apartado b), declara el atributo edad constante y el objeto c4 constante (corrige los errores si los hay)
   ¿Qué pasa con el operador de asignación (operator=)?

d) Supongamos que queremos que cada socio creado tenga un número de socio único. Indica los cambios que hay que realizar en el programa para que, cada vez que se cree un socio, el programa le asigne automáticamente un número diferente a cada socio.
   1. Considera que cuando se crea una copia de un objeto socio la copia debe ser idéntica al original (puede haber 2 socios con mismo número cuando se copia o se asigna)
   2. Considera que cuando se crea una copia de un objeto socio la copia debe tener un número diferente(no puede haber 2 socios con mismo número cuando se copia o se asigna)

e) Si además queremos que todos los socios paguen la misma cuota de abono… ¿qué más cambios habrá que realizar? Implementa métodos que permitan consultar y modificar la cuota de abono común a todos los socios.

SOLUCION:

a) para guardar el nombre usamos un array estático `char nombre[30];`

Como la clase no tiene memoria dinámica, no hace falta programar un destructor (ya que no hay memoria dinámica que liberar), ni el constructor de copia, ni el operador de asignación, ya que los que genera de oficio el compilador (hacen una copia binaria de los datos) funcionan perfectamente al no haber punteros.

```
#include <iostream> //cin, cout
#include <cstring>  //strcpy, strlen
#include <cstdlib>  //system

using namespace std;

class socio {
  char nombre[30];
  int edad;
public:
    socio(const char *nombre, int e);
    socio() { strcpy(nombre,""); edad=0; } //necesario para poder crear array de socios
    void setnombre(const char *cad) { strcpy(nombre, cad);  } //si cad tiene mas de 30 no cabría...
    void ver() const { cout << nombre << "(" << edad << ")\n"; }
    bool operator==(const socio &s); //lo gris es opcional
    friend ostream& operator<<(ostream &s, const socio &soc);
};
ostream& operator<<(ostream &s, const socio &soc);

socio::socio(const char *nombre, int e) {
  strcpy(this->nombre, nombre);
  edad=e;
}

bool socio::operator==(const socio &s) { //lo gris es opcional
  if (strcmp(nombre, s.nombre)==0 && edad==s.edad)
    return true;
  else
    return false;
}

ostream& operator<<(ostream &s, const socio &soc) {
  s << soc.nombre << "(" << soc.edad << ")";
  return s;
}

int main() {
 socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
 socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
 socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
 c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
 c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
 socio c4(c2);                  //crea el socio c4 con los mismos datos que el socio c2
 c2.ver(); c4.ver(); cout << endl;
 if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
  cout << c4 << " == " << c2 << endl;
 c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
 cout << c1 <<", "<< c2 <<", "<< c3 << endl;
 c3.setnombre("Ana");
 c2 = c2;
 c2 = c1;
 c1.setnombre("Luis");
 socio T[3];    //para crear array de objetos socio la clase debe tener constructor sin parametros
 T[0]=c1; T[1]=c2; T[2]=c3;
 for(int i=0; i<3; i++) {
   cout << "T[" << i << "] objeto "
        << (long)&T[i] <<":" << T[i] << endl;
 }
 system("PAUSE"); return EXIT_SUCCESS;
}
```

b) para guardar el nombre usamos un array dinámico `char *nombre;`

Como la clase tiene memoria dinámica, hay que programar:

- **Un destructor (para liberar la memoria dinámica reservada con new en el constructor).**
- **Un constructor de copia y un operador de asignación, ya que los que genera de oficio el compilador (hacen una copia binaria de los datos) no sirven, ya que harían que los punteros de los atributos de los objetos que se copian o asignan apuntaran a la misma zona de memoria, con lo que los cambios que se hagan en un objeto afectan al otro objeto asignado o copiado al compartir (apuntar) a la misma zona de memoria.**

```
#include <iostream> //cin, cout
#include <cstring> //strcpy, sgtrlen
#include <cstdlib> //system

using namespace std;

class socio {
  char *nombre;
  int edad;
public:
    socio(const char *nombre, int e);
    socio() { this->nombre = new char[1]; strcpy(nombre,""); edad=0; }
    ~socio() { delete [] nombre; }       //destructor
    socio(const socio& s);               //constructor de copia
    socio& operator=(const socio &s);    //operador de asignacion
    void setnombre(const char *cad) {
      if (strlen(cad)>strlen(nombre)) {
        delete [] nombre;
        nombre = new char[strlen(cad)+1];
      }
      strcpy(nombre, cad);
    }
    void ver() const { cout << nombre << "(" << edad << ")\n"; }
    bool operator==(const socio &s); //lo gris es opcional
    friend ostream& operator<<(ostream &s, const socio &soc);
};
ostream& operator<<(ostream &s, const socio &soc);

socio::socio(const char *nombre, int e) {
  this->nombre = new char[strlen(nombre)+1];
  strcpy(this->nombre, nombre);
  edad=e;
}

socio::socio(const socio& s) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  edad=s.edad;
}

socio& socio::operator=(const socio &s) {
  if(this != &s) {
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    edad=s.edad;
  }
  return *this;
}

bool socio::operator==(const socio &s) { //lo gris es opcional
  if (strcmp(nombre, s.nombre)==0 && edad==s.edad)
    return true;
  else return false;
}

ostream& operator<<(ostream &s, const socio &soc) {
  s << soc.nombre << "(" << soc.edad << ")";
  return s;
}
```

```
int main() {
 socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
 socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
 socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
 c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
 c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
 socio c4(c2);                  //crea el socio c4 con los mismos datos que el socio c2
 c2.ver(); c4.ver(); cout << endl;
 if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
  cout << c4 << " == " << c2 << endl;
 c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
 cout << c1 <<", "<< c2 <<", "<< c3 << endl;
 c3.setnombre("Ana");
 c2 = c2;
 c2 = c1;
 c1.setnombre("Luis");
 socio T[3];     //para crear array de objetos socio la clase debe tener constructor sin parametros
 T[0]=c1; T[1]=c2; T[2]=c3;
 for(int i=0; i<3; i++)
  cout << "T[" << i << "] objeto "
       << (long)&T[i] <<":" << T[i] << endl;
 system("PAUSE"); return EXIT_SUCCESS;
}
```

c) Tanto para el apartado a) como para el apartado b), declara el atributo edad constante y el objeto c4 constante (corrige los errores si los hay)
¿Qué pasa con el operador de asignación (operator=)?

Si el objeto c4 es constante → solo puede invocar métodos const

`c4.ver();` **← ver() tiene que ser const para que pueda ser invocado por c4**

`if (c4 == c2)` **← operator= =() tiene que ser const para que pueda ser invocado por c4**

Si el atributo edad es constante en los constructores hay que inicializarlo en la zona de los inicializadores y en el operador de asignación no se va a poder modificar su valor

```
#include <iostream> //cin, cout
#include <cstring> //strcpy, sgtrlen
#include <cstdlib> //system

using namespace std;

class socio {
  char *nombre;
  const int edad;
public:
    socio(const char *nombre, int e);
    socio():edad(0){ this->nombre = new char[1]; strcpy(nombre,"");  }
    ~socio() { delete [] nombre; }      //destructor
    socio(const socio& s);              //constructor de copia
    socio& operator=(const socio &s);   //operador de asignacion
    void setnombre(const char *cad) {
      if (strlen(cad)>strlen(nombre)) {
        delete [] nombre;
        nombre = new char[strlen(cad)+1];
      }
      strcpy(nombre, cad);
    }
    void ver() const { cout << nombre << "(" << edad << ")\n"; }
    bool operator==(const socio &s) const;
    friend ostream& operator<<(ostream &s, const socio &soc);
};
ostream& operator<<(ostream &s, const socio &soc);

socio::socio(const char *nombre, int e): edad(e) {
  this->nombre = new char[strlen(nombre)+1];
  strcpy(this->nombre, nombre);
  //edad=e;
}

socio::socio(const socio& s):edad(s.edad) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  //edad=s.edad;
}

socio& socio::operator=(const socio &s) { //podemos copiar todo excepto la edad ya que es contantes
  if(this != &s) {                        //y no se puede modificar
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    //edad=s.edad; //no se puede cambiar el valor ya que el atributo edad es constante
  }
  return *this;
}

bool socio::operator==(const socio &s) const {
  if (strcmp(nombre, s.nombre)==0 && edad==s.edad)
    return true;
  else
    return false;
}
```

```
ostream& operator<<(ostream &s, const socio &soc) {
  s << soc.nombre << "(" << soc.edad << ")";
  return s;
}

int main() {
 socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
 socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
 socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
 c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
 c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
 const socio c4(c2);            //crea el socio c4 con los mismos datos que el socio c2
 c2.ver(); c4.ver(); cout << endl;
 if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
  cout << c4 << " == " << c2 << endl;
 c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
 cout << c1 <<", "<< c2 <<", "<< c3 << endl;
 c3.setnombre("Ana");
 c2 = c2;
 c2 = c1;
 c1.setnombre("Luis");
 socio T[3];    //para crear array de objetos socio la clase debe tener constructor sin parametros
 T[0]=c1; T[1]=c2; T[2]=c3;
 for(int i=0; i<3; i++)
  cout << "T[" << i << "] objeto "
       << (long)&T[i] <<":" << T[i] << endl;
 system("PAUSE"); return EXIT_SUCCESS;
}
```

d) Supongamos que queremos que cada socio creado tenga un número de socio único. Indica los cambios que hay que realizar en el programa para que, cada vez que se cree un socio, el programa le asigne automáticamente un número diferente a cada socio

1. Considera que cuando se crea una copia de un objeto socio, la copia debe ser idéntica al original (puede haber 2 socios con mismo número cuando se copia o se asigna)
2. Considera que cuando se crea una copia de un objeto socio, la copia debe tener un número diferente (no puede haber 2 socios con mismo número cuando se copia o se asigna)

```
#include <iostream> //cin, cout
#include <cstring> //strcpy, sgtrlen
#include <cstdlib> //system

using namespace std;

class socio {
  char *nombre;
  const int edad;
  static int n; //atributo estatico para llevar la cuenta de los socios que se van creando
  int numsocio; //a cada socio creado le asignamos un numsocio igual al valor del contador n
public:
    socio(const char *nombre, int e);
    socio():edad(0){ this->nombre = new char[1]; strcpy(nombre,""); n++; numsocio=n;  }
    ~socio() { delete [] nombre; }      //destructor
    socio(const socio& s);              //constructor de copia
    socio& operator=(const socio &s);   //operador de asignacion
    void setnombre(const char *cad) {
      if (strlen(cad)>strlen(nombre)) {
        delete [] nombre;
        nombre = new char[strlen(cad)+1];
      }
      strcpy(nombre, cad);
    }
    void ver() const { cout << nombre << "(" << edad << ")\n"; }
    bool operator==(const socio &s) const;
    friend ostream& operator<<(ostream &s, const socio &soc);
};
ostream& operator<<(ostream &s, const socio &soc);

int socio::n=0; //inicializo a 0 el atributo estatico que cuento los socios que voy creando

socio::socio(const char *nombre, int e): edad(e) {
  this->nombre = new char[strlen(nombre)+1];
  strcpy(this->nombre, nombre);
  //edad=e;
  n++;     //cada vez que creo un socio incremento n
  numsocio=n; //cada socio tendrá un valor distinto
}

socio::socio(const socio& s):edad(s.edad) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  //edad=s.edad;
  n++;
  numsocio=n;
}

socio& socio::operator=(const socio &s) {
  if(this != &s) {
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    //edad=s.edad;
    //numsocio no lo cambio para que no sea igual
  }
  return *this;
}

bool socio::operator==(const socio &s) const {
  if (strcmp(nombre, s.nombre)==0 && edad==s.edad)
    return true;
  else    return false;
}
```

```
socio::socio(const socio& s):edad(s.edad) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  //edad=s.edad;
  numsocio=s.numsocio;
}

socio& socio::operator=(const socio &s) {
  if(this != &s) {
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    //edad=s.edad;
    numsocio=s.numsocio;
  }
  return *this;
}

bool socio::operator==(const socio &s) const {
  if (strcmp(nombre, s.nombre)==0 &&
      edad==s.edad && numsocio==s.numsocio)
    return true;
  else
    return false;
}
```

```
ostream& operator<<(ostream &s, const socio &soc) {
  s << soc.nombre << "(" << soc.edad << ")" << " n° " << soc.numsocio;
  return s;
}

int main() {
 socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
 socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
 socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
 c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
 c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
 const socio c4(c2);            //crea el socio c4 con los mismos datos que el socio c2
 c2.ver(); c4.ver(); cout << endl;
 if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
  cout << c4 << " == " << c2 << endl;
 c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
 cout << c1 <<", "<< c2 <<", "<< c3 << endl;
 c3.setnombre("Ana");
 c2 = c2;
 c2 = c1;
 c1.setnombre("Luis");
 socio T[3];    //para crear array de objetos socio la clase debe tener constructor sin parametros
 T[0]=c1; T[1]=c2; T[2]=c3;
 for(int i=0; i<3; i++)
  cout << "T[" << i << "] objeto "
       << (long)&T[i] <<":" << T[i] << endl;
 system("PAUSE"); return EXIT_SUCCESS;
}
```

e) Si además queremos que todos los socios paguen la misma cuota de abono… ¿qué más cambios habrá que realizar? Implementa métodos que permitan consultar y modificar la cuota de abono común a todos los socios.

```
#include <iostream> //cin, cout
#include <cstring> //strcpy, sgtrlen
#include <cstdlib> //system

using namespace std;

class socio {
  char *nombre;
  const int edad;
  static int n; //atributo estatico para llevar la cuenta de los socios que se van creando
  int numsocio; //a cada socio creado le asignamos un numsocio igual al valor del contador n
  static float cuota; //atributo estatico común a todos los socios
public:
    socio(const char *nombre, int e);
    socio():edad(0){ this->nombre = new char[1]; strcpy(nombre,""); n++; numsocio=n;  }
    ~socio() { delete [] nombre; }      //destructor
    socio(const socio& s);              //constructor de copia
    socio& operator=(const socio &s);   //operador de asignacion
    void setnombre(const char *cad) {
      if (strlen(cad)>strlen(nombre)) {
        delete [] nombre;
        nombre = new char[strlen(cad)+1];
      }
      strcpy(nombre, cad);
    }
    void ver() const { cout << nombre << "(" << edad << ")\n"; }
    bool operator==(const socio &s) const;
    static getCuota() { return cuota; }
    static setCuota(float valor) { cuota=valor; }
    friend ostream& operator<<(ostream &s, const socio &soc);
};
ostream& operator<<(ostream &s, const socio &soc);

int socio::n=0; //inicializo a 0 el atributo estatico que cuento los socios que voy creando
float socio::cuota=50; //cuota de abono de 50 euros común a todos los socios

socio::socio(const char *nombre, int e): edad(e) {
  this->nombre = new char[strlen(nombre)+1];
  strcpy(this->nombre, nombre);
  //edad=e;
  n++;     //cada vez que creo un socio incremento n
  numsocio=n; //cada socio tendrá un valor distinto
}

socio::socio(const socio& s):edad(s.edad) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  //edad=s.edad;
  n++;
  numsocio=n;
}

socio& socio::operator=(const socio &s) {
  if(this != &s) {
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    //edad=s.edad;
    //numsocio no lo cambio para que no sea igual
  }
  return *this;
}

bool socio::operator==(const socio &s) const {
  if (strcmp(nombre, s.nombre)==0 && edad==s.edad)
    return true;
  else    return false;
}
```

```
socio::socio(const socio& s):edad(s.edad) {
  nombre = new char[strlen(s.nombre)+1];
  strcpy(nombre, s.nombre);
  //edad=s.edad;
  numsocio=s.numsocio;
}

socio& socio::operator=(const socio &s) {
  if(this != &s) {
    delete [] nombre;
    nombre = new char[strlen(s.nombre)+1];
    strcpy(nombre, s.nombre);
    //edad=s.edad;
    numsocio=s.numsocio;
  }
  return *this;
}

bool socio::operator==(const socio &s) const {
  if (strcmp(nombre, s.nombre)==0 &&
      edad==s.edad && numsocio==s.numsocio)
    return true;
  else
    return false;
}
```

```
ostream& operator<<(ostream &s, const socio &soc) {
  s << soc.nombre << "(" << soc.edad << ")" << " n° " << soc.numsocio;
  return s;
}

int main() {
 socio c1("Juan Jose", 27);     //socio de nombre "Juan Jose" y edad 27 años
 socio c2("Paloma", 20);        //socio de nombre "Paloma" y edad 20 años
 socio c3 = c2;                 //el socio c3 tendrá el mismo nombre y edad que c2
 c2.setnombre("Paloma Maria"); //cambia el nombre del socio c2
 c1.ver(); c2.ver(); c3.ver(); cout<<endl; //muestra por pantalla nombre y edad de socios c1, c2 y c3
 const socio c4(c2);            //crea el socio c4 con los mismos datos que el socio c2
 c2.ver(); c4.ver(); cout << endl;
 if (c4 == c2)                  //pregunta si los socios c4 y c2 se llaman igual y tienen la misma edad
  cout << c4 << " == " << c2 << endl;
 c3 = c2 = c1; //c2=c1; c3=c2; //los socios c3, c2 y c1 se van a llamar igual y tendrán la misma edad
 cout << c1 <<", "<< c2 <<", "<< c3 << endl;
 c3.setnombre("Ana");
 c2 = c2;
 c2 = c1;
 c1.setnombre("Luis");
 socio T[3];    //para crear array de objetos socio la clase debe tener constructor sin parametros
 T[0]=c1; T[1]=c2; T[2]=c3;
 for(int i=0; i<3; i++)
  cout << "T[" << i << "] objeto "
       << (long)&T[i] <<":" << T[i] << endl;
 system("PAUSE"); return EXIT_SUCCESS;
}
```