

# 10 (DIEZ)

/\*La clase LISTA1 esta formada por un puntero a la clase NODO (privado) y las funciones asociadas.

La clase NODO esta compuesta por los miembros públicos NOM (string de 20) y el puntero SIG (NODO \*), más las funciones asociadas.

La lista contiene nombres que no guardan ningún orden y aparecen repetidos correspondiendo cada nodo a un voto en un certamen de popularidad.

La clase LISTA2 esta formada por un puntero a la clase CANDIDATO (privado) y las funciones asociadas.

La clase CANDIDATO esta compuesta por los miembros públicos NOM (string de 20), CANT(int) y el puntero SIG (CANDIDATO \*), más las funciones asociadas.

La lista2 contiene nombres (que no se repiten) y la cantidad de votos recibidos .

Se pide construir la función void AGREGA(LISTA1& , LISTA2&), tal que agregue a LISTA2 los votos de LISTA1. Cada NODO que reciba incrementará 1 voto en el NODO2 correspondiente,

o darlo de alta en caso de no existir ese nombre en LISTA2.

Indicar que característica debe tener la función AGREGA() para acceder a ambas listas. \*/

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <iostream>
```

```
using namespace std ;
```

```
class LISTA2;
```

```
//CLASES DE LAS LISTAS CON LOS VOTOS
```

```
class NODO{
    public:
        NODO * SIG;
        char NOM[20];
};
```



```

class LISTA1{
    private:
        NODO * INICIO;
    public:
        friend void AGREGAR (LISTA1 &, LISTA2 &);
        LISTA1(); // SOLO TEST!!!!
};

LISTA1 :: LISTA1 ()
{
    int l ;
    NODO * P ;
    char NOM[][20] = { "FELIPE" , "LOLA" , "LAURA" , "FELIPE" , "ALEJO" ,
"MARK" , "CASI" , "APRUEBO" , "ANA" , "ANA" , "ANA" , "ANA" ,
"LOLA" , "LOLA" , "MIRTA" , "SUSANA" , "FELIPE" ,
"ENZO" , "APRUEBO" , "APRUEBO" } ;
    INICIO = NULL ;
    for ( l=0 ; l<20 ; l++ ) {
        P = new NODO ;
        strcpy ( P->NOM , NOM [ l ] ) ;

        P->SIG = INICIO ;
        INICIO = P ;
    }
}

```

//CLASES DE LAS LISTAS CON LOS CANDIDATOS

```


class CANDIDATO{
    public:
        CANDIDATO * SIG;
        int CANT;
        char NOM[20];
};

```

```

class LISTA2{
    private:
        CANDIDATO * INICIO;
        CANDIDATO * BUSCAR(char *);
};

```



```

public:
    void AGREGAR_CANDIDATO (char *);
    friend void AGREGAR (LISTA1 &, LISTA2 &);
    void MIRAR (); // SOLO TEST!!!!

};

//BUSCO EL CANDIDADO EN LA LISTA PARA SABER SI TENGO QUE
//AGREGARLO O YA EXISTE
CANDIDATO * LISTA2::BUSCAR ( char * S_CAND )
{
    CANDIDATO * PLC ;
    PLC = INICIO ;

    while (PLC) {
        if ( strcmp(PLC->NOM,S_CAND)==0 )
            return PLC ;
        PLC = PLC->SIG ;
    }
    return NULL ;
}

void LISTA2::AGREGAR_CANDIDATO (char * CAND_NOM){

    CANDIDATO * PC;



    PC = new CANDIDATO();
    strcpy(PC->NOM, CAND_NOM);
    PC->CANT = 1;

    PC->SIG = INICIO ;
    INICIO = PC ;

}

void LISTA2 :: MIRAR ()
{
    CANDIDATO * PIPI ;

```



```
PIPI = INICIO ;
cout << "\n\n\n";
while ( PIPI ) {
    printf ( "    %-10s%5d" , PIPI->NOM , PIPI->CANT) ;
    PIPI = PIPI->SIG ;
}
getch();
}
```

```
void AGREGAR (LISTA1 &, LISTA2 &);
```

```
int main(){

    LISTA1 LISTA_VOTOS;
    LISTA2 LISTA_CANDIDATOS;

    AGREGAR (LISTA_VOTOS, LISTA_CANDIDATOS);
    LISTA_CANDIDATOS.MIRAR();

}
```

```
//FUNCION QUE AGREGAR LOS VOTOS DE LA PRIMER LISTA EN LA
SEGUNDA LISTA
```

```
void AGREGAR (LISTA1 & LV, LISTA2 & LC){
```

```
    NODO * PN = LV.INICIO;
    CANDIDATO * PC = LC.INICIO;
```

```
    while(PN){
        PC = LC.BUSCAR(PN->NOM);
        if(!PC){
            LC.AGREGAR_CANDIDATO(PN->NOM);
        }else{
            PC->CANT++;
        }
        PN = PN->SIG;
    }
```



} }