L'exemple du projet spectacle en Classes

1 Introduction

Ici le but est reprendre un sujet connu et de voir comment les classes s'imbriquent entre elles. Ici vous êtes dans la peau d'un gérant d'une salle, et ainsi elle contient des sièges et des reservations. Il est à noter ici que l'algorithmique n'est pas le point pédagogique et donc l'attention n'est pas donnée au fait de trouver si des places sont libres ou non, en fin de ligne ou non,... Il manque également beaucoup de fonctions ici car il ne s'agit pas réellement de réaliser ce projet mais de donner une notion d'imbrication de classes.
Attention de nombreuses notions sont ici importantes.

- Destructeur (Lorsqu'il y a des pointeurs)
- Classes contenant des pointeurs vers d'autres classes
- Tableaux de pointeurs (Pointeurs de pointeurs)
- Attention la classe Reservation utilise une donnée static que nous n'avons pas encore vu dans ce cours

2 La classe Salle

"Salle.h"

```
#include "Siege.h"
#include "Reservation.h"


#ifndef Salle_h
#define Salle_h

class Salle{

private:
    string Nom;  //Une salle a un nom
    int lignes; //une salle a un nombre de lignes
    int colonnes; //et un nombre de colonnes
    Siege * ListeSiege; //Une liste de sièges
    Reservation ** ListeReservation; //Une liste de pointeurs de reservations
    int nb_reservations; //Un nombre de reservations


public:

    Salle(string NomS,int l, int c); //Constructeur avec nom ligne collones (pas de constructeur vide
    ~Salle(); //Destructeur a cause des listes de pointeurs
    void Affiche(); //Afichage
    void Affiche_Siege_nb(); //Affiche le numero d'un siège
    void Add_Reservation(string Nom,int l,int c,int nb_Sieges); //Ajoute rune reservation avec une posision x et y puis un
        nombre de sièges contigus
    Reservation& get_reservation(int res_num); //Renvoyer une reservation a partir un d'un numero de reservation
    Siege& get_Siege(int Siege_num); // Renvoie un siège associé à un numero

};
```

"Salle.cpp"

```
Salle::Salle(string NomS, int l, int c)
{ Nom=NomS;
    lignes=l;
    colonnes=c;
    nb_reservations=0;
    ListeSiege=new Siege[l*c];//obligaiton d'un constructeur par defaut de Siege
    for (int i=0;i<l*c;i++)
    {(*(ListeSiege+i)).set_number(i+1);}  //Remplir les numeros des sièges
    ListeReservation =  new Reservation*[c*l];
    for(int i=0;i<c*l;i++) //Ici ce n'st  pas propre mais ca simplifie les choses on aurait pu faire une allocation à chaque
        ajout comme dans les reservations pour la liste de sièges.
    { *(ListeReservation+i)=NULL;}

}


void Salle::Affiche()
{   cout<<"Salle:"<<Nom<<endl;
    for(int i=0;i<lignes;i++)
    {
        for(int j=0;j<colonnes;j++)
        {
            (*(ListeSiege+j+i*colonnes)).Affiche();//on utilise l'affichage de siege directement

        }
        cout<<endl; //pour faire un affichage ligne colonne.
    }
}


void Salle::Affiche_Siege_nb()  //Affiche la salle avec les numeros de sièges
{   cout<<"Salle:"<<Nom<<endl;
    for(int i=0;i<lignes;i++)
    {
        for(int j=0;j<colonnes;j++)
        {
           cout<< (*(ListeSiege+j+i*colonnes)).get_number()<< " "; //On utilise la fonction siege pour avoir son numero

        }
        cout<<endl;
    }
}


Reservation& Salle::get_reservation(int res_num)  //Renvoie une reservation. Le passage par référence ou par pointeur est
    necessaire afin qu'il ne crée pas un object supplémentaire pour le renoiv. On veut renvoyer la reference de la
    reservation trouvée (notion avancée)
{
    return(**(ListeReservation+res_num));
}

void Salle::Add_Reservation(string Nom,int l,int c,int nb_Sieges) //Ajoute une reservation
{

    *(ListeReservation+nb_reservations)= new Reservation(Nom,ListeSiege+(l-1)*colonnes+(c-1),nb_Sieges); //On iutilise
        directement le constructeur car ListeReservation est un pointeur de pointeurs... chaque case du tableau est un
        pointeur qu'on peut allouer directement en utilisant le constructeur.
    nb_reservations=nb_reservations+1;
}

Siege& Salle::get_Siege(int Siege_num)  //renvoyer le siège associé au numero
{
    return  *(ListeSiege+Siege_num-1);

}

Salle::~Salle()
{

    for(int i=0; i<lignes*colonnes;i++)
    {if (*(ListeReservation+i)!=NULL) //si une reservation pointe sur autre chose que null
    {delete *(ListeReservation+i);} //on la désalloue
    }
    delete [] ListeReservation; //on désalloue le pointeur de pointeur
    delete [] ListeSiege; //on désalloue la liste de sieges
}
```

3 La classe Siege

"Siege.h"

```
#include <string.h>

#ifndef Siege_h
#define Siege_h

class Siege{

private:
    int number;    //Un siège a une numero
    bool reserved; //Dit si il est reservé
public:

    Siege(); //Constructeur Vide
    Siege(int number0); //Constructeur avec un numero
    bool get_reserved(); //Savoir si il est reservé
    int get_number(); //connaitre son numero
    void set_reserved(); //le reserver
     void set_reserved(bool res); //surcharge pour la reserve (pour le déreserver);
    void set_number(int number0); //mettre un numero à un siege
    void Affiche(); ////Afficher


};
```

"Siege.cpp"

```
Siege::Siege()
{number=0; reserved=0;}

Siege::Siege(int number0)   //constructeur avec un numero
{number=number0;
 reserved=0;}

bool Siege::get_reserved()
{return reserved;}

int Siege::get_number()
{return number;}

void Siege::set_reserved()
{reserved=1;}

void Siege::set_number(int number0)
{number=number0;}


void Siege::set_reserved(bool res)
{reserved=res;}


void Siege::Affiche() //Affiche X si reserve et 0 sinon
{if(reserved) {cout<<'X';}
    else{cout<<'0';}
}
```

4 La classe Reservation

"Reservation.h"

```
#include <string.h>
#include "Siege.h"
#ifndef Reservation_h
#define Reservation_h


class Reservation{


private:

string Nom;    //Nom auquel est attribué la reservartion
int nb_Sieges;  //nombre de sièges de la réservation
int num_reservation; //Numero de la réservation
Siege **Liste_Sieges; // Sièges vers lesquels pointe la réservation
static int nb_reservation; //static du nombre de reservations (notion avancée)

public:

Reservation(); //constructeur par défaut (ne sera pas utilisé ici
Reservation(string UnNom, Siege *UnTableauDeSieges,int nb_SiegesP); //Crée une reservation avec un nom un pointeur de siège
    et un nombre de sièges
int get_nb_reservation(); //récupérer le nombre de réservations
void Add_Siege(Siege & UnSiege); //Ajouter un siège à une réservation
void Affiche(); //Afficher une réservation
~Reservation(); //Détruire une réservation
};
```

"Reservation.cpp"

```
int Reservation::nb_reservation=0;  //initialisation d'un ombnre static

Reservation::Reservation()
{
    Nom ="";
    nb_Sieges=0;
    num_reservation=0;
    Liste_Sieges=NULL;
    nb_reservation=nb_reservation+1;
}

int Reservation::get_nb_reservation()
{
    return(nb_reservation);
}


Reservation::Reservation(string UnNom, Siege *UnTableauDeSieges,int nb_SiegesP)
{
    Nom =UnNom;
    nb_Sieges=nb_SiegesP;
    nb_reservation=nb_reservation+1;
    num_reservation=nb_reservation;
    Liste_Sieges= new Siege *[nb_SiegesP]; //allouer le tableau de pointeurs de sièges
    for(int i=0;i <nb_SiegesP;i++)   //faire pointer le tableau vers les sièges réservés
    { *(Liste_Sieges+i)= UnTableauDeSieges+i;
    (*(Liste_Sieges+i))->set_reserved(1); //reserver le siège
    }
}


void Reservation::Add_Siege(Siege & UnSiege)  //ajouter un siège
{
    if(UnSiege.get_reserved())  //regarer si il est déjà réservé
    {cout<<"Siege "<<UnSiege.get_number()<<  "Deja reserve"<<endl;
    }
    else{     //sinon
    nb_Sieges=nb_Sieges+1;  //ajouter un siège à la reservation
    UnSiege.set_reserved();  //reserver le siège

    //Attention il va falloir désallouer la précédent liste de siège pour en faire une nouvelle
    Siege** Liste_Sieges_dummy= new Siege *[nb_Sieges];  //créer un dummy avec un siège en plus que liste_sièges (nb_Siege a
        été incrémenté avant .
    for(int i=0;i<nb_Sieges-1;i++)
    {
       *(Liste_Sieges_dummy+i) = *(Liste_Sieges+i);  //copier termes à terms
    }
    *(Liste_Sieges_dummy+nb_Sieges-1) =&UnSiege; //ajouter le siège
   delete [] Liste_Sieges; //Effacer la liste de sièges
    Liste_Sieges=Liste_Sieges_dummy; //faire pointer la liste de siège vers le nouveau siège
    }
}
void Reservation::Affiche() //Afichage
{
    cout<<"Reservation numero: "<< num_reservation<<endl;
    cout<<"Au Nom de "<< Nom<<endl;
    cout<<"Avec les sièges ";
    for(int i=0;i<nb_Sieges;i++)
    {cout<<(*(Liste_Sieges+i))->get_number()<<"\t";} //affichage des sièges
    cout<<endl;
}

Reservation::~Reservation() //Destructeur qui décrémente la valeur static et efface la liste de Sieges
{
nb_reservation=nb_reservation-1;
    for(int i=0;i<nb_Sieges;i++)
    {(*(Liste_Sieges+i))->set_reserved(0);}//déreserver les sièges associés
   if (Liste_Sieges!=NULL)
   {delete [] Liste_Sieges;
   }
}
```

5 "Main.cpp

```cpp
#include <iostream>
using namespace std;
#include "Siege.h"
#include "Salle.h"

int main(int argc, const char * argv[]) {


    Salle S("tutu",5,10); //Je crée un salle de 5 par 10
    S.Affiche(); //Je l'affiche (tous les sièges sont libres)
    S.Affiche_Siege_nb();  //J'affiche les numeros des sièges
    S.Add_Reservation("Toto",3,4, 4); //je reserve pour Toto 3eme ligen 4eme colonne 4 places. La reservation s'applique à
        une salle
    S.Add_Reservation("blabla",1,1, 2); //je reserve pour blbla 1ere ligne 1ere colonne 2 places
    S.Add_Reservation("fifi",5,1,3); //je reserve pour Fifi 5eme ligen 1eme colonne 3 places

    (S.get_reservation(0)).Affiche(); //J'affiche la reservation 0
     S.get_reservation(1).Affiche(); //J'affiche la reservation 1
     S.get_reservation(2).Affiche(); //J'affiche la reservation 2
    S.Affiche(); //J'affiche la salle a nouveau

    S.get_reservation(2).Add_Siege(S.get_Siege(1)); //J'ajoute à la réservation 2 pour le tableau le siège 1 qui est déja
        réservé
    S.get_reservation(2).Add_Siege(S.get_Siege(8)); //j'ajoute le siège 8 à la reservation 2
    S.Affiche(); //J'affiche la salle
    S.get_reservation(2).Affiche(); //J'affiche la réservation


    return 0;
}
```

Execution

```
Salle:tutu
0000000000
0000000000
0000000000
0000000000
0000000000
Salle:tutu
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
Reservation numero: 1
Au Nom de Toto
Avec les sièges 24  25  26  27
Reservation numero: 2
Au Nom de blabla
Avec les sièges 1   2
Reservation numero: 3
Au Nom de fifi
Avec les sièges 41  42  43
Salle:tutu
XX00000000
0000000000
000XXXX000
0000000000
XXX0000000
Siege 1Deja reserve
Salle:tutu
XX00000X00
0000000000
000XXXX000
0000000000
XXX0000000
Reservation numero: 3
Au Nom de fifi
Avec les sièges 41  42  43  8
Program ended with exit code: 0
```