République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Carthage

ECOLE NATIONALE D'INGENIEURS DE CARTHAGE

Rapport Projet Programmation Orientée Objet

Réalisé par :

Mekni Jihed
Dia Modou
Info 1D

Encadrement :

Madame Nouira Sana

Nom du projet :

Gestion d'un tournoi de danse

Année Universitaire : 2023 - 2024

Table des matières

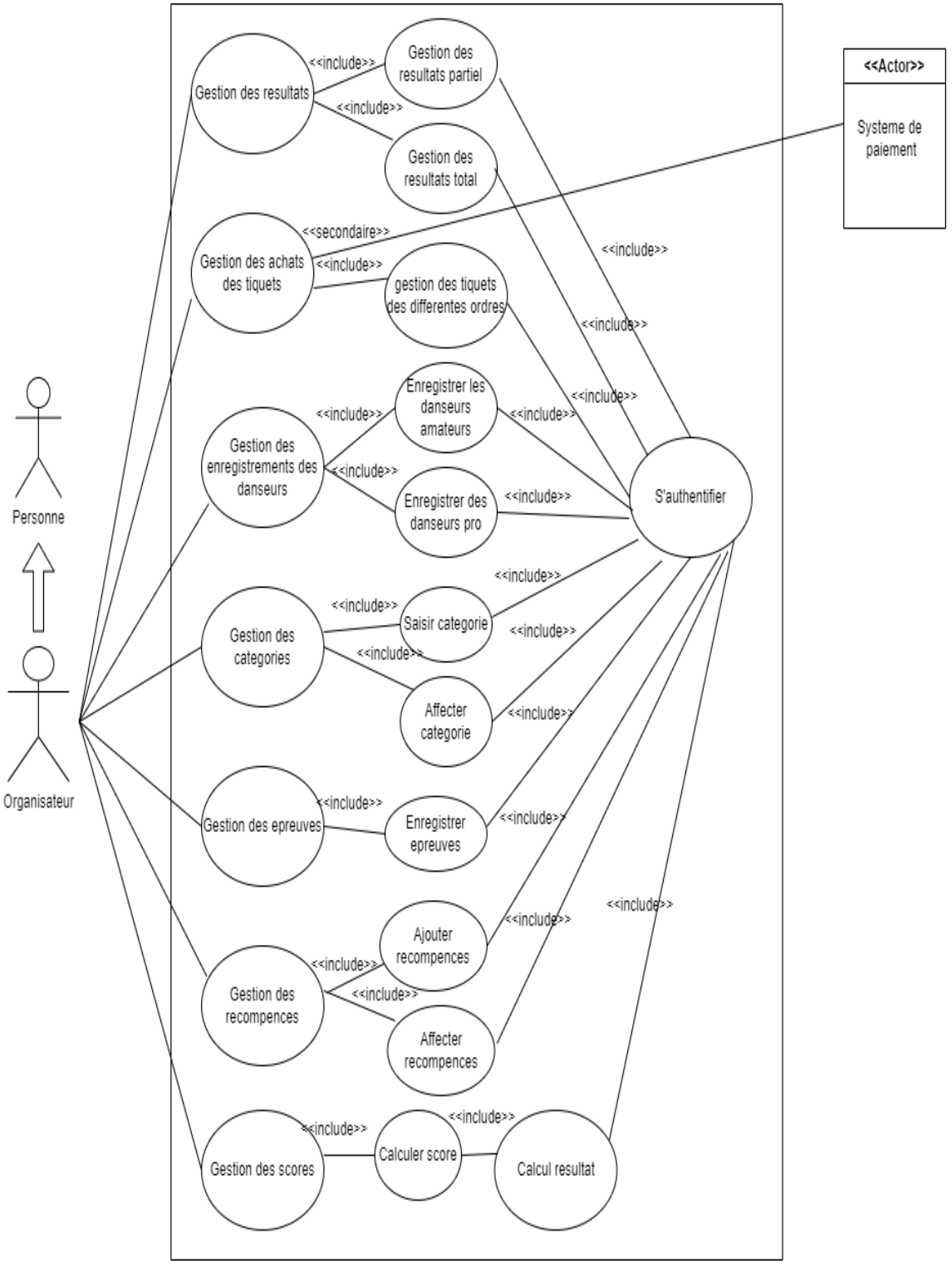
1 Introduction

L'objectif principal de ce projet est de concevoir et de mettre en œuvre un système informatisé capable de gérer différents aspects des tournois de danse, notamment l'enregistrement des danseurs, l'attribution des scores, la gestion des résultats et la remise des récompenses. L'application vise à simplifier et à automatiser les tâches administratives associées à l'organisation d'un tournoi de danse, tout en offrant aux organisateurs un moyen efficace de suivre et d'analyser les performances des participants.

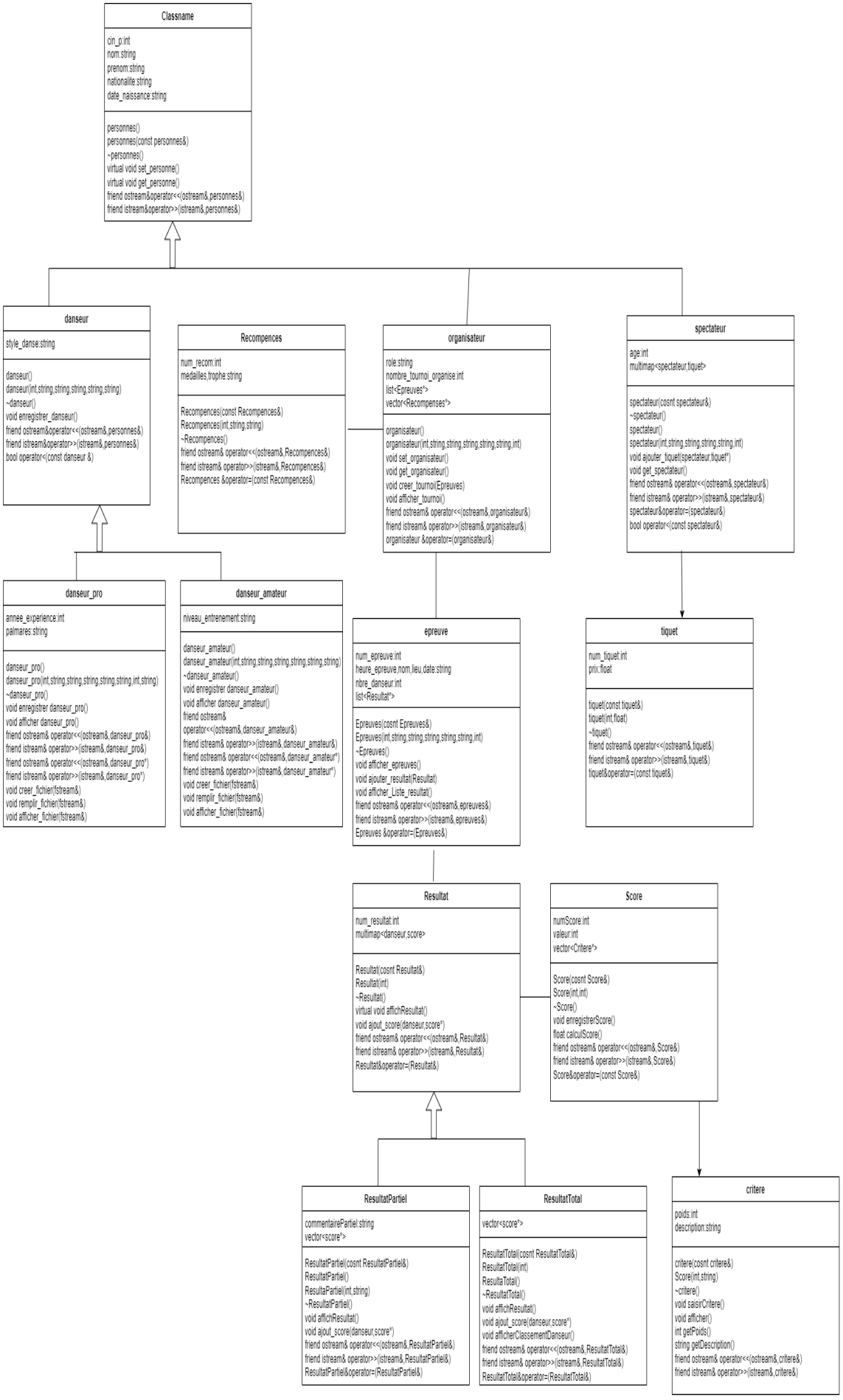
Le rapport détaillera les différentes fonctionnalités de l'application, y compris sa conception, son implémentation et ses principales caractéristiques. Il fournira également des exemples d'utilisation de l'application.

En résumé, ce projet vise à fournir une solution informatique complète et efficace pour la gestion des tournois de danse, offrant aux organisateurs un outil puissant pour orchestrer ces événements de manière transparente et réussie.

2 Diagramme de cas d'utilisation



3 Diagramme de classe



4 Les classes

4.1 Classe personnes

personnes.h

```
    #ifndef personnes_h
#define personnes_h
using namespace std;
#include <string>
#include<ios>

class personnes
{
protected:
    int cin_p;
    string nom;
    string prenom;
    string nationalite;
    string date_naissance;


public:
   personnes();
   personnes(const personnes&);
   ~personnes();
   personnes(int,string,string,string,string);
   virtual  void set_personne();
   virtual  void  get_personne() const ;
   string getNom() const {return nom;}
   friend ostream& operator<<(ostream& , personnes&);
   friend istream& operator>>(istream& , personnes&);



};
#endif
```

personnes.cpp

```
  #include <iostream>
using namespace std;
#include "personnes.h"
#include <string>

// constructeur personnes pour initialiser les attributs
personnes::personnes()
{
    cin_p = 0;
    nom = "";
```

```
        prenom = "";
        nationalite = "";
        date_naissance = "";
    }
    //const recopie
    personnes::personnes(const personnes& p){
        this->cin_p = p.cin_p;
        this->nom = p.nom;
        this->prenom = p.prenom;
        this->nationalite = p.nationalite;
        this->date_naissance = p.date_naissance;

    }
    //destrecteur
    personnes::~personnes(){}
    // constructeur personnes pour saisir les attributs
    personnes::personnes(int cin_p,string  nom , string prenom,
       string nationalite, string date_naissance)
    {
        this->cin_p = cin_p;
        this->nom = nom;
        this->prenom = prenom;
        this->nationalite = nationalite;
        this->date_naissance = date_naissance;
    }
    // methode seter personne
    void personnes::set_personne ()
    {
        cout << "donner le cin " << endl;
        cin >> cin_p;
        cout << "donner le nom " << endl;
        cin >> nom;
        cout << "donner le prenom " << endl;
        cin >> prenom;
        cout << "donner le nationalite " << endl;
        cin >> nationalite;
        cout << "donner date naissance " << endl;
        cin >> date_naissance;
    }
    // methode geter personne
    void personnes::get_personne() const
    {
        cout << "******Affichage ******" << endl;
        cout << "le cin  est :  " << cin_p << endl;
        cout << "le nom  est :  " << nom << endl;
        cout << "le prenom  est :  " << prenom << endl;
        cout << "la nationalite est :  " << nationalite << endl;
        cout << "la date de naissance  est :  " << date_naissance <<
       endl;
    }
    //surcharge de operateur<<
    ostream& operator<<(ostream& out, personnes& p){
        out<<"le cin est : "<<p.cin_p<<endl;
        out<<"le nom est : "<<p.nom<<endl;
        out<<"le prenom est : "<<p.prenom<<endl;
        out<<"la nationalite est : "<<p.nationalite<<endl;
        out<<"le date naissance est : "<<p.date_naissance<<endl;
        return out;
```

```
}
//surcharge de operateur>>
istream& operator>>(istream& in, personnes& p){
    cout << "donner le cin " << endl;
    in >> p.cin_p;
    cout << "donner le nom " << endl;
    in >> p.nom;
    cout << "donner le prenom " << endl;
    in >> p.prenom;
    cout << "donner le nationalite " << endl;
    in >> p.nationalite;
    cout << "donner date naissance " << endl;
    in >> p.date_naissance;
    return in;
}
```

Test execution classe personnes

```
******Affichage ******
le cin  est :  0
le nom  est :
le prenom  est :
la nationalite est :
la date de naissance  est :
donner le cin
12416
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisien
donner date naissance
25fev2003
******Affichage ******
le cin  est :  12416
le nom  est :  mekni
le prenom  est :  jihed
la nationalite est :  tunisien
la date de naissance  est :  25fev2003
donner le cin
23659
donner le nom
dia
donner le prenom
modou
donner le nationalite
senegalais
donner date naissance
23mars20001
le cin est : 23659
le nom est : dia
le prenom est : modou
la nationalite est : senegalais
le date naissance est : 23mars20001
PS C:\Users\mekni\Desktop\version finale poo>
```

4.2 Classe danseur

danseur.h

```
#ifndef danseur_h
#define danseur_h
#include "personnes.cpp"
#include <string>
#include <ios>
class danseur : public personnes
{
protected:
    string style_danse;

public:
    danseur();
    ~danseur();
    danseur(int, string, string, string, string, string);
    void enregistrer_danseur();

    void get_danseur() const;
    friend ostream& operator<<(ostream& , danseur&);
    bool operator<(const  danseur& other)const { return nom <
   other.nom;}
    friend istream& operator>>(istream& , danseur&);

    friend ostream& operator<<(ostream& , danseur*);
     friend istream& operator>>(istream& , danseur*);


};

#endif
```

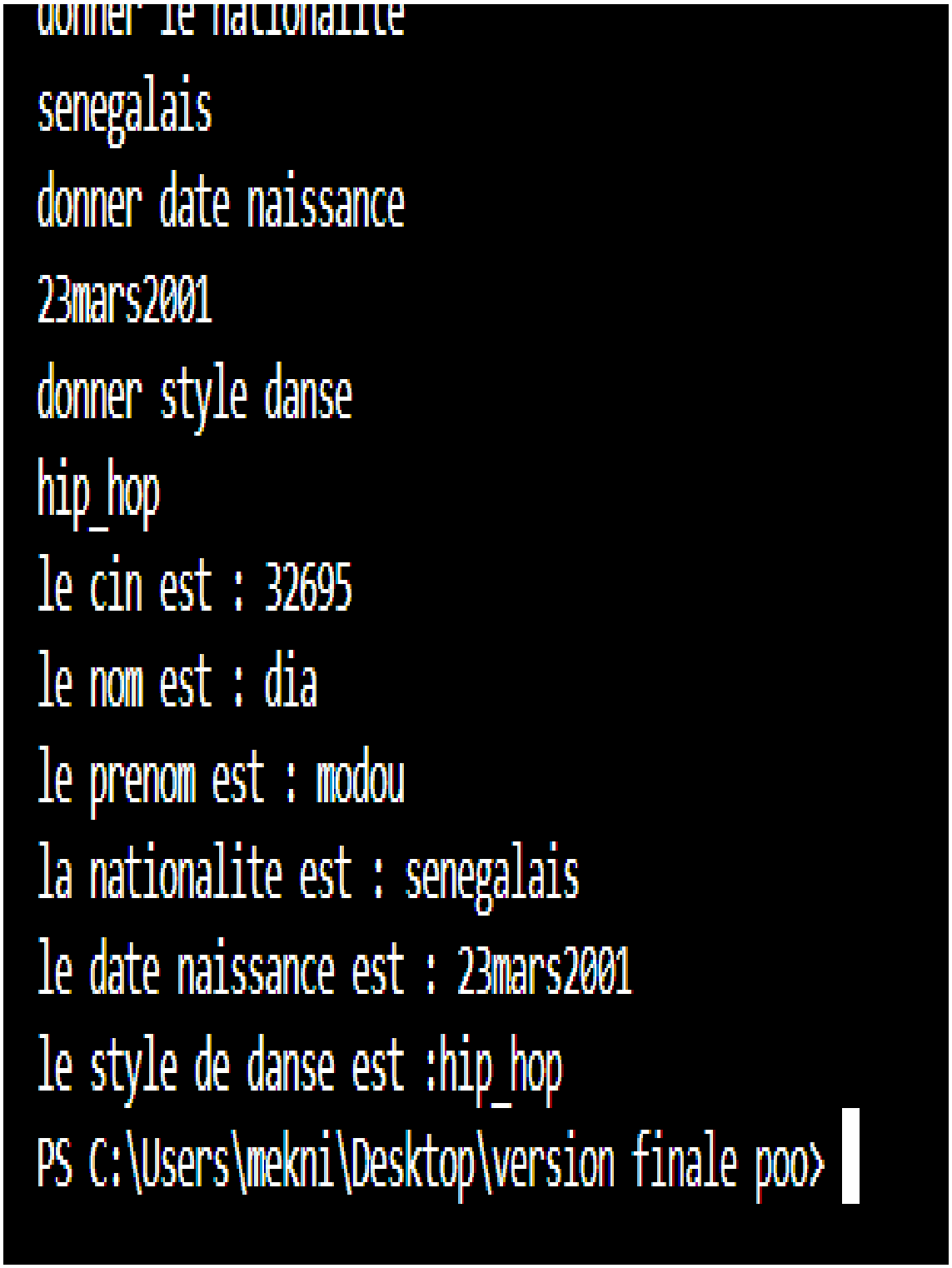
danseur.cpp

```
   #include <iostream>
using namespace std;
#include "danseur.h"
#include <string>
// constructeur par deffaut de danseur
danseur::danseur()
{
    personnes();
    style_danse = "";
}
//destrecteur pour danseur
danseur::~danseur(){}
// constructeur pour danseur
```

```
danseur::danseur(int cin_p, string nom, string prenom, string
    nationalite, string date_naissance, string style_danse)
{
    personnes(cin_p, nom, prenom, nationalite, date_naissance);
    this->style_danse = style_danse;
}
// methode pour enregistrer danseur
void danseur::enregistrer_danseur()
{
    cout << "*******enregistration des danseurs*********" << endl
    ;
    personnes::set_personne();
    cout << "donner le style de danse" << endl;
    cin >> style_danse;
}
// methode geter pour danseur
void danseur::get_danseur() const
{
     personnes::get_personne();
    cout << "le style de danse est :" << style_danse << endl;
}
//surchager l operateur <<
ostream& operator<<(ostream& out,danseur& d){
    out<<"le cin est : "<<d.cin_p<<endl;
    out<<"le nom est : "<<d.nom<<endl;
    out<<"le prenom est : "<<d.prenom<<endl;
    out<<"la nationalite est : "<<d.nationalite<<endl;
    out<<"le date naissance est : "<<d.date_naissance<<endl;
    out<<"le style de danse est :"<<d.style_danse<<endl;
    return out;


}
//surcharger operateur>>
istream& operator>>(istream& in, danseur& d){
    cout << "donner le cin " << endl;
    in >> d.cin_p;
    cout << "donner le nom " << endl;
    in >> d.nom;
    cout << "donner le prenom " << endl;
    in >> d.prenom;
    cout << "donner le nationalite " << endl;
    in >> d.nationalite;
    cout << "donner date naissance " << endl;
    in >> d.date_naissance;
    cout << "donner style danse " << endl;
    in >> d.style_danse;
    return in;
}
```

Text d'exexution classe danseur

```
*******enregistration des danseurs*********
donner le cin
12416
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisien
donner date naissance
25fev20003
donner le style de danse
salsa
******Affichage ******
le cin  est :  12416
le nom  est :  mekni
le prenom  est :  jihed
la nationalite est :  tunisien
la date de naissance  est :  25fev20003
le style de danse est :salsa
donner le cin
32695
donner le nom
dia
donner le prenom
modou
donner le nationalite
senegalais
donner date naissance
23mars2001
donner style danse
```



4.3 Classe danseur amateur

danseur amateur.h

```
  #ifndef danseur_amateur_h
#define danseur_amateur_h
#include"danseur.h"
#include<string>
```

```
class danseur_amateur:public danseur
{
    string niveau_entrenement;
    public:
    danseur_amateur();
    ~danseur_amateur();
    danseur_amateur(int, string, string, string, string, string,
   string);
    void enregistrer_danseur_amateur();
    void afficher_danseur_amateur();
    friend ostream& operator<<(ostream& , danseur_amateur&);
    friend istream& operator>>(istream& , danseur_amateur&);
    friend ostream& operator<<(ostream& , danseur_amateur*);
    friend istream& operator>>(istream& , danseur_amateur*);
    static void creer_fichier(fstream&);
    static void remplir_fichier(fstream&);
    static void afficher_fichier(fstream&);

};

#endif
```

danseur amateur.cpp

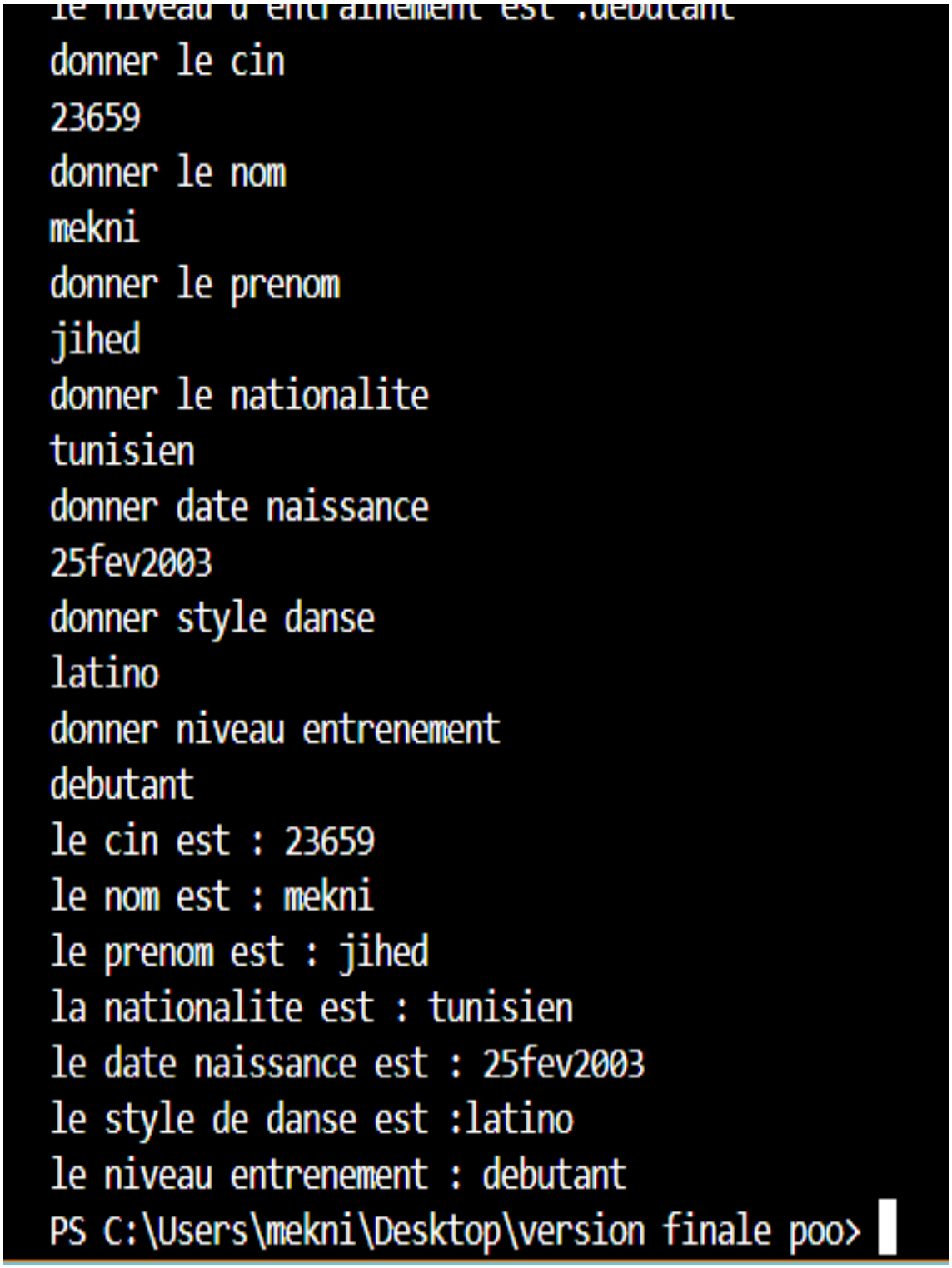
```
  #include <iostream>
using namespace std;
#include "danseur_amateur.h"
#include <string>
#include"fstream"
// constructeur par deffaut pour danseur amateur
danseur_amateur::danseur_amateur()
{
    danseur();
    niveau_entrenement = "";
}
// constructeur pour danseur amateur
danseur_amateur::danseur_amateur(int cin_p, string nom, string
   prenom, string nationalite, string date_naissance, string
   style_danse, string niveau_entrenement)
{
    danseur(cin_p, nom, prenom, nationalite, date_naissance,
   style_danse);
    this->niveau_entrenement = niveau_entrenement;
}
//destrecteur
danseur_amateur::~danseur_amateur(){}
// methode pour enregistrer danseur amateur
void danseur_amateur::enregistrer_danseur_amateur()
{
    danseur::enregistrer_danseur();
    cout << "donner le niveau_entrenement" << endl;
    cin >> niveau_entrenement;
}
//methode pour affichage d un danseur amateur
```

```
void danseur_amateur::afficher_danseur_amateur(){
    get_danseur();
    cout << "le niveau d entrainement est :" <<
   niveau_entrenement << endl;
}
//surchager l operateur <<
ostream& operator<<(ostream& out,danseur_amateur& d){
    out<<"le cin est : "<<d.cin_p<<endl;
    out<<"le nom est : "<<d.nom<<endl;
    out<<"le prenom est : "<<d.prenom<<endl;
    out<<"la nationalite est : "<<d.nationalite<<endl;
    out<<"le date naissance est : "<<d.date_naissance<<endl;
    out<<"le style de danse est :"<<d.style_danse<<endl;
    out<<"le niveau entrenement : "<<d.niveau_entrenement<<endl;
    return out;


}
//surcharger l operator >>
istream& operator>>(istream& in, danseur_amateur& d){
    cout << "donner le cin " << endl;
    in >> d.cin_p;
    cout << "donner le nom " << endl;
    in >> d.nom;
    cout << "donner le prenom " << endl;
    in >> d.prenom;
    cout << "donner le nationalite " << endl;
    in >> d.nationalite;
    cout << "donner date naissance " << endl;
    in >> d.date_naissance;
    cout << "donner style danse " << endl;
    in >> d.style_danse;
    cout << "donner niveau entrenement " << endl;
    in >> d.niveau_entrenement;
    return in;
}
//creation de fichier
void danseur_amateur::creer_fichier(fstream& f){
    f.open("C:\\Users\\mekni\\Desktop\\version finale poo\\
   danseur_amateur.txt",ios::in|ios::out|ios::trunc);
    if(! f.is_open()) {
        exit(-1);
    }
}
//remplissage de fichier
void danseur_amateur::remplir_fichier(fstream& f){
    danseur_amateur d;
    int n;
    cout<<"donner le nombre de danseurs amateurs a remplir"<<endl
   ;
    cin>>n;
    for(int i=0;i<n;i++){
        cout<<"saisir un danseur amateur"<<endl;
        cin>>d;
        f<<d<<endl;
    }
}
//affichage de fichier
```

```
void danseur_amateur::afficher_fichier(fstream& f){
    danseur_amateur d;
    f.seekg(0);

    int arret=0;
    do
    {

        f>>d;
        if(f.eof()) {
            arret=1; break;
        }
        cout<<d<<endl;
    } while (arret==0);




}
//surcharge operator<< avec *danseur_amateur
ostream& operator<<(ostream& out,danseur_amateur* d){
    out<<"le cin est : "<<d->cin_p<<endl;
    out<<"le nom est : "<<d->nom<<endl;
    out<<"le prenom est : "<<d->prenom<<endl;
    out<<"la nationalite est : "<<d->nationalite<<endl;
    out<<"le date naissance est : "<<d->date_naissance<<endl;
    out<<"le style de danse est :"<<d->style_danse<<endl;
    out<<"le niveau entrenement : "<<d->niveau_entrenement<<endl;
    return out;


}
//surchage operateur>> avec *danseur_pro
istream& operator>>(istream& in, danseur_amateur* d){
    cout << "donner le cin " << endl;
    in >> d->cin_p;
    cout << "donner le nom " << endl;
    in >> d->nom;
    cout << "donner le prenom " << endl;
    in >> d->prenom;
    cout << "donner le nationalite " << endl;
    in >> d->nationalite;
    cout << "donner date naissance " << endl;
    in >> d->date_naissance;
    cout << "donner style danse " << endl;
    in >> d->style_danse;
    cout << "donner niveau entrenement " << endl;
    in >> d->niveau_entrenement;
    return in;
}
```

Test execution classe danseur amateur

```
*******enregistration des danseurs*********
donner le cin
12416
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisien
donner date naissance
25fev2003
donner le style de danse
salsa
donner le niveau_entrenement
debutant
******Affichage ******
le cin  est :  12416
le nom  est :  mekni
le prenom  est :  jihed
la nationalite est :  tunisien
la date de naissance  est :  25fev2003
le style de danse est :salsa
le niveau d entrainement est :debutant
donner le cin
23659
```


le niveau d'entrainement est :debutant
donner le cin
23659
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisien
donner date naissance
25fev2003
donner style danse
latino
donner niveau entrenement
debutant
le cin est : 23659
le nom est : mekni
le prenom est : jihed
la nationalite est : tunisien
le date naissance est : 25fev2003
le style de danse est :latino
le niveau entrenement : debutant
PS C:\Users\mekni\Desktop\version finale poo>

Test execution de fichier danseur amateur

danseur.txt

```
1   le cin est : 12416
2   le nom est : mekni
3   le prenom est : jihed
4   la nationalite est : tun
5   le date naissance est : 25fev
6   le style de danse est :salsa
7   le niveau entrenement : debutant
8
9   le cin est : 32654
10  le nom est : dia
11  le prenom est : modou
12  la nationalite est : sen
13  le date naissance est : 23mars
14  le style de danse est :latino
15  le niveau entrenement : debutant
16
17
```

4.4 Classe danseur pro

danseur pro.h

```
#ifndef danseur_pro_h
#define danseur_pro_h
#include"danseur.h"
```

```
#include <string>


class danseur_pro:public danseur
{
    int annee_experience;
    string palmares;

public:
    danseur_pro();
    ~danseur_pro();
    danseur_pro(int, string, string, string, string, string, int,
    string);
    void enregistrer_danseur_pro();
    void afficher_danseur_pro();
    friend ostream& operator<<(ostream& , danseur_pro&);
    friend istream& operator>>(istream& , danseur_pro&);
    static void creer_fichier(fstream&);
    static void remplir_fichier(fstream&);
    static void afficher_fichier(fstream&);
    friend ostream& operator<<(ostream& , danseur_pro*);
    friend istream& operator>>(istream& , danseur_pro*);
};

#endif
```

danseur pro.cpp

```
   #include <iostream>
using namespace std;
#include "danseur_pro.h"
#include <string>
#include"fstream"
// constructeur par deffaut pour danseur pro
danseur_pro::danseur_pro()
{
    danseur();
    annee_experience = 0;
    palmares = "";
}
//destrecteur
danseur_pro::~danseur_pro(){};
// constructeur pour danseur pro
danseur_pro::danseur_pro(int cin_p, string nom, string prenom,
   string nationalite, string date_naissance, string style_danse,
    int annee_experience, string palmares)
{    danseur(cin_p,nom,prenom,nationalite,date_naissance,
   style_danse);
    this->annee_experience = annee_experience;
    this->palmares = palmares;
}
void danseur_pro::enregistrer_danseur_pro()
{
    danseur::enregistrer_danseur();
    cout << "donner les annee d experience" << endl;
```

```
        cin >> annee_experience;
        cout << "donner le palmares" << endl;
        cin >> palmares;
    }
    // methode afficher danseur_pro
    void danseur_pro::afficher_danseur_pro()
    {
        danseur::get_danseur();
        cout << "les annees d experience sont :" << annee_experience
       << endl;
        cout << "le palmares est :" << palmares << endl;
    }
    //surchager l operateur <<
    ostream& operator<<(ostream& out,danseur_pro& d){
        out<<"le cin est : "<<d.cin_p<<endl;
        out<<"le nom est : "<<d.nom<<endl;
        out<<"le prenom est : "<<d.prenom<<endl;
        out<<"la nationalite est : "<<d.nationalite<<endl;
        out<<"le date naissance est : "<<d.date_naissance<<endl;
        out<<"le style de danse est :"<<d.style_danse<<endl;
        out<<"les annees experience : "<<d.annee_experience<<endl;
        out<<"les palmares : "<<d.palmares<<endl;

        return out;


    }
    //surcharger l operator >>
    istream& operator>>(istream& in, danseur_pro& d){
        cout << "donner le cin " << endl;
        in >> d.cin_p;
        cout << "donner le nom " << endl;
        in >> d.nom;
        cout << "donner le prenom " << endl;
        in >> d.prenom;
        cout << "donner le nationalite " << endl;
        in >> d.nationalite;
        cout << "donner date naissance " << endl;
        in >> d.date_naissance;
        cout << "donner style danse " << endl;
        in >> d.style_danse;
        cout << "donner les annees experience " << endl;
        in >> d.annee_experience;
        cout << "donner les palmares " << endl;
        in >> d.palmares;
        return in;
    }
    //creation de fichier
    void danseur_pro::creer_fichier(fstream& f){
        f.open("C:\\Users\\mekni\\Desktop\\version finale poo\\
       danseur_pro.txt",ios::in|ios::out|ios::trunc);
        if(! f.is_open()) {
            exit(-1);
        }
    }
    //remplissage de fichier
    void danseur_pro::remplir_fichier(fstream& f){
        danseur_pro d;
```

```
        int n;
        cout<<"donner le nombre de danseurs pro a remplir"<<endl;
        cin>>n;
        for(int i=0;i<n;i++){
            cout<<"saisir un danseur pro"<<endl;
            cin>>d;
            f<<d<<endl;
        }
    }
    //affichage de fichier
    void danseur_pro::afficher_fichier(fstream& f){
        danseur_pro d;
        f.seekg(0);

        int arret=0;
        do
        {

            f>>d;
            if(f.eof()) {
                arret=1; break;
            }
            cout<<d<<endl;
        } while (arret==0);




    }
    //surcharger operateur << avec *danseur pro
    ostream& operator<<(ostream& out,danseur_pro* d){
        out<<"le cin est : "<<d->cin_p<<endl;
        out<<"le nom est : "<<d->nom<<endl;
        out<<"le prenom est : "<<d->prenom<<endl;
        out<<"la nationalite est : "<<d->nationalite<<endl;
        out<<"le date naissance est : "<<d->date_naissance<<endl;
        out<<"le style de danse est :"<<d->style_danse<<endl;
        out<<"les annees experience : "<<d->annee_experience<<endl;
        out<<"les palmares : "<<d->palmares<<endl;
        return out;


    }
    //surcharger operateur >> avec *danseur pro
    istream& operator>>(istream& in, danseur_pro* d){
        cout << "donner le cin " << endl;
        in >> d->cin_p;
        cout << "donner le nom " << endl;
        in >> d->nom;
        cout << "donner le prenom " << endl;
        in >> d->prenom;
        cout << "donner le nationalite " << endl;
        in >> d->nationalite;
        cout << "donner date naissance " << endl;
        in >> d->date_naissance;
        cout << "donner style danse " << endl;
        in >> d->style_danse;
        cout << "donner les annees experience " << endl;
```

```
        in >> d->annee_experience;
        cout << "donner les palmares " << endl;
        in >> d->palmares;
        return in;
    }
```

Test execution classe danseur pro

```
*******enregistration des danseurs*********
donner le cin
1241
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisein
donner date naissance
25fev2003
donner le style de danse
salsa
donner les annee d experience
5
donner le palmares
gold
******Affichage ******
le cin  est :  1241
le nom  est :  mekni
le prenom  est :  jihed
la nationalite est :  tunisein
la date de naissance  est :  25fev2003
le style de danse est :salsa
les annees d experience sont :5
le palmares est :gold
donner le cin
```

```
le palmares est :gold
donner le cin
23659
donner le nom
mekni
donner le prenom
jihed
donner le nationalite
tunisien
donner date naissance
25fev2003
donner style danse
lationo
donner les annees experience
6
donner les palmares
silver
le cin est : 23659
le nom est : mekni
le prenom est : jihed
la nationalite est : tunisien
le date naissance est : 25fev2003
le style de danse est :lationo
les annees experience : 6
les palmares : silver
PS C:\Users\mekni\Desktop\version finale poo>
```

4.5 Classe Template

Code classe Template

```
        #define ARCHIVE_H

#include "danseur_amateur.cpp"
#include "danseur_pro.cpp"

using namespace std;
// Classe template de base
template <typename T>
class DanseurTemplate : public T {
public:
    DanseurTemplate() : T() {}
    DanseurTemplate(int cin,  string nom, string prenom,
                    string nationalite,  string date_naissance,
   string style)
        : T(cin, nom, prenom, nationalite, date_naissance,style)
   {}

};

// Sp cialisation pour les danseurs amateurs
class DanseurAmateurTemplate : public DanseurTemplate<
   danseur_amateur> {
public:
    DanseurAmateurTemplate() : DanseurTemplate<danseur_amateur>()
    {}
    DanseurAmateurTemplate(int cin, string nom,  string prenom,
                           string nationalite, string
   date_naissance,
                           string style, string
   niveau_entrenement="")
        : DanseurTemplate<danseur_amateur>(cin, nom, prenom,
   nationalite, date_naissance,style) {
            //this->niveau_entrenement = niveau_entrenement;

        }
};

// Sp cialisation pour les danseurs professionnels
class DanseurProTemplate : public DanseurTemplate<danseur_pro> {
public:
    DanseurProTemplate() : DanseurTemplate<danseur_pro>() {}
    DanseurProTemplate(int cin, string nom,string prenom,
                       string nationalite, string date_naissance,
   string style,
                       int annee_experience=5, string palmares=""
   )
        : DanseurTemplate<danseur_pro>(cin, nom, prenom,
   nationalite, date_naissance,style) {
            //this->annee_experience = annee_experience;
            //this->palmares = palmares;
        }
};
```

Test execution Classe Template

```
-------TESTER CLASSE TEMPLATE---------
Saisir danseur amateur
donner le cin
1
donner le nom
jihed
donner le prenom
mekni
donner le nationalite
tun
donner date naissance
11nov
donner style danse
salsa
donner niveau entrenement
moyen
le cin est : 1
le nom est : jihed
le prenom est : mekni
la nationalite est : tun
le date naissance est : 11nov
le style de danse est :salsa
le niveau entrenement : moyen
Saisir danseur amateur
donner le cin
2
donner le nom
dia
donner le prenom
modou
donner le nationalite
sen
```

```
le prenom est : mekni
la nationalite est : tun
le date naissance est : 11nov
le style de danse est :salsa
le niveau entrenement : moyen
Saisir danseur amateur
donner le cin
2
donner le nom
dia
donner le prenom
modou
donner le nationalite
sen
donner date naissance
23juin
donner style danse
salsa
donner les annees experience
3
donner les palmares
champion
le cin est : 2
le nom est : dia
le prenom est : modou
la nationalite est : sen
le date naissance est : 23juin
le style de danse est :salsa
les annees experience : 3
les palmares : champion
PS C:\Users\DELL\Desktop\ProjetC++2024\Application-de-ge
```

4.6 Classe Organisateur

organisateur.h

```
    #ifndef organisateur_h
#define organisateur_h
#include <string>
#include "personnes.h"
#include <list>
#include<vector>
#include "epreuve.cpp"
#include"Recompenses.cpp"
#include "iterator"
class organisateur : public personnes
{
    string role;
    int nombre_tournoi_organisee;
    list<Epreuves *> list_Epreuve;
    vector<Recompenses*> list_recompenses;


public:
    organisateur();
    organisateur(const organisateur &);
    ~organisateur();
    organisateur(int, string, string, string, string, string, int
   );
    void set_organisateur();
    void get_organisateur();
    void creer_tournoi(Epreuves);
    void afficher_tournoi();
    friend ostream &operator<<(ostream &, organisateur &);
    friend istream &operator>>(istream &, organisateur &);
    organisateur &operator=(organisateur &);
};

#endif
```

organisateur.cpp

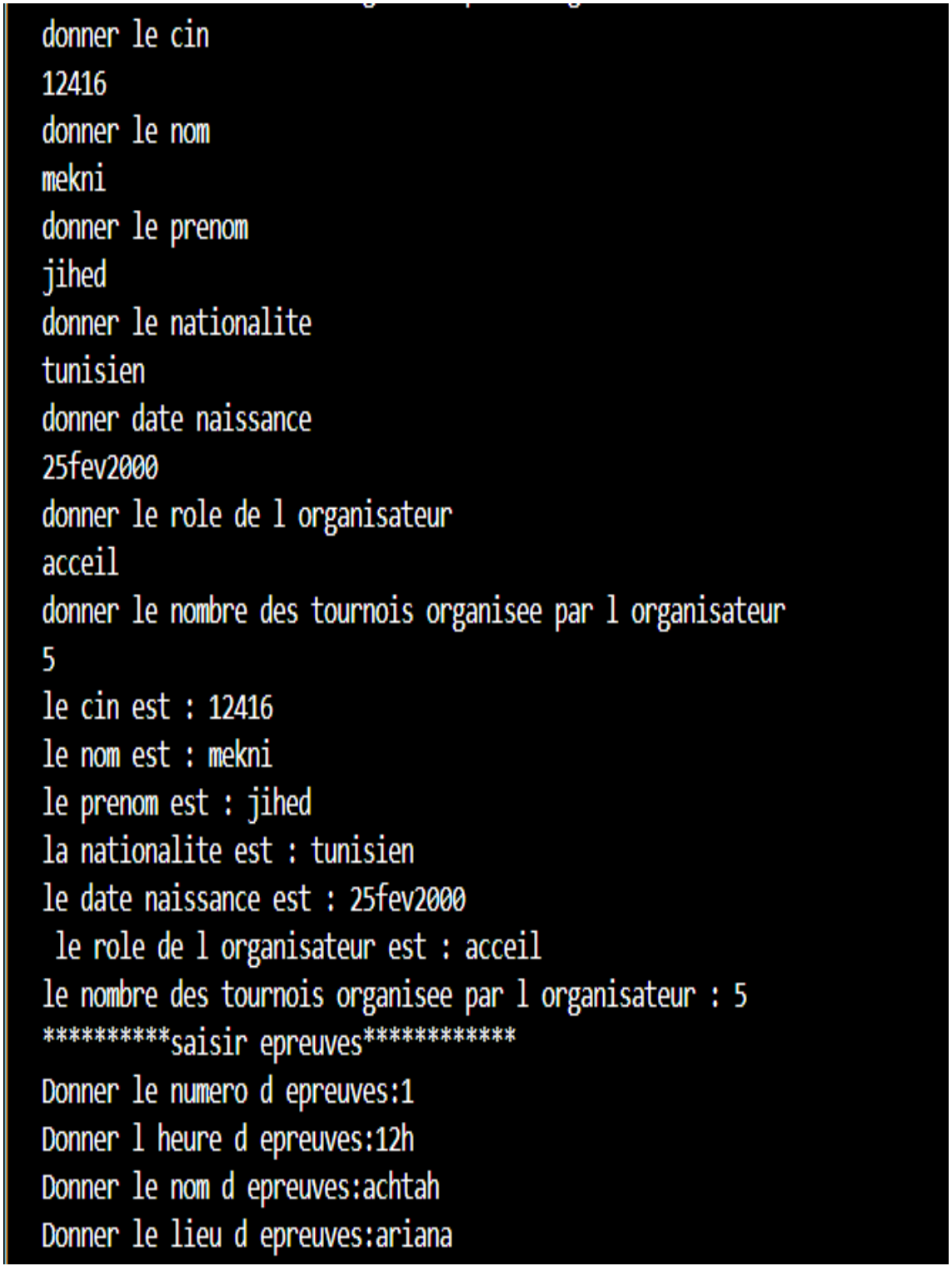
```
   #include <iostream>
using namespace std;
#include "organisateur.h"
#include <string>
// constructeur par deffaut pour oragnisateur
organisateur::organisateur()
{
    personnes();
    role = "";
    nombre_tournoi_organisee = 0;
}
```

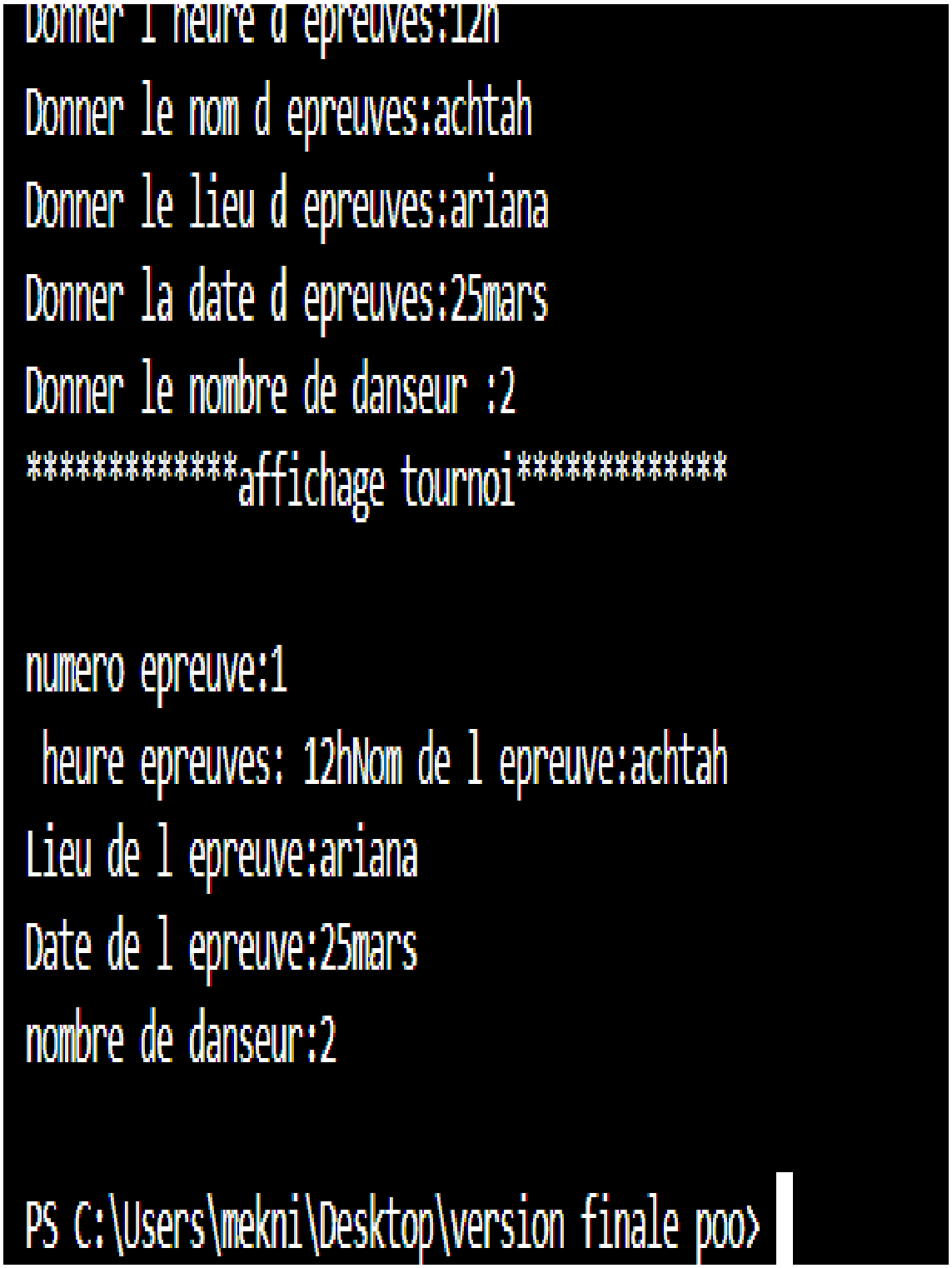
```
// constructeur par recopie
 organisateur::organisateur(const organisateur &o) : personnes(o)
{
    role = o.role;
    nombre_tournoi_organisee = o.nombre_tournoi_organisee;

    list<Epreuves *>::const_iterator it;
    for (it = o.list_Epreuve.begin(); it != o.list_Epreuve.end();
    ++it)
    {
        // Cr ation d'un nouvel objet Epreuves et ajout    la
   liste
        Epreuves *e = new Epreuves(**it); // Assurez-vous que le
   constructeur de copie de Epreuves est correctement d fini
        list_Epreuve.push_back(e);
    }
}

// destrecteur
organisateur::~organisateur() {
    list<Epreuves *>::iterator it;
    for (it = list_Epreuve.begin(); it != list_Epreuve.end(); ++
   it)
    {
        delete *it;
    }
    list_Epreuve.clear();
    for(unsigned int i=0;i<list_recompenses.size();i++){
        delete list_recompenses[i];
    }
    list_recompenses.clear();

}
// constructeur pour organisateur
organisateur::organisateur(int cin_p, string nom, string prenom,
   string nationalite, string date_naissance, string role, int
   nombre_tournoi_organisee)
{
    personnes(cin_p, nom, prenom, nationalite, date_naissance);
    this->role = role;
    this->nombre_tournoi_organisee = nombre_tournoi_organisee;
}
// methode seter pour organisateur
void organisateur::set_organisateur()
{
    set_personne();
    cout << "donner le role de l organisateur" << endl;
    cin >> role;
    cout << "donner le nombre des tournois organisee par l
   organisateur" << endl;
    cin >> nombre_tournoi_organisee;
}
// methode geter pour organisateur
void organisateur::get_organisateur()
{
    get_personne();
    cout << " le role de l organisateur est : " << role << endl;
```

```
        cout << "le nombre des tournois organisee par l organisateur
       : " << nombre_tournoi_organisee << endl;
}
// methode pour creer un tournoi
void organisateur::creer_tournoi(Epreuves e)
{
    Epreuves *ep = new Epreuves(e);
    this->list_Epreuve.push_back(ep);
}
// meth afficher tournoi
void organisateur::afficher_tournoi()
{
    list<Epreuves *>::iterator it;
    for (it = list_Epreuve.begin(); it != list_Epreuve.end(); ++
   it)
    {
        cout << **it;
        cout << endl;
    }
}
//surchager operateur<<
ostream &operator<<(ostream &out, organisateur &o)
{
    out << "le cin est : " << o.cin_p << endl;
    out << "le nom est : " << o.nom << endl;
    out << "le prenom est : " << o.prenom << endl;
    out << "la nationalite est : " << o.nationalite << endl;
    out << "le date naissance est : " << o.date_naissance << endl
   ;
    out << " le role de l organisateur est : " << o.role << endl;
    out << "le nombre des tournois organisee par l organisateur :
    " << o.nombre_tournoi_organisee << endl;

    return out;
}
//surcharge operateur>>
istream &operator>>(istream &in, organisateur &o)
{
    cout << "donner le cin " << endl;
    in >> o.cin_p;
    cout << "donner le nom " << endl;
    in >> o.nom;
    cout << "donner le prenom " << endl;
    in >> o.prenom;
    cout << "donner le nationalite " << endl;
    in >> o.nationalite;
    cout << "donner date naissance " << endl;
    in >> o.date_naissance;
    cout << " donner le role de l organisateur est : " << endl;
    in >> o.role;
    cout << "donner le nombre des tournois organisee par l
   organisateur : " << endl;
    in >> o.nombre_tournoi_organisee;
    return in;
}
//surcharger operateur =
organisateur &organisateur::operator=(organisateur &o)
{
```

```
        if (this != &o)
        {
            personnes *p1 = this;
            personnes *p2 = &o;
            *p1 = *p2;
            list<Epreuves *>::iterator it;
            for (it = list_Epreuve.begin(); it != list_Epreuve.end();
        ++it)
            {
                delete *it;
            }
            list_Epreuve.clear();
            Epreuves *e;
            for (it = o.list_Epreuve.begin(); it != o.list_Epreuve.
       end(); ++it)
            {
                e = new Epreuves(**it);
                list_Epreuve.push_back(e);
            }
        }
    }
```

Test execution classe organisateur





4.7 Classe recompences

recompences.h

```
#include<string>
using namespace std;
class Recompenses
{
private:
```

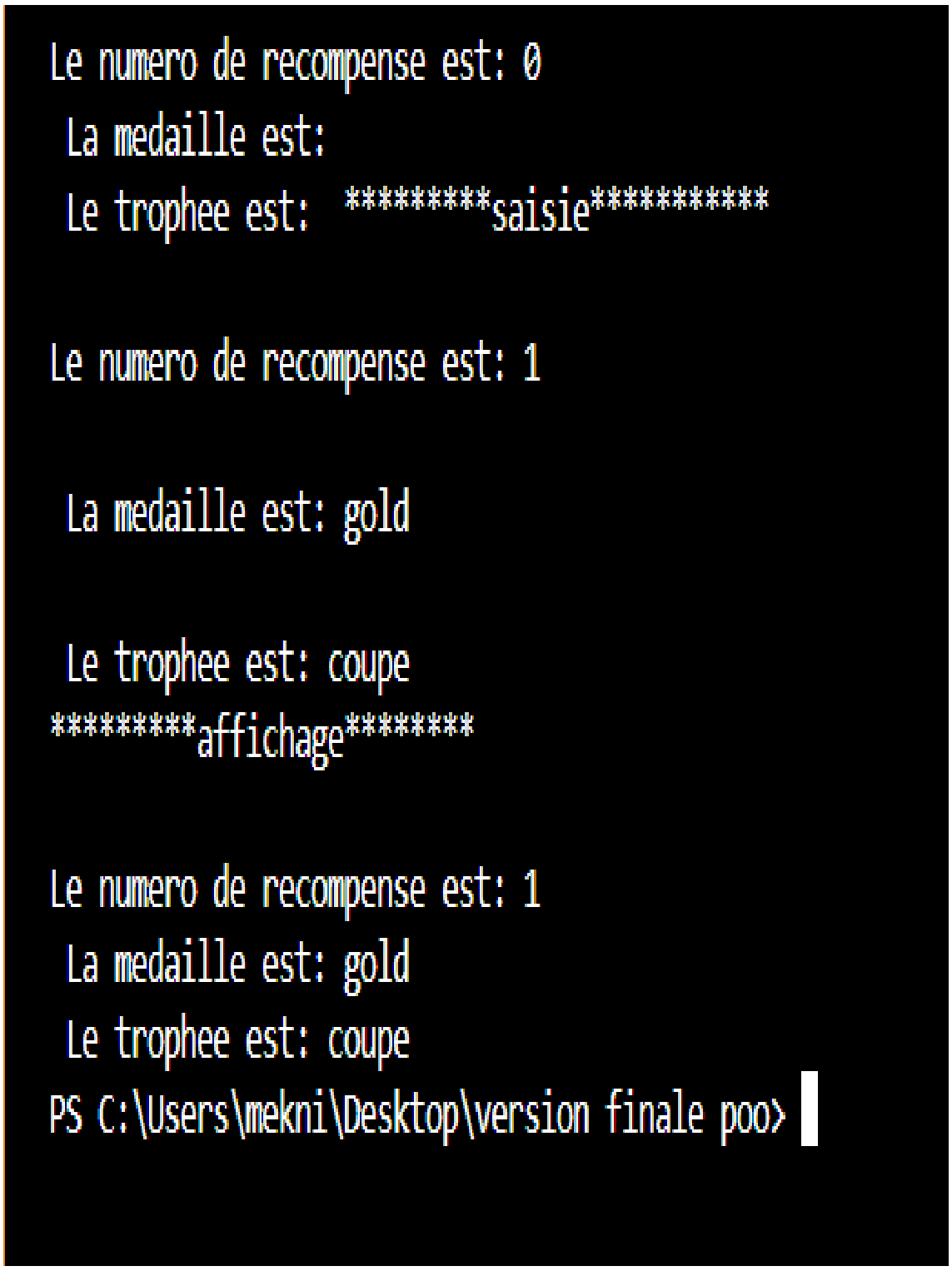
```
        int num_recom;
        string medailles, trophee;

    public:
        Recompenses(int=0,string=" ",string=" ");
        Recompenses(const Recompenses&);
        Recompenses& operator=(const Recompenses&);
        friend ostream& operator<<(ostream&, Recompenses&);
        friend istream& operator>>(istream&, Recompenses&);
        ~Recompenses();
    };
```

recompences.cpp

```
     #include "Recompenses.h"
    #include <iostream>
    //constrecteur
    Recompenses::Recompenses(int num, string med, string trop)
    {
        this->num_recom= num;
        this->medailles=med;
        this->trophee=trop;
    }
    //destrecteur

    Recompenses::~Recompenses()
    {
    }
    //constrecteur par recopie
    Recompenses::Recompenses(const Recompenses& r)
    {
        num_recom=r.num_recom;
        medailles= r.medailles;
        trophee= r.trophee;
    }
    //surcharge operateur =
    Recompenses& Recompenses::operator=(const Recompenses& r)
    {
        if(this!=&r)
        {
            num_recom=r.num_recom;
            medailles=r.medailles;
            trophee=r.trophee;
        }
        return *this;
    }
    //surcharge operateur <<
    ostream& operator<<(ostream& out, Recompenses& r)
    {
        out<<"\nLe numero de recompense est: "<<r.num_recom;
        out<<"\n La medaille est: "<<r.medailles;
        out<<"\n Le trophee est: "<<r.trophee;
        return out;
    }
    //surcharge operateur>>
```

```
istream& operator>>(istream& in , Recompenses& r)
{
    cout<<"\nLe numero de recompense est: ";
    in>>r.num_recom;
    cout<<"\n La medaille est: ";
    in>>r.medailles;
    cout<<"\n Le trophee est: ";
    in>>r.trophee;
    return in;

}
```

Test execution classe recompences



4.8 Classe Epreuves

Epreuves.h

```
#include<string>
#include<list>
#include "Resultat.h"
#include "iterator"
#include<ios>
#include <fstream>
using namespace std;
class Epreuves
{
private:

    int num_epreuves;
    string heure_epreuves;
    string nom, lieu, date;
    int nbre_danseur;
    list<Resultat *> listRsultat;
public:
    //Constructeur par defaut
    Epreuves(int=0,string=" ",string=" ",string=" ",string=" ",
   int=0);
    //Constructeur par recopie
    Epreuves(const Epreuves &e);
    //AFFICHAGE EPREUVE
    void afficher_epreuves();
    //SURCHARGE OPERATEUR =
    Epreuves& operator=(Epreuves&);
    //SURCHARGE OPERATEUR << ET >> POUR SAISIR ET AFFICHER
    friend ostream& operator<<(ostream&, Epreuves&);
    friend istream& operator>>(istream&, Epreuves&);
    //SURCHARGE OPERATEUR << ET >> POUR LIRE ET ECRIRE UN FICHIER
    friend istream &operator>>(istream &in, Epreuves *e);
    friend ostream &operator<<(ostream &out, Epreuves *e);
    void ajouter_resultat(Resultat);
    void afficher_Liste_Resultat();

    //PARTIE FICIHER POUR METTRE TOUS LES RESULTATS DANS UN
   FICHIER
    void creerFichier(fstream& f);
    void remplirFiciher(fstream& f);
    void afficherFiciher(fstream& f);

    //PARTIE FICIHER POUR remplir EPREUVES
    void creerFichierEpreuve(fstream& f);
    void remplirFiciherEpreuve(fstream& f);
    void afficherFiciherEpreuve(fstream& f);



    ~Epreuves();
};
```

Epreuves.cpp

```
 #include "epreuve.h"
#include <iostream>

Epreuves::Epreuves(int num, string heure, string nom, string lieu
    , string date, int nbr)
{
    this->num_epreuves = num;
    this->heure_epreuves = heure;
    this->nom = nom;
    this->lieu = lieu;
    this->date = date;
    this->nbre_danseur = nbr;
}

Epreuves::Epreuves(const Epreuves &e)
{
    this->num_epreuves=e.num_epreuves;
    this->heure_epreuves=e.heure_epreuves;
    this->nom= e.nom;
    this->lieu = e.lieu;
    this->date=e.date;
    this->nbre_danseur= e.nbre_danseur;
    list<Resultat>::iterator it;
    for(auto it=e.listRsultat.begin();it!=e.listRsultat.end();++
   it)
    {
        listRsultat.push_back(*it);
        //(*it)->affichResulat();

        cout<<"\n";
    }

}

void Epreuves::afficher_epreuves()
{
    cout << "numero epreuve:" << this->num_epreuves << "\n heure
   epreuves: " << this->heure_epreuves << endl;
    cout << "Nom de l epreuve:" << nom << endl;
    cout << "Lieu de l epreuve:" << lieu << endl;
    cout << "Date de l epreuve:" << date << endl;
    cout << "nombre de danseur:" << nbre_danseur << endl;

}

ostream &operator<<(ostream &out, Epreuves &e)
{
    out << "\nnumero epreuve:" << e.num_epreuves;
```

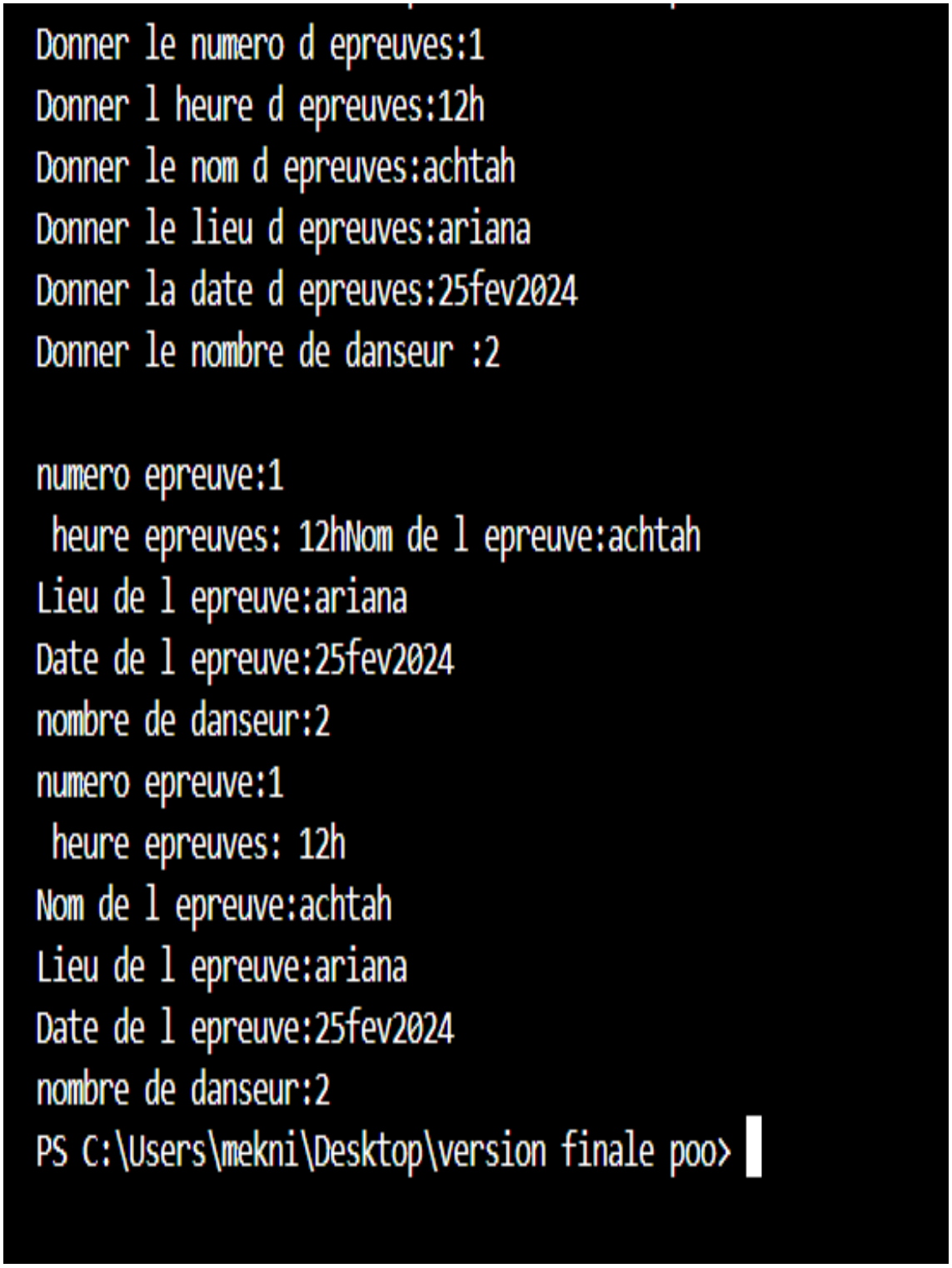
```
        out << "\n heure epreuves: " << e.heure_epreuves<<endl;
        out << "Nom de l epreuve:" << e.nom << endl;
        out << "Lieu de l epreuve:" << e.lieu << endl;
        out << "Date de l epreuve:" << e.date << endl;
        out << "nombre de danseur:" << e.nbre_danseur << endl;

        return out;
    }

    ostream &operator<<(ostream &out, Epreuves *e)
    {
        out << "\nnumero epreuve:" << e->num_epreuves;
        out << "\n heure epreuves: " << e->heure_epreuves<<endl;
        out << "Nom de l epreuve:" << e->nom << endl;
        out << "Lieu de l epreuve:" << e->lieu << endl;
        out << "Date de l epreuve:" << e->date << endl;
        out << "nombre de danseur:" << e->nbre_danseur << endl;

        return out;
    }

    istream &operator>>(istream &in, Epreuves &e)
    {
        cout << "Donner le numero d epreuves:";
        in >> e.num_epreuves;
        cout << "Donner l heure d epreuves:";
        in >> e.heure_epreuves;
        cout << "Donner le nom d epreuves:";
        in >> e.nom;
        cout << "Donner le lieu d epreuves:";
        in >> e.lieu;
        cout << "Donner la date d epreuves:";
        in >> e.date;
        cout << "Donner le nombre de danseur :";
        in >> e.nbre_danseur;



        return in;
    }

    istream &operator>>(istream &in, Epreuves *e)
    {
        in.seekg(0);
        cout << "Donner le numero d epreuves:";
        in >> e->num_epreuves;
        cout << "Donner l heure d epreuves:";
        in >> e->heure_epreuves;
        cout << "Donner le nom d epreuves:";
        in >> e->nom;
        cout << "Donner le lieu d epreuves:";
        in >> e->lieu;
        cout << "Donner la date d epreuves:";
        in >> e->date;
        cout << "Donner le nombre de danseur :";
        in >> e->nbre_danseur;
        return in;
    }
```

```
Epreuves &Epreuves::operator=(Epreuves &e)
{
    if (this != &e)
    {
        this->num_epreuves = e.num_epreuves;
        this->heure_epreuves = e.heure_epreuves;
        this->nom = e.nom;
        this->lieu = e.lieu;
        this->date = e.date;
        this->nbre_danseur = e.nbre_danseur;
        list<Resultat *>::iterator it;
        for ( auto it =listRsultat.begin(); it!=listRsultat.end()
   ; ++it)
        {
            delete *it;
        }

        listRsultat.clear();
        Resultat *r;
        for ( auto it = e.listRsultat.begin(); it != e.
   listRsultat.end(); ++it)
        {
            r = new Resultat(**it);
            listRsultat.push_back(r);
        }
        delete [] r;

     }
     return *this;
}

void Epreuves::ajouter_resultat(Resultat r)
{
    Resultat *res= new Resultat(r);
    listRsultat.push_back(res);

}

Epreuves::~Epreuves()
{
     list<Resultat *>::iterator it;
        for ( auto it =listRsultat.begin(); it!=listRsultat.end()
   ; ++it)
        {
            delete *it;
        }

        listRsultat.clear();
}

void Epreuves::afficher_Liste_Resultat()
{
     cout << "\n-------AFFICHAGE DE LA LISTE DES RESULTATS
   -------\n";
    list<Resultat *>::const_iterator it;
    for (auto it = listRsultat.begin(); it!= listRsultat.end();
   ++it)
```

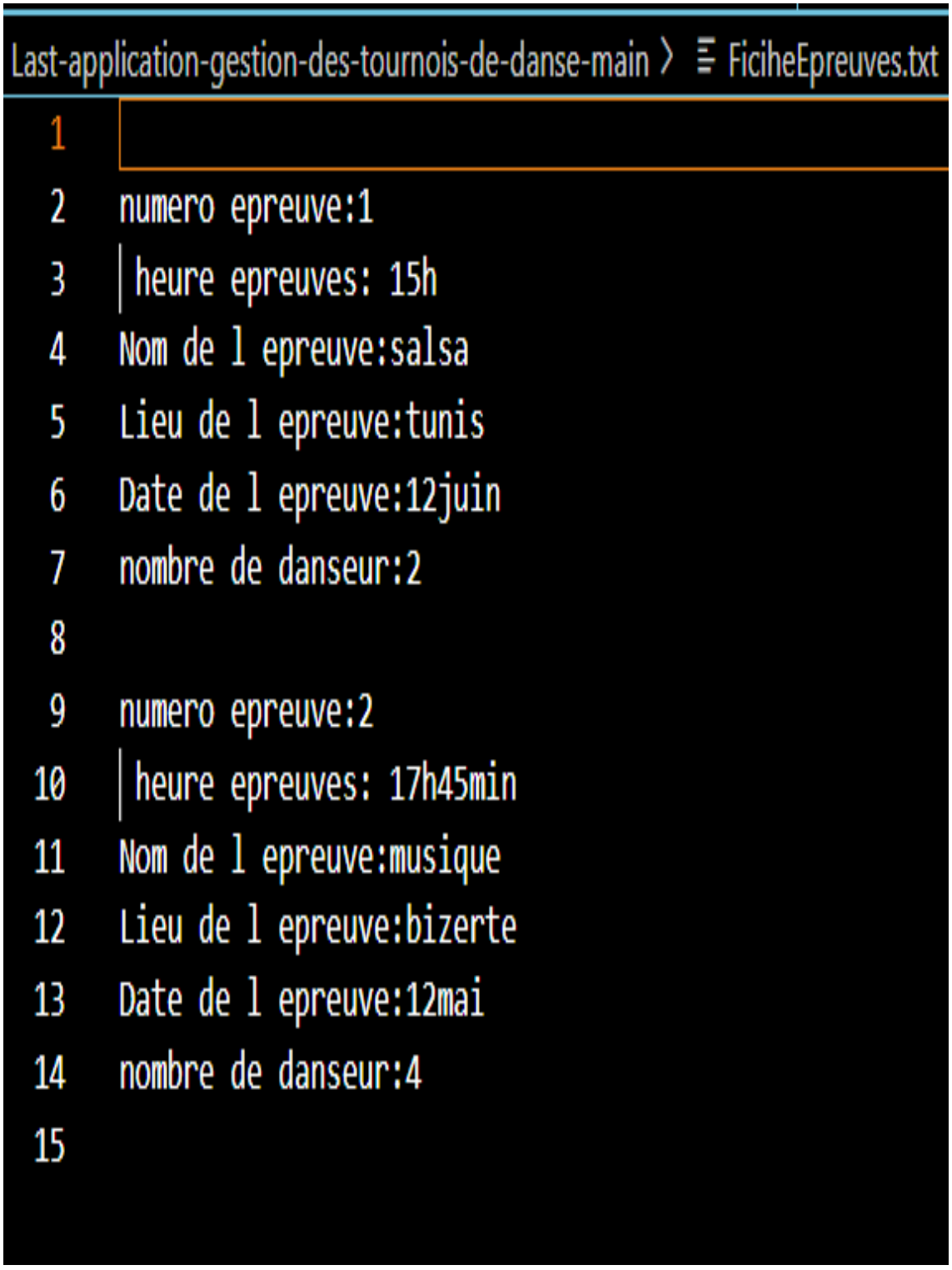
```
        {
            // (*it)->affichResulat();
            cout<<**it;
             cout << "\n";
        }
    }

    void Epreuves::creerFichier(fstream& f)
    {
        f.open("C:\\Users\\DELL\\Desktop\\ProjetC++2024\\Application-
       de-gestion-des-tournoi-de-danse\\FiciheRresultats.txt" , ios::
       out | ios::trunc);
        if (!f.is_open()) {
            cerr << "Erreur : Impossible d'ouvrir le fichier." <<
       endl;
            exit(-1);// Quitter le programme avec un code d'erreur
        }
    }

    void Epreuves::remplirFiciher(fstream& f)
    {
        list<Resultat *>::iterator it;
        /*
        for(it=listRsultat.begin();it!=listRsultat.end();++it)
        {
            f<<**it;
             f << "----------------------------------------" << endl;
        }*/

        for ( it = listRsultat.begin(); it != listRsultat.end(); ++it
       ) {
            Resultat* resultat = *it;
              const multimap<danseur, score>& myMap = resultat->
       getMyMap(); // Obtenir la multimap du r sultat

            multimap<danseur, score>::iterator mapIt;
            for (auto mapIt = myMap.begin(); mapIt != myMap.end(); ++
       mapIt) {
                //danseur& d = mapIt->first;
                score s = mapIt->second;

                //  crire  les donn es dans le fichier
                f << "Numero de resultat: " << resultat->getNum_res()
        << endl;
                f << "Nom du danseur: " <<  mapIt->first.getNom() <<
       endl;
                f << "Score: " << s.calculScore() << endl;
                f << "--------------------" << endl; // Marquer la fin
        des donn es pour chaque paire
            }
            f << "--------------------" << endl;
        }


    }

    void Epreuves::creerFichierEpreuve(fstream& f)
    {
```

```
        f.open("C:\\Users\\DELL\\Desktop\\ProjetC++2024\\Application-
    de-gestion-des-tournoi-de-danse\\FiciheEpreuves.txt" ,ios::in
    | ios::out | ios::trunc);
        if (!f.is_open()) {
            cerr << "Erreur : Impossible d'ouvrir le fichier." <<
    endl;
            exit(-1);// Quitter le programme avec un code d'erreur
        }
}

void Epreuves::remplirFiciherEpreuve(fstream& f)
{
    Epreuves e;
    int nbr;
    cout<<"Donner le nombre epreuves enregister:";
    cin>>nbr;
    for(int i=0;i<nbr;i++)
    {
        cout<<"\nSaisir Epreuves"<<endl;
        cin>>e;
        f<<&e;
    }
}

void Epreuves::afficherFiciherEpreuve(fstream& f)
{
    Epreuves e;
    f.seekg(0);
    while (1)
    {
        f>>&e;
        if(f.eof()) break;
        cout<<e<<endl;

    }

}
```

Test execution Epreuves



Test execution de fichier Epreuves



4.9 Classe Resultat

Resultat.h

```
#include <ios>
#include <map>
#include "danseur.h"
#include "score.h"
class Resultat
{
protected:

    int num_res;
    multimap<danseur,score>  myMap;

public:

    Resultat(int=2);
    Resultat(const Resultat&);
    Resultat& operator=(Resultat&);
    int getNum_res() {return num_res;}
    const multimap<danseur, score>& getMyMap()  const{return
   myMap;}
    friend ostream& operator<<(ostream&, Resultat&);
    friend istream& operator>>(istream&, Resultat&);
    void affichResulat();
    void ajout_score(danseur d, score *s);
     ~Resultat();
};
```

Resultat.cpp

```
    #include "Resultat.h"
#include <iostream>
using namespace std;
using namespace std;
#include <ios>
#include <map>

#include "danseur.h"
#include "score.cpp"



Resultat::Resultat(int num)
{
    this->num_res = num;
}

Resultat::~Resultat()
{
}
```

```
Resultat::Resultat(const Resultat &r)
{
    this->num_res = r.num_res;
}

Resultat &Resultat::operator=(const Resultat &r)
{
    if (this != &r)
    {
        this->num_res = r.num_res;
    }
    return *this;
}

istream &operator>>(istream &in, Resultat &r)
{
    cout << "Donner le numero du resultat:";
    in >> r.num_res;

    return in;
}

ostream &operator<<(ostream &out, const Resultat &r)
{

    out<<"numero du resultat:"<<r.num_res<<endl;
    multimap<danseur,score>::iterator  it;
    for(it=r.myMap.begin();it!=r.myMap.end();++it)
    {

        //out<<(it*)->first.getNom();
        out<<"Le nom du danseur est:"<<it->first.getNom()<<endl;
        out<<"Le resultat sur 20 est:";
        out<<it->second.calculScore();


    }
    return out;
}
for(it=r.myMap.begin();it!=r.myMap.end();++it)
{

   //out<<(*it).first;
    out<<"Le resultat sur 20 est:\n";
    out<<it->second.calculScore();
}

void Resultat::ajout_score(danseur d, score *s){
    myMap.insert(pair<danseur,score>(d,*s));

}

void Resultat::affichResulat()
{

    multimap<danseur,score>::iterator  it;
    for(it=myMap.begin();it!=myMap.end();++it)
```

```
        {

            //cout<< it->first;
            it->first.get_danseur();
            cout<<"Le resultat sur 20 est:\n";
            cout<<it->second.calculScore()<<endl;
        }


     }




    Resultat::Resultat(int num)
    {
        this->num_res = num;
    }

    Resultat::~Resultat()
    {
    }

    Resultat::Resultat(const Resultat &r)
    {
        this->num_res = r.num_res;
    }

    Resultat& Resultat::operator=( Resultat &r)
    {
        if (this != &r)
        {
            this->num_res = r.num_res;
        }
        return *this;
    }

    istream &operator>>(istream &in, Resultat &r)
    {
        cout << "Donner le numero du resultat:";
        in >> r.num_res;

        return in;
    }

    ostream& operator<<(ostream& out,  Resultat &r)
    {

        out<<"numero du resultat:"<<r.num_res<<endl;
        multimap<danseur,score>::iterator  it;
        for(it=r.myMap.begin();it!=r.myMap.end();++it)
        {

            //out<<(it*)->first.getNom();
            out<<"Le nom du danseur est:"<<it->first.getNom()<<endl;
            out<<"Le resultat sur 20 est:";
            out<<it->second.calculScore();
```

```
    }
    return out;
}
/*for(it=r.myMap.begin();it!=r.myMap.end();++it)
{

   //out<<(*it).first;
    out<<"Le resultat sur 20 est:\n";
    out<<it->second.calculScore();
}*/

void Resultat::ajout_score(danseur d, score *s){
    myMap.insert(pair<danseur,score>(d,*s));

}

void Resultat::affichResulat()
{

    multimap<danseur,score>::iterator  it;
    for(it=myMap.begin();it!=myMap.end();++it)
    {

        //cout<< it->first;
        it->first.get_danseur();
        cout<<"Le resultat sur 20 est:\n";
        cout<<it->second.calculScore()<<endl;
    }


 }
```

Test execution classe Resultat

4.10 Classe Resulatet partiel

ResultatPartiel.h

```
#include "Resultat.cpp"
#include <string>
#include<vector>
using namespace std;
class ResultatPartiel:public Resultat
{
private:

    string commentairePartiel;
    vector<score *> scorePartiel;

public:
    ResultatPartiel(){}
    ResultatPartiel(int,  string=" ");
    ResultatPartiel& operator=(  ResultatPartiel&);
    friend ostream& operator<<(ostream&, ResultatPartiel&);
    friend istream& operator>>(istream&, ResultatPartiel&);
    ResultatPartiel(const ResultatPartiel&);

    void affichResulat();
    void ajout_score(danseur d, score *s);
    ~ResultatPartiel();
};
```

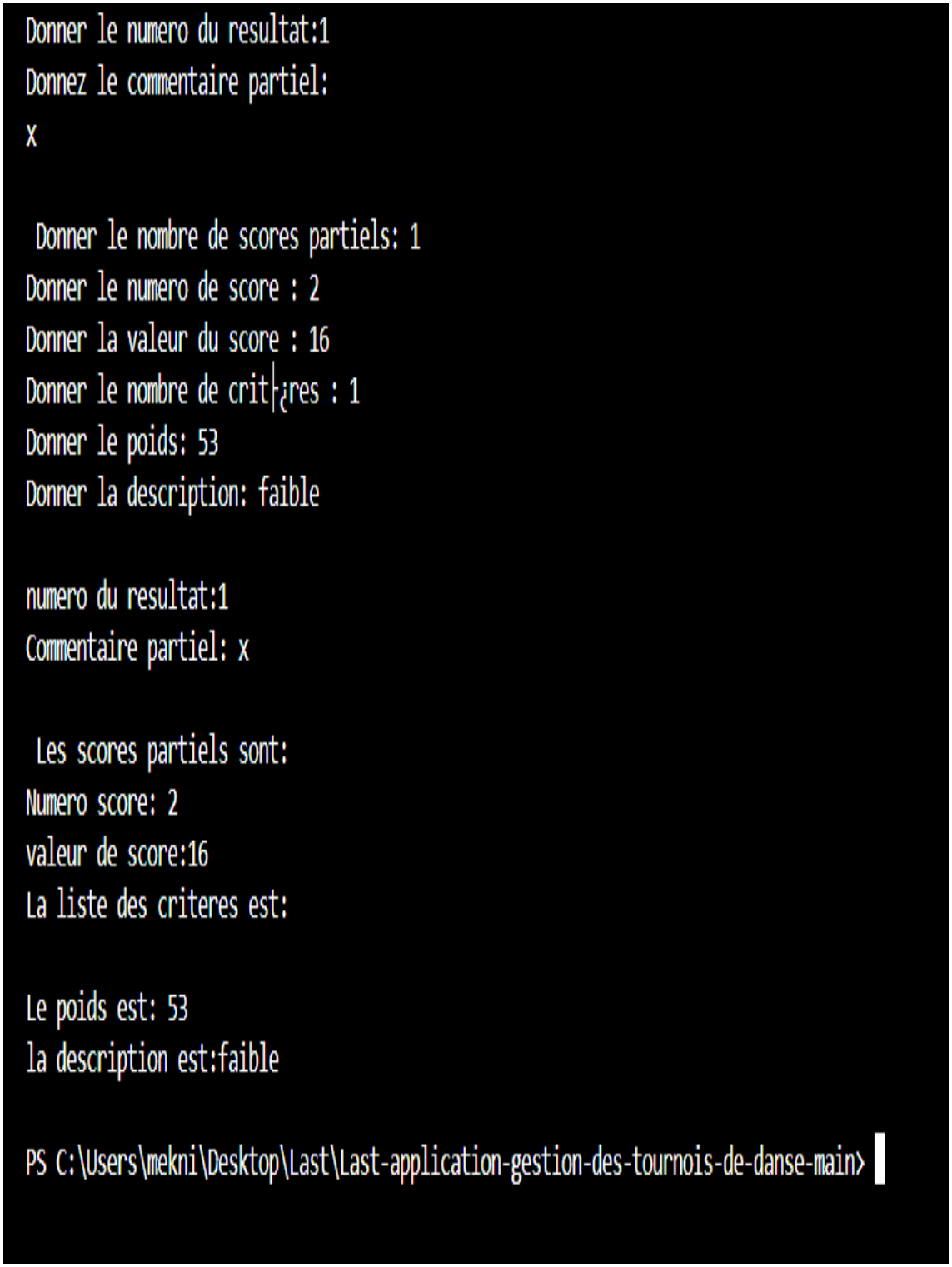
ResultatTotal.cpp

```
#include "ResultatPartiel.h"
#include<iostream>
#include<ios>

//constrecteur
ResultatPartiel::ResultatPartiel(int num,  string com):Resultat(
   num)
{
    this->commentairePartiel = com;
    int choix;
    score *s;
    do{
        //s=new score(*scorePartiel[i]);
        s->enregistrerScore();
        this->scorePartiel.push_back(s);
        cout<<"Tapez 1 pour continuer ou autre pour arreter";
        cin>>choix;
    }while (choix==1);
```

```
}
//destrecteur
ResultatPartiel::~ResultatPartiel()
{

    for(unsigned int i=0;i<scorePartiel.size();i++)
    {
        delete scorePartiel[i];
    }
    scorePartiel.clear();
}
//constrecteur par recopie
ResultatPartiel::ResultatPartiel(const ResultatPartiel& r):
   Resultat(r)
{
    this->commentairePartiel= r.commentairePartiel;
    score *s;
    for(unsigned int i=0;i<r.scorePartiel.size();i++)
    {
        s=new score(*r.scorePartiel[i]);
        this->scorePartiel.push_back(s);
    }
}
//surcharger operateur>>
istream& operator>>(istream& in , ResultatPartiel& r)
{
    unsigned int nbr;
    Resultat *re= &r;
    in>>*re;
    cout<<"Donnez le commentaire partiel:\n";
    in>>r.commentairePartiel;
    cout<<"\n Donner le nombre de scores partiels: ";
    in>>nbr;
    for(unsigned int i=0;i<nbr;i++)
    {
        score *s = new score();
        in>>*s;
        r.scorePartiel.push_back(s);
    }
    cout<<"\n";

    return in;
}
//surcharger operateur<<
ostream& operator<<(ostream& out, ResultatPartiel& r)
{
    Resultat *re=&r;
    out<<*re;
    out<<"Commentaire partiel: "<<r.commentairePartiel<<endl;

    out<<"\n Les scores partiels sont: "<<endl;
    for(unsigned int i=0;i<r.scorePartiel.size();i++)
    {
        out<<*r.scorePartiel[i]<<"\t";
    }
    out<<"\n";
```

```
        return out;
    }
    //surcharger operateur=
    ResultatPartiel& ResultatPartiel::operator=(  ResultatPartiel& rp
       )
    {
        if(this!=&rp)
        {
            Resultat *r1=this;
            Resultat *r2= &rp;
            *r1= *r2;

            for(unsigned int i=0;i<scorePartiel.size();i++)
            {
                delete scorePartiel[i];
            }
            scorePartiel.clear();
            score *s;
            for(unsigned int i=0;i<scorePartiel.size();i++)
            {
                s= new score(*rp.scorePartiel[i]);
                scorePartiel.push_back(s);
            }
        }
        return *this;
    }

    //meth pour l'ajout de score
    void ResultatPartiel::ajout_score(danseur d, score *s) {
        // Ajouter le score au tableau de scores
        scorePartiel.push_back(s);

        // Ajouter le couple (danseur, score)    la map dans la
       classe de base Resultat
        this->Resultat::ajout_score(d, s);
    }
    //meth pour afficher resultat
    void ResultatPartiel::affichResulat()
    {
        Resultat *re=this;
        cout<<*re;
        cout<<"Commentaire partiel: "<<commentairePartiel<<endl;

        cout<<"\n Les scores partiels sont: "<<endl;
        for(unsigned int i=0;i<scorePartiel.size();i++)
        {
            cout<<*scorePartiel[i]<<"\t";
        }
        cout<<"\n";
    }
```

Test execution classe ResultatPartiel



4.11 Classse ResultatTotal

ResultatTotal.h

```
      #include "Resultat.cpp"
#include <string>
#include<vector>
using namespace std;
class ResultatTotal:public Resultat
{
private:


    vector<score *> scoretotal;

public:
    ResultatTotal(){}
    ResultatTotal(int);
    ResultatTotal& operator=(  ResultatTotal&);
    friend ostream& operator<<(ostream&, ResultatTotal&);
    friend istream& operator>>(istream&, ResultatTotal&);
    ResultatTotal(const ResultatTotal&);

    void affichResulat();
    void ajout_score(danseur d, score *s);
    void afficherClassementDanseur();
    ~ResultatTotal();
};
```

ResultatTotal.cpp

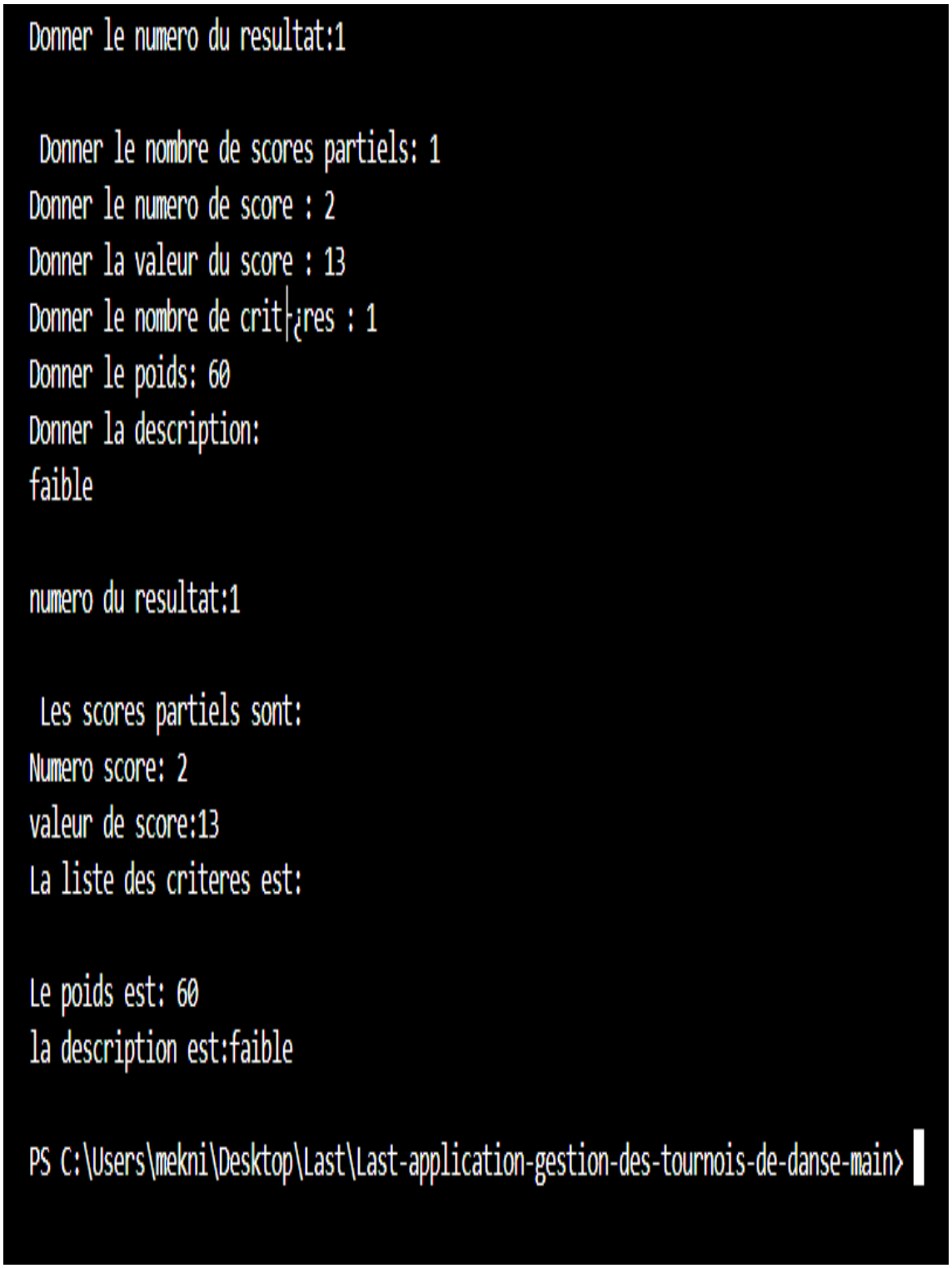
```
#include "ResultTotal.h"
#include<iostream>
#include<ios>
//constrecteur
ResultatTotal::ResultatTotal(int num):Resultat(num)
{

    int choix;
    score *s;
    do{
        //s=new score(*scorePartiel[i]);
        s->enregistrerScore();
        this->scoretotal.push_back(s);
        cout<<"Tapez 1 pour continuer ou autre pour arreter";
        cin>>choix;
    }while (choix==1);
```

```
}
//destrecteur
ResultatTotal::~ResultatTotal()
{
    for(unsigned int i=0;i<scoretotal.size();i++)
    {
        delete scoretotal[i];
    }
    scoretotal.clear();
}
//constrecteur par recopie
ResultatTotal::ResultatTotal(const ResultatTotal& r):Resultat(r)
{

    score *s;
    for(unsigned int i=0;i<r.scoretotal.size();i++)
    {
        s=new score(*r.scoretotal[i]);
        this->scoretotal.push_back(s);
    }
}
//surcharger operateur>>
istream& operator>>(istream& in , ResultatTotal& r)
{
    int nbr;
    Resultat *re= &r;
    in>>*re;

    cout<<"\n Donner le nombre de scores partiels: ";
    in>>nbr;
    for(  int i=0;i<nbr;i++)
    {
        score *s = new score();
        in>>*s;
        r.scoretotal.push_back(s);
    }
    cout<<"\n";

    return in;
}
//surcharger operateur<<
ostream& operator<<(ostream& out, ResultatTotal& r)
{
    Resultat *re=&r;
    out<<*re;


    out<<"\n Les scores partiels sont: "<<endl;
    for(unsigned int i=0;i<r.scoretotal.size();i++)
    {
        out<<*r.scoretotal[i]<<"\t";
    }
    out<<"\n";

    return out;
}
//surcharger operateur=
```

```
ResultatTotal& ResultatTotal::operator=(  ResultatTotal& rp)
{
    if(this!=&rp)
    {
        Resultat *r1=this;
        Resultat *r2= &rp;
        *r1= *r2;

        for(unsigned int i=0;i<scoretotal.size();i++)
        {
            delete scoretotal[i];
        }
        scoretotal.clear();
        score *s;
        for(unsigned int i=0;i<rp.scoretotal.size();i++)
        {
            s= new score(*rp.scoretotal[i]);
            scoretotal.push_back(s);
        }
    }
    return *this;
}

//meth pour ajout des scores
void ResultatTotal::ajout_score(danseur d, score *s) {
    // Ajouter le score au tableau de scores
    scoretotal.push_back(s);

    // Ajouter le couple (danseur, score)    la map dans la
   classe de base Resultat
    this->Resultat::ajout_score(d, s);
}


//meth pour affiche classement
void ResultatTotal::afficherClassementDanseur() {
    // Cr er une copie de la multimap pour pouvoir la trier
    multimap<int, danseur> classement;
     multimap<danseur,score>::iterator it;
    // Remplir la copie avec les scores totaux et les danseurs
    for (auto it = this->myMap.begin(); it != myMap.end(); ++it)
   {
        int totalScore = 0;
        // Calculer le score total du danseur
        for (score* s : scoretotal) {
            totalScore += s->calculScore() ;
        }
        classement.insert({ totalScore, it->first });
    }

    // Afficher le classement des danseurs
    int rank = 1;
    cout << "Classement des danseurs :" << endl;
    for (auto it = classement.rbegin(); it != classement.rend();
   ++it) {
        cout << "Position " << rank++ << " : " << it->second << "
    - Score total : " << it->first << endl;
    }
```

```
}
//meth pour affiche de resultat
void ResultatTotal::affichResulat()
{
    Resultat::affichResulat();
}
```

Test execution ResultatTotal



4.12 Classe Score

Score.h

```
  #include<iostream>
#include<vector>
#include "critere.cpp"
#include <ios>
using namespace std;
class score
{
private:

    int numScore;
    int valeur;
    vector<Critere *>  listCrit;

public:
    score(int=0,int=0);
    score(const score&);
    void enregistrerScore();
    score& operator=(const score&);
    float calculScore();
    friend istream& operator>>(istream&, score&);
    friend ostream& operator<<(ostream&, score&);

    ~score();
};
```

Score.cpp

```
  #include<iostream>
#include "score.h"
#include<vector>
#include "critere.cpp"
#include <ios>
using namespace std;
score::score(int num, int val)
{
    this->valeur=val;
    this->numScore=num;
}

score::~score()
{
    for(unsigned int i=0;i<listCrit.size();i++)
    {
        delete listCrit[i];
    }
    listCrit.clear();
}
score::score(const score&s)
{
    this->numScore=s.numScore;
    this->valeur= s.valeur;
```

```
        Critere *c;
        for(unsigned  int i=0;i<s.listCrit.size();i++)
        {
            c= new Critere(*s.listCrit[i]);
            listCrit.push_back(c);
        }
    }


    void score::enregistrerScore()
    {
        cout<<"Donner le numero score: ";
        cin>>numScore;
        cout<<"Donner la valeur: ";
        cin>>valeur;
        cout<<"Donner les criteres "<<endl;
        Critere *c;
        int choix;
        do{
            c=new Critere;
             c->saisirCritere();
            listCrit.push_back(c);

            cout<<"Voulez vous ajouter d autres critere: Tapez 1 pour
        continuer ou autres pour arreter: ";
            cin>>choix;

        }while(choix==1);
    }
    score& score::operator=(const score& s)
    {
        cout<<"\n DEBUT DE LA SURDEFINITION\n";
        if(this!=&s)
        {
            this->numScore= s.numScore;
            this->valeur=s.valeur;
             for(unsigned int i=0;i<listCrit.size();i++)
            {
                delete listCrit[i];
            }
            listCrit.clear();
            for (unsigned int i=0; i<s.listCrit.size();i++)
            {
                listCrit[i]=s.listCrit[i];
            }

        }
        cout<<"\n FIN DE LA SURDEFINITION\n";
        return *this;
    }

    istream& operator>>(istream& in, score& s) {
        cout << "Donner le numero de score : ";
        in >> s.numScore;
        cout << "Donner la valeur du score : ";
        in >> s.valeur;

        // Saisie des crit res
```

```
        int nbCrit;
        cout << "Donner le nombre de crit res : ";
        in >> nbCrit;
        s.listCrit.clear();
        for (int i = 0; i < nbCrit; ++i) {
            Critere* critere = new Critere();
            in >> *critere;
            s.listCrit.push_back(critere);
        }

        return in;
    }

    ostream& operator<<(ostream& out, score& s)
    {
        out<<"Numero score: "<<s.numScore<<endl;
        out<<"valeur de score:"<<s.valeur<<endl;
        out<<"La liste des criteres est:\n";
        for(unsigned int i=0;i<s.listCrit.size();i++)
        {
            out<<*s.listCrit[i]<<endl;

        }
        return out;
    }

    float score::calculScore()
    {
        float res=0.0;
        res+= valeur *0.65;
        int n= listCrit.size();
        float rcri=0;
        for(unsigned int  i=0;i<listCrit.size();i++)
        {
            if(listCrit[i]->getDescription()=="faible")
                rcri+=4;
                else
                if(listCrit[i]->getDescription()=="moyen")
                    rcri+=7;
                    else if(listCrit[i]->getDescription()=="fort")
                         rcri+=9;
                         else
                         rcri+=2;

            if(listCrit[i]->getPoids() > 50)
                 rcri+=4;
                 else
                 rcri+=8;
        }
        float rescri=0.35*( rcri /n);

        res+=rescri;


        return res;

    }
```

Test execution classe Score

```
--------Enregistrement score--------
Donner le numero score: 1
Donner la valeur: 18
Donner les criteres
Donner le poids: 78
Donner la description: faible
Voulez vous ajouter d autres critere: Tapez 1 pour continuer ou autres pour arreter: 1
Donner le poids: 100
Donner la description: moyen
Voulez vous ajouter d autres critere: Tapez 1 pour continuer ou autres pour arreter: 0

--------affichage score par operator <<--------
Numero score: 1
valeur de score:18
La liste des criteres est:

Le poids est: 78
la description est:faible

Le poids est: 100
la description est:moyen

--------Enregistrement score par operator >>--------
Donner le numero de score : 2
Donner la valeur du score : 16
Donner le nombre de crit¿res : 2
Donner le poids: 110
Donner la description: faible
Donner le poids: 50
Donner la description: fort
```

4.13 Classe Critere

Critere.h

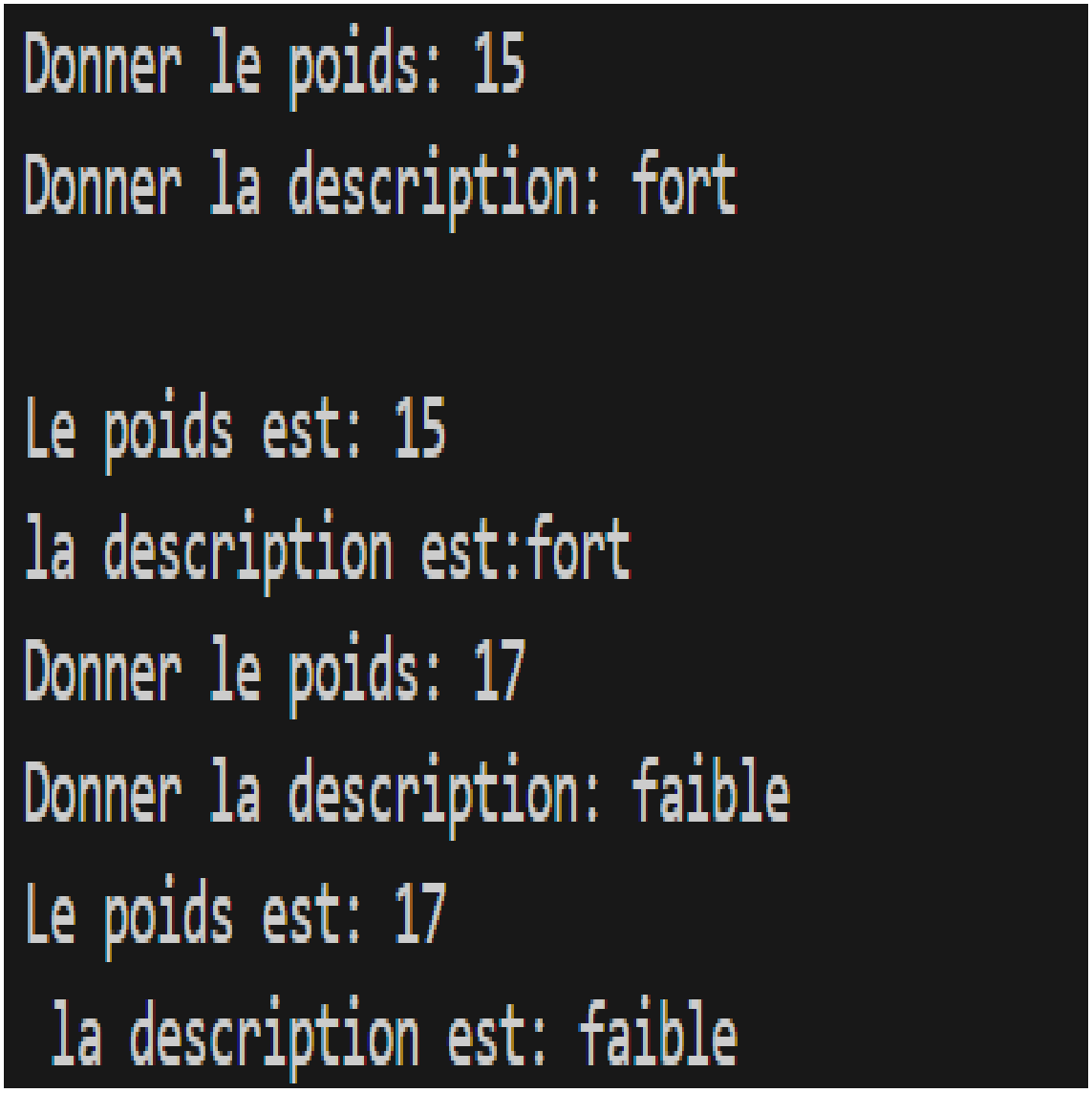
```
#include <string>
#include<ios>
using namespace std;
class Critere
{
private:

    int poids;
    string description;
public:
    Critere(int=0, string=" ");
    Critere(const Critere&);
    void saisirCritere();
    friend ostream& operator<<(ostream&, Critere&);
    friend istream& operator>>(istream&, Critere&);
    void afficher();
    int getPoids(){return poids;}
    string getDescription(){return description;}

    ~Critere(){}
};
```

Critere.cpp

```
  #include<iostream>
#include "critere.h"
#include <string>
#include<ios>
using namespace std;


Critere::Critere(int poid, string descrip)
{
    this->poids= poid;
    this->description= descrip;
}


Critere::Critere(const Critere& c)
{
    poids=c.poids;
    description=c.description;
}
void Critere::afficher()
{
    cout<<"Le poids est: "<<poids<<"\n la description est: "<<
   description<<endl;
}
```

```
void Critere::saisirCritere()
{
    cout<<"Donner le poids: ";
    cin>>poids;
    cout<<"Donner la description: ";
    cin>>description;
}

ostream& operator<<(ostream& out, Critere& c)
{
    out<<"\nLe poids est: "<<c.poids;
    out<<"\nla description est:"<<c.description;
    return out;
}

istream& operator>>(istream& in , Critere& c)
{
    cout<<"Donner le poids: ";
    in>>c.poids;
    cout<<"Donner la description: ";
    in>>c.description;
    return in;

}
```

Test execution classe Critere



4.14 Classe spectateur

spectateur.h

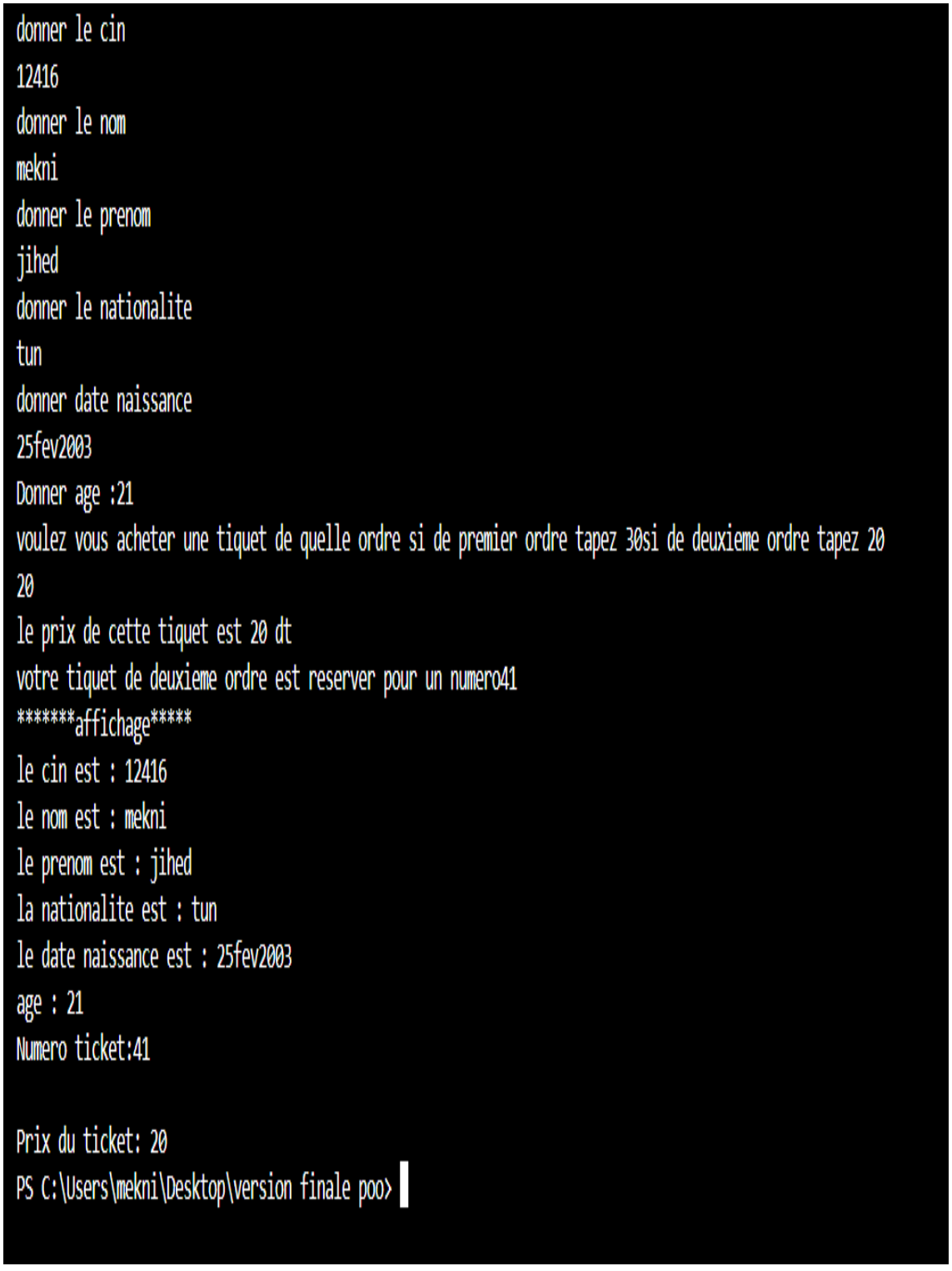
```
   #ifndef spectateur_h
#define spectateur_h
#include "personnes.cpp"
#include "Ticket.cpp"
#include <string>
#include <map>
#include <ios>
class spectateur : public personnes
{
  int age;
 multimap<spectateur, Ticket> Mymap;

public:
  spectateur();
  spectateur(const spectateur &);
  ~spectateur();
  spectateur(int, string, string, string, string, int = 0);
  void ajouter_ticket(spectateur s, Ticket *t)
  {
    Mymap.insert(pair<spectateur, Ticket>(s, *t));
  }
  void get_spectateur();
  friend ostream &operator<<(ostream &, spectateur &);
  friend istream &operator>>(istream &, spectateur &);
  spectateur &operator=(spectateur &);
  bool operator<(const spectateur &other) const { return nom <
   other.nom; }

};

#endif
```

spectateur.cpp

```
 #include <iostream>
using namespace std;
#include "spectateur.h"

#include <string>
#include "iterator"
// constructeur par deffaut pour spectateur
spectateur::spectateur()
{
    personnes();
}
// constructeur pour spectateur
spectateur::spectateur(int cin_p, string nom, string prenom,
   string nationalite, string date_naissance, int a)
{
    personnes(cin_p, nom, prenom, nationalite, date_naissance);
    age = a;
```

```
}
// destrecteur
spectateur::~spectateur() {}
spectateur::spectateur(const spectateur &r)
{
    this->age = r.age;
}

// methode geter pour spectateur
void spectateur::get_spectateur()
{
    personnes::get_personne();
}
//surcharger operateur<<
ostream &operator<<(ostream &out, spectateur &s)
{
    out << "le cin est : " << s.cin_p << endl;
    out << "le nom est : " << s.nom << endl;
    out << "le prenom est : " << s.prenom << endl;
    out << "la nationalite est : " << s.nationalite << endl;
    out << "le date naissance est : " << s.date_naissance << endl
   ;
    out << "age : " << s.age<<endl;
    /*for (auto &pair : s.Mymap)
    {

        auto x = pair.first;
        out << x;
        auto s = pair.second;
        out << s;
    }*/

    return out;
}
//surcharger operateur>>
istream &operator>>(istream &in, spectateur &s)
{
    personnes *p = &s;
    in >> *p;
    cout << "Donner age :";
    in >> s.age;

    return in;
}
//surcharger operateur=
spectateur &spectateur::operator=(spectateur &s)
{
    if (this != &s)
    {
        this->cin_p = s.cin_p;
    }
    return *this;
}
```

Test execution classe spectateur



4.15 Classe Tiquet

Tiquet.h

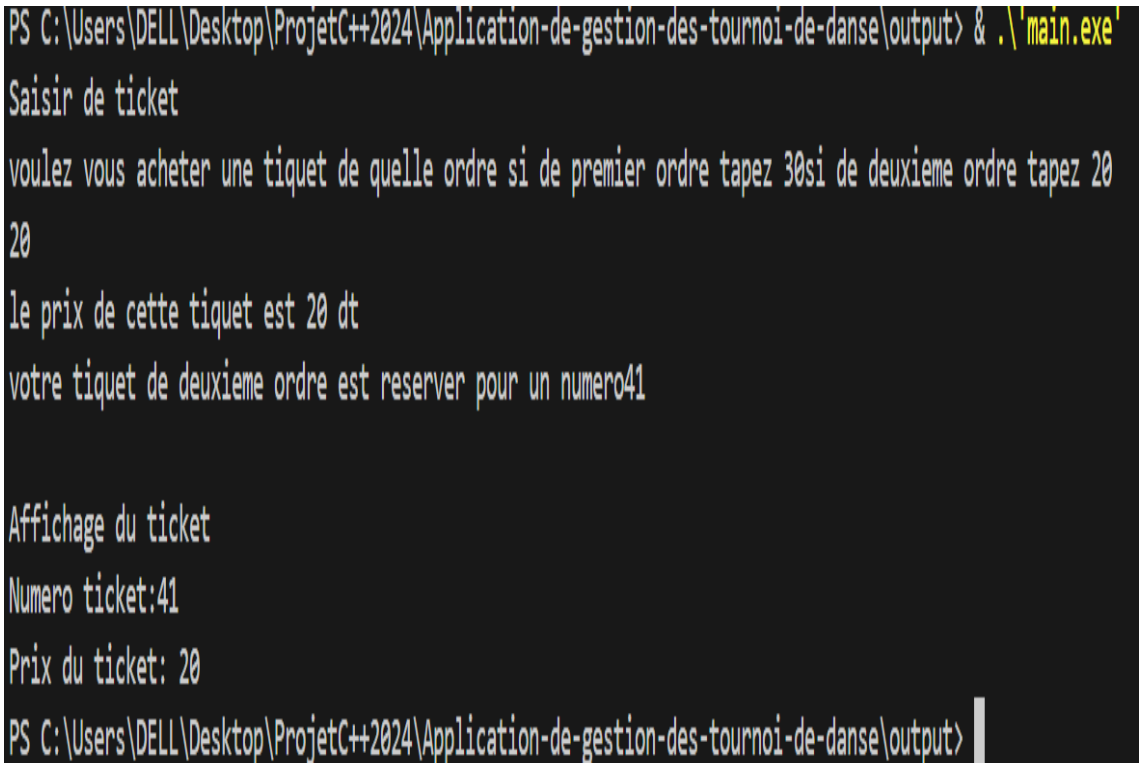
```
    #include<ios>
#include <random>
class Ticket
{
private:

    int num_ticket;
    float prix;
public:
    Ticket(int=0,float=100);
    ~Ticket();
    Ticket(const Ticket&);
    Ticket& operator=(const Ticket&);
    friend istream& operator>>(istream& in , Ticket&);
    friend ostream& operator<<(ostream& , Ticket&);
};
```

Tiquet.cpp

```
#include "Ticket.h"
#include<iostream>
using namespace std;
//constrecteur
Ticket::Ticket(int num ,float prix)
{
    this->num_ticket= num;
    this->prix= prix;
}
//destrecteur
Ticket::~Ticket()
{
}
//constrecteur par recopie
/// @param t
Ticket::Ticket(const Ticket& t)
{
    this->num_ticket = t.num_ticket;
    this->prix = t.prix;
}
//surcharger operateur=
Ticket& Ticket::operator=(const Ticket& t)
{
    if(this!=&t)
    {
        this->num_ticket=t.num_ticket;
        this->prix=t.prix;
    }
    return *this;
```

```
}
//surcharger operateur>>
istream& operator>>(istream& in, Ticket& t)
{

    cout<<"voulez vous acheter une tiquet de quelle ordre si de
   premier ordre tapez 30si de deuxieme ordre tapez 20"<<endl;
    in>>t.prix;

    if(t.prix==30){
        cout<<"le prix de cette tiquet est 30 dt"<<endl;
        t.num_ticket=rand();
        cout<<"votre tiquet de premier ordre est reserver pour un
    numero"<<t.num_ticket<<endl;

    }
    if(t.prix==20){
        cout<<"le prix de cette tiquet est 20 dt"<<endl;
        t.num_ticket=rand();
        cout<<"votre tiquet de deuxieme ordre est reserver pour
   un numero"<<t.num_ticket<<endl;

    }
    return in ;
}
//surcharger operateur<<
ostream& operator<<(ostream& out, Ticket& t)
{
    out<<"Numero ticket:"<<t.num_ticket<<endl;
    out<<"\nPrix du ticket: "<<t.prix;
    return out;
}
```

Test execution classe Tiquet



5 MENU

5.1 Menu Principal

```
int main()
{


     int choix;
    do {
        // Affichage du menu principal
        cout << "\nMenu de l application de gestion de tournoi de
 danse" << endl;
        cout << "1. Gestion des epreuves" << endl;
        cout << "2. Gestion des categories" << endl;
        cout << "3. Gestion des danseurs " << endl;
        cout<<  "4. Gestion des  spectateur"<<endl;
        cout << "5. Attribution des resultats" << endl;
        cout << "6. Gestion des recompenses" << endl;
        cout << "0. Quitter l application" << endl;
        cout << "Votre choix : ";
        cin >> choix;


        switch (choix) {
            case 1:
                // Organisation des  preuves
                organiserEpreuves() ;

                break;
            case 2:
                gestionCategories();
                break;
            case 3:
                //Gestion des danseurs
                enregistrerDanseur();

                break;
            case 4:
                     //Gestion des spectateurs
                    enregistrerSpectateur();
                break;
            case 5:
                    // Attribution des scores
                    gestionScores(listEpreuves, listDanseur);

                break;
            case 6:
                // Gestion des r compenses
                gestionRecompense( listRecompenses,listEpreuves,
 listDanseur);
                break;
            case 0:
                cout << "Merci d'avoir utilise l'application. Au
revoir !" << endl;
                break;
            default:
```

```
                cout << "Choix invalide. Veuillez entrer un
    nombre entre 0 et 6." << endl;
                break;
            }
        } while (choix != 0);

        for(unsigned int i=0;i<listDanseur.size();i++)
        {
            delete listDanseur[i];
        }
        listDanseur.clear();

        for(unsigned int i=0;i<listSpectateur.size();i++)
        {
            delete listSpectateur[i];
        }
        listSpectateur.clear();

        for(unsigned int i=0;i<listEpreuves.size();i++)
        {
            delete listEpreuves[i];
        }
        listEpreuves.clear();
        return 0;
    }
```

5.2 Fonction utilises Dans le Menu

```
    #include <iostream>
    #include <algorithm> // Pour la fonction sort()
    #include<typeinfo> //pour le casting
    #include <vector>
    #include <string>
    #include "personnes.cpp"
    #include "danseur_amateur.cpp"
    #include "danseur_pro.cpp"
    #include "Resultat.h"
    #include "danseur.cpp"
    #include "epreuve.cpp"
    #include "spectateur.cpp"
    #include "Recompenses.cpp" //  le fichier d'en-t te des
       r compenses


    using namespace std;

    vector<danseur*> listDanseur;
    vector<spectateur*> listSpectateur;
    vector<Epreuves*> listEpreuves;
    vector<Recompenses*> listRecompenses;

    void organiserEpreuves() {
        // Cr ation d'un objet pour g rer les  preuves
        Epreuves epreuves;
```

```
    int choix;
    do {
        cout << "\nOrganisation des epreuves :" << endl;
        cout << "1. Creer une nouvelle epreuve" << endl;
        cout << "2. Afficher les epreuves existantes" << endl;
        cout << "0. Retour au menu principal" << endl;
        cout << "Votre choix : ";
        cin >> choix;

        switch (choix) {
            case 1:
                {
                    cin.ignore(); // Pour  viter  les probl mes
    avec getline()
                    cout << "Entrez les d tails de la nouvelle
     preuve  :" << endl;
                    cin >> epreuves; // Utilisation de l'
    op rateur de saisie surcharg

                    Epreuves *e=new Epreuves(epreuves);
                    listEpreuves.push_back(e);
                    cout << " preuve  cr  e avec succ s !" <<
    endl;
                }
                break;
            case 2:
                {
                    cout << " preuves  existantes :" << endl;
                    //Trier la liste des epreuves pour afficher
    en ordre alphabetique
                    sort(listEpreuves.begin(), listEpreuves.end()
    , Epreuves::compareNomsEpreuves);
                    for(unsigned int i=0;i<listEpreuves.size();i
    ++)
                    {
                        cout<<*listEpreuves[i];
                        cout<<"------------------------------\n";
                    }
                }
                break;
            case 0:
                cout << "Retour au menu principal." << endl;
                break;
            default:
                cout << "Choix invalide. Veuillez entrer un
    nombre entre 0 et 2." << endl;
                break;
        }
    } while (choix != 0);
}

void enregistrerDanseur() {
    int choixType;
    do{
    cout << "Choisissez le type de danseur  enregistrer : " <<
    endl;
    cout << "1. Danseur " << endl;
```

```
          cout << "2.Danseur Amateur" << endl;
         cout << "3. Danseur Professionnel" << endl;
         cout<< "4: Afficher la liste de tous les danseurs"<<endl;
         cout<< "0: Retour au menu principal"<<endl;
         cout << "Votre choix : ";
         cin >> choixType;

         switch (choixType) {
             case 1:
                {
                     danseur d1;
                     d1.enregistrer_danseur();
                     danseur *d=new danseur(d1);
                     listDanseur.push_back(d);
                     cout << "Danseur amateur enregistre avec succes !
        " << endl;
                 }
                 break;
             case 2:
                 {
                     danseur_amateur nouveauDanseurAmateur;
                     nouveauDanseurAmateur.enregistrer_danseur_amateur
        ();
                     danseur_amateur *da=new danseur_amateur(
        nouveauDanseurAmateur);
                     listDanseur.push_back(da);

                     cout << "Danseur amateur enregistre avec succes !
        " << endl;
                 }
                 break;
             case 3:
                     {
                     danseur_pro nouveauDanseurPro;
                     nouveauDanseurPro.enregistrer_danseur_pro();
                     danseur_pro *dp=new danseur_pro(nouveauDanseurPro
        );
                     listDanseur.push_back(dp);
                     cout << "Danseur professionnel enregistr  avec
        succ s !" << endl;
                 }
                 break;
             case 4:
                             //Trier la liste des danseurs
                             danseur *d;
                        sort(listDanseur.begin(),listDanseur.end());
                        for(unsigned int i=0;i<listDanseur.size();i
        ++)
                        {
                             if(typeid(*listDanseur[i])==typeid(
        danseur))
                                 cout<<*listDanseur[i];
                             else
                             if(typeid(*listDanseur[i])==typeid(
        danseur_amateur))
                             {
                                     d=new danseur_amateur(static_cast
        <const danseur_amateur&>(*listDanseur[i]));
```

```
                                    cout<<d;

                            }
                            else
                            {
                                d=new danseur_pro(static_cast<const
    danseur_pro&>(*listDanseur[i]));
                                cout<<d;
                            }
                            cout<<"-----------------\n";
                        }
            default:
                cout << "Choix invalide. Veuillez entrer un chiffre
    de 1 a 4." << endl;
                break;
     }
     }while (choixType!=0);


}


void enregistrerSpectateur() {
    int choixType;
    spectateur s;
     do{
    cout << "1. enregistrer spectateur " << endl;
     cout << "2. Acheter ticket" << endl;
    cout<< "3. Affichage de tous les spectateurs"<<endl;
    cout << "0. Retour au menu principal" << endl;
    cout << "Votre choix : ";
    cin >> choixType;

    switch (choixType) {
        case 1:
           {


                cin>>s;
                spectateur *s1=new spectateur(s);
                listSpectateur.push_back(s1);
                cout << "Spectateur enregistre avec succes !" <<
    endl;
            }
            break;
        case 2:
            {
                 Ticket t;
                 cin>>t;
                 s.ajouter_ticket(s,&t);
            }
            break;
        case 3:
            for(unsigned int i=0;i<listSpectateur.size();i++)
            {
                cout<<*listSpectateur[i];
                cout<<"-----------------\n";
            }
```

```
                break;
            case 0:
                    cout << "Retour au menu principal." << endl;
                    break;
            default:
                cout << "Choix invalide. Veuillez entrer 1 ou 2." <<
    endl;
                break;
         }
         }while(choixType!=0);
    }

    void gestionCategories() {
         vector<string> categories;

         int choix;
         do {
             cout << "\nGestion des categories de danse :" << endl;
             cout << "1. Ajouter une categorie" << endl;
             cout << "2. Afficher les categories existantes" << endl;
             cout << "0. Retour au menu principal" << endl;
             cout << "Votre choix : ";
             cin >> choix;

             switch (choix) {
                 case 1:
                     {
                         string nouvelleCategorie;
                         cout << "Entrez le nom de la nouvelle
    categorie : ";
                         cin >> nouvelleCategorie;
                         categories.push_back(nouvelleCategorie);
                         cout << "Cat gorie ajoutee avec succes !" <<
         endl;
                     }
                     break;
                 case 2:
                     {
                         if (categories.empty()) {
                             cout << "Aucune cat gorie n'a   t
    ajout e pour le moment." << endl;
                         } else {
                             cout << "Cat gories existantes :" <<
    endl;
                             for (const auto& categorie : categories)
    {
                                 cout << "- " << categorie << endl;
                             }
                         }
                     }
                     break;
                 case 0:
                     cout << "Retour au menu principal." << endl;
                     break;
                 default:
                     cout << "Choix invalide. Veuillez entrer un
    nombre entre 0 et 2." << endl;
                     break;
```

```
            }
        } while (choix != 0);
    }

    // D finition de la fonction gestionScores
    void gestionScores(const std::vector<Epreuves*>& listeEpreuves,
        const std::vector<danseur*>& listeDanseurs) {
        int choix;
        do {
            // Afficher le sous-menu de gestion des scores
            std::cout << "\n===== GESTION DES SCORES =====" << std::
        endl;
            std::cout << "1. Attribuer un score    un danseur" << std
        ::endl;
            std::cout << "2. Afficher les r sultats des danseurs d'
        une epreuve" << std::endl;
            std::cout << "0. Retour au menu principal" << std::endl;
            std::cout << "Choix : ";
            std::cin >> choix;

            // Effectuer une action en fonction du choix de l'
        utilisateur
            switch (choix) {
                case 1: {
                    // Attribuer un score    un danseur
                    if (listeEpreuves.empty()) {
                        std::cout << "Aucune epreuve enregistree.
        Veuillez enregistrer une epreuve avant d'ajouter un resultat."
        << std::endl;
                    } else {
                        unsigned int  index;
                        std::cout << "Choisissez l epreuve auquel
        attribuer un resultat : " << std::endl;
                        for (size_t i = 0; i < listeEpreuves.size();
        ++i) {
                            std::cout << i+1 << ". " << listeEpreuves
        [i]->getNom() << std::endl;
                        }
                        cout << "Indice de l epreuve : ";
                        cin >> index;
                        if (index >= 1 && index <= listeEpreuves.size
        ()) {
                            if (listeDanseurs.empty()) {
                        std::cout << "Aucun danseur enregistr .
        Veuillez enregistrer un danseur avant d'attribuer un score."
        << std::endl;
                    } else {
                        unsigned int  indexDanseur;
                        std::cout << "Choisissez le danseur auquel
        attribuer un score : " << std::endl;
                        for (size_t i = 0; i < listeDanseurs.size();
        ++i) {
                            std::cout << i+1 << ". " << listeDanseurs
        [i]->getNom() << std::endl;
                        }
                        cout << "Indice du danseur : ";
                        cin >> indexDanseur;
```

```
                    if (indexDanseur >= 1 && indexDanseur <=
listeDanseurs.size()) {
                        Resultat resultat;
                        cin>>resultat;
                        score s;
                        cin>>s;
                        score *nouveauScore = new score(s);
                        resultat.ajout_score(*listeDanseurs[
indexDanseur - 1], nouveauScore);
                        delete nouveauScore;
                         listeEpreuves[index-1]->ajouter_resultat
(resultat);

                        std::cout << "Score attribue avec succes
au danseur : " << listeDanseurs[indexDanseur - 1]->getNom() <<
 std::endl;
                        std::cout << "Resultat ajouter avec
succes a l epreuve : " << listeEpreuves[index - 1]->getNom()
<< std::endl;
                    } else {
                        std::cout << "Indice danseur invalide."
<< std::endl;
                    }
                }

                    } else {
                        std::cout << "Indice resultat invalide."
<< std::endl;
                    }
                }

                break;
            }
            case 2: {
                // Afficher les r sultats de chaque danseureur
                unsigned int  index;
                    std::cout << "Choisissez l epreuve auquel
vous voulez afficher ces resultats : " << std::endl;
                    for (size_t i = 0; i < listeEpreuves.size();
++i) {
                        std::cout << i+1 << ". " << listeEpreuves
[i]->getNom() << std::endl;
                    }
                     cout << "Indice de l epreuve : ";
                     cin >> index;
                     if (index >= 1 && index <= listeEpreuves.
size()) {

                        listeEpreuves[index-1]->
afficher_Liste_Resultat();

                    } else {
                        std::cout << "Indice invalide." << std::
endl;
                    }

                break;
            }
```

```
                case 0:
                    std::cout << "Retour au menu principal." << std::
    endl;
                    break;
                default:
                    std::cout << "Choix invalide. Veuillez reessayer.
    " << std::endl;
                    break;
            }
        } while (choix != 0);
    }

    //Fonction pour afficher le classe des danseurs dans une epreuve
    void afficherClassementDanseur( Epreuves e,int indiceResultat) {
        // Cr er une copie de la multimap pour pouvoir la trier
        multimap<int, danseur> classement;
         multimap<danseur,score>::iterator it;
         multimap<danseur,score> mp;
         list<Resultat *>::iterator it1;
        // Remplir la copie avec les scores totaux et les danseurs

            list<Resultat *> lr=e.getListResultat();

            int arret=0;
            for(it1=lr.begin();it1!=lr.end();++it1)
            {
                if(indiceResultat==arret) break;
                arret++;
            }
            Resultat *r=new Resultat(**it1);
           mp=r->getMyMap();
        for (auto it = mp.begin(); it !=mp.end(); ++it) {
            int totalScore = it->second.calculScore();
            classement.insert({ totalScore, it->first });
        }

        // Afficher le classement des danseurs
        int rank = 1;
        cout << "Classement des danseurs :" << endl;
        for (auto it = classement.rbegin(); it != classement.rend();
       ++it) {
            cout << "Position " << rank++ << " : " << it->second << "
        - Score total : " << it->first << endl;
        }
    }

    void gestionRecompense(vector<Recompenses*>& listeRecompenses,
       const vector<Epreuves*>& listeEpreuves,  const vector<danseur
       *>& listeDanseurs) {
        int choix;
        do {
            // Afficher le sous-menu de gestion des r compenses
            std::cout << "\n===== GESTION DES RECOMPENSES =====" <<
       std::endl;
            std::cout << "1. Ajouter une recompense" << std::endl;
            std::cout << "2. Afficher les recompenses" << std::endl;
            std::cout << "3. Afficher le classement des danseur d une
        epreuve" << std::endl;
```

```
        std::cout << "0. Retour au menu principal" << std::endl;
        std::cout << "Choix : ";
        std::cin >> choix;

        // Effectuer une action en fonction du choix de l'
    utilisateur
        switch (choix) {
            case 1: {
                // Ajouter une r compense

                Recompenses r;
                cin>>r;
                Recompenses* nouvelleRecompense = new Recompenses
    (r);
                listeRecompenses.push_back(nouvelleRecompense);

                std::cout << "Recompense ajoutee avec succes." <<
     std::endl;
                break;
            }
            case 2: {
                // Afficher les r compenses
                std::cout << "Liste des r compenses :" << std::
    endl;
                for (size_t i = 0; i < listeRecompenses.size();
    ++i) {
                    std::cout << "Recompense " << i+1 << ": " <<
    *listeRecompenses[i] << std::endl;
                }
                break;
            }
            case 3:{
                //AFFICHAGE DU CLASSEMENT DES DANSEURS
                if (listeEpreuves.empty()) {
                    std::cout << "Aucune epreuve enregistree.
    Veuillez enregistrer une epreuve avant d'ajouter un resultat."
     << std::endl;
                } else {
                    unsigned int  index;
                    std::cout << "Choisissez l epreuve auquel
    afficher le classement : " << std::endl;
                    for (size_t i = 0; i < listeEpreuves.size();
    ++i) {
                        std::cout << i+1 << ". " << listeEpreuves
    [i]->getNom() << std::endl;
                    }
                     cout << "Indice de l epreuve : ";
                     cin >> index;
                     if (index >= 1 && index <= listeEpreuves.
    size()) {
                        if (listeDanseurs.empty()) {
                    std::cout << "Aucun danseur enregistr .
    Veuillez enregistrer un danseur avant d'attribuer un score."
    << std::endl;
                } else {
                    unsigned int  indexresultat;
                    std::cout << "Choisissez lequel des resultats
     voulez vous afficher son classement: " << std::endl;
```

```
                        cout << "Indice du resultat : ";
                        cin >> indexresultat;
                       if (indexresultat >= 1 ) {
                           afficherClassementDanseur(*listeEpreuves[
    index],indexresultat);
                       } else {
                           std::cout << "Indice resultats invalide."
     << std::endl;
                       }
                   }

                       } else {
                           std::cout << "Indice resultat invalide."
    << std::endl;
                       }
                   }

               }
               case 0:
                   std::cout << "Retour au menu principal." << std::
    endl;
                   break;
               default:
                   std::cout << "Choix invalide. Veuillez r essayer
    ." << std::endl;
                   break;
           }
       } while (choix != 0);
   }
```

5.3 Test Execution du menu

```
Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
2. Gestion des categories
3. Gestion des danseurs
4. Gestion des  spectateur
5. Attribution des resultats
6. Gestion des recompenses
0. Quitter l application
Votre choix : 1

Organisation des epreuves :
1. Creer une nouvelle epreuve
2. Afficher les epreuves existantes
0. Retour au menu principal
Votre choix : 1
Entrez les d├®tails de la nouvelle ├®preuve :
Donner le numero d epreuves:1
Donner l heure d epreuves:16h00min
Donner le nom d epreuves:festivalDeDanse
Donner le lieu d epreuves:Tunis
Donner la date d epreuves:15juillet
Donner le nombre de danseur :10
├ëpreuve cr├®├®e avec succ├¿s !

Organisation des epreuves :
1. Creer une nouvelle epreuve
2. Afficher les epreuves existantes
0. Retour au menu principal
Votre choix : 2
```

```
Organisation des epreuves :
1. Creer une nouvelle epreuve
2. Afficher les epreuves existantes
0. Retour au menu principal
Votre choix : 2
├ëpreuves existantes :

numero epreuve:1
 heure epreuves: 16h00min
Nom de l epreuve:festivalDeDanse
Lieu de l epreuve:Tunis
Date de l epreuve:15juillet
nombre de danseur:10
-----------------------------

Organisation des epreuves :
1. Creer une nouvelle epreuve
2. Afficher les epreuves existantes
0. Retour au menu principal
Votre choix : 0
Retour au menu principal.

Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
```

```
0. Quitter l application
Votre choix : 2

Gestion des categories de danse :
1. Ajouter une categorie
2. Afficher les categories existantes
0. Retour au menu principal
Votre choix : 1
Entrez le nom de la nouvelle categorie : jazz
Cat├®gorie ajoutee avec succes !

Gestion des categories de danse :
1. Ajouter une categorie
2. Afficher les categories existantes
0. Retour au menu principal
Votre choix : 1
Entrez le nom de la nouvelle categorie : hipphopp
Cat├®gorie ajoutee avec succes !

Gestion des categories de danse :
1. Ajouter une categorie
2. Afficher les categories existantes
0. Retour au menu principal
Votre choix : 2
Cat├®gories existantes :
- jazz
- hipphopp

Gestion des categories de danse :
1. Ajouter une categorie
2. Afficher les categories existantes
0. Retour au menu principal
```

```
Gestion des categories de danse :
1. Ajouter une categorie
2. Afficher les categories existantes
0. Retour au menu principal
Votre choix : 0
Retour au menu principal.

Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
2. Gestion des categories
3. Gestion des danseurs
4. Gestion des  spectateur
5. Attribution des resultats
6. Gestion des recompenses
0. Quitter l application
Votre choix : 3
Choisissez le type de danseur ├á enregistrer :
1. Danseur
2.Danseur Amateur
3. Danseur Professionnel
4: Afficher la liste de tous les danseurs
0: Retour au menu principal
Votre choix : 1
*******enregistration des danseurs*********
donner le cin
1
donner le nom
dia
donner le prenom
jihed
donner le nationalite
Sen
```

```
jihed
donner le nationalite
Sen
donner date naissance
12mars
donner le style de danse
salsa
Danseur amateur enregistre avec succes !
Choisissez le type de danseur ├á enregistrer :
1. Danseur
2.Danseur Amateur
3. Danseur Professionnel
4: Afficher la liste de tous les danseurs
0: Retour au menu principal
Votre choix : 1
*******enregistration des danseurs*********
donner le cin
2
donner le nom
mekni
donner le prenom
saoussi
donner le nationalite
tun
donner date naissance
23avril
donner le style de danse
hipp
Danseur amateur enregistre avec succes !
Choisissez le type de danseur ├á enregistrer :
1. Danseur
2.Danseur Amateur
```

```
Danseur professionnel enregistr├® avec succ├¿s !
Choisissez le type de danseur ├á enregistrer :
1. Danseur
2.Danseur Amateur
3. Danseur Professionnel
4: Afficher la liste de tous les danseurs
0: Retour au menu principal
Votre choix : 4
le cin est : 2
le nom est : mekni
le prenom est : saoussi
la nationalite est : tun
le date naissance est : 23avril
le style de danse est :hipp
-----------------
le cin est : 3
le nom est : rochdi
le prenom est : friddi
la nationalite est : tun
le date naissance est : 23jan
le style de danse est :jazz
-----------------
le cin est : 5
le nom est : bouaffif
le prenom est : mohamed
la nationalite est : Tun
le date naissance est : 25mai
le style de danse est :rock
-----------------
le cin est : 1
le nom est : dia
le prenom est : jihed
```

```
Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
2. Gestion des categories
3. Gestion des danseurs
4. Gestion des  spectateur
5. Attribution des resultats
6. Gestion des recompenses
0. Quitter l application
Votre choix : 4
1. enregistrer spectateur
2. Acheter ticket
3. Affichage de tous les spectateurs
0. Retour au menu principal
Votre choix : 1
donner le cin
1
donner le nom
aicha
donner le prenom
ba
donner le nationalite
sen
donner date naissance
17juin
Donner age :21
Spectateur enregistre avec succes !
1. enregistrer spectateur
2. Acheter ticket
3. Affichage de tous les spectateurs
```

```
2. Acheter ticket
3. Affichage de tous les spectateurs
0. Retour au menu principal
Votre choix : 2
voulez vous acheter une tiquet de quelle ordre si de premier ordre tapez 30si de deuxieme ordre tapez 20
30
le prix de cette tiquet est 30 dt
votre tiquet de premier ordre est reserver pour un numero41
1. enregistrer spectateur
2. Acheter ticket
3. Affichage de tous les spectateurs
0. Retour au menu principal
Votre choix : 3
le cin est : 1
le nom est : aicha
le prenom est : ba
la nationalite est : sen
le date naissance est : 17juin
age : 21
----------------
1. enregistrer spectateur
2. Acheter ticket
3. Affichage de tous les spectateurs
0. Retour au menu principal
Votre choix : 0
Retour au menu principal.
```

```
Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
2. Gestion des categories
3. Gestion des danseurs
4. Gestion des  spectateur
5. Attribution des resultats
6. Gestion des recompenses
0. Quitter l application
Votre choix : 5

===== GESTION DES SCORES =====
1. Attribuer un score ├á un danseur
2. Afficher les r├®sultats des danseurs d'une epreuve
0. Retour au menu principal
Choix : 1
Choisissez l epreuve auquel attribuer un resultat :
1. festivalDeDanse
Indice de l epreuve : 1
Choisissez le danseur auquel attribuer un score :
1. mekni
2. rochdi
3. bouaffif
4. dia
Indice du danseur : 1
Donner le numero du resultat:1
Donner le numero de score : 1
Donner la valeur du score : 15
Donner le nombre de crit├¿res : 1
Donner le poids: 67
Donner la description: fort
Score attribue avec succes au danseur : mekni
Resultat ajouter avec succes a l epreuve : festivalDeDanse
```

```
===== GESTION DES SCORES =====
1. Attribuer un score ├á un danseur
2. Afficher les r├®sultats des danseurs d'une epreuve
0. Retour au menu principal
Choix : 1
Choisissez l epreuve auquel attribuer un resultat :
1. festivalDeDanse
Indice de l epreuve : 1
Choisissez le danseur auquel attribuer un score :
1. mekni
2. rochdi
3. bouaffif
4. dia
Indice du danseur : 4
Donner le numero du resultat:1
Donner le numero de score : 4
Donner la valeur du score : 12
Donner le nombre de crit├¿res : 1
Donner le poids: 89
Donner la description: faible
Score attribue avec succes au danseur : dia
Resultat ajouter avec succes a l epreuve : festivalDeDanse

===== GESTION DES SCORES =====
1. Attribuer un score ├á un danseur
2. Afficher les r├®sultats des danseurs d'une epreuve
0. Retour au menu principal
```

```
Menu de l application de gestion de tournoi de danse
1. Gestion des epreuves
2. Gestion des categories
3. Gestion des danseurs
4. Gestion des  spectateur
5. Attribution des resultats
6. Gestion des recompenses
0. Quitter l application
Votre choix : 6

===== GESTION DES RECOMPENSES =====
1. Ajouter une recompense
2. Afficher les recompenses
3. Attribuer une recompense au meilleur danseur
4. Afficher le classement des danseur d une epreuve
0. Retour au menu principal
Choix : 1

Le numero de recompense est: 1

 La medaille est: or

 Le trophee est: coupe
Recompense ajoutee avec succes.

===== GESTION DES RECOMPENSES =====
1. Ajouter une recompense
2. Afficher les recompenses
3. Attribuer une recompense au meilleur danseur
4. Afficher le classement des danseur d une epreuve
0. Retour au menu principal
Choix : 1
```

```
Le numero de recompense est: 2

 La medaille est: bronze

 Le trophee est: aucun
Recompense ajoutee avec succes.

===== GESTION DES RECOMPENSES =====
1. Ajouter une recompense
2. Afficher les recompenses
3. Attribuer une recompense au meilleur danseur
4. Afficher le classement des danseur d une epreuve
0. Retour au menu principal
Choix : 2
Liste des r├®compenses :
Recompense 1:
Le numero de recompense est: 1
 La medaille est: or
 Le trophee est: coupe
Recompense 2:
Le numero de recompense est: 2
 La medaille est: bronze
 Le trophee est: aucun
```

6 Classes Templates

6.1 Templates pour la classe mére danseur

Templatedanseur.h

```
#ifndef ARCHIVE_H
#define ARCHIVE_H

#include "danseur_amateur.cpp"
#include "danseur_pro.cpp"

using namespace std;
// Classe template de base
template <typename T>
class DanseurTemplate : public T {
public:
    DanseurTemplate() : T() {}
```

```
        DanseurTemplate(int cin,  string nom, string prenom,
                        string nationalite,  string date_naissance,
    string style)
            : T(cin, nom, prenom, nationalite, date_naissance,style)
    {}

};

// Sp cialisation pour les danseurs amateurs
class DanseurAmateurTemplate : public DanseurTemplate<
    danseur_amateur> {
public:
    DanseurAmateurTemplate() : DanseurTemplate<danseur_amateur>()
    {}
    DanseurAmateurTemplate(int cin, string nom,  string prenom,
                           string nationalite, string
    date_naissance,
                           string style, string
    niveau_entrenement="")
            : DanseurTemplate<danseur_amateur>(cin, nom, prenom,
    nationalite, date_naissance,style) {
                //this->niveau_entrenement = niveau_entrenement;

            }
};

// Sp cialisation pour les danseurs professionnels
class DanseurProTemplate : public DanseurTemplate<danseur_pro> {
public:
    DanseurProTemplate() : DanseurTemplate<danseur_pro>() {}
    DanseurProTemplate(int cin, string nom,string prenom,
                       string nationalite, string date_naissance,
    string style,
                       int annee_experience=5, string palmares=""
    )
            : DanseurTemplate<danseur_pro>(cin, nom, prenom,
    nationalite, date_naissance,style) {
                //this->annee_experience = annee_experience;
                //this->palmares = palmares;
            }
};

#endif // ARCHIVE_H
```

Test execution template classe

```
-------TESTER CLASSE TEMPLATE---------
Saisir danseur amateur
donner le cin
1
donner le nom
jihed
donner le prenom
mekni
donner le nationalite
tun
donner date naissance
11nov
donner style danse
salsa
donner niveau entrenement
moyen
le cin est : 1
le nom est : jihed
le prenom est : mekni
la nationalite est : tun
le date naissance est : 11nov
le style de danse est :salsa
le niveau entrenement : moyen
Saisir danseur amateur
donner le cin
2
donner le nom
dia
donner le prenom
modou
donner le nationalite
sen
```

```
le prenom est : mekni
la nationalite est : tun
le date naissance est : 11nov
le style de danse est :salsa
le niveau entrenement : moyen
Saisir danseur amateur
donner le cin
2
donner le nom
dia
donner le prenom
modou
donner le nationalite
sen
donner date naissance
23juin
donner style danse
salsa
donner les annees experience
3
donner les palmares
champion
le cin est : 2
```

6.2 Templates pour la classe mére resultat

TemplateResultat.h

```
// Classe template de base
#include "Resultat.h"
#include<string>
#include <vector>
using namespace std;

template <typename T>
class ResultatTemplate  {
protected:
    int num_res;
    multimap<danseur, score> myMap;

public:
    ResultatTemplate(int = 2) : num_res(2) {}
    ResultatTemplate(const ResultatTemplate &) {}
    ResultatTemplate &operator=(const ResultatTemplate &) {
   return *this; }

    virtual void affichResulat() = 0;
    virtual void ajout_score(danseur d, score *s) = 0;
    virtual ~ResultatTemplate() {}
};

// Sp cialisation pour les r sultats partiels
template <typename T>
class ResultatPartielTemplate : public ResultatTemplate<T> {
private:
    string commentairePartiel;
    vector<score *> scorePartiel;

public:
    ResultatPartielTemplate(int num_res, string commentaire = "")
        : ResultatTemplate<T>(num_res), commentairePartiel(
   commentaire) {}
    ResultatPartielTemplate(const ResultatPartielTemplate &) {}
    string getCommentairePartiel() {return commentairePartiel;}
    ResultatPartielTemplate &operator=(const
   ResultatPartielTemplate &) { return *this; }
    void affichResulat() override {}
    void ajout_score(danseur d, score *s) override {}
    ~ResultatPartielTemplate() {}
};

// Sp cialisation pour les r sultats totaux
template <typename T>
class ResultatTotalTemplate : public ResultatTemplate<T> {
private:
    vector<score *> scoretotal;

public:
    ResultatTotalTemplate(int num_res) : ResultatTemplate<T>(
   num_res) {}
    ResultatTotalTemplate(const ResultatTotalTemplate &) {}
```

```
        ResultatTotalTemplate &operator=(const ResultatTotalTemplate
       &) { return *this; }
        void affichResulat() override {}
        void ajout_score(danseur d, score *s) override {}
        ~ResultatTotalTemplate() {}
        int getNum_res(){return this->num_res;}
    };
```

Test execution template Resultat

```
// Création d'un résultat partiel
ResultatPartielTemplate<Resultat> resultatPartiel(1, "Commentaire sur le resultat partiel");
// Ajout d'un score au résultat partiel
danseur d1;
score s1;
resultatPartiel.ajout_score(d1, &s1);
// Affichage du numéro de résultat et du commentaire du résultat partiel
cout << "Commentaire du resultat partiel : " << resultatPartiel.getCommentairePartiel() << endl;
// Création d'un résultat total
ResultatTotalTemplate<Resultat> resultatTotal(2);
// Ajout d'un score au résultat total
danseur d2 ;
score s2 ;
resultatTotal.ajout_score(d2, &s2);
// Affichage du numéro de résultat total
cout << "Numéro de resultat total : " << resultatTotal.getNum_res() << endl;
```

```
PS C:\Users\DELL\Desktop\ProjetC++2024\Application-de-gestion-des-tournoi-de-danse\output> &
-------TESTER CLASSE TEMPLATE---------
Commentaire du resultat partiel : Commentaire sur le resultat partiel
Num-®ro de resultat total : 2
```

7 Conclusion et perspectives

Dans cette conclusion, nous avons souligné le succès du projet de gestion des tournois de danse, mettant en lumière ses fonctionnalités essentielles. Malgré les défis rencontrés lors du développement, l'application répond aux besoins des organisateurs d'événements de danse.

Pour l'avenir, plusieurs perspectives d'amélioration ont été identifiées, notamment l'ajout de fonctionnalités avancées, l'amélioration des performances et l'extension de la portée de l'application pour couvrir différents types de tournois de danse.

En poursuivant le développement et l'amélioration de l'application, nous visons à continuer à améliorer la gestion des tournois de danse et à offrir des outils efficaces aux organisateurs et aux participants pour réussir dans ce domaine dynamique.