Lycée La Fayette
Champagne-sur-Seine • Fontaineroux

Rapport de Travaux Pratiques
BTS SN-IR

CNotes

Flavian Laxenaire
24 novembre 2021

Forme / 20 [coef. 1] →		Fond / 20 [coef. 2] →		Note / 20	
Qualité du rapport		Méthodologie			
Expression écrite		Respect du Cahier des Charges			
Pertinence de la rédaction		Qualité technique			
Respect des standards de codage		État d'avancement		Malus	

Sommaire

1. Modèle de base de la classe CNotes

1.1. Méthode de calcul de moyenne :

Cnotes.cpp

```
double CNotes::Moyenne() const
{
pElem p;
p = prem;
double moy;
while (p != NULL)
{
moy += p->note;
p = p->suiv;
}
return moy /= nbElem;
}
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ g++ -Wall -o main main.cpp cnotes.cpp
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ ./main
math     : 2 notes, moy. = 11
```

2. Extension des moyens de saisie de notes…

2.1. Créer une surcharge de l'opérateur += afin de pouvoir ajouter une note à la chaîne.

Cnotes.h

```
CNotes &operator+=(double v);
```

Cnotes.cpp

```
CNotes &CNotes::operator+=(double v)
{
    Ajoute(v);
    return *this;
}
```

2.2. Tester la classe avec le programme de test suivant :

main.cpp

```
int main(void)
{
    CNotes math;
    math.Ajoute(12);
    math.Ajoute(10);
    affiche("math    ", math);

    CNotes physique(7);
    physique += 13;
    physique += 10;
    affiche("physique", physique);
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ ./main
math     : 2 notes, moy. = 11
physique : 3 notes, moy. = 10
```

2.3. Surcharger l'opérateur []

Cnotes.h

```
CNotes &operator[](int indice);
CNotes &operator=(double v);
```

Cnotes.cpp

```
CNotes &CNotes::operator=(double v)
{
    {
        // cout << "note : " << v << endl;
        pElem p = prem;
        for (int i = 0; i < indice_current; i++)
        {
            p = p->suiv;
        }
        Ajoute(v);
        return *this;
    }
}
CNotes &CNotes::operator[](int indice)
{
    pElem p = prem;
    for (int i = 0; i < indice; i++)
    {
        if (p->suiv == NULL && i + 1 < indice)
            Ajoute(0);

        p = p->suiv;
    }
    indice_current = indice;
    return *this;
}
```

main.cpp

```
    CNotes anglais;
    anglais[0] = 11;
    anglais[1] = 15;
    anglais = anglais + 13;
    //cout << anglais.print();
    affiche("anglais ", anglais);

    CNotes francais(12);
    francais[2] = 14;
  //  cout << francais.print();
    affiche("francais", francais);
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ g++ -Wall -o main main.cpp cnotes.cpp
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ ./main
n°0 11 n°1 15 n°2 13
anglais  : 3 notes, moy. = 13
n°0 12 n°1 0 n°2 14
francais : 3 notes, moy. = 8.66667
```

2.4. fonction print note

cnotes.cpp

```
string CNotes::print()
{
    std::ostringstream oss;
    pElem p;
    p = prem;
    oss << std::setfill(' ');
    for (int i = 0; i < nbElem; i++)
    {
        if (p != NULL)
        {
            oss << "n°" << i << ' ' << p->note << ' ';
            p = p->suiv;
        }
    }
    oss << endl;
    return oss.str();
}
```

3. Manipulations de listes complètes...

3.1. Créer une surcharge de la méthode Ajoute()

Cnotes.h

```
void Ajoute(const CNotes &cn);
```

Cnotes.cpp

```
void CNotes::Ajoute(const CNotes &cn)
{
   pElem c = cn.prem;
   for (int i = 0; i < cn.nbElem; i++)
   {
       Ajoute(c->note);
       c = c->suiv;
   }
}
```

3.2. Implémenter un constructeur de copie

Cnotes.h

```
CNotes(const CNotes &cn)
    {
        Ajoute(cn);
    }
```

3.3. Ajouter une surcharge de l'opérateur +=

Cnotes.h

```
CNotes &operator+=(const CNotes &cn);
```

Cnotes.cpp

```
CNotes &CNotes::operator+=(const CNotes &cn)
{
    Ajoute(cn);
    return *this;
}
```

3.4. Valider les méthodes en ajoutant le code de test suivant :

Main.cpp

```
CNotes anglais;
    anglais[0] = 11;
    anglais[1] = 15;
    anglais = anglais + 13;
    cout << anglais.print();
    affiche("anglais ", anglais);

    CNotes francais(12);
    francais[2] = 14;
    cout << francais.print();
    affiche("francais", francais);

    CNotes langues;
    langues += francais;
    langues += anglais;
    cout << langues.print();
    affiche("langues ", langues);
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ g++ -Wall -o main main.cpp cnotes.cpp
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ ./main
n°0 11 n°1 15 n°2 13
anglais  : 3 notes, moy. = 13
n°0 12 n°1 0 n°2 14
francais : 3 notes, moy. = 8.66667
n°0 12 n°1 0 n°2 14 n°3 11 n°4 15 n°5 13
langues  : 6 notes, moy. = 10.8333
```

main.cpp

```
int main(void)
{
CNotes math;
math.Ajoute(12);
math.Ajoute(10);
affiche("math ", math);

CNotes physique(7);
physique += 13;
physique += 10;
affiche("physique", physique);

CNotes anglais;
anglais[0] = 11;
anglais[1] = 15;
```

```
anglais = anglais + 13;
cout << anglais.print();
affiche("anglais ", anglais);

CNotes francais(12);
francais[2] = 14;
cout << francais.print();
affiche("francais", francais);
CNotes langues;
langues += francais;
langues += anglais;
cout << langues.print();
affiche("langues ", langues);

CNotes bilan;
bilan += math + physique;
bilan += anglais;
bilan += francais;
affiche("bilan ", bilan);
// cout << bilan.print();
}
```

```
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ g++ -Wall -o main main.cpp cnotes.cpp
snir@snir-21-22:~/Documents/bts_2nd_annee/dev/cpp/4-CNotes$ ./main
math     : 2 notes, moy. = 11
physique : 3 notes, moy. = 10
n°0 11 n°1 15 n°2 13
anglais  : 3 notes, moy. = 13
n°0 12 n°1 0 n°2 14
francais : 3 notes, moy. = 8.66667
n°0 12 n°1 0 n°2 14 n°3 11 n°4 15 n°5 13
langues  : 6 notes, moy. = 10.8333
bilan    : 11 notes, moy. = 10.6364
```

4. Annexe

Cnotes.h

```
#ifndef CNOTES_H
#define CNOTES_H
#include <iostream>

using namespace std;

class CElem;
typedef CElem *pElem;
class CElem
{
public:
    double note;
    pElem suiv;
};

class CNotes
{
    pElem prem, dern;
    int nbElem;
    int indice_current;

public:
    CNotes(double n = -1)
        : prem(NULL), dern(NULL), nbElem(0)
    {
        if (n != -1)
            Ajoute(n);
    }
    CNotes(const CNotes &cn)
    {
        Ajoute(cn);
    }
    ~CNotes();
    CNotes &operator+=(double v);
    CNotes &operator[](int indice);
    CNotes &operator=(double v);
    CNotes &operator+(double v);
    CNotes &operator+=(const CNotes &cn);
    CNotes &operator+(const CNotes& cn);
    int nbNotes() const { return nbElem; }
    void Ajoute(double v);
    void Ajoute(const CNotes &cn);
    double Moyenne() const;
    string print();
};
#endif
```

Cnotes.cpp

```
#include "cnotes.h"
#include <string>
#include <sstream>
#include <iomanip>

CNotes::~CNotes()
{
    pElem p = prem;
    while (prem != NULL)
    {
        p = p->suiv;
        delete prem;
        prem = p;
```

```
    }
    prem = dern = NULL;
    nbElem = 0;
}

void CNotes::Ajoute(double v)
{
    pElem p = new CElem;
    p->note = v;
    p->suiv = NULL;
    if (prem == NULL)
        prem = dern = p;
    else
    {
        dern->suiv = p;
        dern = dern->suiv;
    }
    nbElem++;
}

void CNotes::Ajoute(const CNotes &cn)
{
    pElem c = cn.prem;
    for (int i = 0; i < cn.nbElem; i++)
    {
        Ajoute(c->note);
        c = c->suiv;
    }
}

double CNotes::Moyenne() const
{
    pElem p;
    p = prem;
    double moy;
    while (p != NULL)
    {
        moy += p->note;
        p = p->suiv;
    }
    return moy /= nbElem;
}
string CNotes::print()
{
    std::ostringstream oss;
    pElem p;
    p = prem;
    oss << std::setfill(' ');
    for (int i = 0; i < nbElem; i++)
    {
        if (p != NULL)
        {
            oss << "n°" << i << ' ' << p->note << ' ';
            p = p->suiv;
        }
    }
    oss << endl;
    return oss.str();
}
CNotes &CNotes::operator+(double v)
{
    Ajoute(v);
    return *this;
}

CNotes &CNotes::operator+(const CNotes& cn)
{
    Ajoute(cn);
```

```
    return *this;
}

CNotes &CNotes::operator=(double v)
{
    {
        // cout << "note : " << v << endl;
        pElem p = prem;
        for (int i = 0; i < indice_current; i++)
        {
            p = p->suiv;
        }
        Ajoute(v);
        return *this;
    }
}
CNotes &CNotes::operator[](int indice)
{
    pElem p = prem;
    for (int i = 0; i < indice; i++)
    {
        if (p->suiv == NULL && i + 1 < indice)
            Ajoute(0);

        p = p->suiv;
    }
    indice_current = indice;
    return *this;
}
CNotes &CNotes::operator+=(double v)
{
    Ajoute(v);
    return *this;
}
CNotes &CNotes::operator+=(const CNotes &cn)
{
    Ajoute(cn);
    return *this;
}
```

main.cpp

```
#include "cnotes.h"

void affiche(const string &m, const CNotes &c)
{
    cout << m << " : " << c.nbNotes() << " note";
    if (c.nbNotes() > 1)
        cout << 's';
    cout << ", moy. = " << c.Moyenne() << endl;
}

int main(void)
{
    CNotes math;
    math.Ajoute(12);
    math.Ajoute(10);
    affiche("math    ", math);

    CNotes physique(7);
    physique += 13;
    physique += 10;
    affiche("physique", physique);

    CNotes anglais;
    anglais[0] = 11;
```

```
    anglais[1] = 15;
    anglais = anglais + 13;
    cout << anglais.print();
    affiche("anglais ", anglais);

    CNotes francais(12);
    francais[2] = 14;
    cout << francais.print();
    affiche("francais", francais);

    // CNotes test2;
    // CNotes test(math);
    //test += math;
    // cout << test.print();
    // affiche("test", test);

    CNotes langues;
    langues += francais;
    langues += anglais;
    cout << langues.print();
    affiche("langues ", langues);

    CNotes bilan;
    bilan += math + physique;
    bilan += anglais;
    bilan += francais;
    affiche("bilan   ", bilan);
   // cout << bilan.print();
}
```