Atelier 4

Exercice1:

```cpp
#include <iostream>
using namespace std;

class Complexe {
private:
    double re;
    double im;
public:
    Complexe(double r = 0, double i = 0) : re(r), im(i) {}
    Complexe operator+(const Complexe& other) {
        return Complexe(re + other.re, im + other.im);
    }
    Complexe operator-(const Complexe& other) {
        return Complexe(re - other.re, im - other.im);
    }
    Complexe operator*(const Complexe& other) {
        return Complexe(re * other.re - im * other.im,re * other.im + im * other.re);
    }
    Complexe operator/(const Complexe& other) {
        double denom = other.re * other.re + other.im * other.im;
        if (denom == 0) {
            throw "Division par zéro.";
        }
        return Complexe((re * other.re + im * other.im) / denom,(im * other.re - re * other.im) / denom);
    }
    bool operator==(const Complexe& other) {
        return (re == other.re && im == other.im);
    }
    void afficher() const {
        cout << re << " + " << im << "i" <<endl;
    }
};
```

```cpp
void menu() {
    cout << "Choisissez une opération:" <<endl;
    cout << "1. Addition" <<endl;
    cout << "2. Soustraction" <<endl;
    cout << "3. Multiplication" <<endl;
    cout << "4. Division" <<endl;
    cout << "5. Égalité" <<endl;
    cout << "6. Quitter" <<endl;
}
int main() {
    double re1, im1, re2, im2;
    cout << "Entrez la partie réelle du premier nombre complexe: ";
    cin >> re1;
    cout << "Entrez la partie imaginaire du premier nombre complexe: ";
    cin >> im1;
    Complexe z1(re1, im1);
    cout << "Entrez la partie réelle du deuxième nombre complexe: ";
    cin >> re2;
    cout << "Entrez la partie imaginaire du deuxième nombre complexe: ";
    cin >> im2;
    Complexe z2(re2, im2);
    int choix;
    while (true) {
        menu();
        cout << "Votre choix: ";
        cin >> choix;
        switch (choix) {
            case 1: {
                Complexe resultat = z1 + z2;
                cout << "Résultat de l'addition: ";
                resultat.afficher();
                break;
            }
```

```
            case 2: {
                Complexe resultat = z1 - z2;
                cout << "Résultat de la soustraction: ";
                resultat.afficher();
                break;
            }
            case 3: {
                Complexe resultat = z1 * z2;
                cout << "Résultat de la multiplication: ";
                resultat.afficher();
                break;
            }
            case 4: {
                try {
                    Complexe resultat = z1 / z2;
                    cout << "Résultat de la division: ";
                    resultat.afficher();
                } catch (const runtime_error& e) {
                    cerr << e.what() <<endl;
                }
                break;
            }
            case 5: {
                if (z1 == z2) {
                    cout << "Les nombres complexes sont égaux." <<endl;
                } else {
                    cout << "Les nombres complexes ne sont pas égaux." <<endl;
                }
                break;
            }
```

```
            case 6:
                cout << "Au revoir!" <<endl;
                return 0;
            default:
                cout << "Choix invalide, veuillez réessayer." <<endl;
        }
    }
    return 0;
}
```

```
Entrez la partie réelle du premier nombre complexe: 6
Entrez la partie imaginaire du premier nombre complexe: 2
Entrez la partie réelle du deuxième nombre complexe: 4
Entrez la partie imaginaire du deuxième nombre complexe: 2
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. Égalité
6. Quitter
Votre choix: 1
Résultat de l'addition: 10 + 4i
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. galité
6. Quitter
Votre choix: 2
Résultat de la soustraction: 2 + 0i
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. Égalité
6. Quitter
Votre choix: 3
```

```
Votre choix: 3
Résultat de la multiplication: 20 + 20i
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. Égalité
6. Quitter
Votre choix: 4
Résultat de la division: 1.4 + -0.2i
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. Égalité
6. Quitter
Votre choix: 5
Les nombres complexes ne sont pas égaux.
Choisissez une opération:
1. Addition
2. Soustraction
3. Multiplication
4. Division
5. Égalité
6. Quitter
Votre choix: 6
Au revoir!
```

Exercice 2 :

```cpp
#include <iostream>
using namespace std;

class Animal {
protected:
    string nom;
    int age;
public:
    void set_value(const string& nomAnimal, int ageAnimal) {
        nom = nomAnimal;
        age = ageAnimal;
    }
};
class Zebra : public Animal {
public:
    void afficher_info() {
        cout << "Zèbre - Nom: " << nom << ", Âge: " << age << " ans, Origine: Savane." <<endl;
    }
};
class Dolphin : public Animal {
public:
    void afficher_info() {
        cout << "Dauphin - Nom: " << nom << ", Âge: " << age << " ans, Origine: Océan." <<endl;
    }
};
int main() {
    Zebra z1, z2;
    Dolphin d1;
    z1.set_value("Zébra 1", 2);
    z2.set_value("Zébra 2", 3);
    d1.set_value("Dauphin 1", 7);
    z1.afficher_info();
    z2.afficher_info();
    d1.afficher_info();
    return 0;
}
```

```
Zèbre - Nom: Zébra 1, Âge: 2 ans, Origine: Savane.
Zèbre - Nom: Zébra 2, Âge: 3 ans, Origine: Savane.
Dauphin - Nom: Dauphin 1, Âge: 7 ans, Origine: Océan.


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 3 :

```cpp
#include <iostream>
using namespace std;

class Personne {
private:
    string nom;
    string prenom;
    string date;
public:
    Personne(const string& n, const string& p, const string& d): nom(n), prenom(p), date(d) {}
    virtual void Afficher() const {
        cout << "Nom: " << nom << ", Prénom: " << prenom << ", Date de Naissance: " << date <<endl;
    }
    virtual ~Personne() {}
};
class Employe : public Personne {
private:
    double salaire;
public:
    Employe(const string& n, const string& p, const string& d, double s) : Personne(n, p, d), salaire(s) {}
    void Afficher() const {
        Personne::Afficher();
        cout << "Salaire: " << salaire <<endl;
    }
};
class Chef : public Employe {
private:
    string service;
public:
    Chef(const string& n, const string& p, const string& d, double s, const string& serv) : Employe(n, p, d, s), service(serv) {}
    void Afficher() const {
        Employe::Afficher();
        cout << "Service: " << service <<endl;
    }
};
```

```cpp
class Directeur : public Chef {
private:
   string societe;
public:
    Directeur(const string& n, const string& p, const string& d, double s, const string& serv, const string& soc) : Chef(n, p, d, s, serv), societe(soc) {}
    void Afficher() const {
       Chef::Afficher();
       cout << "Société: " << societe <<endl;
    }
};
int main() {
    Personne p("Borki", "Mayssae","04/07/2001");
    p.Afficher();
    Employe e("Jamson","Kay","28/03/1994",3000);
    e.Afficher();
    Chef c("Idrissi","Saad","20/12/1990",4000,"Informatique");
    c.Afficher();
    Directeur d("Simon", "Alex","12/09/1985",6000,"Resources humaines","Techno");
    d.Afficher();
    return 0;
}
```

```
Nom: Borki, Prénom: Mayssae, Date de Naissance: 04/07/2001
Nom: Jamson, Prénom: Kay, Date de Naissance: 28/03/1994
Salaire: 3000
Nom: Idrissi, Prénom: Saad, Date de Naissance: 20/12/1990
Salaire: 4000
Service: Informatique
Nom: Simon, Prénom: Alex, Date de Naissance: 12/09/1985
Salaire: 6000
Service: Resources humaines
Société: Techno


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 4 :

```cpp
#include <iostream>
#include <cmath>
using namespace std;

class Vecteur3D {
private:
    float x, y, z;
public:
    Vecteur3D(float x = 0.0, float y = 0.0, float z = 0.0) : x(x), y(y), z(z) {}
    void afficher() const {
        cout << "(" << x << ", " << y << ", " << z << ")" <<endl;
    }
    Vecteur3D somme(const Vecteur3D& autre) const {
        return Vecteur3D(x + autre.x, y + autre.y, z + autre.z);
    }
    float produitScalaire(const Vecteur3D& autre) const {
        return (x * autre.x) + (y * autre.y) + (z * autre.z);
    }
    bool coincide(const Vecteur3D& autre) const {
        return (x == autre.x && y == autre.y && z == autre.z);
    }
    float norme() const {
        return sqrt(x * x + y * y + z * z);
    }
    Vecteur3D normax(const Vecteur3D& autre) const {
        return (norme() > autre.norme()) ? *this : autre;
    }
    Vecteur3D* normaxAdresse(Vecteur3D* autre) {
        return (norme() > autre->norme()) ? this : autre;
    }
    const Vecteur3D& normaxReference(const Vecteur3D& autre) const {
        return (norme() > autre.norme()) ? *this : autre;
    }
};
```

```cpp
int main() {
    Vecteur3D v1(1, 2, 3);
    Vecteur3D v2(4, 5, 6);
    cout << "Vecteur 1: ";
    v1.afficher();
    cout << "Vecteur 2: ";
    v2.afficher();
    Vecteur3D somme = v1.somme(v2);
    cout << "Somme: ";
    somme.afficher();
    float produit = v1.produitScalaire(v2);
    cout << "Produit scalaire: " << produit <<endl;
    if (v1.coincide(v2)) {
        cout << "Les vecteurs coïncident." <<endl;
    }
    else {
        cout << "Les vecteurs ne coïncident pas." <<endl;
    }
    cout << "Norme de v1: " << v1.norme() <<endl;
    cout << "Norme de v2: " << v2.norme() <<endl;
    Vecteur3D plusGrand = v1.normax(v2);
    std::cout << "Vecteur avec la plus grande norme (valeur): ";
    plusGrand.afficher();
    Vecteur3D* plusGrandAdresse = v1.normaxAdresse(&v2);
    std::cout << "Vecteur avec la plus grande norme (adresse): ";
    plusGrandAdresse->afficher();
    const Vecteur3D& plusGrandReference = v1.normaxReference(v2);
    std::cout << "Vecteur avec la plus grande norme (référence): ";
    plusGrandReference.afficher();
    return 0;
}
```

```
Vecteur 1: (1, 2, 3)
Vecteur 2: (4, 5, 6)
Somme: (5, 7, 9)
Produit scalaire: 32
Les vecteurs ne coïncident pas.
Norme de v1: 3.74166
Norme de v2: 8.77496
Vecteur avec la plus grande norme (valeur): (4, 5, 6)
Vecteur avec la plus grande norme (adresse): (4, 5, 6)
Vecteur avec la plus grande norme (référence): (4, 5, 6)


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 5 :

```cpp
#include <iostream>
using namespace std;

class Test {
public:
    void call() {
        static int compteur = 0;
        compteur++;
        cout << "call a été appelé " << compteur << " fois." <<endl;
    }
};

int main() {
    Test t1;
    t1.call();
    t1.call();
    t1.call();
    return 0;
}
```

```
call a été appelé 1 fois.
call a été appelé 2 fois.
call a été appelé 3 fois.


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 6 :

main.cpp | Point.h | Point.cpp

```cpp
#ifndef POINT_H
#define POINT_H

class Point {
private:
    float x;
    float y;
public:
    Point(float xCoord, float yCoord);
    void deplace(float dx, float dy);
    void affiche() const;
};
#endif
```

main.cpp | Point.h | Point.cpp

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

Point::Point(float xCoord, float yCoord) : x(xCoord), y(yCoord) {}
void Point::deplace(float dx, float dy) {
    x += dx;
    y += dy;
}
void Point::affiche() const {
    cout << "Point(" << x << ", " << y << ")" <<endl;
}

```

main.cpp | Point.h | Point.cpp

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

int main() {
    Point p(2.0f, 3.0f);
    cout << "Point initial : ";
    p.affiche();
    p.deplace(1.0f, 1.0f);
    cout << "Point après déplacement : ";
    p.affiche();
    return 0;
}

```

```
Point initial : Point(2, 3)
Point après déplacement : Point(3, 4)


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 7 :

```cpp
#include <iostream>
#include <vector>
#include <exception>
using namespace std;

class Pile {
private:
    vector<int> elements;
public:
    Pile() = default;
    void push(int element) {
        elements.push_back(element);
    }
    int pop() {
        if (elements.empty()) {
            throw out_of_range("La pile est vide, impossible de dépiler.");
        }
        int element = elements.back();
        elements.pop_back();
        return element;
    }
    bool vide() const {
        return elements.empty();
    }
    void afficher() const {
        if (elements.empty()) {
            cout << "La pile est vide." <<endl;
            return;
        }
        cout << "Éléments de la pile : ";
        for (int i = elements.size() - 1; i >= 0; --i) {
            cout << elements[i] << " ";
        }
        cout <<endl;
    }
};
```

```
37 int main() {
38     Pile p1;
39     Pile p2;
40     p1.push(30);
41     p1.push(20);
42     p1.push(10);
43     cout << "Pile p1 : ";
44     p1.afficher();
45     cout << "Dépile de p1 : " << p1.pop() <<endl;
46     cout << "Après dépilement, p1 : ";
47     p1.afficher();
48     p2.push(50);
49     p2.push(40);
50     cout << "Pile p2 : ";
51     p2.afficher();
52     cout << "Dépile de p2 : " << p2.pop() <<endl;
53     cout << "Après dépilement, p2 : ";
54     p2.afficher();
55     p2.pop();
56     cout << "Après dépilement, p2 : ";
57     p2.afficher();
58     try {
59         p2.pop();
60     } catch (const out_of_range& e) {
61         cout << "Erreur : " << e.what() <<endl;
62     }
63     return 0;
64 }
65
```

```
Pile p1 : Éléments de la pile : 10 20 30
Dépile de p1 : 10
Après dépilement, p1 : Éléments de la pile : 20 30
Pile p2 : Éléments de la pile : 40 50
Dépile de p2 : 40
Après dépilement, p2 : Éléments de la pile : 50
Après dépilement, p2 : La pile est vide.
Erreur : La pile est vide, impossible de dépiler.


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 8 :

```cpp
#include <iostream>
#include <exception>
using namespace std;

class Fichier {
private:
    char* P;
    size_t longueur;
public:
    Fichier() : P(nullptr), longueur(0) {}
    void Creation(size_t taille) {
        longueur = taille;
        P = new char[longueur];
        if (!P) {
            throw "Échec de l'allocation mémoire.";
        }
    }
    void Remplit() {
        if (P) {
            for (size_t i = 0; i < longueur; ++i) {
                P[i] = 'A' + (i % 26);
            }
        }
        else {
            throw "Espace mémoire non alloué.";
        }
    }
    void Affiche() const {
        if (P) {
            cout << "Contenu du fichier : ";
            for (size_t i = 0; i < longueur; ++i) {
                cout << P[i];
            }
            cout <<endl;
        }
        else {
            cout << "Aucun contenu à afficher." <<endl;
        }
    }
    ~Fichier() {
        delete[] P;
    }
};
int main() {
    try {
        Fichier* fichier = new Fichier();
        fichier->Creation(8);
        fichier->Remplit();
        fichier->Affiche();
        delete fichier;

    }
    catch (const exception& e) {
        cerr << "Erreur : " << e.what() <<endl;
    }
    return 0;
}

```

```
Contenu du fichier : ABCDEFGH


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 9 :

```cpp
#include <iostream>
using namespace std;

struct Element {
    int valeur;
    Element* suivant;
    Element(int val) : valeur(val), suivant(nullptr) {}
};
class Liste {
private:
    Element* premier;
public:
    Liste() : premier(nullptr) {}
    void ajouter(int val) {
        Element* nouvelElement = new Element(val);
        nouvelElement->suivant = premier;
        premier = nouvelElement;
    }
    void supprimer() {
        if (premier) {
            Element* temp = premier;
            premier = premier->suivant;
            delete temp;
        }
        else {
            cout << "La liste est vide, impossible de supprimer." <<endl;
        }
    }
```

```
    void afficher() const {
        Element* courant = premier;
        if (!courant) {
            cout << "La liste est vide." <<endl;
            return;
        }
        cout << "Liste : ";
        while (courant) {
            cout << courant->valeur << " ";
            courant = courant->suivant;
        }
        cout <<endl;
    }
    ~Liste() {
        while (premier) {
            supprimer();
        }
    }
};
int main() {
    Liste liste;
    liste.ajouter(30);
    liste.ajouter(20);
    liste.ajouter(10);
    liste.afficher();
    liste.supprimer();
    liste.afficher();
    return 0;
}
```

```
Liste : 10 20 30
Liste : 20 30


...Program finished with exit code 0
Press ENTER to exit console.
```

Exercice 10 :

```cpp
#include <iostream>
using namespace std;

int main() {
    string dateHeure;
    cout << "Entrez la date et l'heure sous la forme JJMMAAAAHHMM : ";
    cin >> dateHeure;
    if (dateHeure.length() != 12) {
        cerr << "Erreur : La chaîne doit avoir exactement 12 caractères." <<endl;
        return 1;
    }
    string jour = dateHeure.substr(0,4);
    string mois = dateHeure.substr(0,7);
    string annee = dateHeure.substr(20,01);
    string heure = dateHeure.substr(1,2);
    string minutes = dateHeure.substr(1, 2);
    cout << "Date : " << jour << "/" << mois << "/" << annee <<endl;
    cout << "Heure : " << heure << ":" << minutes <<endl;
    return 0;
}
```

```
Entrez la date et l'heure sous la forme JJMMAAAAHHMM : 1234567890
Erreur : La chaîne doit avoir exactement 12 caractères.


...Program finished with exit code 1
Press ENTER to exit console.
```