Exercice1 : Déclaration d'une classe et définition de ses méthodes

Examinez ce code attentivement. Chaque étape du programme doit être détaillée dans un commentaire. Expliquez le résultat de l'exécution:

```
#ifndef DATE_H
#define DATE_H
#include <iostream>
#include <string>

using namespace std;


class Date
{
    int jour,annee;
    string mois;


    public:
        Date();
        void DefinirDate(int, string, int);
        void lireDate();
        virtual ~Date();

};

#endif // DATE_H
```

- Définition des méthodes d'accès

```
#include "Date.h"
#include <string>

using namespace std;

void Date::DefinirDate(int j, string m, int a)
{
    jour=j;
    mois=m;
    annee=a;
}

void Date::lireDate()
{
    cout<<"Date: "<<jour<<" "<<mois<<" "<<annee;

}
```

- Fonction principale

```
#include <iostream>
#include <string>
#include "Date.h"

using namespace std;

int main()
{
    Date date_du_jour;
    date_du_jour.DefinirDate(3,"Novembre",2017);
    date_du_jour.lireDate();
    return 0;
```

Exercice2 : Constructeurs et destructeurs

Complétez l'exemple de l'exercice 1. On testera de façon **pertinente** le bon fonctionnement de cette classe. Chaque étape du programme doit être détaillée dans un commentaire. Expliquez le résultat de l'exécution

```
class Date
{
    int jour,annee;
    string mois;


    public:
        Date(int, string, int);
        virtual ~Date();
        void DefinirDate(int, string, int);
        void lireDate();


};
```

- Définition du constructeur et destructeur

```
Date::Date(int jourInitial, string moisInitial, int anneeInitiale)
{
    jour=jourInitial;
    mois=moisInitial;
    annee=anneeInitiale;

}

Date::~Date()
{

}
void Date::DefinirDate(int j, string m, int a)
{
    jour=j;
    mois=m;
    annee=a;
}

void Date::lireDate()
{
    cout<<"Date: "<<jour<<" "<<mois<<" "<<annee;

}
```

- Fonction principale

```
int main()
{
  int j, a;
  string m;
  cout<<"saisissez la date (ex 5 Novembre 2017) ";
  cin>>j;
  cin>>m;
  cin>>a;
  Date date_du_jour(j,m,a);
  date_du_jour.lireDate();
    return 0;
}
```

Exercice3 : Surcharge des constructeurs

Complétez l'exemple de l'exercice 2. Chaque étape du programme doit être détaillée dans un commentaire. Expliquez le principe de la surcharge des constructeurs

```
class Date
{
    int jour,annee;
    string mois;


    public:
        Date(int, string, int);
        Date(int, string);
        Date(int);
        Date();
        virtual ~Date();
        void DefinirDate(int, string, int);
        void lireDate();


};
```

- Définition des constructeurs A complétez

- Fonction principale

```
Date Aujourdhui();  // date courante
Date Cette_annee(5, "Avril"); // 5 Avril de l'année courante
Date Date_dans_Mois(10);  //le 10 mois et année courante
```

Exercice4 : Constructeur de copie

Le compilateur ne génère pas seulement un constructeur et un destructeur par défaut si **l'**utilisateur ne les a pas définis, il génère aussi un constructeur de copie qui est appelé chaque fois qu'une copie d'objet est réalisée en mémoire.

Exemple d'utilisation d'un constructeur de copie à travers la classe Date

```
using namespace std;
class Date
{
    private:
        int *jour, * mois, *annee;
    public:
        Date(int,int,int);// constructeur par défaut
        Date(const Date &);// Constructeur de copie
        virtual ~Date();  // Destructeur
    /*méthodes d'accès*/
    void DefinirDate(int j, int m, int a) {*jour=j; *mois=m; *annee=a;}
    int LireDate()const
    {cout<<*jour<<" "<<*mois<<" "<<*annee<<endl;}
    int LireMois()const {return *mois;}
    int LireJour()const {return *jour;}
    int LireAnnee()const {return *annee;}

};

Date::Date(int j,int m, int a)
{
    //reservation de l'espace mémoire dans le tas, les valeurs des membres
    //sont ensuite stockées aux adresses correspondantes
    jour= new int;*jour= j;
    mois= new int;*mois=m;
    annee=new int; *annee=a;
}
Date::Date(const Date &source)
{
    jour=new int; *jour=source.LireJour();
    mois=new int; *mois=source.LireMois();
    annee= new int; *annee=source.LireAnnee();
}

//définition des destructeur
Date::~Date()
{
    delete jour; jour=0;// libération de la mémoire
    delete mois; mois=0;
    delete annee; annee=0;
}
```

Commentez les résultats d'exécution du programme principal :

Expliquer le principe du constructeur de copie

```
using namespace std;

int main()
{
    Date Date_initiale(1,1,2017);
    Date Copie_date=Date_initiale;
    Date_initiale.DefinirDate(31,12,2017);
    cout<<" Date_initiale= ";
    Date_initiale.LireDate();
    Copie_date.LireDate();
    cout<<endl;

    return 0;
}
```

Exercice5 : Surcharge des opérateurs

Tapez, commentez et expliquez le code suivant :

```
class Date
{
    private:
        int jour, mois, annee;
    public:
        Date(int,int,int);// constructeur par défaut
        virtual ~Date();  // Destructeur

    void DefinirDate(int j, int m, int a) {jour=j; mois=m; annee=a;}
    int LireDate()const
    {cout<<jour<<" "<<mois<<" "<<annee<<endl;}
    int LireMois()const {return mois;}
    int LireJour()const {return jour;}
    int LireAnnee()const {return annee;}
    Date operator+ (const Date &);

};
```

```
Date::Date(int j,int m, int a)
{jour= j; mois=m; annee=a;}

Date::~Date(){}
```

```
Date Date::operator+(const Date &source)
{
    int j,m,a;
    j=jour+source.LireJour();
    m=mois+source.LireMois();
    a=annee+source.LireAnnee();
    if(j>30)
    {
        j-=30;
        m+=1;
    }
    if(m>12)
    {
        m-=12;
        a+=1;
    }
    return Date(j,m,a);
}
```