# LAB 4. NẠP CHỒNG TOÁN TỬ

THỜI LƯỢNG : 4 TIẾT

## A. Mục tiêu

- Giúp sinh viên biết cách định nghĩa các lớp và các toán tử áp dụng cho các thể hiện của lớp.
- Biết cách định nghĩa phép ép kiểu thông qua việc nạp chồng toán tử
- Biết và hiểu rõ hơn về từ khóa static
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Nắm vững các toán tử được phép nạp chồng
  - Hiểu rõ cú pháp cũng như cách định nghĩa các phương thức nạp chồng toán tử
  - Cài đặt (định nghĩa) được các phương thức để nạp chồng toán tử một ngôi, hai ngôi
  - Hiểu rõ phép ép kiểu và cách định nghĩa phép ép kiểu
  - Biết cách sử dụng các toán tử đã định nghĩa

## B. Yêu cầu

- Sinh viên phải trả lời đầy đủ các câu hỏi (nếu có) hoặc chụp màn hình kết quả, ghi lại vào tập tin Word theo yêu cầu ghi trong phần hướng dẫn thực hành.
- Đặt tên tập tin Word theo dạng: Lab04_MSSV_HoVaTen.rar.
- Tạo thư mục, đặt tên là MSSV_Lab04 để lưu tất cả bài làm trong bài thực hành này.
- Sinh viên phải hoàn thành tối thiểu 2 bài tập bắt buộc.
- Phải tạo một dự án riêng cho mỗi phần hoặc mỗi bài tập.
- Cuối buổi thực hành, nén thư mục trên & nộp bài qua email: phucnguyen5555@gmail.com

## C. Hướng dẫn thực hành

### 1. Tạo dự án

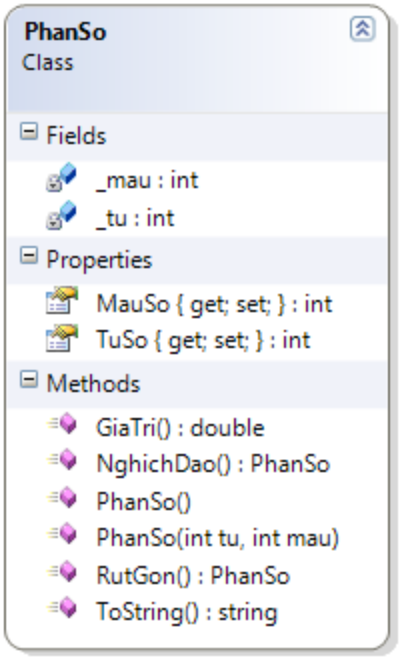
Trong bài thực hành này, ta sẽ định nghĩa các phép toán cơ bản như cộng, trừ, nhân, chia và các phép toán so sánh, phép toán dịch bit, phép ép kiểu, … trên phân số.

Bước 1. Tạo một thư mục, đặt tên theo dạng MSSV-Lab04. Ví dụ: 1210234-Lab04.

Bước 2. Tạo dự án mới, dạng Console Application, đặt tên là Lab04_Bai01_ToanTuHaiNgoi. Sau đó, lưu vào thư mục đã tạo ở bước 1.

Bước 3. Tham khảo lớp PhanSo trong Lab03_Bai01 để tạo lớp phân số như hình dưới đây:

Bước 4. Viết mã lệnh trong hàm Main để tạo đối tượng và kiểm tra các hàm đã định nghĩa như sau




```
// Tạo các thể hiện của lớp phân số
PhanSo haiBon = new PhanSo(2, 4);
PhanSo motBa = new PhanSo(1, 3);

// Rút gọn phân số haiBon
PhanSo toiGian = haiBon.RutGon();
// Xuất kết quả
Console.WriteLine("{0} = {1}", haiBon, toiGian);

// Nghịch đảo phân số motBa
PhanSo dao = motBa.NghichDao();
// Xuất kết quả
Console.WriteLine("{0} ==nghich dao==> {1}", motBa, dao);
```

Bước 5. Chạy chương trình và ghi nhận lại kết quả.

## 2. Nạp chồng toán tử hai ngôi

Phần nãy hướng dẫn cách định nghĩa các phương thức nạp chồng các toán tử hai ngôi. Phép toán hai ngôi bao gồm: +, -, *, /, %, ==, !=, <, >, <=, >=, <<, >>, &, |. Đầu tiên là các phép toán số học.

Bước 6. Trong lớp phân số, cài đặt các phương thức sau để nạp chồng toán tử +, -.

```
// Phương thức nạp chồng toán tử cộng hai phân số
public static PhanSo operator + (PhanSo x, PhanSo y)
{
    int tuMoi = x.TuSo * y.MauSo + x.MauSo * y.TuSo;
    int mauMoi = x.MauSo * y.MauSo;
    PhanSo tong = new PhanSo(tuMoi, mauMoi);
    return tong.RutGon();
}

// Phương thức cộng phân số với một số
public static PhanSo operator +(PhanSo x, int k)
{
    PhanSo y = new PhanSo(k, 1);
    return x + y;
}

// Phương thức nạp chồng toán tử trừ hai phân số
public static PhanSo operator -(PhanSo x, PhanSo y)
{
    int tuMoi = x.TuSo * y.MauSo - x.MauSo * y.TuSo;
    int mauMoi = x.MauSo * y.MauSo;
    return new PhanSo(tuMoi, mauMoi).RutGon();
}
```

```
// Phương thức trừ phân số cho một số
public static PhanSo operator -(PhanSo x, int k)
{
    return x + (-k);
}
```

Bước 7. Trong hàm Main, bổ sung đoạn mã sau để kiểm tra các phép toán đã định nghĩa

```
// Tính tổng hai phân số
PhanSo tong = haiBon + motBa;

// Xuất kết quả
Console.WriteLine("{0} + {1} = {2}", haiBon, motBa, tong);

// Tính tổng phân số với một số
tong = motBa + 3;

// Xuất kết quả
Console.WriteLine("{0} + 3 = {1}", motBa, tong);
```

Bước 8. Chạy chương trình và ghi lại kết quả.

Bước 9. Viết tiếp mã lệnh để kiểm tra phép toán trừ (-) và ghi lại kết quả.

Bước 10. Quay trở lại lớp PhanSo, định nghĩa tiếp các phép toán *, / theo mẫu sau:

```
// Phương thức nạp chồng toán tử nhân hai phân số
public static PhanSo operator *(PhanSo x, PhanSo y)...

// Phương thức nhân phân số với một số
public static PhanSo operator *(PhanSo x, int k)...

// Phương thức nạp chồng toán tử chia hai phân số
public static PhanSo operator /(PhanSo x, PhanSo y)...

// Phương thức chia phân số cho một số
public static PhanSo operator /(PhanSo x, int k)...
```

Bước 11. Trong hàm Main, viết mã lệnh để kiểm tra hoạt động của phép * và /. Ghi lại kết quả.

Tiếp theo, ta sẽ định nghĩa các phép toán so sánh. Lưu ý rằng, các toán tử so sánh phải được định nghĩa theo cặp. Ví dụ: == và !=, < và >, <= và >=.

Bước 12. Trong lớp PhanSo, định nghĩa tiếp các phương thức sau:

```
// Phương thức nạp chồng toán tử so sánh bằng
public static bool operator ==(PhanSo x, PhanSo y)
{
    return x.GiaTri() == y.GiaTri();
}
```

```
// Phương thức nạp chồng toán tử so sánh khác
public static bool operator !=(PhanSo x, PhanSo y)
{
    return !(x == y);
    // hoặc: return x.GiaTri() != y.GiaTri();
}
```

Bước 13. Trong hàm Main, bổ sung đoạn mã sau để kiểm tra hai phép toán vừa định nghĩa

```
if (toiGian == motBa)
    Console.WriteLine("{0} == {1}", toiGian, motBa);
else
    Console.WriteLine("{0} != {1}", toiGian, motBa);
```

Bước 14. Chạy chương trình và ghi nhận lại kết quả

Bước 15. Tạo hai phân số khác với giá trị tùy ý và kiểm tra hoạt động của phép toán !=

Bước 16. Trong lớp PhanSo, tiếp tục định nghĩa các phép toán <, >, <=, >= theo mẫu sau

```
// Phương thức nạp chồng toán tử so sánh lớn hơn
public static bool operator >(PhanSo x, PhanSo y)...

// Phương thức nạp chồng toán tử so sánh nhỏ hơn
public static bool operator <(PhanSo x, PhanSo y)...

// Phương thức nạp chồng toán tử so sánh >=
public static bool operator >=(PhanSo x, PhanSo y)...

// Phương thức nạp chồng toán tử so sánh <=
public static bool operator <=(PhanSo x, PhanSo y)...
```

Bước 17. Trở lại hàm Main, viết mã lệnh để kiểm tra các toán tử vừa định nghĩa. Chạy chương trình và ghi nhận kết quả.

Các toán tử hai ngôi thao tác bit như << và >> cũng có thể được nạp chồng như sau:

Bước 18. Trong lớp PhanSo, định nghĩa các phương thức sau:

```
// Định nghĩa phương thức nạp chồng toán tử dịch trái
public static PhanSo operator <<(PhanSo x, int n)
{
    int tuMoi = x.TuSo << n;
    return new PhanSo(tuMoi, x.MauSo);
}
// Định nghĩa phương thức nạp chồng toán tử dịch phải
public static PhanSo operator >>(PhanSo x, int n)
{
    int tuMoi = x.TuSo >> n;
    return new PhanSo(tuMoi, x.MauSo);
}
```

Bước 19. Ở hàm Main, bổ sung đoạn mã sau để sử dụng các phép toán trên

```
// Dịch trái tử số 4 bit => nhân 2 liên tiếp 4 lần
PhanSo p = motBa << 4;

// Xuất kết quả
Console.WriteLine("{0} << 4 = {1}", motBa, p);

// Dịch phải tử số 3 bit => chia 2 liên tiếp 3 lần
PhanSo q = motBa >> 3;

// Xuất kết quả
Console.WriteLine("{0} >> 3 = {1}", motBa, q);
```

Bước 20. Chạy chương trình và ghi lại kết quả. Mô tả vắn tắt hoạt động của chương trình trên.
Bước 21. Nhấp phải chuột vào tên dự án, chọn View Class diagram để xem sơ đồ lớp.
Bước 22. Bấm chuột vào các dấu mũi tên ở góc trên bên phải mỗi lớp trong sơ đồ
Bước 23. Nhấp phải chuột vào vùng trống, chọn Change Members Format > Display Full Sig…
Bước 24. Nhấp phải chuột vào vùng trống, chọn Adjust Shapes Width
Bước 25. Chụp hình và lưu lại sơ đồ lớp vừa tạo.

### 3. Nạp chồng toán tử một ngôi

Phần nãy hướng dẫn cách định nghĩa các phương thức nạp chồng các toán tử một ngôi. Phép toán một ngôi có thể nạp chồng bao gồm: -, !, ~, ++, --, true, false. Ở đây, ta dùng toán tử ! làm phép rút gọn phân số, toán tử ~ dùng để lấy phân số nghịch đảo.

Bước 1. Tạo dự án mới, dạng Console Application, đặt tên là Lab04_Bai02_ToanTuMotNgoi.
Bước 2. Tạo lớp PhanSo như mục 1 rồi thêm đoạn mã sau để nạp chồng toán tử !

```
// Phương thức nạp chồng toán tử phủ định !
// Ta dùng toán tử này để lấy phân số tối giản
public static PhanSo operator !(PhanSo x)
{
    return x.RutGon();
}
```

Bước 3. Trong hàm Main, nhập đoạn mã sau:

```
// Tạo một thể hiện của lớp PhanSo
PhanSo baSau = new PhanSo(3, 6);

// Rút gọn phân số
PhanSo toiGian = !baSau;

// Xuất kết quả
Console.WriteLine("{0} = {1}", baSau, toiGian);
```

Bước 4. Chạy chương trình và ghi lại kết quả.

Bước 5. Trở lại lớp PhanSo, định nghĩa thêm các toán tử ~ và – như sau

```
// Phương thức nạp chồng toán tử phủ định ~
// Ta dùng toán tử này để lấy nghịch đảo
public static PhanSo operator ~(PhanSo x)
{
    return x.NghichDao();
}

// Phương thức nạp chồng toán tử dấu âm -
public static PhanSo operator -(PhanSo x)
{
    return new PhanSo(-x.TuSo, x.MauSo);
}
```

Bước 6. Trong hàm Main, nhập đoạn mã sau để kiểm tra các phương thức vừa định nghĩa

```
// Nghịch đảo phân số
PhanSo dao = ~baSau;

// Xuất kết quả
Console.WriteLine("{0} ==nghich dao==> {1}", baSau, dao);

// Toán tử dấu âm
PhanSo am = -dao;

// Xuất kết quả
Console.WriteLine(am);
```

Bước 7. Chạy chương trình và cho biết kết quả.

Bước 8. Tiếp tục định nghĩa phép toán tự tăng, tự giảm cho trong lớp PhanSo như sau

```
// Phương thức nạp chồng toán tử tự tăng ++
public static PhanSo operator ++(PhanSo x)
{
    return new PhanSo(x.TuSo + x.MauSo, x.MauSo);
}

// Phương thức nạp chồng toán tử tự giảm
public static PhanSo operator --(PhanSo x)
{
    return new PhanSo(x.TuSo - x.MauSo, x.MauSo);
}
```

Bước 9. Trong hàm Main, bổ sung các lệnh sau

```
// Toán tử tự tăng = cộng phân số với 1
toiGian++;
```

```
// Xuất kết quả
Console.WriteLine(toiGian);

// Toán tử tự tăng/giảm có thể dùng trước hoặc sau biến
--baSau;

// Xuất kết quả
Console.WriteLine(baSau);
```

Bước 10. Chạy chương trình, ghi nhận kết quả và giải thích tại sao có kết quả đó.

## 4. Các toán tử ép kiểu

Ta có thể ép kiểu một đối tượng về một giá trị true/false bằng cách nạp chồng toán tử true/false. Khi đó, có thể đặt đối tượng đó làm điều kiện trong lệnh if hay while. .Net framework sẽ tự động ép kiểu sang giá trị true/false. Tuy nhiên, ghi nhớ rằng, không thể dùng phép so sánh một đối tượng với giá trị true/false.

Trong ví dụ này, ta giả sử một phân số x là true nếu giá trị khác 0 và là false nếu ngược lại. Việc định nghĩa toán tử true/false được thực hiện như sau:

Bước 1. Tạo dự án mới, dạng Console Application, đặt tên là Lab04_Bai03_ToanTuEpKieu

Bước 2. Tạo lớp PhanSo như mục 1 rồi thêm đoạn mã sau để nạp chồng toán tử true/false

```
// Phương thức nạp chồng toán tử true
public static bool operator true(PhanSo x)
{
    return x.TuSo != 0;
}

// Phương thức nạp chồng toán tử false
public static bool operator false(PhanSo x)
{
    return x.TuSo == 0;
}
```

Bước 3. Trong hàm Main, bổ sung đoạn mã sau

```
// Minh họa toán tử true/false. Phân số k = 0/1.
PhanSo k = new PhanSo();

if (k)
    Console.WriteLine("Phan so co gia tri khac 0");
else
    Console.WriteLine("Phan so co gia tri bang 0");

Console.ReadKey();
```

Bước 4. Chạy chương trình, ghi nhận kết quả và giải thích.

Ta cũng có thể ép kiểu từ kiểu phân số sang một kiểu khác (chẳng hạn như int, double) và ngược lại. Việc ép kiểu có thể thực hiện theo 2 cách: ép kiểu tường minh và ép kiểu không tường minh.

Bước 5. Trong lớp PhanSo, định nghĩa các phương thức sau để ép một số nguyên hoặc một số thực sang một phân số.

```
// Ép kiểu không tường minh (implicit)

// Phương thức nạp chồng toán tử ép kiểu
// từ một số nguyên sang một phân số
public static implicit operator PhanSo(int k)
{
    return new PhanSo(k, 1);
}

// Phương thức nạp chồng toán tử ép kiểu
// từ một số thực sang một phân số
public static implicit operator PhanSo(double k)
{
    int mauMoi = 1;
    while (k - (int)k > 0.0000001)
    {
        k *= 10;
        mauMoi *= 10;
    }

    return new PhanSo((int)k, mauMoi);
}
```

Bước 6. Sau khi định nghĩa 2 phương thức trên, ta có thể gán một giá trị số nguyên hay một số thực vào một biến PhanSo như sau (viết trong hàm Main):

```
// Tạo một thể hiện của phân số bằng cách
// ép kiểu không tường minh từ một số nguyên
PhanSo haiMot = 2;

// Xuất kết quả
Console.WriteLine(haiMot);

// Tạo một thể hiện của phân số bằng cách
// ép kiểu không tường minh từ một số thực
PhanSo haiBa = 2.345;

// Xuất kết quả
Console.WriteLine(haiBa);
```

Bước 7. Chạy chương trình và ghi nhận kết quả.

Bước 8. Giải thích từng bước cách tạo một phân số từ một số thực như trong đoạn mã trên.

Bước 9. Tiếp theo, ta định nghĩa một phương thức (trong lớp PhanSo) để ép kiểu tường minh từ một phân số về một giá trị thực (số thực). Ở đây, ta dùng từ khóa explicit.

```
// Ép kiểu tường minh (explicit)

// Phương thức nạp chồng toán tử ép kiểu
// từ một phân số sang một số thực
public static explicit operator double (PhanSo x)
{
    return x.GiaTri();
}
```

Bước 10. Với việc định nghĩa phương thức này, ta có thể gán một phân số cho một biến số thực một cách tường minh (thể hiện rõ phép ép kiểu) như sau:

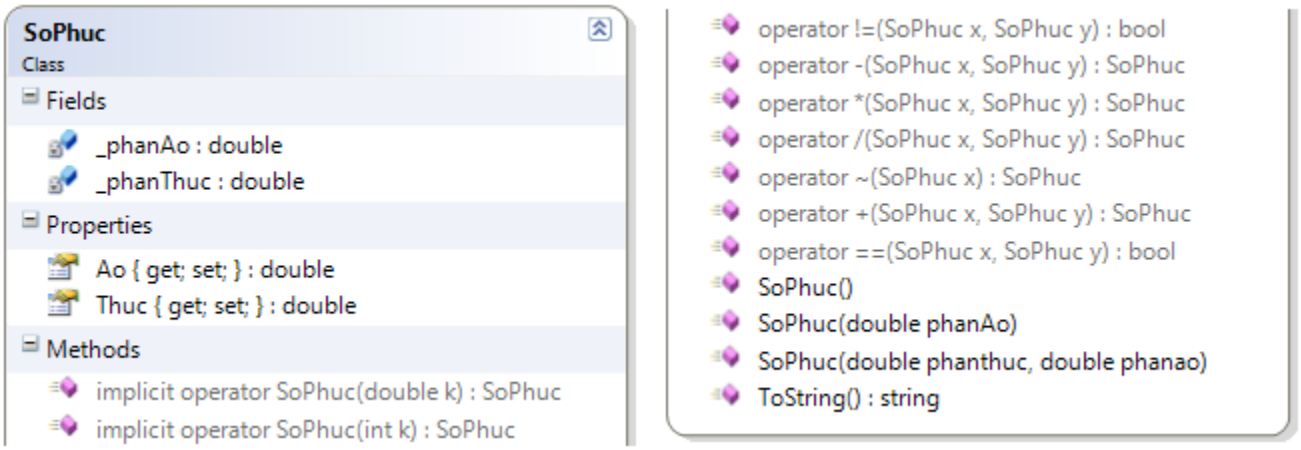
```
// Ép kiểu tường minh từ một phân số sang số thực
double x = (double) haiMot;
Console.WriteLine("{0} = {1}", haiMot, x);

x = (double)haiBa;
Console.WriteLine("{0} = {1}", haiBa, x);
```

Bước 11. Chạy chương trình, ghi nhận lại kết quả và giải thích tại sao có kết quả đó.

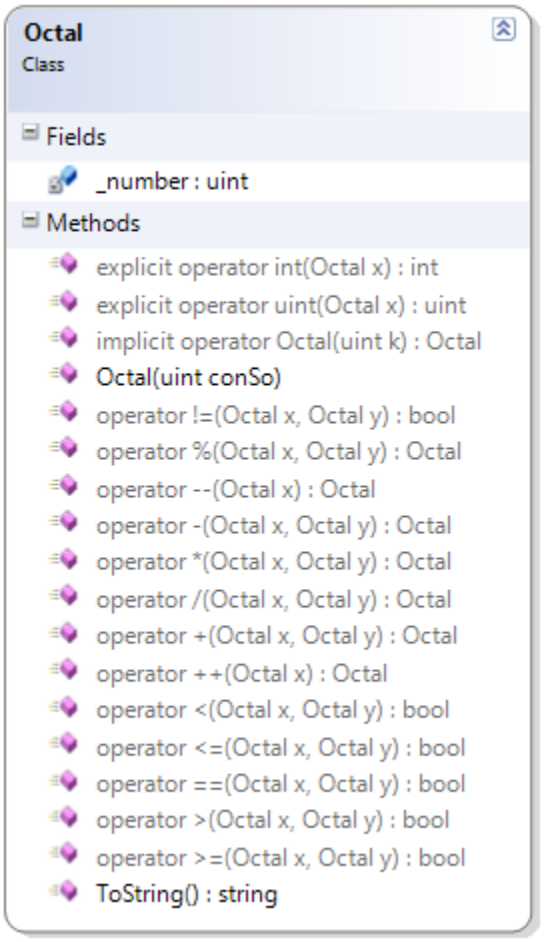
## D. Bài tập bắt buộc

1. Viết chương trình thực hiện các phép toán cộng, trừ, nhân, chia trên trường số phức. Xem lại bài tập 6 ở Lab 2 để biết định nghĩa và các phép toán trên số phức.

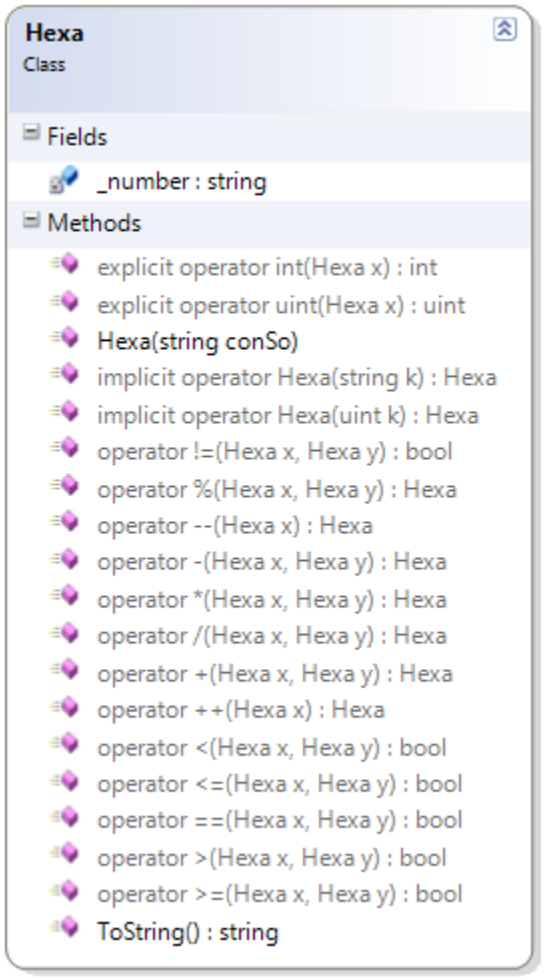



- Phép toán ~ dùng để tìm số phức liên hợp của một số phức x cho trước.
- Khi ép một số nguyên/thực sang số phức, giá trị này được gán cho phần thực, phần ảo = 0

2. Viết chương trình thực hiện các phép toán cộng, trừ, nhân, chia, so sánh trên các số hệ bát phân (cơ số 8). Ngoài ra, định nghĩa các phương thức để , ép kiểu từ số nguyên dương hệ 10 sang số hệ 8 (chẳng hạn: Octal oc = 100;) và ngược lại (ví dụ: uint so = (uint)oc;). Các số hệ 8 được biểu diễn bởi lớp Octal như sau:



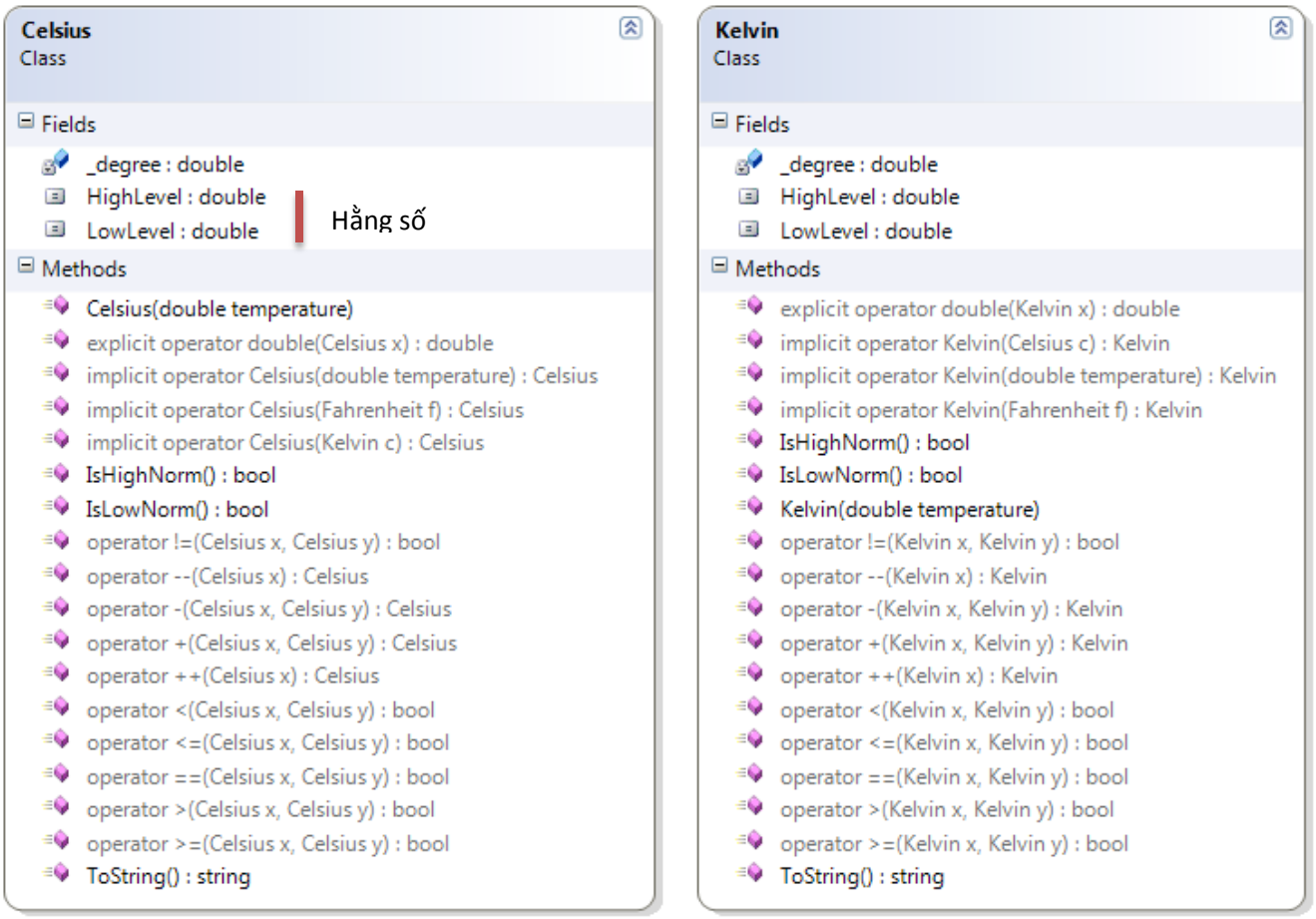

3. Viết chương trình thực hiện các phép toán cộng, trừ, nhân, chia, so sánh trên các số hệ thập lục phân (cơ số 16). Ngoài ra, định nghĩa các phương thức để , ép kiểu từ số nguyên dương hệ 10 sang số hệ 16 (chẳng hạn: Hexa h = 100;) và ngược lại (ví dụ: uint so = (uint)h;). Các số hệ 16 được biểu diễn bởi lớp Hexa như sau:




Ví dụ về cách sử dụng các toán tử đã định nghĩa:

- Octal x = 109; // chuyển số 109 sang số ở hệ 8 rồi lưu vào đối tượng x.
- uint k = (uint)x; // Chuyển số ở hệ 8 được lưu trong đối tượng x thành một số hệ 10.
- Hexa h = k, z = "A0B23"; // Chuyển số k ở hệ 10 và chuỗi A0B23 thành số ở hệ 16.
- Hexa du = z % h; // Tính số dư (ở hệ 16) khi lấy số z chia cho số h.
- bool b = z >= h; // Kiểm tra có phải z lớn hơn h

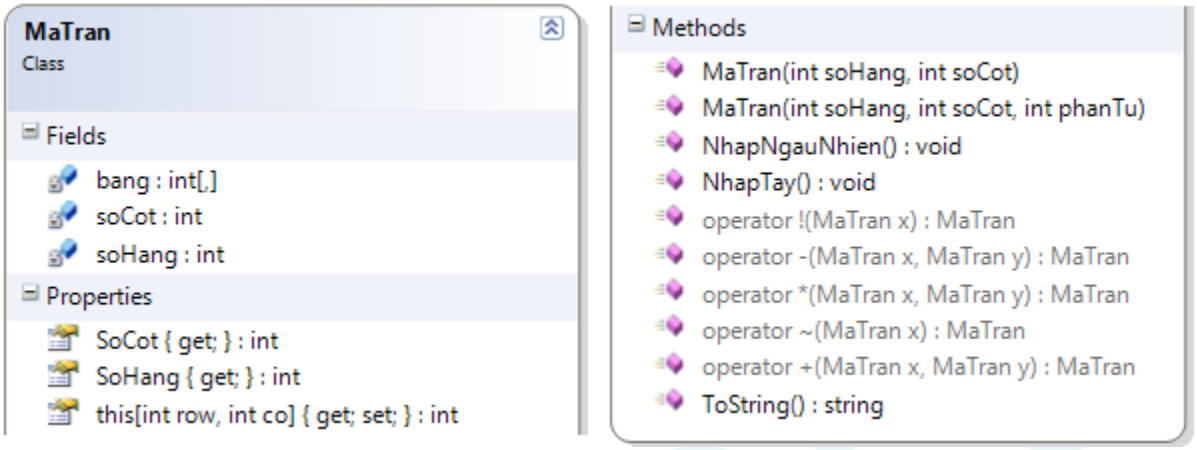
4. Hiện nay, có nhiều thang đo nhiệt độ khác nhau. Trong đó, phổ biến nhất là độ C (Celsius), độ K (Kelvin) và độ F (Fahrenheit). Chi tiết về các thang đo này có thể xem thêm tại: http://vi.wikipedia.org/wiki/%C4%90%E1%BB%99_Fahrenheit. Hãy cài đặt các lớp để biểu diễn các thang đo nhiệt độ này và các phép toán để chuyển một giá trị từ thang đo này sang thang đo khác theo sơ đồ lớp dưới đây:




- Lớp Fahrenheit tương tự 2 lớp trên, chỉ khác các phương thức đổi từ độ C, độ K sang độ F.
- LowLevel là nhiệt độ tại điểm chuẩn thứ nhất, HighLevel là nhiệt độ tại điểm chuẩn thứ hai.
- Phương thức IsLowNorm: kiểm tra nhiệt độ lưu trong đối tượng có bằng LowLevel?
- Phương thức IsHighNorm: kiểm tra nhiệt độ lưu trong đối tượng có bằng HighLevel?
- Kiểm tra các phép ép kiểu như sau:

```
Fahrenheit f = 100.0; // hoặc new Fahrenheit(100.0);
Console.Write("{0} Fahrenheit", f);
Celsius c = f;
Console.Write(" = {0} Celsius", c);
Kelvin k = c;
Console.WriteLine(" = {0} Kelvin", k);
```

5. Viết chương trình thực hiện các phép toán cộng, trừ, nhân hai ma trận và tìm ma trận chuyển vị, ma trận nghịch đảo của một ma trận cho trước.




Trong đó, phép toán ! được dùng để lấy ma trận nghịch đảo, phép toán ~ được dùng để tìm ma trận chuyển vị. Để tìm ma trận nghịch đảo của ma trận A, ta phải tính định thức của A. Công thức tính định thức của ma trận được cho như sau:

Cho ma trận $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$ Khi đó: $|A|=\det(A)= \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$

Khi khai triển theo dòng i, định thức của ma trận A được tính bởi

$$|A| = (-1)^i a_{i1}|A_{i1}|+(-1)^{i+1}a_{i2}|A_{i2}|+ \dots +(-1)^{i+j-1}a_{ij}|A_{ij}|+ \dots +(-1)^{i+n-1}a_{in}|A_{i+n}|$$

Trong đó, $|A_{ij}|$ là định thức của ma trận thu được sau khi đã bỏ đi hàng i, cột j khỏi ma trận A.

6. Viết chương trình thực hiện các phép toán cộng, trừ, nhân, chia hai đa thức. Mỗi đa thức được biểu diễn bởi một mảng một chiều chứa các hệ số của đa thức. Xem lại bài tập 7, Lab 3 để biết cách định nghĩa lớp DaThuc theo dạng này. Tạo 2 thể hiện của lớp DaThuc để biểu diễn 2 đa thức sau: $f = 3x^5 - 2x^3 - 7x^2 - 1$, $g = x^4 + 5x - 2$.

   Cho biết kết quả của các phép toán sau:

   a. DaThuc h = f + g;
   b. DaThuc p = f * g;
   c. DaThuc q = f / g;
   d. DaThuc k = f % g;

## E. Bài tập làm thêm

1. Cài đặt lại lớp Octal trong bài tập 2 ở phần trên để biểu diễn được số thực (lấy tối đa 3 chữ số bên phải dấu chấm thập phân). Định nghĩa thêm các phương thức để thực hiện việc ép kiểu một số thực sang số Octal và ngược lại.
2. Cài đặt lại lớp Hexa trong bài tập 3 ở phần trên để biểu diễn được số thực (lấy tối đa 3 chữ số bên phải dấu chấm thập phân). Định nghĩa thêm các phương thức để thực hiện việc ép kiểu một số thực sang số Hexa và ngược lại.
3. Viết chương trình thực hiện các phép toán cộng, trừ, nhân, chia hai đa thức. Trong đó, đa thức được biểu diễn như trong bài tập 3, phần E của Lab03.
4. Cài đặt lại lớp NgayThang trong Lab02 để bổ sung thêm các toán tử ++ (tăng thêm 1 ngày), -- (giảm bớt 1 ngày), các phép toán so sánh (==, !=, <, >, <=, >=), phép ép kiểu từ một số nguyên dương sang kiểu NgayThang. Số nguyên đó chính là số ngày tính từ ngày 1/1/1970.
5. Cài đặt lớp ThoiGian gồm 3 thuộc tính Giờ, Phút, Giây. Bổ sung các phép toán ++ (tăng 1 giây), -- (giảm 1 giây) và các phép toán so sánh.