



# Thiết kế lớp hỗn số

# THIẾT KẾ LỚP CHonSo

# THIẾT KẾ LỚP CHonSo

## - Thuộc tính:

- Phần nguyên.
- Phần tử số.
- Phần mẫu số.

## - Phương thức:

- Nhóm các phương thức khởi tạo.
- Nhóm các phương thức cung cấp thông tin.
- Nhóm phương thức cập nhật thông tin.
- Nhóm các phương thức kiểm tra.
- Nhóm các phương thức xử lý.

# THIẾT KẾ LỚP CHonSo

- Nhóm các phương thức khởi tạo:
  - 1. Phương thức khởi tạo mặc định.
  - 2. Phương thức khởi tạo sao chép.
  - 3. Phương thức khởi tạo khi biết đầy đủ thông tin.
  - 4. Phương thức thiết lập mặc định.
  - 5. Phương thức thiết lập sao chép.
  - 6. Phương thức thiết lập khi biết đầy đủ thông tin.
  - 7. Phương thức Nhập.
  - 8. Toán tử vào.

# THIẾT KẾ LỚP CHonSo

- Nhóm các phương thức cung cấp thông tin:
  - 1. Phương thức Xuất.
  - 2. Toán tử ra.
  - 3. Phương thức cung cấp phần nguyên.
  - 4. Phương thức cung cấp tử số.
  - 5. Phương thức cung cấp mẫu số.

# THIẾT KẾ LỚP CHonSo

- Nhóm các phương thức cập nhật thông tin:
  - 1. Toán tử gán.
  - 2. Phương thức cập nhật phần nguyên.
  - 3. Phương thức cập nhật phần tử số.
  - 4. Phương thức cập nhật phần mẫu số.

# THIẾT KẾ LỚP CHonSo

- Phương thức kiểm tra:
  - Kiểm tra có phải hỗn số không?

# THIẾT KẾ LỚP CHonSo

- Nhóm các phương thức xử lý:
  - 1. Toán tử so sánh bằng
  - 2. Toán tử so sánh khác
  - 3. Toán tử so sánh lớn hơn
  - 4. Toán tử so sánh nhỏ hơn
  - 5. Toán tử so sánh lớn hơn bằng
  - 6. Toán tử so sánh nhỏ hơn bằng



# THIẾT KẾ LỚP CHonSo

- Nhóm các phương thức xử lý:
  - 7. Toán tử cộng
  - 8. Toán tử trừ
  - 9. Toán tử nhân
  - 10. Toán tử chia

# KHAI BÁO LỚP CHonSo

# KHAI BÁO LỚP CHonSo

- Thuộc tính:
  - Phần nguyên.
  - Phần tử số.
  - Phần mẫu số.

# KHAI BÁO LỚP CHonSo

```
class CHonSo
{
    private:
        int nguyen;
        int tu;
        int mau;
    public:
```

# KHAI BÁO LỚP CHonSo

## - Thuộc tính:

- Phần nguyên.
- Phần tử số.
- Phần mẫu số.

## - Phương thức:

- Nhóm các phương thức khởi tạo.
- Nhóm các phương thức cung cấp thông tin.
- Nhóm phương thức cập nhật thông tin.
- Nhóm các phương thức kiểm tra.
- Nhóm các phương thức xử lý.

# KHAI BÁO LỚP CHonSo

```
class CHonSo
{
    private:
        int nguyen;
        int tu;
        int mau;
    public:
        // Nhóm phương thức khởi tạo
        // Nhóm phương thức cung cấp thông tin
        // Nhóm phương thức cập nhật thông tin
        // Nhóm phương thức kiểm tra
        // Nhóm phương thức xử lý
```

# KHAI BÁO LỚP CHonSo

## - Nhóm các phương thức khởi tạo:

- 1. Phương thức khởi tạo mặc định.
- 2. Phương thức khởi tạo sao chép.
- 3. Phương thức khởi tạo khi biết đầy đủ thông tin.
- 4. Phương thức thiết lập mặc định.
- 5. Phương thức thiết lập sao chép.
- 6. Phương thức thiết lập khi biết đầy đủ thông tin.
- 7. Phương thức Nhập.
- 8. Toán tử vào.

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**const** CHonSo&);

**void** KhoiTao(**int**, **int**, **int**);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream&** operator>>(**istream&**, CHonSo&);



# KHAI BÁO LỚP CHonSo

- Nhóm các phương thức cung cấp thông tin:
  - 1. Phương thức Xuất.
  - 2. Toán tử ra.
  - 3. Phương thức cung cấp phần nguyên.
  - 4. Phương thức cung cấp tử số.
  - 5. Phương thức cung cấp mẫu số.

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức cung cấp thông tin

**void** Xuat();

**friend ostream& operator<<**(ostream&, CHonSo&);

**int** GetNguyen();

**int** GetTu();

**int** GetMau();

# KHAI BÁO LỚP CHonSo

- Nhóm các phương thức cập nhật thông tin:
  - 1. Toán tử gán.
  - 2. Phương thức cập nhật phần nguyên.
  - 3. Phương thức cập nhật phần tử số.
  - 4. Phương thức cập nhật phần mẫu số.

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cập nhật thông tin  
CHonSo& operator = (const CHonSo&);  
void SetNguyen(int);  
void SetTu(int);  
void SetMau(int);
```

# KHAI BÁO LỚP CHonSo

- Phương thức kiểm tra:
  - Kiểm tra có phải hỗn số không?

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức kiểm tra  
int isHonSo();
```

# KHAI BÁO LỚP CHonSo

- Nhóm các phương thức xử lý:
  - 1. Toán tử so sánh bằng
  - 2. Toán tử so sánh khác
  - 3. Toán tử so sánh lớn hơn
  - 4. Toán tử so sánh nhỏ hơn
  - 5. Toán tử so sánh lớn hơn bằng
  - 6. Toán tử so sánh nhỏ hơn bằng

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
    int operator == (const CHonSo&);  
    int operator != (const CHonSo&);  
    int operator > (const CHonSo&);  
    int operator >= (const CHonSo&);  
    int operator < (const CHonSo&);  
    int operator <= (const CHonSo&);
```



# KHAI BÁO LỚP CHonSo

- Nhóm các phương thức xử lý:
  - 7. Toán tử cộng
  - 8. Toán tử trừ
  - 9. Toán tử nhân
  - 10. Toán tử chia

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```

# PHƯƠNG THỨC KHỞI TẠO

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**const** CHonSo&);

**void** KhoiTao(**int**, **int**, **int**);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& operator>>(**istream**&, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
void CHonSo::KhoiTao()  
{  
    nguyen = 0;  
    tu = 0;  
    mau = 1;  
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**const** CHonSo&);

**void** KhoiTao(**int**, **int**, **int**);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& operator>>(**istream**&, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
void CHonSo::KhoiTao(const CHonSo &P)
{
    nguyen = P.nguyen;
    tu = P.tu;
    mau = P.mau;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**const** CHonSo&);

**void** KhoiTao(**int**, **int**, **int**);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& operator>>(**istream**&, CHonSo&);



# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
void CHonSo::KhoiTao(int nn, int tt, int mm)
{
    nguyen = nn;
    tu = tt;
    mau = mm;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**int**, **int**, **int**);

**void** KhoiTao(**const** CHonSo&);

**CHonSo**();

**CHonSo**(**int**, **int**, **int**);

**CHonSo** KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& **operator**>>(**istream**&, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
CHonSo::CHonSo()  
{  
    nguyen = 0;  
    tu = 0;  
    mau = 1;  
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**int**, **int**, **int**);

**void** KhoiTao(**const** CHonSo&);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& operator>>(**istream**&, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
CHonSo::CHonSo(int nn, int tt, int mm)
{
    nguyen = nn;
    tu = tt;
    mau = mm;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**int**, **int**, **int**);

**void** KhoiTao(**const** CHonSo&);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream**& operator>>(**istream**&, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
CHonSo::CHonSo(const CHonSo &P)
{
    nguyen = P.nguyen;
    tu = P.tu;
    mau = P.mau;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức khởi tạo  
void KhoiTao();  
void KhoiTao(int, int, int);  
void KhoiTao(const CHonSo&);  
CHonSo();  
CHonSo(int, int, int);  
CHonSo KhoiTao(const CHonSo&);  
void Nhap();  
friend istream& operator>>(istream&, CHonSo&);
```



# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
void CHonSo::Nhap()  
{  
    cout << "Nhap phan nguyen: ";  
    cin >> nguyen;  
    cout << "Nhap tu: ";  
    cin >> tu;  
    cout << "Nhap mau: ";  
    cin >> mau;  
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức khởi tạo

**void** KhoiTao();

**void** KhoiTao(**int**, **int**, **int**);

**void** KhoiTao(**const** CHonSo&);

CHonSo();

CHonSo(**int**, **int**, **int**);

CHonSo KhoiTao(**const** CHonSo&);

**void** Nhap();

**friend** **istream&** operator>>(**istream&**, CHonSo&);

# NHÓM PHƯƠNG THỨC KHỞI TẠO

```
istream& operator >> (istream& is, CHonSo &P)
{
    cout << "Nhap phan nguyen: " ;
    is >> P.nguyen;
    cout << "Nhap tu: ";
    is >> P.tu;
    cout << "Nhap mau: ";
    is >> P.mau;
    return is;
}
```

# PHƯƠNG THỨC CUNG CẤP THÔNG TIN

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức cung cấp thông tin

**void** Xuat();

**friend ostream& operator<<**(ostream&, CHonSo&);

**int** GetNguyen();

**int** GetTu();

**int** GetMau();

# Nhóm phương thức cung cấp thông tin

```
void CHonSo::Xuat()  
{  
    cout<<nguyen<<" + ("<< tu<<" / "<<mau<< " ) "<<";  
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức cung cấp thông tin

**void** Xuat();

**friend ostream& operator<<**(ostream&,CHonSo&);

**int** GetNguyen();

**int** GetTu();

**int** GetMau();

# Nhóm phương thức cung cấp thông tin

```
ostream& operator << (ostream& os, CHonSo& P)
{
    os<<P.nguyen<<" + ("<<P.tu<<" / "<< P.mau<<" ) ";
    return os;
}
```



# KHAI BÁO LỚP CHonSo

// Nhóm phương thức cung cấp thông tin

**void** Xuat();

**friend** ostream& **operator**<<(ostream&,CHonSo&);

**int** GetNguyen();

**int** GetTu();

**int** GetMau();

# Nhóm phương thức cung cấp thông tin

```
int CHonSo::GetNguyen()  
{  
    return nguyen;  
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức cung cấp thông tin

**void** Xuat();

**friend** ostream& **operator**<<(ostream&,CHonSo&);

**int** GetNguyen();

**int** GetTu();

**int** GetMau();

# Nhóm phương thức cung cấp thông tin

```
int CHonSo::GetTu()  
{  
    return tu;  
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cung cấp thông tin  
void Xuat();  
friend ostream& operator<<(ostream&,CHonSo&);  
int GetNguyen();  
int GetTu();  
int GetMau();
```

# Nhóm phương thức cung cấp thông tin

```
int CHonSo::GetMau()  
{  
    return mau;  
}
```

# PHƯƠNG THỨC CẬP NHẬT THÔNG TIN

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cập nhật thông tin  
CHonSo& operator = (const CHonSo&);  
void SetNguyen(int);  
void SetTu(int);  
void SetMau(int);
```



# Nhóm phương thức cập nhật thông tin

```
CHonSo& CHonSo::operator = (const CHonSo &P)
{
    nguyen = P.nguyen;
    tu = P.tu;
    mau = P.mau;
    return *this;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cập nhật thông tin  
CHonSo& operator = (const CHonSo&);  
void SetNguyen(int);  
void SetTu(int);  
void SetMau(int);
```

# Nhóm phương thức cập nhật thông tin

```
void CHonSo::SetNguyen(int nn)
{
    nguyen = nn;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cập nhật thông tin  
CHonSo& operator = (const CHonSo&);  
void SetNguyen(int);  
void SetTu(int);  
void SetMau(int);
```

# Nhóm phương thức cập nhật thông tin

```
void CHonSo::SetTu(int tt)
{
    tu = tt;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức cập nhật thông tin  
CHonSo& operator = (const CHonSo&);  
void SetNguyen(int);  
void SetTu(int);  
void SetMau(int);
```

# Nhóm phương thức cập nhật thông tin

```
void CHonSo::SetMau(int mm)
{
    mau = mm;
}
```

# PHƯƠNG THỨC KIỂM TRA



# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức kiểm tra  
int isHonSo();
```

# Nhóm phương thức kiểm tra

```
int CHonSo::isHonSo()  
{  
    if (mau != 0)  
        return 1;  
    return 0;  
}
```

# PHƯƠNG THỨC XỬ LÝ

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);  
int operator != (const CHonSo&);  
int operator > (const CHonSo&);  
int operator >= (const CHonSo&);  
int operator < (const CHonSo&);  
int operator <= (const CHonSo&);
```

# Nhóm phương thức xử lý

```
int CHonSo::operator==(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a == b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);
```

```
int operator != (const CHonSo&);
```

```
int operator > (const CHonSo&);
```

```
int operator >= (const CHonSo&);
```

```
int operator < (const CHonSo&);
```

```
int operator <= (const CHonSo&);
```

# Nhóm phương thức xử lý

```
int CHonSo::operator!=(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a != b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);
```

```
int operator != (const CHonSo&);
```

```
int operator > (const CHonSo&);
```

```
int operator >= (const CHonSo&);
```

```
int operator < (const CHonSo&);
```

```
int operator <= (const CHonSo&);
```



# Nhóm phương thức xử lý

```
int CHonSo::operator>(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a > b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);
```

```
int operator != (const CHonSo&);
```

```
int operator > (const CHonSo&);
```

```
int operator >= (const CHonSo&);
```

```
int operator < (const CHonSo&);
```

```
int operator <= (const CHonSo&);
```

# Nhóm phương thức xử lý

```
int CHonSo::operator>=(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a >= b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);
```

```
int operator != (const CHonSo&);
```

```
int operator > (const CHonSo&);
```

```
int operator >= (const CHonSo&);
```

```
int operator < (const CHonSo&);
```

```
int operator <= (const CHonSo&);
```

# Nhóm phương thức xử lý

```
int CHonSo::operator<(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a < b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

// Nhóm phương thức xử lý

```
int operator == (const CHonSo&);  
int operator != (const CHonSo&);  
int operator > (const CHonSo&);  
int operator >= (const CHonSo&);  
int operator < (const CHonSo&);  
int operator <= (const CHonSo&);
```

# Nhóm phương thức xử lý

```
int CHonSo::operator<=(const CHonSo &P)
{
    float a = (float)(((nguyen*mau)+tu)/mau);
    float b = (float)(((P.nguyen*P.mau)+P.tu)/P.mau);
    if(a <= b)
        return 1;
    return 0;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```



# Nhóm phương thức xử lý

```
CHonSo CHonSo::operator+(CHonSo P)
{
    CHonSo temp;
    temp.nguyen = nguyen + P.nguyen;
    temp.tu = tu*P.mau + mau*P.tu;
    temp.mau = mau*P.mau;
    return temp;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```

## Nhóm phương thức xử lý

```
CHonSo CHonSo::operator-(CHonSo P)
{
    CHonSo temp;
    temp.nguyen = nguyen - P.nguyen;
    temp.tu = tu * P.mau - mau * P.tu;
    temp.mau = mau * P.mau;
    return temp;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```

# Nhóm phương thức xử lý

```
CHonSo CHonSo::operator*(CHonSo P)
{
    CHonSo temp;
    temp.tu = ((nguyen*mau)+tu)*((P.nguyen*P.mau)+P.tu);
    temp.mau = mau*P.mau;
    temp.nguyen = 0;
    return temp;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```

# Nhóm phương thức xử lý

```
CHonSo CHonSo::operator/(CHonSo P)
{
    CHonSo temp;
    temp.tu = ((nguyen*mau)+tu)*P.mau;
    temp.mau = mau*((P.nguyen*P.mau)+P.tu);
    temp.nguyen = 0;
    return temp;
}
```

# KHAI BÁO LỚP CHonSo

```
// Nhóm phương thức xử lý  
CHonSo operator+(CHonSo);  
CHonSo operator-(CHonSo);  
CHonSo operator*(CHonSo);  
CHonSo operator/(CHonSo);  
~CHonSo();
```



# Nhóm phương thức xử lý

```
CHonSo::~CHonSo()  
{  
    return;  
}
```

# Kết thúc.

- Nguyễn Văn Linh – 20520613
  - Hồng Gia Hy – 20520561
- Nguyễn Huy Trí Dũng -20520459
  - Đỗ Trọng Tình – 20520318
- Nguyễn Công Đoàn - 20520447