CHƯƠNG 2: CÁC MỞ RỘNG CỦA C++

❒ Bài tập có hướng dẫn

1. Viết hàm GiaiPTBậc2() sau, hàm này trả về số nghiệm n và giá trị các nghiệm x1, x2 (nếu có) của một phương trình bậc 2 với các hệ số a, b, c:

   void GiaiPTBậc2 (float a, float b, float c, int &n, float &x1, float &x2);

   Hướng dẫn:

```
void GiaiPTBậc2 (float a, float b, float c, int &sn, float &x1, float &x2) {
    float delta = b*b – 4*a*c ;
    if(delta < 0)
            sn = 0
    else if(delta == 0) {
            sn = 1;
            x1 = x2 = -b / (2*a);
    }
    else {
            sn = 2;
            x1 = (-b + sqrt(delta)) / (2*a);
            x2 = (-b - sqrt(delta)) / (2*a);
    }
}
```

2. Viết hàm SapXep() sau, hàm thực hiện sắp xếp một mảng số nguyên theo chiều tăng dần hoặc giảm dần, mặc định kiểu sắp xếp là tăng dần.

   void SapXep(int a[], int n, int flag = 0);

   Khi tham số hình thức thứ ba nhận giá trị mặc định bằng 0 hàm sắp xếp tăng dần, ngược lại hàm sắp xếp giảm dần.

   Hướng dẫn:

```
void Nhap(int a[], int n) {
    do {
            <Nhap n>
    } while(n < 1 || n > SIZE)
    for(int i = 0; i < n; i++) {
            cout << “Phan tu thu “ << i <<”:”;
            cin >> a[i];
```

```
    }
}
void Xuat(int a[], int n) {
    for(int i = 0; i < n; i++)
        cout << a[i] <<"\t";
    cout <<"\n";
}
void SapXep(int a[], int n, int flag) {
    int i, j;
    for(i = 0;  i < n-1;  i++)
        for(j = i + 1;  j < n;  j++)
            if(flag == 0 ? a[i] > a[j] : a[i] < a[j]) {
                int tam = a[i];
                a[i] = a[j];
                a[j] = tam;
            }
}
void main() {
int a[SIZE];
    int n;
    Nhap(a, n);
    Xuat(a, n);
    SapXep(a, n); //Sắp tăng dần
    Xuat(a, n);
    SapXep(a, n, 1); //Sắp giảm dần
    Xuat(a, n);
}
```

3. Sử dụng toán tử new để cấp phát vùng nhớ cho mảng động một chiều nguyên, sau đó thực hiện các thao tác sau:
   a. Nhập mảng.
   b. Xuất mảng.
   c. Tính tổng các phần tử là số nguyên tố.
   d. Sắp xếp các phần tử là số nguyên tố tăng dần.
   e. Tìm vị trí phần tử nguyên tố đầu tiên.
   f. Xóa phần tử nguyên tố đầu tiên.

Sau khi thực hiện xong các thao tác thì sử dụng toán tử delete để giải phóng vùng nhớ đã được cấp phát trước đó.

Hướng dẫn:

```
//Hàm kiểm tra nguyên tố
int NguyenTo(int x) {
    int nt;
    if(x < 2)
        nt = 0;          //Không là nguyên tố
    else {
        nt = 1;          //Là nguyên tố
        for(int i = 2; i < x; i++)
            if(x % i == 0) {
                nt = 0;          //Không là nguyên tố
                break;
            }
    }
    return nt;
}
//Câu 2a
void Nhap(int *p, int n) {
    do {
        printf("Nhap so phan tu:");
        scanf("%d", &n);
    } while(n < 1 || n > SIZE);
    for(int i = 0; i < n; i++)  {
        printf("pt thu %d:", i);
        scanf("%d", &p[i]);
    }
}
//Câu 2b
void Xuat(int *p, int n) {
    for(int i = 0; i < n; i++)
        printf("%d\t", p[i]);
    printf("\n");
}
//Câu 2c
```

```
int TôngNT(int p[], int n) {
	int ret = 0;
	int i;
	for(i = 0; i < n; i++)
		if(NguyenTo(p[i]))	ret += p[i];
	return ret;
}
//Câu 2d
void SapXepNTTang(int p[], int n) {
	int i, j;
	for(i = 0; i < n - 1; i++)
		for(j = i + 1; j <n; j++) {
			if(NguyenTo(p[i]) && NguyenTo(p[j]) && p[i] > p[j]) { /*so
sánh p[i] và p[j]*/
			//Hoán vị p[i] và p[j]
			int tam = p[i];
			p[i] = p[j];
			p[j] = tam;
		}
}
//Câu 2e
int TimNTDauTien(int *p, int n) {
	int i, j;
	int ret = -1; //Không tìm thấy
	for(i = 0; i < n; i++)
		if(NguyenTo(p[i])) {
			ret = i;		//Tìm thấy tại vị trí i
			break;
		}
	return ret;
}
//Câu 2f
void xoa(int *p, int &n, int vt) {
	int i;
	if(vt  < 0 || vt >= n)
```

```
            printf("Vi tri %d khong hợp le\n", vt);
    else {
            if(n == 0)
                    printf("Mang rong\n")
            else {
                    // Dời các phần tử từ vị trí vt + 1 đến n - 1 lên một vị trí
                    for(i = vt; i <= n - 1; i++)
                            p[i] = p[i + 1];
                    //Giảm số phần tử xuống 1
                    n--;
            }
    }
}
void XoaNTDauTien(int *p, int &n) {
    int vt = TimNTDauTien(p, n);
    if(vt != -1)
            Xoa(a, &n, vt);
    else
            cout << "Khong tim thay so NT\n";
}
//Hàm chính
void main() {
    int *p;
    int n;
    <Nhap n>
    p = new int[n];          //Cấp phát vùng nhớ động
    Nhap(p, n);
    Xuat(p, n);
    cout << "Tong chinh phuong: " << TongCP(p, n) <<"\n";
    int vitri = TimNTDauTien(p, n);
    if(vitri != -1) {
            cout << " vi tri nguyen to dau tien: " << vitri <<"\n";
            Xoa(p, n, vitri);
            Xuat(p, n);
    }
```

```
    else {
            cout << "Khong tim thay nguyen to\n");
    }
    delete[]p;              //Giải phóng vùng nhớ động
}
```

4. Khai báo kiểu dữ liệu DT để biểu diễn thông tin của một đa thức, sau đó định nghĩa các hàm sau:

```
void Nhap(DT &u);                   //Nhập đa thức theo bậc tăng dần
void Xuat(const DT &u);
DT operator+(const DT &u, const DT &v);   //Toán tử cộng hai đa thức
DT operator-(const DT &u, const DT &v);   // Toán tử trừ hai đa thức
DT operator*(const DT &u, const DT &v);   // Toán tử nhân hai đa thức
void operator-(DT &u);              // Toán tử đảo dấu đa thức
float operator^( const DT &u, float x);  /*Toán tử tính giá trị đa thức
                                            tại x*/
```

Hướng dẫn:

```
#define SIZE 10
struct DT {
    int n;                  //Bậc đa thức
    float arr[SIZE]; //Mảng chứa các hệ số của đa thức
};
void Nhap(DT &u) {
    do {
            cout << "Nhap bac da thuc:";
            cin >> u.n;
    }while(u.n < 1 || u.n > SIZE);
    for(int i = 0; i < u.n + 1; i++) {
            cout << "Nhập hệ số thứ " << i <<":\n";
            cin >> u.arr[i];
    }
}
void Xuat(const DT &u) {
    for(int i = 0; i < u.n + 1; i++)
            cout << u.arr[i] <<"\t";
    cout << "\n";
}
```

```
DT operator+(const DT &u, const DT &v) {
    int k = (u.n > v.n ? u.n : v.n);
    DT ret;
    ret.n = k;
    for(int i = 0; i < ret.n + 1; i++) {
        if(i <= u.n && i <= v.n)
            ret.arr[i] = u.arr[i] + v.arr[i];
        else if(i <= u.n)
            ret.arr[i] = u.arr[i];
        else ret.arr[i] = v.arr[i];
    }
    return ret;
}
DT operator*(const DT &u, const DT &v) {
    int i, j;
    int k =  u.n + v.n;
    DT ret;
    ret.n = k;
    //Khởi tạo các hệ số của đa thức ret bằng 0
    for(i = 0; i < ret.n + 1; i++)
        ret.arr[i] = 0;
    //Nhân hai đa thức
    for(i = 0; i < u.n + 1; i++)
        for(j = 0; j < v.n + 1; j++)
            ret.arr[i+j] += u.arr[i] * v.arr[j];
    retunr ret;
}
float operator^ (DT u, float x) {
    float ret  = 0;
    float t = 1;
    for(int i = 0; i < u.n + 1; i++) {
        ret += u.arr[i] * t;
        t *= x;
    }
    return ret;
}
```

```
void operator-(DT &u) {
    for(int i = 0; i < u.n + 1; i++)
            u.arr[i] = -u.arr[i];
}
```

❒ Bài tập luyện tập

5. Viết hàm GiaiPTTrungPhuong() sau, hàm này trả về số nghiệm sn và các nghiệm x1, x2 , x3, x4 (nếu có) của một phương trình trùng phương với các hệ số a, b và c:

   void GiaiPTTrungPhuong (float a, float b, float c, int &sn, float &x1, float &x2, float &x3, float &x4);

6. Sử dụng toán tử new để cấp phát vùng nhớ động cho mảng hai chiều nguyên, sau đó thực hiện các thao tác sau:

   a. Nhập mảng.
   b. Xuất mảng.
   c. Tính tổng các phần tử là số chính phương.
   d. Sắp xếp các phần tử trên dòng thứ k tăng dần.
   e. Sắp xếp các phần tử là số nguyên tố trên dòng thứ k tăng dần.
   f. Tìm vị trí phần tử lớn nhất.
   g. Tìm vị trí phần tử lớn nhất trên dòng thứ k.

   Sau khi thực hiện xong các thao tác thì sử dụng toán tử delete để giải phóng vùng nhớ đã được cấp phát trước đó.

7. Khai báo kiểu dữ liệu PS để biểu diễn thông tin của một phân số, sau đó định nghĩa các hàm sau:

```
void Nhap(PS &u);
void Xuat(const PS &u);
int USCLN(int x, int y);
void RutGon(PS &u) ;
 //Toán tử số học
PS operator+(const PS &u, const PS &v);
PS operator-(const PS &u, const PS &v);
PS operator*(const PS &u, const PS &v);
 PS operator/(const PS &u, const PS &v);
 //Toán tử quan hệ
 int operator>(const PS &u, const PS &v);
 int operator>=(const PS &u, const PS &v);
 int operator<(const PS &u, const PS &v);
```

```
int operator<=(const PS &u, const PS &v);
int operator==(const PS &u, const PS &v);
int operator!=(const PS &u, const PS &v);
//Toán tử số học mở rộng
void operator+=(PS &u, const PS &v);
void operator-=(PS &u, const PS &v);
void operator*=(PS &u, const PS &v);
void operator/=(PS &u, const PS &v);
```

CHƯƠNG 3: ĐỐI TƯỢNG VÀ LỚP

❒ Bài tập có hướng dẫn

1. Xây dựng lớp sau:

```
class TG {                    //Lớp tam giác
private:
    float a, b, c;      //Ba cạnh tam giác
public:
    TG(float aa = 0, float bb = 0, float cc = 0);
    void Nhap();      //Nhập ba cạnh
    void Xuat();      //Xuất thông tin tam giác
    int HopLe();      //Kiểm tra ba cạnh tam giác hợp lệ không?
    void PhanLoai(); //Phân loại tam giác
    float ChuVi();    //Tính chu vi tam giác
    float DienTich(); //Tính diện tích tam giác
};
```

Hướng dẫn:

```
void TG ::Nhap() {
    cout << "Nhap 3 canh tam giac:";
    cin >> a >> b >> c;
}
void TG ::Xuat() {
    cout << "Chu vi: "<< ChuVi() << "\n";
    cout << "Dien tich: "<< DienTich() << "\n";
    Phanloai();
}
int TG ::HopLe() {
    if(a+ b > c && b + c > a && c + a > b)
            return 1;
    else return 0;
}
void TG ::PhanLoai() {
    if(a == b || b == c || c == a)
            if(a == b && b == c)
```

```
                cout << "Day la tam giac deu\n";
        else if(a * a == b * b + c * c || b * b == a * a + c * c || c *  c == a * a + b
* b)
                cout << "Day la tam giac vuong can\n";
        else cout << "Day la tam giac can\n";
    else if(a * a == b * b + c * c || b * b == a * a + c * c || c *  c == a * a + b * b)
        cout << "Day la tam giac vuong\n";
    else cout << "Day la tam giac thuong\n";
}
float TG::ChuVi() {
    return a + b + c;
}
float TG::DienTich() {
    float p = (a + b + c)/2;
    return sqrt(p* (p – a) * (p – b) * (p – c));
}
```

2. Xây dựng lớp sau:

```
class Ngay {
private:
    int d, m, y;            //ngày, tháng, năm
public:
    Ngay(int dd = 1, int mm = 1, int yy = 1);
    void Nhap();
    void Xuat();
    int Nhuan();    //Kiểm tra năm nhuần
    int SNTrongThang();    //Tình số ngày trong tháng
    int HopLe();            //Kiểm tra ngày hợp lệ
    void TangNgay();        //Tăng ngày lên một ngày
    void GiamNgay();        // Giảm ngày xuống một ngày
    void TangTuan();        //Tăng ngày lên một tuần
    void GiamTuan();        // Giảm ngày xuống một tuần
};
```

Hướng dẫn

```
int Ngay::Nhuan() {
    if(y % 400 == 0) || (y % 4 == 0 && y %100 != 0))
        return 1;
    else return 0;
}
int Ngay::SNTrongThang() {
    int a[12] = { //Mảng số ngày trong 12 tháng
        31, Nhuan() ? 29 : 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
    };
    return a[m – 1];
}
int Ngay::HopLe() {
    if(d > 0 && d <= SNTrongThang() && m > 0 && m < 13 && y > 0)
        return 1;
    else return 0;
}
void Ngay::TangNgay() {
    d++;
    if(d > SNTrongThang()) {
        d = 1;
        m++;
        if(m > 12) {
            m = 1;
            y++;
        }
    }
}
void Ngay::GiamNgay() {
    d--;
    if(d < 1) {
        m--;
        if(m < 1) {
            m = 12;
            y--;
        }
```

```
        d = SNTrongThang();
    }
}
void Ngay::TangTuan() {
    d += 7;
    if(d > SNTrongThang ()) {
            d = d - SNTrongThang ();
            m++;
            if(m > 12) {
                    m = 1;
                    y++;
            }
    }
}
int Ngay::GiamTuan() {
    d -= 7;
    if(d < 1) {
            m--;
            if(m < 1) {
                    m = 12;
                    y--;
            }
            d = d + SNTrongThang ();
    }
}
```

3. Xây dựng lớp sau:

```
class DT {          //Lớp đa thức
private:
    int n;                  //Bậc đa thức
    float *p;               //con trỏ tới vùng nhớ động chứa các hệ số
public:
    DT();
    DT(int nn);
    DT (const DT &u);
    ~ DT ();
```

```
    void Nhap();
    void Xuat();
    float GiaTri(float x);      //Tính giá trị đa thức tại x
    DT Cong(const DT &u); //Cộng hai đa thức
    DT Tru(const DT &u);    //Trừ hai đa thức
    DT Nhan(const DT &u); //Nhân hai đa thức
};
```

❖ **Hướng dẫn:**

```
DT::DT(int nn) {
    n = nn;
    p = new float[n + 1];    //Một đa thức bậc n sẽ có n + 1 hệ số
}
DT DT::Cong(const DT &u) {
    int k = (n > u.n ? n : u.n);
    DT ret(k);                  //Gọi hàm DT::DT(int)
    for(int i = 0; i < ret.n + 1; i++) {
            if(i <= n && i <= u.n)
                    ret.p[i] = p[i] + u.p[i];
            else if(i <= n)
                    ret.p[i] = p[i];
            else ret.p[i] = u.p[i];
    }
    return ret;
}
DT DT::Nhan(const DT &u) {
    int i, j;
    int k =  n + u.n;
    DT ret(k);
    //Khởi tạo các hệ số của đa thức ret bằng 0
    for(i = 0; i < ret.n + 1; i++)
            ret.p[i] = 0;
    //Nhân hai đa thức
    for(i = 0; i < n + 1; i++)
            for(j = 0; j < u.n + 1; j++)
                    ret.p[i+j] += p[i] * u.p[j];
```

```
    return ret;
}
float DT::GiaTri(float x) {
    float ret  = 0;
    float t = 1;
    for(int i = 0; i < n + 1; i++) {
            ret += p[i] * t;
            t *= x;
    }
    return ret;
}
```

4. Với hai lớp **Vector** và **MT** đã có, hãy xây dựng một hàm thực hiện việc nhân ma trận với Vector theo giải pháp: Khai báo hàm này là **hàm thành phần của lớp MT** và là **bạn của lớp Vector**.

Hướng dẫn

```
class Vector;          //Khai báo trước lớp Vector
class MT {
private:
    . . .
public:
    . . .
    Vector NhanMV(const Vector &u);
};
class Vector {
private:
    . . .
public:
    . . .
    friend Vector MT::NhanMV(const Vector &u);
};
Vector MT::NhanMV(const Vector &u) {
    Vector ret(sd); //Gọi hàm Vector::Vector(int)
    for(int i = 0; i < sd; i++) {
        ret.p[i] = 0;
        for(int j = 0; j < u.n; j++)
            ret.p[i] += p[i][j] * u.p[j];
```

```
    }
    return ret;
}
```

5. Xây dựng lớp các sau:

```
class Ngay {
private:
    int d, m, y;                        //ngày, tháng, năm
public:
    Ngay(int dd = 1, int mm = 1, int yy = 1);
    void Nhap();
    void Xuat();
    int Nhuan();                        //Kiểm tra năm nhuần
    int SNTrongThang();     //Tình số ngày trong tháng
    int HopLe();                        //Kiểm tra ngày hợp lệ
    void TangNgay();                    //Tăng ngày lên một ngày
    void GiamNgay();                    // Giảm ngày xuống một ngày
    void TangTuan();                    //Tăng ngày lên một tuần
    void GiamTuan();                    // Giảm ngày xuống một tuần
};
class Gio {
private:
    int h, m, s;                        //giờ, phút, giây
public:
    Gio(int hh = 0, int mm = 0, int ss = 0);
    void Nhap();
    int HopLe();                //Kiểm tra giờ hợp lệ
    void Xuat24();              //xuất giờ theo 24 tiếng
    void Xuat12();              //xuất giờ theo 12 tiếng
    void TangGiay(int n);               //Tăng giờ lên n giây
    Giơ Cong(const Gio &u);             //Cộng hai giờ
    friend class NgayGio;               //Lớp NgayGio là bạn của lớp Giơ
};
class NgayGio {
private:
    Gio A;              //A là đối tượng thuộc lớp Gio
```

```
    Ngay B;          //B là đối tượng thuộc lớp Ngay
public:
    NgayGio(int hh = 0, int mm = 0, int ss = 0, int dd = 1, int mm_ = 1,     int yy = 1);
    NgayGio(Gio AA, Ngay BB);
    void Nhap();
    void Xuat();
    void TangGiay(int n);    //Tăng thời gian lên n giây
};
```

Hướng dẫn:

```
void NgayGio::TangGiay(int n) {
    A.TangGiay(n);          //Gọi hàm Gio::TangGiay(int)
    if(A.h > 23) {
        A.h %= 24;          //Truy nhập thành phần private của lớp Gio
        B.TangNgay();       //Gọi hàm Ngay::TangNgay()
    }
}
```

6. Xây dựng các lớp sau:

```
class MH {                          //Lớp môn học
private:
    char tenmh[21];         //Tên môn học
    int st;                         //Số tiết của môn học
public:
    MH();
    void Nhap();            //Nhập môn học
    void Xuat();            //Xuất môn học
    int LayST();            //Lấy số tiết
};
class GV {                          //Lớp Giáo Viên
private:
    char tengv[31];         //Họ tên
    int ns;                         //Năm sinh
    int sm;                         //Số môn học giáo viên có thể dạy
    MH *p;          //Con trỏ chỉ tới vùng nhớ động chứa các môn học
public:
    GV();
```

```
    GV(const GV &u);
    ~GV();
    void Nhap();            //Nhập giáo viên
    void Xuat();            //Xuất giáo viên
    void SapXep();          //Sắp xếp các môn học giảm dần theo số tiết
};
```

Hướng dẫn:

```
MH::MH() {
    tenmh[0] = 0;
    st = 0;
}
void MH::Nhap() {
    cout << "Nhap ten mon hoc:";
    cin.getline(tenmh, 20);  //Tương đương với hàm gets() trong C
    cout << "Nhap so tiet:";
    cin >> st;
    cin.ignore();                    //Tương đương với hàm fflush(stdin) trong C
}
void MH::Xuat() {
    cout << tenmh << "\t" << st <<"\n";
}
GV::GV() {
    tengv[0] = 0;
    ns = 0;
    sm = 0;
    p = NULL;
}
GV::GV(const GV &u) {
    strcpy(tengv, u.tengv);
    ns = u.ns;
    sm = u.sm;
    p = new MH[sm];
    for(int i = 0; i < sm; i++)
            p[i] = u.p[i];
}
```

```
void GV::Nhap() {
    cout << "Ten giao vien:"; cin.getline(tengv, 30);
    cout << "Nam sinh:"; cin >> ns;
    cout << "So mon:"; cin >> sm;
    cin.ignore();
    p = new MH[sm];
    for(int i = 0; i < sm; i++) {
            cout << "Mon hoc thu " << i << ":";
            p[i].Nhap();      //gọi MH::Nhap()
    }
}
void GV::Xuat() {
    cout << tengv << "\t" << ns <<"\t" << sm <<"\n";
    for(int i = 0; i < sm; i++)
            p[i].Xuat();              //gọi MH::Xuat()
}
void GV::SapXep() {
    for(int i = 0; i < sm – 1; i++)
        for(int j = i + 1; j < sm; j++)
            if(p[i].layST() < p[j].LayST()) {
                MH tam = p[i];
                p[i] = p[j];
                p[j] = tam;
            }
}
```

7. Xây dựng các lớp sau:

```
class PS { //Lớp phân số
private:
    int ts, ms;                         //tử số và mẫu số
public:
    PS();
    void Nhap();            //Nhập phân số
    void Xuat();            //Xuất phân số
    int USCLN(int a, int b);  /*Tìm ước số chung lớn nhất của hai số
                    nguyên*/
```

```
    void RutGon();          //Rút gọn phân số
    int SoSanh(const PS &u);        //So sánh hai phân số
};
class DSPS { //Lớp danh sách phân số
private:
    int n;          //Số phân số
    PS *p;          //Con trỏ tới vùng nhớ chứa danh sách phân số
public:
    DSPS();
    ~DSPS();
    DSPS(DSPS &u);
    DSPS& operator=(DSPS &u);
    void Nhap();    //Nhập danh sách phân số
    void Xuat();            //Xuất danh sách phân số
    void SapXep(); //Sắp xếp danh sách phân số theo thứ tự tăng dần
};
```

Hướng dẫn

```
int PS::SoSanh(const PS &u) {
    if(ts * u.ms < ms * u.ts)
            return -1;
    else if(ts * u.ms == ms * u.ts)
            return 0;
    else
            return 1;
}
DSPS::DSPS(const DSPS &u) {
    ts = u.ts;
    ms = u.ms;
    n = u.n;
    p = new PS[n];
    for(int i = 0; i < n; i++)
            p[i] = u.p[i];
}
DSPS& DSPS::operator=(const DSPS &u) {
    cout <<"Goi ham DSPS::operator=(const DSPS &) \n";
    if(p != NULL)
```

```
        delete[]p;
    n = u.n;
    p = new PS[n];
    for(i = 0; i < n; i++)
        p[i] = u.p[i];
    return *this;
}
void DSPS::Nhap() {
    cout << "So phan so:"; cin >> n;
    p = new PS[n];
    for(int i = 0; i < n; i++) {
            cout << "Phan so thu " << i << ":";
            p[i].Nhap(); //gọi PS::Nhap()
    }
}
void DSPS::Xuat() {
    for(int i = 0; i < n; i++)
            p[i].Xuat();            //gọi PS::Xuat()
}
void DSPS::SapXep() {
    for(int i = 0; i < n – 1; i++)
        for(int j = i + 1; j < n; j++)
            if(p[i].SoSanh(p[j]) == 1) {
                PS tam = p[i];
                p[i] = p[j];
                p[j] = tam;
            }
}
```

8. Xây dựng một lớp sau:

```
class HDBH {            //Lớp hoá đơn bán hàng
private:
    char tenmh[21];     //Tên mặt hàng
    float tb;                   //Tiền bán
    static int tshd;    //Tổng số hóa đơn
    static float tstb;  //Tổng số tiền bán
```

```
public:
    HDBH(char *tenmh1 = NULL, float tb1 = 0.0);
    ~HDBH();
    void SuaTB(float tb1);  //Sửa tiền bán cũ thành tiền bán mới tb1
    static void Xuat();
};
```

Hướng dẫn:

```
int HDBH::tshd = 0;
float HDBH::tstb = 0;
HDBH::HDBH(char *tenmh1, float tb1) {
    cout << "HDBH::HDBH(char *, float)\n";
    strcpy(tenmh, tenmh1);
    tb = tb1;
    tshd++;
    tstb += tb;
}
HDBH ::~HDBH() {
    cout << "HDBH::~HDBH()\n";
    tshd--;
    tstb -= tb;
}
void HDBH::SuaTB(float tb1) {
    tstb -= tb;
    tstb += tb1;
}
void HDBH::Xuat() {
    cout << "Tong so tien ban:" <<tstb<<"\n";
    cout << "Tong so hoa don:" <<tshd<<"\n";
}
```

❒ Bài tập luyện tập

9. Xây dựng một lớp **Gio** để mô tả các đối tượng thời gian (giờ, phút, giây) như sau:

```
class Gio {
private:
    int h, m, s;                    //giờ, phút, giây
```

```
public:
    Gio(int hh = 0, int mm = 0, int ss = 0);
    void Nhap();
    int HopLe();              //Kiểm tra giờ hợp lệ
    void Xuat24();            //xuất giờ theo 24 tiếng
    void Xuat12();            //xuất giờ theo 12 tiếng
    void TangGiay(int n);     //Tăng giờ lên n giây
    Giơ Cong(const Gio &u);          //Cộng hai giờ
};
```

10. Xây dựng các lớp sau:

```
struct Ngay {
    int d, m, y;               //Ngày, tháng, năm
};
struct Gio {
    int h, m, s;               //Giờ, phút, giây
};
class CB {                     //Lớp chuyến bay
private:
    Ngay ngay_bay;             //Ngày bay
    Gio gio_bay;               //Giờ bay
    char Noi_di[30];           //Nơi đi
    char Noi_den[30];          //Nơi đến
public:
    ChuyenBay ();
    void Nhap();
    void Xuat();
    int SoSanh(const ChuyenBay &u); /*So sánh ngày bay của hai chuyến
                                      bay*/
};
class DSCB {          //Lớp danh sách chuyến bay
private:
    int n;            //Số chuyến bay
    CB *p;            //Con trỏ tới vùng nhớ động chứa các chuyến bay
public:
    DSCB ();
```

```
    DSCB (int nn);
    DSCB (const DSCB &u);
    DSCB & operator=(const DSCB &u);
    ~ DSCB ();
        void Nhap();
    void Xuat();
    void SapXep(); //Sắp xếp danh sách chuyến bay tăng theo ngày bay
};
```

11. Xây dựng các lớp sau:

```
struct Ngay {
    int d, m, y;            //Ngày, tháng, năm
};
class SV {                  //Lớp sinh viên
public:
    char ma[20];    //Mã số
    char ten[30];   //Họ tên
    Ngay ns;                //Ngày sinh
    float diem;             //Điểm
private:
    SV();
    void Nhap();    //Nhập một sinh viên
    void Xuat();            //Xuất một sinh viên
    int SoSanh(const SV &u); //So sánh ngày sinh của hai sinh viên
};
class DSSV {                //Lớp danh sách sinh viên
public:
    int n;                  //Số sinh viên
    SV *p;                  //Con trỏ tới vùng nhớ chứa danh sách sinh viên
private:
    DSSV();
    DSSV(int nn);
    DSSV(const DSSV &u);
    DSSV & operator=(const DSSV &u);
    ~DSSV();
        void Nhap();    //Nhập danh sách sinh viên
```

```
    void Xuat();                //Xuất danh sách sinh viên
    void SapXep(); /*Sắp xếp danh sách sinh viên tăng dần theo ngày
    sinh*/
};
```

CHƯƠNG 4: TOÁN TỬ TRÊN LỚP

❒ Bài tập có hướng dẫn

1. Xây dựng lớp sau:

```
class PS {              //lớp phân số
private:
    int ts, ms;
public:
    PS(int ts1 = 0, int ms1 = 1);
    int USCLN(int x, int y);
    void RutGon();
    void Nhap();
    void Xuat();
    //Toán tử số học
    friend PS operator+(const PS &u, const PS &v);
    friend PS operator-(const PS &u, const PS &v);
    friend PS operator*(const PS &u, const PS &v);
    friend PS operator/(const PS &u, const PS &v);
    //Toán tử quan hệ
    friend int operator>(const PS &u, const PS &v);
    friend int operator>=(const PS &u, const PS &v);
    friend int operator<(const PS &u, const PS &v);
    friend int operator<=(const PS &u, const PS &v);
    friend int operator==(const PS &u, const PS &v);
    friend int operator!=(const PS &u, const PS &v);
    //Toán tử số học mở rộng
    PS operator+=(const PS &u);
    PS operator-=(const PS &u);
    PS operator*=(const PS &u);
    PS operator/=(const PS &u);
    //Toán tử nhập xuất
    friend istream& operator>>(istream &is, PS &u);
    friend ostream& operator<<(ostream &os, const PS &u);
    //Toán tử tăng giảm
    PS operator++();
```

```
    PS operator++(int);
};
```

Hướng dẫn:

```
PS::PS(int ts1, int ms1) {
    ts = ts1 ; ms = ms1 ;
    RutGon();
    if(ms < 0) {
            ts *= -1; ms *= -1;
    }
}
void PS::RutGon() {
    int uscln = USCLN(ts, ms);
    ts /= uscln;
    ms /= uscln;
}
PS operator+( const PS &u, const PS &v) {
    PS ret(u.ts * v.ms + u.ms * v.ts,  u.ms * v.ms);
    return ret;
}
int operator>(const PS &u, const PS &v) {
    if(u.ts * v.ms > u.ms * v.ts)
            return 1;
    else return 0;
}
PS PS::operator+=(const PS &u) {
    ts = ts * u.ms + ms * u.ts;
    ms = ms * u.ms;
    return *this;
}
PS PS::operator++() {
    cout << "PS::operator++()\n";
    ts = ts + ms;
    PS ret(ts, ms);
    return ret;
}
```

```
PS PS::operator++(int) {
    cout << "PS::operator++(int)\n";
    PS ret(ts, ms);
    ts = ts + ms;
    return ret;
}
```

2. Xây dựng lớp **Vector** để mô tả các đối tượng Vector trong không gian n chiều như sau:

```
class Vector {
private:
    int n;          //Số chiều
    float *p;       //Con trỏ tới vùng nhớ chứa các tọa độ
public:
    Vector();
    Vector(int nn);
    Vector(const Vector &u);
    ~Vector();
    void Nhap();
    void Xuat();
    int GetN();             //Lấy số chiều
    friend istream& operator>>(istream &is, Vector &u); //Toán tử  nhập
    friend ostream& operator<<(ostream &os, const Vector &u); /*Toán tử  xuất*/
    Vector& operator=(const Vector & u);    //Toán tử gán
    Vector operator+(const Vector & u);     //Cộng hai Vector
    Vector operator-(const Vector &u);              //Trừ hai Vector
    Vector operator*(float x);          //Nhân vô hướng Vector với số thực
    float operator*(const Vector &u);           //Nhân vô hướng hai Vector
};
```

Hướng dẫn:

```
istream& operator>>(istream &is, Vector &u) {
    cout << "Goi ham operator>>(istream &, Vector &)\n";
    if(u.p != NULL) {
            cout << "Nhap so chieu:";
            is >> u.n;
            u.p = new float[u.n];
```

```
    }
    for(i = 0; i < u.n; i++) {
            cout << "Toa do thu " << i <<":";
            is >> u.p[i];
    }
    return is;
}
ostream& operator<<(ostream &os, const Vector &u) {
    cout << "Goi ham operator<<(ostream &, const Vector&)\n";
    for(i = 0; i < u.n; i++)
        os << u.p[i] <<"t";
    os << "\n";
    return os;
}
```

3. Xây dựng lớp sau:

```
class MT {          //Lớp ma trận
private:
    int sd, sc;     //số dòng và số cột
    float **p;      //con trỏ tới vùng nhớ chứa các phần tử
public:
    MT ();
    MT(int sd1, int sc1);
    MT (const MT &u);
    ~ MT ();
    void Nhap();
    void Xuat();
    float TongDong(int k);  //Tổng các phần tử ở dòng thứ k
    int GetM();             //lấy số dòng
    int GetN();             //lấy số cột
    friend istream& operator>>(istream &is, MT &u); /*Toán tử nhập*/
    friend ostream& operator<<(ostream &os, const MT &u); /*Toán
                                                          tử xuất*/
    MT& operator=(const MT  &u);  //Toán tử gán
    MT operator+(const MT  &u);   // Toán tử cộng hai ma tran
    MT operator-(const MT  &u);   // Toán tử trừ hai ma tran
```

```
    MT operator*(const MT  &u);     // Toán tử nhân hai ma tran
};
```

Hướng dẫn:

```
MT& MT::operator=(const MT &u) {
    cout << "MT::operator=(MT &)\n";
    int i, j;
    sd = u.sd; sc = u.sc;
    if(p != NULL) {
            //Xóa vùng nhớ động đã có trong đối tượng vế trái
            for(i = 0; i < sd; i++)
                    delete[]p[i];
            delete[]p
    }
    //Cấp phát vùng nhớ động mới
    p = new float*[sd];
    for(i = 0; i < sd; i++)
    p[i] = new float[sc];
    //Gán
    for(i = 0; i < sd; i++)
    for(j = 0; j < sc; j++)
        p[i][j] = u.p[i][j];
    return *this;
}
istream& operator>>(istream &is, MT &u) {
    int i, j;
    cout << "Goi ham operator>>(istream &, MT &)\n";
    if(u.p == NULL) {
            cout << "Nhap so dong:"; is >> u.sd;
            cout << "Nhap so cot:"; is >> u.sc;
            u.p = new float*[u.sd];
            for(i = 0; i < u.sd; i++)
                    u.p[i] = new float[u.sc];
    }
    for(int i = 0; i < u.sd ; i++)
            for(int j = 0; j < u.sc ; j++) {
```

```
                cout << "phan tu [" << i << "][" << j <<"]=";
                is >> u.p[i][j];
        }
    return is;
}
ostream& operator<<(ostream &os, const MT &u) {
    int i, j;
    cout << "Goi ham operator<<(ostream &, const MT &)\n";
    for(int i = 0; i < u.sd ; i++)
        for(int j = 0; j < u.sc ; j++) {
            os << u.p[i][j] <<"\t"
        }
    return os;
}
```

4. Xây dựng các lớp sau:

```
class MH {                  //Lớp môn học
private:
    char tenmh[21];         //Tên môn học
    int st;                 //Số tiết
public:
    MH();
    void Nhap();
    void Xuat();
    int LayST();
    friend istream& operator>>(istream &is, MH &u); //Toán tử nhập
    friend ostream& operator<<(ostream &os, const MH &u); /*Toán
                                                            tử xuất*/
};
class GV {                  //Lớp giáo viên
private:
    char tengv[31];         //Họ tên
    int ns;                 //Năm sinh
    int sm;                 //Số môn học có thể dạy
    MH *p;                  //Con trỏ tới vùng nhớ động chứa các môn học
public:
```

```
    GV();
    ~GV();
    GV(const GV &u);
    GV& operator=(const GV &u);
    LaySM();
    void Nhap();
    void Xuat();
    friend istream& operator>>(istream &is, GV &u);
    friend ostream& operator<<(ostream &os, const GV u);
    void SapXep();  //Sắp xếp danh sách môn học giảm dần theo số tiết
};
class BM {                  //Lớp bộ môn
private:
    char tenbm[21];         //Tên bộ môn
    int sgv;                        //Số giáo viên
    GV *p;                  //Con trỏ chỉ tới vùng nhớ động chứa các giáo viên
public:
    BM();
    ~BM();
    BM(const BM &u);
    BM &operator=(const BM &u);
    void Nhap();
    void Xuat();
    friend istream& operator>>(istream &is, BM &u);
    friend ostream& operator<<(ostream &os, const BM &u);
    void SapXep();  /*Sắp xếp danh sách giáo viên giảm dần theo số môn mà mỗi giáo viên có thể dạy*/
};
```

Hướng dẫn:

```
void BM::Nhap() {
    cout << "Ten bo mon:";
    cin.getline(tenbm, 20);
    cout << "So giao vien:";
    cin >> sgv;
    cin.ignore();
    p = new GV[sgv];
```

```
    for(int i = 0; i < sgv; i++) {
    cout << "** Giao vien thu " << i <<": **\n";
        p[i].Nhap();          //Gọi GV::Nhap()
    }
}
BM::BM(const BM &u) {
    cout << "BM::BM(BM &)\n";
    strcpy(tenbm, u.tenbm);
    sgv = u.sgv;
    p = new GV[sgv];
    for(int i = 0; i < sgv; i++)
            p[i] = u.p[i];    //Gọi GV::operator=()
}
BM & BM::operator=(const BM &u) {
    cout << "BM::operator=(BM &)\n";
    if(p != NULL)
            delete[]p;
    strcpy(tenbm, u.tenbm);
    sgv = u.sgv;
    p = new GV[sgv];
    for(int i = 0; i < sgv; i++)
            p[i] = u.p[i]; //Gọi hàm GV::operator=()
    return *this;
}
void BM::SapXep() {
    for(int i = 0; i < sgv – 1; i++)
    for(int j = i + 1; j < sgv; j++)
    if(p[i].laySM() < p[j].LaySM()) {
            GV tam = p[i];          //Gọi GV::GV(const GV &)
            p[i] = p[j];                    //Gọi GV::operator=(const GV &)
            p[j] = tam;
    }
}
```

❒ Bài tập luyện tập

5. Xây dựng lớp sau:

```
class SP {              //Lớp số phức
private:
    float re, im;
public:
    SP (int r = 0, int i = 0);
    void Nhap();
    void Xuat();
    float Module();             //Tính độ dài số phức
    //Toán tử số học
    friend SP operator+(const SP &u, const SP &v);
    friend SP operator-(const SP &u, const SP &v);
    friend SP operator*(const SP &u, const SP &v);
    friend SP operator/(const SP &u, const SP &v);
    //Toán tử quan hệ
    friend int operator>(SP &u, SP &v);
    friend int operator>=(SP &u, SP &v);
    friend int operator<(SP &u, SP &v);
    friend int operator<=(SP &u, SP &v);
    friend int operator==(SP &u, SP &v);
    friend int operator!=(SP &u,  SP &v);
    //Toán tử số học mở rộng
    SP operator+=(const SP &u);
    SP operator-=( const SP &u);
    SP operator*=( const SP &u);
    SP operator/=( const SP &u);
    //Toán tử nhập xuất
    friend istream& operator>>(istream &is, SP &u);
    friend ostream& operator<<(ostream &os, const SP &u);
};
```

6. Xây dựng lớp sau:

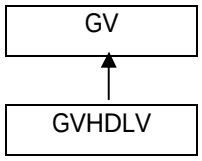
```
class DT {              //Lớp đa thức
private:
    int n;              //Bậc đa thức
    float *p;           //con trỏ chỉ tới vùng nhớ động chứa các hệ số
```

```
public:
    DT();
    DT(int nn);
    DT (const DT &u);
    ~ DT ();
    void Nhap();
    void Xuat();
    friend istream& operator>>(istream &is, DT &u); /*Toán tử nhập*/
    friend ostream& operator<<(ostream &os, const DT &u); /*Toán
                                                              tử xuất*/
    float operator^(float x);              //Toán tử tính giá trị đa thức tại x
    DT& operator=(const DT &u);    //Toán tử gán
    DT operator+(const DT &u);     //Toán tử cộng hai đa thức
    DT operator-(const DT &u);             //Toán tử trừ hai đa thức
    DT operator*(const DT &u);     //Toán tử nhân hai đa thức
    void operator-();                          //Toán tử đảo dấu đa thức
};
```

CHƯƠNG 5: KỸ THUẬT KẾ THỪA

❑ Bài tập có hướng dẫn

1. Xây dựng các lớp có quan hệ thừa kế như sau:




```
class MH {                    //Lớp môn học
private:
    char tenmh[21];          //Tên môn học
    int st;                          //Số tiết
public:
    MH ();
    void Nhap();
    void Xuat();
    int LayST();
};
class GV {              //Lớp giáo viên
private:
    char tengv[31];          //Họ tên
    int ns;                          //Năm sinh
    int sm;                          //Số môn học có thể dạy
    MH *p;                   //Con trỏ tới vùng nhớ động chứa các môn học
public:
    GV();
    ~GV();
    GV(const GV &u);
    GV& operator=(const GV &u);
    LaySM();
    void Nhap();
    void Xuat();
    void SapXep();  //Sắp xếp danh sách môn học giảm dần theo số tiết
};
class LV {    //Lớp luận văn
```

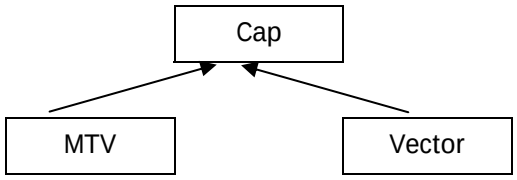
```
private:
    char tenlv[31];          //Tên luận văn
    char tensv[31];          //Tên sinh viên
    int nbv;                 //Năm bảo vệ
public:
    LV();
    void Nhap();
    void Xuat();
};
class GVHDLV:public GV {    //Lớp giáo viên hướng dẫn luận văn
private:
    int slv;            //số luận văn
    LV *p;              //Con trỏ tới vùng nhớ động chứa các luận văn
public:
    GVHDLV();
    ~GVHDLV();
    GVHDLV(const GVHDLV &u);
    void Nhap();
    void Xuat();
    GVHDLV& operator=(const GVHDLV &u);  //Toán tử gán
};
```

Hướng dẫn:

```
void GVHDLV::Nhap() {
    GV::Nhap();
    cout << "So luan van";
    cin >> slv;
    cin.ignore();
    p = new LV[slv];
    for(int i = 0; i < slv; i++) {
        cout << "Luan van thu << i <<"\n";
        p[i].Nhap();          //Goi LV::Nhap()
    }
}
void GVHDLV::Xuat() {
    GV::Xuat();
```

```
    for(int i = 0; i < slv; i++)
        p[i].Xuat();            //Gọi hàm LV::Xuat()
}
GVHDLV ::GVHDLV( const GVHDLV &u) : GV(u) {
    cout << "GVHDLV ::GVHDLV( GVHDLV &u)\n";
    slv = u.slv;
    p = new LV[slv];
    for(int i = 0; i < slv; i++)
            p[i] = u.p[i];
}
GVHDLV & GVHDLV::operator=(const GVHDLV &u) {
    cout << "goi ham GVHDLV::operator=()\n";
    if(p!= NULL) {
        delete[]p;
        p = NULL;
    }
    //Gán các thành phần mà lớp GVHDLV thừa kế
    *((GV *)this) = (GV&)u;          //Gọi GV::operator=(GV&)
    //Gán các thành phần bổ sung của lớp GVHDLV
    slv = u.slv;
    p = new LV[slv];
    for(int i = 0; i < slv; i++)
        p[i] = u.p[i];
    return *this;
}
```

2. Xây dựng các lớp theo phân cấp thừa kế như sau:




```
class Cap {
private:
    static int n;       /*Thành phần dữ liệu dùng chung cho các lớp dẫn xuất
    từ lớp Cap */
public:
```

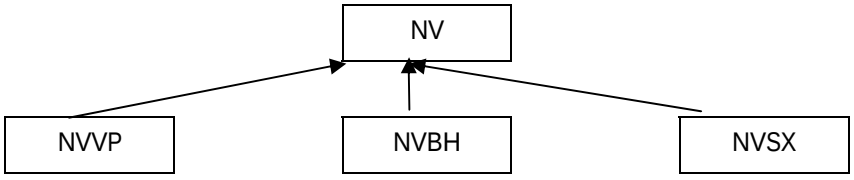
```
    void Nhap();
    int LayN();
};
int Cap::n = 0;                 //Khởi tạo thành phần dữ liệu static

class Vector;                   //Khai báo trước lớp Vector
class MTV:public Cap {          //Lớp ma trận vuông
private:
    float arr[SIZE][SIZE];
public:
    void Nhap();
    void Xuat();
    Vector operator*(const Vector &u); //Toán tử nhân MTV với Vector
};
class Vector:public Cap {
private:
    float arr[SIZE];
public:
    void Nhap();
    void Xuat();
    friend Vector MTV::operator*(const Vector &u); /*khai báo bạn
                                                        lớp Vector*/
};
```

Hướng dẫn:

```
void MTV::Nhap() {
    int n;              //Lưu cấp ma trận vuông
    n = LayN();
    if(n == 0) {
            Cap::Nhap();
            n = LayN();
    }
    for(int i = 0; i < n; i++)
            for(int i = 0; i < n; i++) {
                    cout <<"pt[" << i <<"][" << j <<"]=";
                    cin >>arr[i][j];
```

```
            }
}
void Vector::Nhap() {
    int n;          //Lưu số chiều của vector
    n = LayN();
    if(n == 0) {
            Cap::Nhap();
            n = LayN();
    }
    for(int i = 0; i < n; i++) {
        cout <<"pt[" << i <<"]=";
        cin >>arr[i];
    }
}
void MTV::operator*(const Vector &u) {
    Vector ret;
    int n = LayN();         //Cấp ma trận vuông cũng là số chiều vector
    for(int i = 0; i < n; i++) {
            ret.arr[i] = 0;
            for(int j = 0; j < n; j++)
                    ret.arr[i] += arr[i][j] * u.arr[j];
    }
    return ret;
}
```

3. Xây dựng các lớp theo phân cấp thừa kế như sau:




```
class NV {          //Lớp nhân viên
private:
    char msnv[10];  //Mã số nhân viên
    char ht[30];    //Họ tên
    float ml;               //Mức lương
```

```
public:
    NV();
    virtual void Nhap();
    virtual void Xuat();
    virtual float TienThuong() = 0;
};
class NVVP  : public NV {      //Lớp nhân viên văn phòng
private:
    float tgct;                 //Thời gian công tác tính theo tháng
public:
    NVVP ();
    void Nhap();
    void Xuat();
    float TienThuong();
};
class NVBH : public NV {      //Lớp nhân viên bán hàng
private:
    float hst;                  //Hệ số thưởng
public:
    NVBH ();
    void Nhap();
    void Xuat();
    float TienThuong() ;
};
class NVSX  : public NV {     //Lớp nhân viên sản xuất
private:
    float tssp;         //Tổng sàn phẩm tính từ đầu năm
public:
    NVSX ();
    void Nhap();
    void Xuat();
    float TienThuong();
};
```

Lưu ý: Tiền thưởng được tính theo quy tắc sau:

- Nhân viên văn phòng

  nếu tgct < 6 thì tiền thưởng = 100000

ngược lại tiền thưởng = 200000 + ml * 10% * tgct/6

- Nhân viên bán hàng

  tiền thưởng = 150000 * hst

- Nhân viên sản xuất

  tiền thưởng = 20000 * tssp

```
//Lớp Nhân viên
NV::NV() {
    msnv[0] = 0;
    ht[0] = 0;
    ml = 0;
}
void NV::Nhap() {
    cout <<"Nhap ma so:";
    cin.ignore();
    cin.getline(msnv,20);
    cout <<"Nhap ho ten:";
    cin.getline(ht,30);
    cout <<"nhap muc luong:";
    cin >>ml;
    cin.ignore();
}
void NV::Xuat() {
    cout <<msnv<<"\n"<<ht<<"\n"<<ml<<"\t"<<"\n";
}
//Lớp Nhân viên văn phòng
NVVP::NVVP() {
    tgct = 0;
}
void NVVP::Nhap() {
    NV::Nhap();
    cout <<"Nhap thoi gian cong tac:";
    cin >> tgct;
    cin.ignore();
}
void NVVP::Xuat() {
```

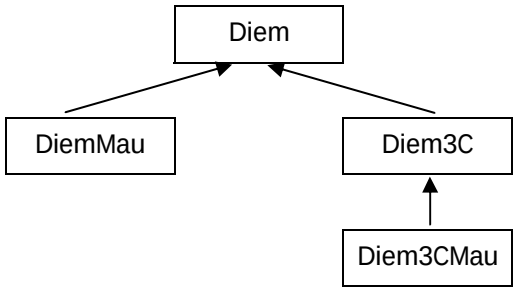
```
    NV::Xuat();
    cout <<"thoi gian cong tac:"<<tgct<<"\n";
}
double NVVP::TienThuong() {
    if(tgct<6)
        return 100000;
    else return 200000 + ml*(10/100)*tgct/6;
}
double NVVP::TienLuong() {
    return ml + TienThuong();
}
//Lớp nhân viên bán hang
NVBH::NVBH() {
    hst = 0;
}
void NVBH::Nhap() {
    NV::Nhap();
    cout <<"nhap he so thuong:";
    cin >>hst;
    cin.ignore();
}
void NVBH::Xuat() {
    NV::Xuat();
    cout <<"he so thuong:"<<hst<<"\n";
}
double NVBH::TienThuong() {
    return 150000 * hst;
}
double NVBH::TienLuong() {
    return ml + TienThuong();
}
//Lớp nhân viên sản xuất
NVSX::NVSX() {
    tssp = 0;
}
```

```
void NVSX::Nhap() {
	NV::Nhap();
	cout <<"nhap tong so san pham:";
	cin >> tssp;
	cin.ignore();
}
void NVSX::Xuat() {
	NV::Xuat();
	cout <<"tong so san pham:"<<tssp<<"\n";
}
double NVSX::TienThuong() {
	return 20000*tssp;
}
double NVSX::TienLuong() {
	return ml + TienThuong();
}
//Hàm chính
#define MAX_NV	20
void main() {
	NV *pnv[MAX_NV];		//Mảng các con trỏ kiểu lớp trừu tượng
	int n;			//Lưu số nhân viên
	int loai;		//0: nvvp, 1: nvbh, 2:nvsx
	int i;
	do {
		<Nhap n>
	} while (n < 1 || n > MAX_NV);
	for(i = 0; i < n; i++) {
		cout << “Nhan vien thu “ << i <<”\n:”;
		do {
			<Nhap loai>
		} while(loai < 0 || loai > 2);
		if(loai == 0)
			pnv[i] = new NVVP;
		else if(loai == 1)
			pnv[i] = new NVBH;
```

```
            else pnv[i] = new NVSX;
            pnv[i] -> Nhap();          //Thể hiện sự kết nối động
    }
    for(i = 0; i < n; i++) {
            cout << " ** Nhan vien thu " << i <<" **\n";
            pnv[i]->Xuat();            //Thể hiện sự kết nối động
            cout <<"Tien thuong: " << pnv[i]->TienThuong() << "\n"; /*Thể
                                                                hiện sự kết nối động*/
    }
}
```

❑ Bài tập luyện tập

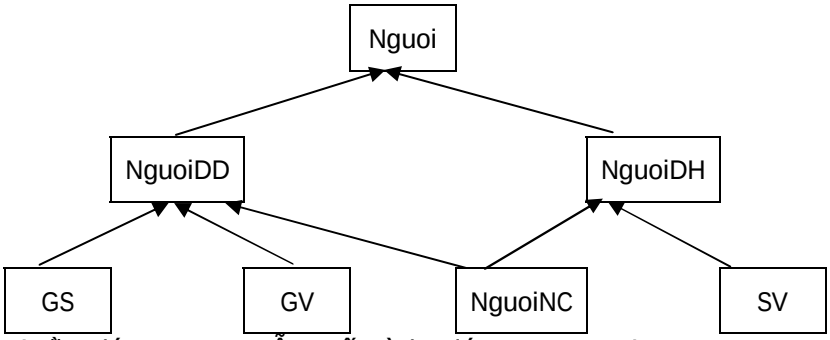
4. Xây dựng các lớp theo phân cấp thừa kế như sau:




```
class Diem3C : public Diem {
private:
    int z;              //Toạ độ chiều thứ 3
public:
    Diem3C();
    Diem3C(int xx, int yy, int zz);
    Diem3C(const &Diem3C);
    void Xuat();
};
class Diem3CMau : public Diem3C {
private:
    int m;              //Màu
public:
    Diem3CMau ();
    Diem3CMau (int xx, int yy, int zz, int mm);
```

```
    Diem3CMau (const & Diem3CMau);
    void Xuat();
};
```

5. Xây dựng các lớp theo phân cấp thừa kế như sau:



Lưu ý rằng lớp nguoiNC dẫn xuất từ hai lớp NguoiDD và NguoiDH theo kiểu đa thừa kế (Người nghiên cứu vừa thuộc dạng người đi dạy vừa thuộc dạng người đi học).

```
class Nguoi {
private:
    char ten[30];
    char ma[10];
public:
    Nguoi();
    void Nhap();
    void Xuat();
    char *LayTen();
    char *LayMa();
    virtual int KhenThuong() = 0 ;   //Hàm ảo thuần tuý
};
class NguoiDD : virtual public Nguoi{          //Lớp người đi dạy
private:
    char hocvi[10];          //Học vị
    int sobaibao;            //Số bài báo đã công bố
public:
    NguoiDD();
    Nhap();
    char* LayHV();
    int LaySoBaiBao();
};
```

```
class NguoiDH : virtual public Nguoi {          //Lớp người đi học
private:
    float Diem;
public:
    NguoiDH();
    void Nhap();
    float LayDiem();
};
class GS:public NguoiDD {              //Lớp giáo sư
private:
    int soncs;                  //Số nghiên cứu sinh đã hướng dẫn
    int sosach;                 //Số sách đã xuất bản
public:
    GS();
    void Nhap();
    void Xuat();
    virtual int KhenThuong();
};
class GV : public NguoiDD {            //Lớp giảng viên
private:
    int sosv;                   //Số sinh viên đã hướng dẫn
public:
    GV();
    void Nhap();
    void Xuat();
    virtual int KhenThuong();
};
class NguoiNC: public NguoiDD, public NguoiDH { //Lớp người nghiên cứu
private:
    int loai;           //0: Cao học, 1:nghiên cứu sinh
public:
    NguoiNC();
    void Nhap();
    void Xuat();
    virtual int KhenThuong();
};
```

```
class SV : public NguoiDH() {
private:
    int  nc;            //0: không nghiên cứu, 1: có nghiên cứu
public:
    SV();
    void Nhap();
    void Xuat();
    virtual int KhenThuong();
};
```

Lưu ý: Khen thưởng được tính theo quy tắc sau:

- Giáo sư chỉ được khen thưởng khi có sobaibao > 2 và soncs > 2 và sosach > 2
- Giảng viên chỉ được khen thưởng khi có sobaibao > 0 và sosv > 10
- Người nghiên cứu

  Với cao học: chỉ được khen thưởng khi có Diem > 8

  Với nghiên cứu sinh: chỉ được khen thưởng khi có Diem > 8 và sobaibao > 0
- Sinh viên chỉ được khen thưởng khi nc = 1 và diem > 9

CHƯƠNG 6: KHUÔN HÌNH

❑ Bài tập có hướng dẫn

1. Xây dựng các khuôn hình hàm sau:
   - Khuôn hình hàm **HoanVi** để hoán vị hai số truyền vào.
   - Khuôn hình hàm **SapXep** dùng để sắp xếp một mảng theo thứ tự tăng trong đó có sử dụng khuôn hình hàm **HoanVi.**
   - Khuôn hình hàm **Nhap** để nhập một mảng từ bàn phím.
   - Khuôn hình hàm **Xuat** để xuất một mảng ra màn hình.

   Sử dụng các khuôn hình đã xây dựng để viết một chương trình nhập, sắp xếp và xuất 4 mảng: một mảng nguyên, một mảng thực, một mảng ký tự và một mảng chuỗi.

Hướng dẫn:

```
#define SIZE        10
#define MAX_CHAR 20
template <class T>
void HoanVi(T &u, T &v) {
    T tam = u ;
    u = v ;
    v = tam ;
}
template <class T>
void SapXep(T arr[], int n) {
    for(int i = 0 ; i < n ; i++)
            for(int j = 0 ; j < n ; j++)
                    if(arr[i] > arr[j]) {
                            T tam = arr[i];
                            arr[i] = arr[j];
                            arr[j] = tam;
                    }
}
template <class T>
void Nhap(T arr[], int &n) {
    cout << " Nhap so phan tu :"
    cin >> n;
```

```
    for(int i = 0; i < n; i++) {
            cout << "Phan tu thu " << i <<":";
            cin >> arr[i];
    }
}
template <class T>
void Xuat(T arr[], int n) {
    for(int i = 0; i < n; i++)
            cout << arr[i] <<"\t";
    cout << "\n";
}
/*Cụ thể hóa hàm thể hiện void SapXep(char *a[], int n) trong khuôn hình SapXep*/
void SapXep(char arr[][MAX_CHAR], int n) {
    for(int i = 0 ; i < n ; i++)
            for(int j = 0 ; j < n ; j++)
                    if(strcmp(arr[i], arr[j]) > 0) {
                            T tam[MAX_CHAR];
                            strcpy(tam, arr[i]);
                            strcpy(arr[i], arr[j]);
                            strcpy(arr[j], tam);
                    }
}
void main() {
    int a[SIZE]; int n;                 //Mảng nguyên
    float b[SIZE]; int m ;    //Mảng thực
    char c[SIZE]; int k ;               //Mảng ký tự
    char d[SIZE][MAX_CHAR]; int t ;         //Mảng chuỗi
    Nhap(a, n) ; Xuat(a, n) ; SapXep(a, n) ; Xuat(a, n) ;
    Nhap(b, m) ; Xuat(b, m) ; SapXep(b, m) ; Xuat(b, m) ;
    Nhap(c, k) ; Xuat(c, k) ; SapXep(c, k) ; Xuat(c, k) ;
    Nhap(d, t) ; Xuat(d, t) ; SapXep(d, t) ; Xuat(d, t) ;
}
```

2. Xây dựng khuôn hình lớp sau:

```
template <class T >
class Vector {
```

```
private:
    int n;                  //số chiều
    T arr[SIZE];            //Mảng chứa các tọa độ Vector
public:
    Vector();
    Vector(int nn);
    Vector(const Vector<T>& u);
    Vector<T>& operator=(const Vector<T>& u);  /*Toán tử gán hai
                                                    Vector*/
    void Nhap();
    void Xuat();
};
```

Hướng dẫn:

```
template <class T > Vector<T>::Vector() {
    n = 0;
}
template <class T >
Vector<T>::Vector(int nn) {
    n = nn;
}
template <class T > Vector<T>::Vector(const Vector<T>& u) {
    n = u.n;
    for(int i = 0; i < n; i++)
            arr[i] = u.arr[i];
}
template <class T >
Vector<T>& Vector<T>::operator=(const Vector<T>& u) {
    n = u.n;
    for(int i = 0; i < n; i++)
            arr[i] = u.arr[i];
    return *this;
}
template <class T >
void Vector<T>::Nhap() {
    if(n == 0) {
```

```
            cout << "Nhap so chieu:";
            cin >> n;
    }
    for(int i = 0; i < n; i++) {
            cout << "toa do thu " << i <<":";
            cin >> arr[i];
    }
}
template <class T > void Vector<T>::Xuat() {
    for(int i = 0; i < n; i++)
            cout << arr[i] << "\t";
    cout << "\n";
}
```

❑ Bài tập luyện tập

3. Xây dựng lớp sau:

```
class PS {    //Lớp phân số
private:
        int ts, ms;
public:
        //Toán tử so sánh lớn
        friend int operator>(const PS &u, const PS &v);
        //Toán tử nhập/xuất
        friend istream& operator>>(istream &is, PS &u);
        friend ostream& operator<<(ostream &os, const PS &u);
};
```

Sử dụng các khuôn hình hàm đã xây dựng ở bài 1 để viết một chương trình nhập, sắp xếp và xuất một mảng các phân số.

4. Xây dựng khuôn hình lớp sau:

```
template <class T>
class MTV {          //Lớp ma trận vuông
private:
    int n;                          //Cấp ma trận vuông
    T arr[SIZE][SIZE]        //Mảng chứa các phần tử ma trận vuông
```

```
public:
    MTV ();
    MTV (int nn);
    MTV (const MTV<T>& u);
    MTV<T> & operator=(const MTV<T>& u); /*Toán tử gán hai MTV */
    MTV<T> operator+(const MTV<T>& u); /*Toán tử cộng hai MTV */
    MTV<T> operator*(const MTV<T> & u); /*Toán tử nhân hai MTV */
    void Nhap();
    void Xuat();
};
```