TRUNG TÂM TIN HỌC – ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM

227 Nguyễn Văn Cừ - Quận 5- Tp.Hồ Chí Minh

Tel: 38351056 – Fax 8324466 – Email: ttth@hcmuns.edu.vn

BÀI TẬP

CHƯƠNG TRÌNH CHUYÊN ĐỀ

NGÀNH LẬP TRÌNH

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI C++

Mã tài liệu: DT_NCM_LT_BT_CD_LTHDTC++

Phiên bản 1.0 – Tháng 02/2010

CHƯƠNG 2: CÁC MỞ RỘNG CỦA C++

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 và c:

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

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 tang = 0);

   Khi đối số thứ ba nhận giá trị mặc định bằng không hàm sắp xếp tăng dần, ngược lại hàm sắp xếp giảm dần.

3. Sử dụng các toán tử new và delete thực hiện các thao tác cấp phát và giải phóng bộ nhớ động cho mảng một chiều, sau đó thực hiện một số bài toán cơ bản trên mảng một chiều.

4. Sử dụng các toán tử new và delete thực hiện các thao tác cấp phát và giải phóng bộ nhớ động cho mảng hai chiều, sau đó thực hiện một số bài toán cơ bản trên mảng hai chiều.

5. Khai báo kiểu dữ liệu PS biểu diễn thông tin của một phân số, sau đó định nghĩa các hàm và hàm toán tử 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);
```

❖ **Hướng dẫn**

```
int operator>(const PS &u, const PS &v) {
    if(u.ts * v.ms > u.ms * v.ts)
            return 1;
    else return 0;
}
void operator+=(PS &u, const PS &v) {
    u.ts = u.ts * v.ms + u.ms * v.ts;
    u.ms = u.ms * t.ms;
    RutGon(u);
}
```

6. Khai báo kiểu dữ liệu DT biểu diễn thông tin của một đa thức, sau đó định nghĩa các hàm và hàm toán tử sau:

```
void Nhap(DT &u);
void Xuat(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-(const DT &u);               // Toán tử đảo dấu đa thức
double operator^( const DT &u, double x);  /*Toán tử tính giá trị đa thức
            tại x*/
```

❖ **Hướng dẫn:**

```
struct DT
{
    int n; //Bậc đa thức
    double arr[SIZE]; //Mảng chứa các hệ số của đa thức
};
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;
}
```

CHƯƠNG 3: ĐỐI TƯỢNG VÀ LỚP

1. Xây dựng một lớp *TamGiác* để mô tả các đối tượng tam giác như sau:

```
class TamGiac
{
private:
    double a, b, c;   //Ba cạnh tam giác
public:
    TamGiac(double aa = 0, double bb = 0, double 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
    double ChuVi();           //Tính chu vi tam giác
    double DienTich();        //Tính diện tích tam giác
};
```

❖ **Hướng dẫn:**

```
void TamGiac ::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";
}
```

2. 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
    Giơ Tru(const Gio &u); //Trừ hai giờ
};
```

❖ **Hướng dẫn:**

```
void Gio::TangGiay(int n)
{
    s += n;
    if(s > 59) {
            m += s/60;
            s %= 60;
            if(m > 59) {
                    h += m/60;
                    m %= 60;
            }
    }
}
Gio Gio::Cong(const Gio &u)
{
    Gio ret;
    ret.s = s + u.s;
    if(ret.s > 59)
            ret.s %= 60;
    ret.m = m + u.m + (s + u.s)/60;
    if(ret.m > 59)
            ret.m %= 60;
    ret.h = h + u.h + (m + u.m)/60;
    if(ret.h > 23)
            ret.m %= 24;
```

```
    return ret;
}
```

3. Xây dựng một lớp *Ngay* để mô tả các đối tượng thời gian (ngày, tháng, năm) như 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**

```
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 ();
    }
}
```

4. Xây dựng một lớp *Diem* để mô tả các đối tượng điểm trong mặt phẳng như sau:

```
class Diem
{
private:
    int x, y;           //Tọa độ
public:
```

```
    Diem(int xx = 0, int yy = 0);
    void Nhap();
    void Xuat();
    void DiChuyen(int dx, int dy); //Di chuyển điểm đi một độ dời (dx, dy)
    int Trung(Diem u); //Kiểm tra hai điểm trùng nhau
    Diem DoiXung();    /*Tìm điểm đối xứng với một điểm cho trước qua gốc tọa độ*/
    double KCach(const Diem &u);          //Khoảng cách hai điểm
    int HopLe(const Diem &u, const Diem &v); /*Kiểm tra 3 đỉnh có
            tạo thành tam giác */
    double ChuVi(Diem &u, Diem &v); //Tính Chu vi tam giác qua ba điểm
    double DienTich(Diem &u, Diem &v); /*Tính Diện tích tam giác qua
            ba điểm*/
};
```

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

```
class Vector
{
private:
    int n;          //số chiều
    double *p;      //Con trỏ tới vùng nhớ động chứa các tọa độ
public:
    Vector();                       //Hàm thiết lập không đối số
    Vector(int nn);                 //Hàm thiết lập một đối số
    Vector(const Vector &u);        //Hàm thiết lập sao chép
    ~Vector();                      //Hàm hủy bỏ
    void Nhap();                    //Nhập tọa độ
    void Xuat();                    //Xuất tọa độ
    int LayN();                     //Lấy số chiều
    Vector Cong(const Vector &u);   //Cộng 2 vector
    Vector Tru(const Vector &u);    //Trừ hai vector
    Vector Nhan(double x);          //Nhân một vector với một số thực
    double Nhan(const Vector &u);   //Nhân vô hướng hai vector
};
```

6. Xây dựng một lớp *DaThuc* để mô tả các đối tượng đa thức như sau:

```
class DaThuc
{
```

```
private:
    int n;          //Bậc đa thức
    double *p;      //con trỏ tới vùng nhớ động chứa các hệ số
public:
    DaThuc();                       //Hàm thiết lập không đối số
    DaThuc(int nn);                 //Hàm thiết lập một đối số
    DaThuc (const DaThuc &u);       //Hàm thiết lập sao chép
    ~ DaThuc ();                    //Hàm hủy bỏ
    void Nhap();                    //Nhập đa thức
    void Xuat();                    //Xuất đa thức
    double GiaTri(double x);        //Tính giá trị đa thức tại x
    DaThuc Cong(const DaThuc &u); //Cộng hai đa thức
    DaThuc Tru(const DaThuc &u);  //Trừ hai đa thức
    DaThuc Nhan(const DaThuc &u); //Nhân hai đa thức
};
```

❖ **Hướng dẫn:**

```
DaThuc::DaThuc(int nn) {
    n = nn;
    p = new double[n + 1]; //Một đa thức bậc n có n + 1 hệ số
}
```

7. Xây dựng lớp *MaTran* để mô tả các đối tượng ma trận như sau:

```
class MaTran
{
private:
    int sd, sc;     //Số dòng và số cột
    double **p;     //Con trỏ tới vùng nhớ động chứa các phần tử
public:
    MaTran ();                      //Hàm thiết lập không đối số
    MaTran(int sd1, int sc1);       //Hàm thiết lập hai đối số
    MaTran (const MaTran &u);       //Hàm thiết lập sao chép
    ~ MaTran ();                    //Hàm hủy bỏ
    void Nhap();
    void Xuat();
    double TongDong(int k);         //Tổng các phần tử ở dòng thứ k
    int LaySD();                    //Lấy số dòng
```

```
    int LaySC();                        //Lấy số cột
    MaTran Cong(const MaTran &u); //Cộng hai ma trận
    MaTran Tru(const MaTran &u);  //Trừ hai ma trận
    MaTran Nhan(const MaTran &u); //Nhân hai ma trận
};
```

❖ **Hướng dẫn**

```
MaTran MaTran ::Nhan(const MaTran &u) {
    MaTran ret(sd, u.sc) ; //Gọi MaTran ::MaTran(int, int)
    for(int i = 0 ; i < ret.sd ; i++)
            for(int j = 0 ; j < ret.sc ; j++) {
                    ret.p[i][j] = 0;
                    for(int k = 0; k < sc; k++)
                            ret.p[i][j] += p[i][k] * u.p[k][j];
            }
    return ret;
}
```

8. Với hai lớp *Vector* và *MaTran* đã xây dựng ở trên, hãy xây dựng một hàm thực hiện việc nhân MaTran với Vector theo một trong hai giải pháp sau:

   Giải pháp 1: Khai báo hàm này là hàm tự do và là bạn của cả hai lớp *MaTran* và *Vector*.

   Giải pháp 2: Khai báo hàm này là hàm thành phần của lớp *MaTran* và là bạn của lớp *Vector*.

❖ **Hướng dẫn giải pháp 2:**

```
class Vector; //Khai báo trước lớp Vector

class MaTran {

private:

    . . .

public:

    . . .

    Vector NhanMV(const Vector &u);

};

class Vector {

private:
```

```
    . . .
public:
    . . .
    friend Vector MaTran::NhanMV(const Vector &u);
};
Vector MaTran::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;
}
```

9. Xây dựng lớp *NgayGio* để mô tả các đối tượng thời gian (bao gồm (h, m, s) và (d, m, y)) như sau:

```
class NgayGio
{
private:
    Gio A;
    Ngay B;
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;
        B.TangNgay();        //Gọi hàm Ngay::TangNgay()
    }
}
```

10. Xây dựng một lớp *TamGiac* để mô tả các đối tượng tam giác như sau:

```
class TamGiac
{
private:
    Diem A, B, C;     //Ba đỉnh
public:
    TamGiac(int x1 = 0, int y1 = 0, int x2 = 0, int y2 = 0, int x3 = 0, int y3 = 0);
    TamGiac(Diem AA, Diem BB, Diem CC);
    void Nhap();
    void Xuat();
    double ChuVi();
    double DienTich();
    Diem TrongTam(); //Tìm điểm trọng tâm tam giác
    TamGiac DienTichMax(TamGiac &u); /*Tìm tam giác có diện tích lớn nhất
    trong hai tam giác cho trước*/
};
```

❖ **Hướng dẫn**

```
double TamGiac::ChuVi() {
    return   A.KCach(B) + B.KCach(C) + C.KCach(A);
}
TamGiac TamGiac::DienTichMax(const TamGiac &u) {
    TamGiac ret = *this;
    if(ret.DienTich() < u.DienTich())
            ret = u;
    return ret;
}
```

11. Xây dựng hai lớp MonHoc và GiaoVien để mô tả các đối tượng môn học và giáo viên như sau:

```
class MonHoc
{
```

```
private:
    char tenmh[21];          //Tên môn học
    int st;                  //Số tiết
public:
    MonHoc();
    void Nhap();             //Nhập môn học
    void Xuat();             //Xuất môn học
    int LayST();             //Lấy số tiết
};
class GiaoVien
{
private:
    char tengv[31];          //Họ tên
    int ns;                  //Năm sinh
    int sm;                  //Số môn học giáo viên có thể dạy
    MonHoc *p;               //Con trỏ tới vnđ chứa các môn học
public:
    GiaoVien();
    GiaoVien(const GiaoVien &u);
    ~GiaoVien();
    void Nhap();             //Nhập giáo viên
    void Xuat();             //Xuất giáo viên
    void SapXep();           //Sắp xếp các môn học giảm dần theo số tiết
};
```

❖ **Hướng dẫn:**

```
MonHoc::MonHoc() {
    tenmh[0] = 0;
    st = 0;
}
void MonHoc::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 MonHoc::Xuat() {
    cout << tenmh << "\t" << st <<"\n";
}
GiaoVien::GiaoVien() {
    tengv[0] = 0;
    ns = 0;
    sm = 0;
    p = NULL;
}
GiaoVien::GiaoVien(const GiaoVien &u) {
    strcpy(tengv, u.tengv);
    ns = u.ns;
    sm = u.sm;
    p = new MonHoc[sm];
    for(int i = 0; i < sm; i++)
            p[i] = u.p[i];
}
void GiaoVien::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 MonHoc::Nhap()
    }
}
void GiaoVien::Xuat() {
    cout << tengv << "\t" << ns <<"\t" << sm <<"\n";
    for(int i = 0; i < sm; i++)
            p[i].Xuat(); //gọi MonHoc::Xuat()
}
void GiaoVien::SapXep() {
    for(int i = 0; i < sm – 1; i++)
```

```
        for(int j = i + 1; j < sm; j++)
            if(p[i].layST() < p[j].LayST()) {
                MonHoc tam = p[i];
                p[i] = p[j];
                p[j] = tam;
            }
}
```

12. Xây dựng một lớp HDBH (hóa đơn bán hàng) để mô tả các đối tượng hóa đơn như sau:

```
class HDBH
{
private:
    char tenmh[21];          //Tên mặt hàng
    double tb;               //Tiền bán
    static int tshd;         //Tổng số hóa đơn
    static double tstb;      //Tổng số tiền bán
public:
    HDBH(char *tenmh1 = NULL, double tb1 = 0.0);
    ~HDBH();
    void SuaTB(double tb1);  //Sửa tiền bán cũ thành tiền bán mới tb1
    static void Xuat();
};
```

❖ **Hướng dẫn:**

```
int HDBH::tshd = 0;
double HDBH::tstb = 0;
HDBH::HDBH(char *tenmh1, double tb1) {
    cout << "HDBH::HDBH(char *, double)\n";
    strcpy(tenmh, tenmh1);
    tb = tb1;
    tshd++;
    tstb += tb;
}
HDBH ::~HDBH() {
    cout << "HDBH::~HDBH()\n";
    tshd--;
    tstb -= tb;
```

```
}
void HDBH::SuaTB(double tb1) {
    tstb -= tb;
    tstb += tb1;
}
void HDBH::Xuat() {
    cout << “Tong so tien ban:” <<tstb<<”\n”;
    cout << “Tong so hoa don:” <<tshd<<”\n”;
}
```

CHƯƠNG 4: TOÁN TỬ TRÊN LỚP

1. Xây dựng lớp *SoPhuc* để mô tả các đối tượng số phức như sau:

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

❖ **Hướng dẫn**

```
doule SoPhuc::Module() const {
    return sqrt(re*re + im*im);
}
int operator>(const SoPhuc &u, const SoPhuc &v) {
    if(u.Module() > v.Module())
        return 1;
    else return 0;
}
SoPhuc SoPhuc::operator+=(const SoPhuc &u) {
    re = re + u.re;
    im = im + u.im;
    return *this;
}
```

2. Xây dựng lớp *PhanSo* để mô tả các đối tượng phân số như sau:

```
class PhanSo
{
private:
    int ts, ms;
public:
    PhanSo(int ts1 = 0, int ms1 = 1);
    int USCLN(int x, int y);
    void RutGon();
    void Nhap();
    void Xuat();
    //Toán tử số học
    friend PhanSo operator+(const PhanSo &u, const PhanSo &v);
    friend PhanSo operator-( const PhanSo &u, const PhanSo &v);
    friend PhanSo operator*( const PhanSo &u, const PhanSo &v);
    friend PhanSo operator/( const PhanSo &u, const PhanSo &v);
    //Toán tử quan hệ
    friend int operator>( const PhanSo &u, const PhanSo &v);
    friend int operator>=( const PhanSo &u, const PhanSo &v);
    friend int operator<( const PhanSo &u, const PhanSo &v);
    friend int operator<=( const PhanSo &u, const PhanSo &v);
    friend int operator==( const PhanSo &u, const PhanSo &v);
```

```
    friend int operator!=( const PhanSo &u, const PhanSo &v);
    //Toán tử số học mở rộng
    PhanSo operator+=( const PhanSo &u);
    PhanSo operator-=( const PhanSo &u);
    PhanSo operator*=( const PhanSo &u);
    PhanSo operator/=( const PhanSo &u);
    //Toán tử nhập xuất
    friend istream& operator>>(istream &is, PhanSo &u);
    friend ostream& operator<<(ostream &os, const PhanSo &u);
    //Toán tử tăng giảm
    PhanSo operator++();
    PhanSo operator++(int);
};
```

❖ **Hướng dẫn**

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

```
}
int PhanSo::operator+=(const PhanSo &u) {
    ts = ts * u.ms + ms * u.ts;
    ms = ms * u.ms;
    return *this;
}
PhanSo PhanSo::operator++() {
    cout << "PhanSo::operator++()\n";
    ts = ts + ms;
    PhanSo ret(ts, ms);
    return ret;
}
PhanSo PhanSo::operator++(int) {
    cout << "PhanSo::operator++(int)\n";
    PhanSo ret(ts, ms);
    ts = ts + ms;
    return ret;
}
```

3. 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
    double *p;      //Con trỏ tới vùng nhớ chứa các tọa độ
public:
    Vector();                       //Hàm thiết lập không đối số
    Vector(int nn);                 //Hàm thiết lập một đối số
    Vector(const Vector &u);        //Hàm thiết lập sao chép
    ~Vector();              //Hàm hủy bỏ
    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*(double x);      //Nhân vô hướng Vector với số thực
    double 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 double[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;
}
```

4. Xây dựng lớp *DaThuc* để mô tả các đối tượng đa thức như sau:

```
class DaThuc
{
private:
    int n;          //Bậc đa thức
    double *p;      //con trỏ tới vùng nhớ chứa các hệ số
public:
    DaThuc();               //Hàm thiết lập không đối số
    DaThuc(int nn);         //Hàm thiết lập một đối số
```

```
    DaThuc (const DaThuc &u);       //Hàm thiết lập sao chép
    ~ DaThuc ();                    //Hàm hủy bỏ
    void Nhap();
    void Xuat();
    friend istream& operator>>(istream &is, DaThuc &u); /*Toán tử
                                    nhập*/
    friend ostream& operator<<(ostream &os, const DaThuc &u); /*Toán
                                    tử xuất*/
    double operator^(double x);  //Toán tử tính giá trị đa thức tại x
    DaThuc& operator=(const DaThuc &u);  //Toán tử gán
    DaThuc operator+(const DaThuc &u);   //Toán tử cộng hai đa thức
    DaThuc operator-(const DaThuc &u);   //Toán tử trừ hai đa thức
    DaThuc operator*(const DaThuc &u);   //Toán tử nhân hai đa thức
    void operator-();                    //Toán tử đảo dấu đa thức
};
```

5. Xây dựng lớp *MaTran* để mô tả các đối tượng ma trận như sau:

```
class MaTran
{
private:
    int sd, sc;        //số dòng và số cột
    double **p;        //con trỏ tới vùng nhớ chứa các phần tử
public:
    MaTran ();                      //Hàm thiết lập không đối số
    MaTran(int sd1, int sc1);       //Hàm thiết lập hai đối số
    MaTran (const Matran &u);               //Hàm thiết lập sao chép
    ~ MaTran ();                    //Hàm hủy bỏ
    void Nhap();
    void Xuat();
    double 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, MaTran &u); /*Toán tử
                                    nhập*/
    friend ostream& operator<<(ostream &os, const MaTran &u); /*Toán
                                    tử  xuất*/
```

```
    Matran& operator=(const Matran  &u);   //Toán tử gán
    MaTran operator+(const MaTran  &u);   // Toán tử cộng hai ma tran
    MaTran operator-(const MaTran  &u);    // Toán tử trừ hai ma tran
    MaTran operator*(const MaTran  &u);    // Toán tử nhân hai ma tran
};
```

❖ **Hướng dẫn**

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

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

6. Xây dựng ba lớp MonHoc, GiaoVien và BoMon để mô tả các đối tượng môn học, giáo viên và bộ môn như sau:

```
class MonHoc
{
private:
    char tenmh[21];         //Tên môn học
    int st;                 //Số tiết
public:
    MonHoc();
    void Nhap();
    void Xuat();
    friend istream& operator>>(istream &is, MonHoc &u); //Toán tử nhập
    friend ostream& operator<<(ostream &os, const MonHoc &u); /*Toán
                            tử xuất*/
    int LayST();
};
class GiaoVien
{
```

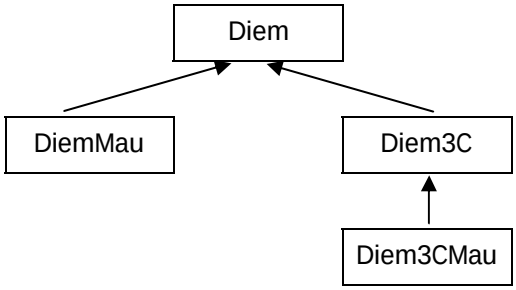
```
private:
    char tengv[31];          //Họ tên
    int ns;                  //Năm sinh
    int sm;                  //Số môn học có thể dạy
    MonHoc *p;               //Con trỏ tới vnđ chứa các môn học
public:
    GiaoVien();
    ~GiaoVien();
    GiaoVien(const GiaoVien &u);
    GiaoVien& operator=(const GiaoVien &u);
    friend istream& operator>>(istream &is, GiaoVien &u);
    friend ostream& operator<<(ostream &os, const GiaoVien u);
    LaySM();
    void Nhap();
    void Xuat();
    void SapXep();           //Sắp xếp ds môn học giảm dần theo số tiết
};
class BoMon
{
private:
    char tenbm[21];          //Tên bộ môn
    int sgv;                 //Số giáo viên
    GiaoVien *p;             //Con trỏ tới vnđ chứa các giáo viên
public:
    BoMon();
    ~BoMon();
    BoMon(const BoMon &u);
    BoMon & operator=(const BoMon &u);
    friend istream& operator>>(istream &is, BoMon &u);
    friend ostream& operator<<(ostream &os, const BoMon &u);
    void Nhap();
    void Xuat();
    void SapXep();  /*Sắp xếp danh sách giáo viên giảm dần theo số môn mà mỗi giáo viên có thể dạy*/
};
```

❖ **Hướng dẫn:**

```
void BoMon::Nhap() {
    cout << “Ten bo mon:”;
    cin.getline(tenbm, 20);
    cout << “So giao vien:”;
    cin >> sgv;
    cin.ignore();
    p = new GiaoVien[sgv];
    for(int i = 0; i < sgv; i++) {
    cout << “** Giao vien thu “ << i <<”: **\n”;
        p[i].Nhap(); //Gọi GiaoVien::Nhap()
    }
}
BoMon::BoMon(const BoMon &u) {
    cout << “BoMon::BoMon(BoMon &)\n”;
    strcpy(tenbm, u.tenbm);
    sgv = u.sgv;
    p = new GiaoVien[sgv];
    for(int i = 0; i < sgv; i++)
            p[i] = u.p[i]; //Gọi GiaoVien::operator=()
}
GiaoVien& BoMon::operator=(const BoMon &u) {
    cout << “BoMon::operator=(BoMon &)\n”;
    if(p != NULL)
            delete[]p;
    strcpy(tenbm, u.tenbm);
    sgv = u.sgv;
    p = new GiaoVien[sgv];
    for(int i = 0; i < sgv; i++)
            p[i] = u.p[i]; //Gọi hàm GiaoVien::operator=()
    return *this;
}
```

CHƯƠNG 5: KỸ THUẬT KẾ THỪA

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




```
class Diem3C : public Diem
{
    int z; //Tọa độ chiều thứ 3
public:
    . . .
}
```

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

```
class Luanvan                   //Lớp luận văn
{
private:
    char tenlv[31];             //Tên luận văn
    char tensv[31];             //Tên sinh viên
    int nbv;                    //Năm bảo vệ
public:
    Luanvan ();
    void Nhap();
    void Xuat();
};
class GVHDLV:public GiaoVien        //Lớp giáo viên hướng dẫn luận văn
{
private:
    int slv;            //số luận văn
    Luanvan *p;         //Con trỏ tới vnđ chứa các luận văn
public:
    GVHDLV();
    ~GVHDLV();
```

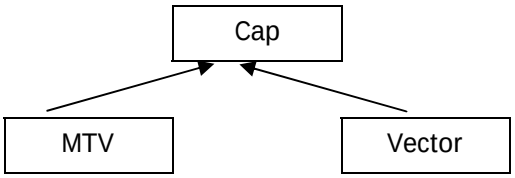
```
    GVHDLV(const GVHDLV &u);
    void Nhap();
    void Xuat();
    GVHDLV operator=(const GVHDLV &u);  //Toán tử gán
};
```

❖ **Hướng dẫn:**

```
void GVHDLV::Nhap() {
    GiaoVien::Nhap();
    cout << "So luan van";
    cin >> slv;
    cin.ignore();
    p = new LuanVan[slv];
    for(int i = 0; i < slv; i++) {
        cout << "Luan van thu << i <<"\n";
        p[i].Nhap(); //Goi luanVan::Nhap()
    }
}
void GVHDLV::Xuat() {
    GiaoVien::Xuat();
    for(int i = 0; i < slv; i++)
        p[i].Xuat(); //Goi luanVan::Xuat()
}
GVHDLV ::GVHDLV( const GVHDLV &u) : GiaoVien(u) {
    cout << "GVHDLV ::GVHDLV( GVHDLV &u)\n";
    slv = u.slv;
    p = new LuanVan[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
    *((GiaoVien *)this) = (GiaoVien&)u; /*Gọi          GiaoVien::operator=(GiaoVien&)*/
    //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;
}
```

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



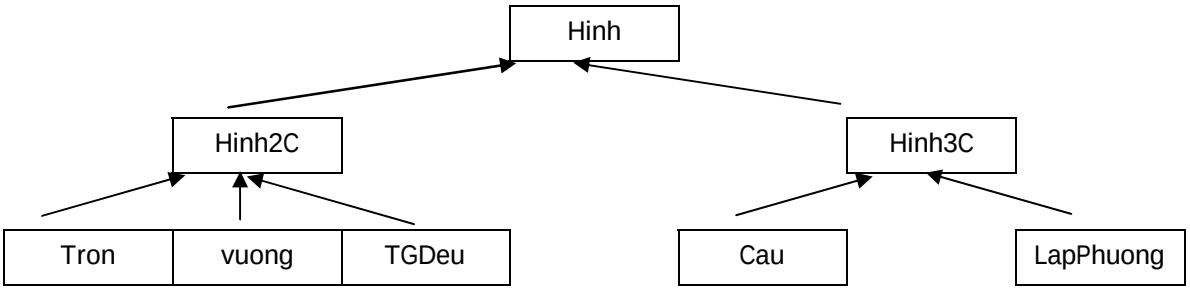

```
class Cap
{
    static int n;      /*Dùng chung cho đối tượng MTV và đối tượng          Vector*/
public:
    void Nhap();
    int LayN();
};
int Cap::n = 0;      //Khởi tạo thành phần dữ liệu static

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

```
private:
        double 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;
    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;
    n = LayN();
    if(n == 0) {
            Cap::Nhap();
            n = LayN();
    }
    for(int i = 0; i < n; i++) {
        cout <<"pt[" << i <<"]=";
        cin >>arr[i];
    }
}
```

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




```
class Hinh {
public:
    virtual double DienTich() = 0;
    virtual void Xuat() = 0; //Xuất thông tin của hình
};
class Hinh2C : public Hinh {
public:
    virtual double ChuVi() = 0;
};
class Hinh3C : public Hinh {
public:
    virtual double TheTich() = 0;
};
class Tron : public Hinh2C {
private:
    double r;
public:
    Tron(double rr);
    double DienTich();
    double ChuVi();
    void Xuat();
};
class Vuong : public Hinh2C {
    . . .
};
class TGDeu : public Hinh2C {
    . . .
};
class Cau : public Hinh3C {
```

```
    private:
        double r;
    public:
        Cau(double rr);
        double DienTich();
        double TheTich();
        void Xuat();
    };
    class LapPhuong : public Hinh3C {
        . . .
    };
```

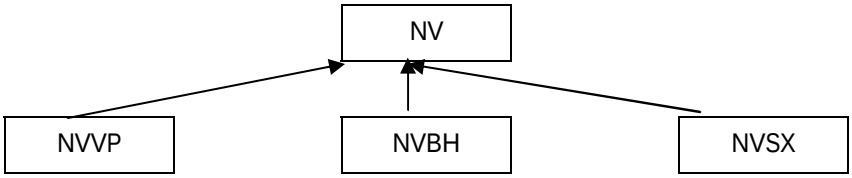
Lưu ý:

DT(Cầu) = $4*PI*r^2$ TT(Cầu) = $PI*r^3/4$

DT(L.phương) = $6*a^2$ TT(L.Phương) = a^3

DT(TG.đều) = $a^2*sqrt(3)/2$ CV(TG.Đều) = 3a

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




```
clas NV { //Lớp nhân viên
private:
        char msnv[21]; //Mã số nhân viên
        char ht[31]; //Họ tên
        double ml; //Mức lương
        int loai; /*Loại nhân viên = 0: nhân viên văn phòng, 1:nhân viên bán hàng, 2:nhân viên sản xuất*/
public:
        NV();
        virtual void Nhap();
        virtual void Xuat();
        virtual double TienThuong() = 0;
};
clas NVVP  : public NV { //Lớp nhân viên văn phòng
```

```
private:
        double tgct; //Thời gian công tác tính theo tháng
public:
        NVVP ();
        void Nhap();
        void Xuat();
        double TienThuong();
};
clas NVBH : public NV { //Lớp nhân viên bán hàng
private:
        double hst; //Hệ số thưởng
public:
        NVBH ();
        void Nhap();
        void Xuat();
        double TienThuong() ;
};
clas NVSX  : public NV { //Lớp nhân viên sản xuất
private:
        double tssp; //Tổng sàn phẩm tính từ đầu năm
public:
        NVSX ();
        void Nhap();
        void Xuat();
        double TienThuong();
};
//Hàm chính
void main() {
        NV *pnv[20]; //Mảng các con trỏ kiểu lớp trừu tượng
        int n; //Lưu số nhân viên
        int loai; //0: nvvp, 1: nvbh, 2:nvsx
        int i;
        <Nhap n>
        for(i = 0; i < n; i++) {
                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();
        }
        for(i = 0; i < n; i++) {
                cout << " ** Nhan vien thu " << i <<" **\n";
                pnv[i]->Xuat();
                cout <<"Tien luong: " << pnv[i]->TienLuong() << "\n";
        }
}
```

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

Đối với nhân viên văn phòng

nếu tgct < 6 thì tiền thưởng = 100000

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

Đối với nhân viên bán hàng

tiền thưởng = 150000 * hst

Đối với nhân viên sản xuất

tiền thưởng = 20000 * tssp

CHƯƠNG 6: KHUÔN HÌNH

1. Xây dựng khuôn hình hàm *HoanVi*() để hoán vị hai số truyền vào.
   Xây dựng khuôn hình hàm *SapXep*() dùng để sắp xếp một mảng theo thứ tự tăng trong đó có sử dụng khuôn hình hàm *hoanVi*().
   Xây dựng khuôn hình hàm *Nhap*() để nhập một mảng từ bàn phím.
   Xây dựng 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 ba mảng: một mảng nguyên, một mảng thực và một mảng ký tự.

   ❖ **Hướng dẫn:**

```
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";
}
```

2. Xây dựng lớp PhanSo

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

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

3. Xây dựng khuôn hình lớp Vector như sau:

```
template <class T >
class Vector {
private:
    int n;          //số chiều
    T arr[SIZE];    //Mảng chứa các tọa độ Vector
public:
    Vector();               //Hàm thiết lập không đối số
    Vector(int nn);         //Hàm thiết lập một đối số
    Vector(const Vector<T>& u);   //Hàm thiết lập sao chép
    Vector<T>& operator=(const Vector<T>& u);  /*Toán tử gán hai
                    Vector*/
    void Nhap();
    void Xuat();
};
```

```
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";
    }
```

4. Xây dựng khuôn hình lớp MTV (ma trận vuông) như sau:

```
    template <class T>
    class MTV {
    private:
        int n;                      //Cấp ma trận vuông
        T arr[SIZE][SIZE]           //Mảng chứa các phần tử ma trận vuông
    public:
        MaTran();                           //Hàm thiết lập không đối số
        MaTran(int nn);                     //Hàm thiết lập một đối số
        MaTran(const MaTran<T>& u); //Hàm thiết lập sao chép
        MaTran<T>& operator=(const MaTran<T>& u); /*Toán tử gán hai
                        MaTran*/
        MaTran<T> operator+(const MaTran<T> &u); /*Toán tử cộng hai
                        MaTran*/
        MaTran<T> operator*(const MaTran<T> &u); /*Toán tử nhân hai
                        MaTran*/
        void Nhap();
        void Xuat();
};
```