Bài tập C++ LẦN 3 – 2017 (nhóm 4,5,6,9)
Kiến thức: Tổng hợp + Class

Bài 1. Class ChuSoNguyen : có thuộc tính là 1 số nguyên, 1 mảng A[] các số có 1 chữ số, số phần tử . Viết các phương thức:

a. Khởi tạo số = 0
b. Nhập 1 số từ bàn phím
c. TachSo() : tách số đó ra thành các chữ số và lưu vào mảng A[]
d. In kết quả ra màn hình, ngăn cách nhau bởi 3 dấu cách.

Ví dụ với số 45321, in ra màn hình:
4 5 3 2 1

Bài 2. Class BMI có thuộc tính là chiều cao, cân nặng và chỉ số BMI. Viết các phương thức

a. Khởi tạo các thuộc tính = 0
b. Khởi tạo các thuộc tính chiều cao, cân nặng
c. Nhập chiều cao cân nặng từ bàn phím
d. Tính Chỉ số BMI (Body Mass Index) được tính theo công thức BMI = cân_nặng (kg)/chiều_cao(m) * chiều_cao(m)
e. Phân loại hiển thị kết quả theo thông số:
   Thieu can: nhỏ hơn 18.5
   Trung binh: từ 18.5 đến dưới 25
   Thua can: từ 25 đến dưới 30
   Beo phi: lớn hơn hoặc bằng 30

Bài 3. Class UCBC có thuộc tính là 2 số integer. Viết các phương thức:
a. khởi tạo 2 số = 0
b. khởi tạo 2 số
c. nhập 2 số
d. Tìm UCLN
e. Tìm BCNN

Bài 4. Class CoSo có thuộc tính là 1 số integer, cơ số b($1< b\leq 36$), mảng A[] để lưu kết quả đổi cơ số sang hệ cơ số b, độ dài mảng A. Viết các phương thức

a. Khởi tạo số và cơ số = 0
b. Khởi tạo số và cơ số
c. Nhập số, cơ số
d. Đổi số sang hệ cơ số b và lưu vào mảng số A
e. Hiển thị kết quả đổi cơ số

Bài 5. Class ThuaSo có thuộc tính là 1 số integer, 1 mảng A lưu các thừa số nguyên tố. Viết các phương thức:

a. khởi tạo số = 0
b. Nhập số
c. Phân tích số thành thừa số nguyên tố và lưu vào mảng A
d. Hiển thị kết quả phân tích

Ví dụ: Số 28 được phân tích thành 2 x 2 x 7

Bài 6. Class Fibonacci có thuộc tính là 1 số integer. Viết các phương thức:

a. Khởi tạo số = 0
b. Nhập 1 số từ bàn phím
c. Tính số Fibonacci theo công thức: F0 =1, F1 = 1; Fn = Fn-1 + Fn-2 với n>=2.

Bài 7. Class DanhSach có thuộc tính là mảng float A và số phần tử n. Viết các phương thức:

a. Nhập n và mảng A, các phần tử của A là những số nguyên lớn hơn 0 và nhỏ hơn 100
b. Tìm phần tử lớn nhất và lớn thứ 2 trong mảng cùng chỉ số
c. Sắp xếp mảng theo giá trị giảm dần
d. Sau khi sắp xếp, chèn x (x nhập từ bàn phím) vào mảng sao cho giá trị vẫn giảm dần
e. Sau khi sắp xếp, xoá x (x nhập từ bàn phím) khỏi mảng nếu có.

Bài 8: Class Dinh có thuộc tính là toạ độ (x,y) của nó. Class DaGiac gồm danh sách đỉnh, số đỉnh (số đỉnh<=5). Viết các phương thức:

a. Dinh: khởi tạo 1 đỉnh
b. Dinh: nhập 1 đỉnh
c. Dinh: hiển thị 1 đỉnh
d. Dinh: tính khoảng cách giữa 2 đỉnh
e. DaGiac: khởi tạo 1 đa giác = 0
f. DaGiac: nhập 1 đa giác
g. DaGiac: hiển thị 1 đa giác
h. DaGiac: Kiểm tra loại của đa giác: tam giác, hình vuông, hình chữ nhật, bình hành, hình thang, hình ngũ giác đều, hình đa giác bình thường.
i. DaGiac: tính diện tích đa giác
j. DaGiac: tính chuvi đa giác

Bài 9. Viết class DaThuc có thuộc tính là **hệ số a[], bậc** n và các phương thức là:

a. Phương thức khởi DaThuc()tạo 1 đa thức trống
b. Phương thức nhập nhap() 1 đa thức từ bàn phím
c. Phương thức hiển thị hienthi() 1 đa thức ra màn hình
d. Phương thức đọc docfile(ifstream) 1 đa thức từ file

e. Phương thức ghi ghiFile(ofstream) 1 đa thức ra file
f. Phương thức tổng tong(Dathuc) đa thức và return kết quả là 1 Dathuc
g. Phương thức hiệu hieu(Dathuc) đa thức và return kết quả là 1 Dathuc
h. Phương thức hiệu tich(Dathuc) đa thức và return kết quả là 1 Dathuc
i. Phương thức tính gia tri giatri(float x) đa thức và return kết quả là 1 số float
j. Chồng toán tử nhập đa thức từ bàn phím
k. Chồng toán tử hiển thị đa thức ra màn hình
l. Chồng toán tử tổng (+), hiệu(-), tích(*) đa thức

Các đa thức được lưu trong file dathuc.inp và kết quả cần lưu vào file dathuc.out
Dathuc.inp: dòng đầu ghi bậc của đa thức; dòng tiếp theo ghi các hệ số từ bậc cao nhất về 0
Ví dụ:
3
1 2 -3 4.5
2
3 2 1.5
dathuc.out: dòng đầu ghi bậc của tổng của 2 đa thức, dòng tiếp ghi hệ số từ bậc cao nhất về 0 của tổng 2 đa thức; tương tự với hiệu2 đa thức.
Ví dụ:
3
1 5 -1 6
3
1 -1 -5 3

Bài 10. Viết class MaTran có thuộc tính là **số dòng n, số cột m, phần tử** và các phương thức là:

a. Phương thức khởi tạo MaTran () tạo 1 ma trận trống
b. Phương thức nhập nhap() 1 ma trận từ bàn phím
c. Phương thức hiển thị hienthi() 1 ma trận ra màn hình
d. Phương thức đọc docfile(ifstream) 1 ma trận từ file
e. Phương thức ghi ghiFile(ofstream) 1 ma trận ra file
f. Phương thức tổng tong(MaTran) ma trận và return kết quả là 1 MaTran
g. Phương thức hiệu hieu(MaTran) và return kết quả là 1 MaTran
h. Phương thức hiệu tich(MaTran) và return kết quả là 1 MaTran
i. Phương thức tìm max các phần tử của ma trận maxMT(float x) và return kết quả là 1 số float
j. Chồng toán tử nhập ma trận từ bàn phím
k. Chồng toán tử hiển thị ma trận ra màn hình
l. Chồng toán tử tổng (+), hiệu(-), tích(*)ma trận

Các ma trận được lưu trong file matran.inp và kết quả cần lưu vào file matran.out

matran.inp: dòng đầu ghi số dòng và cột của ma trận; các dòng tiếp theo ghi các phần tử của ma trận thứ 1; tương tự với ma trận thứ 2

Ví dụ:

3 3
1 2 3
4 5 6
7 8 9
3 3
9 8 7
6 5 4
3 2 1

kqmatran.out: dòng đầu ghi số dòng và cột của ma trận; các dòng tiếp theo ghi các phần tử của ma trận tổng; cách 1 dòng rồi ghi tiếp ma trận hiệu

ví dụ:

3 3
10 10 10
10 10 10
10 10 10
3 3
-8 -6 -4
-2 0 2
4 6 8

Bài 11. Class TicTacToe có thuộc tính là 1 mảng tic 3x3. Viết các phương thức:

a. Phương thức khởi tạo TicTacToe khởi tạo các phần tử mảng tic = 0
b. Phương thức kiểm tra ketthuc() trả về -1 nếu chưa kết thúc, 0 nếu hoà, 1(hoặc 2) nếu người 1 (hoặc 2) thắng.
c. Phương thức capnhat(int x, int y, int luotchoi) trả về true nếu cập nhật được tic[x][y] = luotchoi, false nếu ô tic[x][y] đã được điền trước đó (!=0)
d. Phương thức luuFile(ofstream f, int x, int y, int luotchoi) lưu các bước cập nhật thành công ở câu c vào file tictactoe.dat
e. Phương thức choiGame() cho phép 2 người chơi. Khi người 1 đi, điền 1 vào ô tương ứng. Khi người 2 đi, điền 2 vào ô tương ứng. Các lượt đi chỉ được điền vào ô còn trống. Sau mỗi lần đi, xác định game đã có người thắng hay cần tiếp tục.

Kết quả các bước đi cần lưu ở file tictactoe.dat: mỗi dòng ghi STT của người chơi, dòng và cột người đó đi.

Ví dụ:

1 2 3
2 3 3

Lưu ý: Game Tic-tac-toe giống cờ caro, nhưng chỉ giới hạn bảng 3x3, ai được 3 ô thẳng hàng trước sẽ thắng.

Bài 12. Class Sudoku có thuộc tính là 1 mảng sud 9x9, viết các phương thức:

a. Phương thức khởi tạo Sudoku() đọc file Sudoku.dat và khởi tạo giá trị cho mảng thuộc tính
b. Phương thức ketThuc() kiểm tra xem toàn bộ mảng đã được điền các số từ 1-9 chưa
c. Phương thức kiemTra(int x, int y, int c) trả về true nếu được phép điền vào ô sud[x][y] giá trị c, false nếu không được. Ta được phép điền khi:
   - Ô sud[x][y] hiện tại chưa được đi (=0)
   - Cột y chưa có ô nào bằng c
   - Dòng x chưa có ô nào bằng c
   - Ma trận con 3x3 chứa sud[x][y] chưa có ô nào bằng c
d. Phương thức capnhat(int x, int y, int c) cập nhật Sud[x][y] = c sau khi kiểm tra hợp lệ ở c
e. Phương thức choiGame() cho phép người chơi game

Cách chơi: Điền số từ 1 đến 9 vào những ô trống sao cho mỗi cột dọc, mỗi hàng ngang, mỗi phân vùng nhỏ (ô 3x3) có đủ các số từ 1 đến 9 mà không được lặp lại.
Bảng câu đố hình vuông, mỗi chiều có 9 ô nhỏ, hợp thành 9 cột, 9 hàng và được chia thành 9 ô lớn 3x3. Một vài ô nhỏ được đánh số, đó là những manh mối duy nhất để bạn tìm lời giải. Bảng câu đố lưu ở file text Sudoku.dat. File text này có 9 dòng, mỗi dòng có 9 số một chữ số: 0 tương ứng ô trên bàn cờ còn trống, 1-9 tương ứng với giá trị được điền ở vị trí tương ứng.

003020600
900305001
001806400
008102900
700000008
006708200
002609500
800203009
005010300

Bài 13 Tạo 1 class SinhVien gồm các thuộc tính: mã, họ, đệm, tên, điểm, giới tính và các phương thức thực hiện:

a. Phương thức khởi tạo SinhVien () tạo 1 SinhVien trống
b. Phương thức nhập nhap() 1 SinhVien từ bàn phím
c. Phương thức hiển thị hienthi() 1 SinhVien ra màn hình
d. Phương thức đọc docfile(ifstream) 1 SinhVien từ file sinhvien.dat
e. Phương thức ghi ghiFile(ofstream) 1 SinhVien ra file sinhvien.dat
   Lưu ý: tự xác định cấu trúc file sinhvien.dat
f. Chồng toán tử nhập1 SinhVien từ bàn phím
g. Chồng toán tử hiển thị 1 SinhVien ra màn hình

h. Phương thức sosanhTen(SinhVien sv) so sánh 2 họ tên với nhau (trả về 0 nếu bằng nhau, -1 nếu sv nhỏ hơn, 1 nếu sv lớn hơn)

Bài 14: Tạo class DSSinhVien sử dụng class SinhVien ở bài 6, gồm các thuộc tính số sinh viên, mảng SinhVien. Viết các phương thức thực hiện:

a. Phương thức khởi tạo DSSinhVien() khởi tạo 1 danh sách rỗng
b. Phương thức timkiem(ID) trả về chỉ số sinh mảng của sinh viên có mã tương ứng
c. Phương thức sua(SinhVien sv) sửa sinh viên có ID là sv.ID thành dữ liệu như sv (dùng b để tìm chỉ số mảng)
d. Phương thức xoa(int id) để xoá sinh viên có ID tương ứng (dung b để tìm chỉ số mảng)
e. Phương thức them(SinhVien sv) để them 1 sinh viên vào danh sách
f. Phương thức sapxepHoTen() để sắp xếp lại sinh viên theo họ tên ABC
g. Phương thức Lưu danh sách sinhvien vào file sinhvien.dat
h. Phương thức Đọc danh sách sinh viên từ file sinhvien.dat

Bài 15. Class TracNghiem có thuộc tính là: câu hỏi, 4 phương án trả lời, đáp án, trả lời của người chơi

Class DSTracNghiem có thuộc tính là:: số câu hỏi, mảng TracNghiem, số câu trả lời đúng. Viết các phương thức thực hiện:

a. Class TracNghiem: phương thức đọc doc() 1 câu hỏi từ bcvt.dat vào 1 đối
b. Class TracNghiem: phương thức hiển thị hienthi() 1 câu hỏi
c. Class TracNghiem: Phương thức kiemtra() kiểm tra trả lời của người chơi có giống với đáp án không?
d. Class DSTracNghiem: phương thức đọc doc() 1 danh sách câu hỏi từ bcvt.dat
e. Class DSTracNghiem: Phương thức kiemtra() sử dụng hàm TracNghiem::kiemtra() để kiểm tra người chơi với từng câu hỏi trong danh sách có đúng không và đếm số câu trả lời đúng.
f. Class DSTracNghiem: Phương thức phanloai(): nếu số câu trả lời đúng >80%: xuat sac;

- nếu số câu trả lời đúng >=60%: “Rat tot”
- nếu số câu trả lời đúng <60% : “Can phai tim hieu them ve truong”.

file bcvt.data: dòng đầu ghi số câu hỏi; dòng tiếp ghi nội dung câu hỏi đầu; 4 dòng tiếp ghi 4 phương án trả lời; dòng tiếp ghi STT của đáp án đúng; tương tự với các câu còn lại.

Ví dụ:

3
Khoa CNTT cua PTIT hoc may nam?
3 nam
4 nam

5 nam
4.5 nam
4
Ten tieng Viet cua PTIT la gi?
Hoc vien Cong nghe Buu chinh Vien thong
Truong Cong nghe Buu chinh Vien thong
Hoc vien Buu chinh
Hoc vien Buu chinh Vien thong
1
PTIT truc thuoc bo nao?
Bo Giao duc va Dao tao
Bo Thong tin va Truyen thong
Bo Khoa hoc va Cong nghe
Ca 3 bo tren
2

Bài 16: **(Rational Class)** Tạo một class tên là Rational để thực hiện các thao tác số học với phân số (+,-,*,/). Viết chương trình để kiểm tra class của bạn.
Sử dụng số integer để biển diễn dữ liệu private của class: tử số, mẫu số. Viết hàm tạo để cho phép 1 đối tượng của class này có thể khởi tạo khi nó được khai báo. Hàm tạo cần chứa giá trị mặc định trong trường hợp không có giá trị khởi tạo nào được cung cấp và cần lưu dưới dạng rút gọn. Ví dụ 2/4 cần lưu với tử số =1 và mẫu số =2 (1/2). Viết hàm public để thực hiện các nhiệm vụ sau.
a. Cộng 2 phân số, kết quả cần lưu ở dạng rút gọn
b. Trừ 2 phân số, kết quả cần lưu ở dạng rút gọn
c. Nhân 2 phân số, kết quả cần lưu ở dạng rút gọn
d. Chia 2 phân số, kết quả cần lưu ở dạng rút gọn
e. Hiển thị 1 phân số dưới dạng a/b với a là tử số và b là mẫu số, kết quả cần lưu ở dạng rút gọn
f. Hiển thị phân số dưới dạng số floating point

Bài 17: Viết class Time để hiển thị thời gian hiện tại sử dụng phương thức tick. Phương thức tick tăng thời gian lưu trong 1 đối tượng Time thêm 1 giây. Đối tượng Time phải luôn ở trạng thái hợp lệ. Viết chương trình kiểm tra phương thức tick sử dụng vòng lặp để hiển thị thời gian theo định dạng chuẩn. Nhớ kiểm tra các chương hợp.
a. Tăng sang phút mới
b. Tăng sang giờ mới
c. Tăng sang ngày mới (ví dụ: 11:59:59 PM thành 12:00:00 AM)

Bài 18: Tạo một class HugeInteger sử dụng 1 mảng 40 phần tử để lưu số integer có tối đa 40 chữ số.

a. Viết các phương thức input (nhập), output (hiển thị), add (+) và subtract (-).

b. Để so sánh các đối tượng HugeInteger, viết phương thức isEqualTo (==), isNotEqualTo (!=), isGreaterThan (>), isLessThan(<), isGreaterThanOrEqualTo (>=), và isLessThanOrEqualTo (<=), isZero (==0)- mỗi hàm sẽ trả về giá trị true nếu quan hệ thoả mãn, false nếu ngược lại.

c. Viết các phương thức multiply (*), divide (/) và modules(%)

Bài 19: Áp dụng Class HugeInteger cho Class DaThuc, MaTran, DaySo, viết 1 menu tương ứng

Bài 20: Class So có thuộc tính là một số nguyên K , số chữ số N(N≤15) , hệ cơ số b. Viết các phương thức thực hiện:

a. Kiểm tra K là số nguyên tố;
b. Kiểm tra K là số thuận nghịch (k là số thuận nghịch nếu đọc xuôi hay đọc ngược các chữ số của k ta đều nhận được một số như nhau. Ví dụ số: 30303);
c. Kiểm tra Biểu diễn của K ở hệ cơ số B (B bất kỳ được nhập từ bàn phím cũng là một số thuận nghịch. Ví dụ số k=30303 có biểu diễn ở hệ cơ số 8 là 73137 cũng là một số thuận nghịch;
d. Liệt kê các số có N chữ số thoả mãn a,b,c

Bài 21. Class CapSo có thuộc tính là 2 số S, T. Viết các phương thức:

a. khởi tạo cho S, T = 0
b. kiểm tra nguyento(x) kiểm tra x có phải là số nguyên tố có 4 chữ số hay không
c. nhap() yêu cầu nhập 2 số S, T là 2 số nguyên tố có 4 chữ số, áp dụng câu b
d. dichchuyen(): tìm cách dịch chuyển S thành T thỏa mãn đồng thời những điều kiện dưới đây:

- Mỗi phép dịch chuyển chỉ được phép thay đổi một chữ số của số ở bước trước đó (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1733 là hợp lệ);
- Số nhận được cũng là một số nguyên tố có 4 chữ số (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1833 là không hợp lệ, và S dịch chuyển thành 1733 là hợp lệ);
- Số các bước dịch chuyển là ít nhất.

Dãy các phép dịch chuyển thỏa mãn điều kiện ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các phép dịch chuyển nhỏ nhất;
- Dòng kế tiếp ghi lại các số nguyên tố được dịch chuyển theo chiều ngược lại từ T về S theo từng bước.

Ví dụ với S = 1033, T = 8179 trong file nguyento.in dưới đây sẽ cho ta file ketqua.out như sau:

ketqua.out

6

8179 8779 3779 3739 3733 1733 **1033**

Bài 22: **Hệ thống thông tin vận chuyển**

Công ti vận chuyển cung cấp hàng loạt dịch vụ chuyển đồ với các mức giác khác nhau. Hãy tạo một sơ đồ kế thừa để biểu diễn các loại vận chuyển. Sử dụng Package như class cơ sở, sau đó thêm 2 class TwoDayPackage và OvernightPackage kế thừa từ Package. Class Package gồm các thuộc tính biểu diễn tên, địa chỉ, thành phố, quận cho cả người gửi và người nhận, ngoài ra cần lưu khối lượng (kg) và phí chuyển/kg. Hàm tạo của package cần khởi tạo các thông tin này. Cần đảm bảo khối lượng và phí chuyển là số dương. Package cần có phương thức public calculateCost trả về 1 số double tính phí chuyển tương ứng. class dẫn xuất TwoDayPackage cần kế thừa các chức năng của class Package, và cần thêm thuộc tính để biểu diễn phí lưu kho mà công ty vận chuyển thu cho dịch vụ giao hàng trong 2 ngày. Hàm tạo của TwoDayPackage cần nhận 1 giá trị để khởi tạo cho thuộc tính này. TwoDayPackage cần định nghĩa lại hàm calculateCost để tính phí chuyển thêm cả phí lưu kho. Class OvernightPackage cần kế thừa trực tiếp từ Package và chứa thêm thuộc tính tính phí phụ trội/kg cho dịch vụ giao hàng trong ngày. OvernightPackage cần định nghĩa lại hàm calculateCost để thêm phí giao hàng trong ngày.Viết chương trình tạo các đối tượng thuộc từng loại dịch vụ và tính chi phí chuyển hàng.

Bài 23: **Hệ thống tài khoản**

Tạo 1 sơ đồ kế thừa mà một ngân hàng sẽ dùng để biểu diễn tài khoản của khách hàng. Tất cả các khách hàng có nạp tiền vào tài khoản và rút tiền từ tài khoản. Loại tài khoản tiết kiệm tính lãi dựa trên tiền gửi. Ngược lại, tài khoản tiền gửi thanh toán tính phí cho từng giao dịch (rút tiền/gửi tiền).

Tạo một sơ đồ kế thừa gồm một class cơ sở Account và 2 class dẫn suất Savings-Account và CheckingAccount kế thừa từ class Account. Class cơ sở Account cần có một thuộc tính kiểu double để biểu diễn số dư tài khoản. Class cần có 1 hàm tạo nhận 1 giá trị để khởi tạo cho số dư tài khoản. Hàm tạo cần kiểm tra xem giá trị khởi tạo này phải >=0.0 . Nếu không, số dư tài khoản sẽ được khởi tạo = 0.0 và hàm tạo cần hiển thị 1 thông báo lỗi là giá trị khởi tạo không hợp lệ. Class cần 3 hàm thành viên. Hàm credit thêm 1 lượng tiền vào số dư hiện tại. Hàm debit sẽ trừ bớt 1 lượng tiền và cần đảm bảo lượng tiền trừ bớt không vượt quá số dư tài khoản. Nếu vi phạm, số dư tài khoản cần được giữ nguyên (như trước lúc bị trừ) và hàm cần hiển thị 1 thông báo lỗi "không được rút số tiền vượt quá số dư hiện có". Hàm getBalance sẽ trả về số dư hiện tại.

class dẫn suất SavingsAccount cần kế thừa các chức năng của class Account, nhưng cần thêm 1 thuộc tính kiểu double để lưu lãi suất (dưới dạng %). Hàm tạo cần nhận 1 giá trị khởi tạo cho số dư tài khoản, và 1 giá trị khởi tạo cho lãi suất. SavingsAccount cần có hàm public calculateInterest trả
về 1 số double là lãi suất thu được bằng cách nhân lãi suất với số dư. (Lưu ý: cần kế thừa hàm credit và debit mà không cần phải định nghĩa lại)
class dẫn suất CheckingAccount kế thừ từ class Account và thêm một thuộc tính kiểu double biểu diễn phí giao dịch phải trả cho mỗi giao dịch. Hàm tạo của CheckingAccount cần nhận 1 giá trị khởi tạo cho số dư tài khoản và 1 giá trị khởi tạo cho phí giao dịch. class CheckingAccount cần phải định nghĩa lại hàm credit và debit để trừ thêm phí giao dịch từ số dư tài khoản khi mỗi giao dịch được thực hiện thành công. Hàm credit và debit của class CheckingAccount cần gọi tới hàm tương ứng ở class Account để thực hiện cập nhật số dư tài khoản. Hàm debit của CheckingAccount chỉ tính phí giao dịch KHI VÀ CHỈ KHI tiền thực sự được rút (khi mà số dư tài khoản lớn hơn số tiền định rút). (Gợi ý: định nghĩa hàm debit của Account có kiểu trả và là bool để xác định tiền đã được rút thành công hay chưa. Sau đó, sử dụng giá trị trả về của hàm này để xác định xem có tính phí giao dịch không)
Sau khi định nghĩa các class trên, viết chương trình tạo các đối tượng của từng class và kiểm tra các hàm thành viên. Ngoài ra, thêm 1 đối tượng SavingsAccount thực hiện: gọi hàm calculateInterest, truyền giá trị trả về của hàm này vào hàm credit.

Bài 24: Sử dụng mô hình kế thừa Package tạo ở bài trên để tạo 1 chương trình hiển thị địa chỉ và tính phí vận chuyển cho nhiều Packages. Chương trình cần có 1 vector của các con trỏ Package trỏ tới các đối tượng của các class TwoDayPackage và OvernightPackage. Xử lý các Packages một cách đa hình. Với từng Package, gọi hàm get để nhận địa chỉ của người gửi và người nhận, in 2 địa chỉ này ra màn hình. Đồng thời, gọi hàm calculateCost của Package và hiển thị kết quả ra màn hình. Lưu lại phí vận chuyển của từng Package theo vector và hiển thị tổng phí vận chuyển của các Package.

Class 25: Viết 1 chương trình đa hình sử dụng mô hình kế thừa Account trong bài trên. Tạo 1 vector các con trỏ Account để trỏ vào đối tượng của các class SavingsAccount và CheckingAccount. Với mỗi Account trong vector, cho phép người dùng xác định số tiền để rút khỏi tài khoản sử dụng hàm debit và số tiền gửi thêm vào tài khoản sử dụng hàm credit. Trong quá trình xử lý tài khoản, xác định kiểu của nó. Nếu tài khoản là SavingsAccount, tính lãi suất sử dụng hàm calculateInterest, và thêm lãi suất vào số dư hiện tại sử dụng hàm credit. Sau khi xử lý 1 tài khoản, hiển thị số dư tài khoản mới bằng cách gọi hàm getBalance.

Bài 26: Một khách sạn phân cấp các phòng theo nhiều loại và dựa trên thời gian thuê của từng khách để lập hóa đơn tiền phòng.
Khai báo lớp Nguoi***(Họ tên, email, SĐT)***. Khai báo lớp KhachHang kế thừa từ lớp

Nguoi và có thêm ***(mã khách hàng, kiểu phòng cần thuê, mô tả).***
Khai báo lớp Phong gồm các thuộc tính ***(mã phòng, Kiểu phòng, Mức tiền thuê)*** – với kiểu phòng có thế là: phòng đơn, phòng đôi và phòng VIP, mã phòng là một số nguyên có 3 chữ số.
Khai báo lớp ***Bảng Sắp xếp*** là bạn của lớp KhachHang và lớp Phong trong đó một khách hàng được sắp xếp tại loại phòng phù hợp (nếu thiếu thì đề nghị loại phòng khác) cùng với số ngày thuê. Một khách hàng không được mượn quá 50 phòng.
Viết chương trình trong ngôn ngữ C++ thực hiện các yêu cầu sau:

a. Nhập thêm Phòng vào file PH.DAT. In ra danh sách phòng với mức tiền thuê >1 triệu đã có trong file.
b. Nhập thêm Khách hàng vào file KH.DAT. In ra danh sách các KH đã có trong file.
c. Nhập danh sách sắp xếp phòng cho mỗi khách hàng đã có trong file KH.DAT; lưu vào file BANGSX.DAT và in danh sách ra màn hình.
d. Sắp xếp danh sách đã lưu trong BANGSX.DAT theo kiểu phòng 5. Tính toán và lập hóa đơn cho mỗi khách hàng.