Bài C++ lần 3 (nhóm cô Ngọc)

36. Tạo 1 chương trình toán học với 1 menu là các chức năng tính toán gồm:

a. Tính tổng/tích/phần nguyên, phần dư 2 số (Bài 1/baiLan1)

b. Tính đường kính/chu vi/diện tích hình tròn (Bài 3/baiLan1)

c. Tính chỉ số BMI (bài 8/baiLan1)

d. Tìm ước chung lớn nhất, bội chung nhỏ nhất của 2 số (bài 9/baiLan1)

e. Chuyển cơ số 10 thành cơ số bất kỳ (bài 10/baiLan1)

g. Phân tích 1 số nguyên thành các thừa số nguyên tố (bài 11/baiLan1)

f. Tính giá trị đa thức, đạo hàm, tổng đa thức (bài 16/baiLan1)

g. Tìm phần tử lớn nhất, lớn thứ 2 trong mảng, sắp xếp mảng (bài 17/baiLan1)

h. Tìm từ dài nhất trong xâu (bài 18/baiLan1)

i. Tìm hàng hoặc cột hoặc đường chéo của ma trận có tổng phần tử lớn nhất; tìm ma trận chuyển vị; tìm định thức; tìm ma trận nghịch đảo (bai 19/baiLan1)

j. Hiển thị dòng theo thứ tự ngược (bài 22/baiLan2)

k. Xác định 1 xâu có đối xứng không (bài 23/baiLan2)

l. Chuyển biểu thức trung tố sang hậu tố (bài 24/baiLan2)

m. Tính giá trị biểu thức viết dạng hậu tố (bài 25/baiLan2)

n. Cộng, trừ, nhân, chia đa thức (bài 27/baiLan2)

o. Cộng, trừ, nhân ma trận (bài 28/baiLan2)

p. Tách họ tên (Bài 31/baiLan2)

Lưu ý: chuyển hết các chức năng sang class, có thể gói 1 số chức năng vào chung 1 class.

37. (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

38. 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)

39. (HugeInteger Class) 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(%)

40. 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.

41. 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.

42. 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.

43. 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.

44. 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 ***Người (Họ tên, email, Số ĐT)***. Khai báo lớp ***Khách hàng*** kế thừa từ lớp ***Người*** và có thêm ***(mã khách hàng, kiểu phòng cần thuê, mô tả).***

Khai báo lớp ***Phòng*** 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 **Khách hàng** và lớp **Phòng** 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:

1. Nhập thêm Phòng vào file PH.DAT. In ra danh sách phòng với mức tiền thuê >1 triệu đã có trong file.
2. Nhập thêm Khách hàng vào file KH.DAT. In ra danh sách các KH đã có trong file.
3. 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.

4. Sắp xếp danh sách đã lưu trong BANGSX.DAT theo kiểu phòng
5. Tính toán và lập hóa đơn cho mỗi khách hàng.

45. Khai báo lớp ***Người (Họ tên, Email, Số ĐT)***

Khai báo lớp ***Khách hàng*** kế thừa từ lớp **Người** và bổ sung các thuộc tính ***(mã KH, Loại KH)*** - với loại khách hàng có thể là: cá nhân, tập thể, mã KH là một số nguyên có 5 chữ số.

Khai báo lớp ***Loại lãi suất (loạiLS, lãi suất tiền gửi, mô tả).***

Khai báo lớp ***Sổ tiết kiệm*** là **bạn** của lớp ***Khách hàng*** và lớp ***Loại lãi suất*** trong đó với mỗi khách hàng cho biết số tiền gửi, số tháng định gửi. Một khách hàng có thể lập một hoặc nhiều số tiết kiệm với một hoặc nhiều loại lãi suất khác nhau nhưng tổng số sổ <30.

Viết chương trình trong ngôn ngữ C++ thực hiện các yêu cầu sau:

1. Nhập thêm khách hàng vào file KH.DAT. In ra danh sách khách hàng là tập thể đã có trong file.
2. Nhập thêm loại lãi suất vào file laisuat.DAT. In ra danh sách loại lãi suất đã có trong file.
3. Nhập danh sách Sổ tiết kiệm cho mỗi khách hàng đã có trong file KH.DAT; lưu danh sách vào file SOTK.DAT và in ra màn hình.
4. Sắp xếp danh sách Sổ tiết kiệm đã lưu trong SOTK.DAT Số tiền gửi
5. Lập bảng kê tổng số tiền gửi cho mỗi khách hàng