Проект 1: Command Line Utilities

Този проект представлява серия от инструменти с които ще се обработват текстови и бинарни файлове. Обработката на текстовите файловете ще включва филтриране на съдържанието по дадени критерии, търсене и сортиране.

Домашно 4 Напишете клас представящ филтър, който предоставя възможността да се чете вход от конзолата до край на файл ([ctrl] + D/Z). След това може да изведе последователно тези от прочетените редове, в които се съдържа избрана дума.
Преценете какви член функции и член данни трябва да има този клас.
Демонстрирайте използването му в кратка main функция.
Напишете клас представящ подредена поредица от филтри (FilterChain). Класът ще трябва да може се инициализира с поток за писане и поток за четене. Класът предоставя възможност да се запише филтрираното съдържание от входния поток, преминало последователно през всички филтри, в изходния поток. Добавете подходящи методи за добавяне и премахване на филтрите от класа.
В това домашно ще трябва да предефинирате изброените оператори за съответния клас, както поне още 4 които не са изброени за класа FilterChain.
За клас Filter, следните оператори:
operator= - стандартен
operator== - сравнява два филтъра по стринга който филтрират
operator!= - обратното на ==
operator<< - записва низът за филтриране в поток (низ за филтриране - това което се филтрира от текста)
operator>> - чете си низът за филтриране от поток (низ за филтриране - това което се филтрира от текста)
operator+= - десен аргумент char добавя аргумента към края на низът за филтриране
operator+= - десен аргумент char* добавя дадения низ към края на низът за филтриране
operator| - два аргумента Filter връща FilterChain съставен от аргументите си

За клас FilterChain, следните оператори:
operator= - стандартен
operator== - сравнява само филтрите, но не и подредбата (равни FilterChain-ове дават еднакъв резултат при еднакъв вход)
operator!= - обратното на == (различни FilterChain-ове дават различен резултат при различен вход)
operator+= - добавя Filter към класа
operator| - ляв аргумент FilterChain, десен - Filter - връща FilterChain с добавен десния аргумент
operator+ - два аргумента FilterChain - връща нов, с филтрите от аргументите без повторение

operator-= - десен аргумент char*, премахва всички филтри който филтрират подадения низ
operator[] - десен аргумент int връща филтър на позиция подаденото число
operator[] - десен аргумент char* връща филтър филтриращ подадения низ

Документирайте и подреждайте кода си.
Домашно 5. Сложете следните класове в подходяща йерархия с наследяване:

- WordFilter - досегашният филтър който пропуска редове които съдържат дадена дума
- EncodeFilter - криптира/кодира входен текстов файл по избран от вас начин
- DecodeFilter - декриптира/разкодира входен файл от направен с EncodeFilter
- CapitalizeFilter - променя първата буква на всяка дума в главна от входния текстов файл

Бонус:

- ZeroEscapeFilter
  - работи върху бинарни файлове
  - изхода на филтъра е бинарен
  - в изхода на филтъра не се съдържа byte със стойност 0x00/0
- ZeroUnescapeFilter
  - работи върху бинарни файлове направени от ZeroEscapeFilter
  - изхода е бинарен и е същият какъвто е бил преди да премине през ZeroEscapeFilter

Пояснения:

1. ZeroUnescapeFilter прави обратният процес на ZeroEscapeFilter-а, той трябва да възтанови escpae-натото съдържание.
2. Изхода на EncodeFilter-а може да е текстов или бинарен но DecodeFilter-а трябва да може да възтанови началното съдържание. Начинът по който кодирате или криптирате съдържанието може да е какъвто и да е, стига да е обратим.

- Реализирайте общ интерфейс за всички филтри.
- Направете FilterChain да поддържа всички филтри които имате до сега.
- Добавете в сегашната йерархия клас SortFilter който трябва да връща редовете от входа сортирани лексикографски
- Демонстрирайте как поне 3 от филтрите работят в един FilterChain обект последователно.

Проект 2: Игра с карти

Целта на проекта ви е да реализирате походова игра с карти. Играта се играе от двама играчи, всеки от които притежава предварително зададено тесте. В тестето участват до 10 карти. Всяка карта притежава четири свойства - Атака, Живот, Необходима енергия, Умения (от 1 до 3 на брой). Победител се излъчва когато единия

играч остане без карти. За изваждането на карта на "бойното поле" е необходима енергия. Играчите започват с опередена енергия и след всеки ход получават допълнителна такава.
Домашно 4. Първата ви задача около този проект е да създадете клас карта. Той трябва да притежава горепосочените свойства като на този етап ще игнорираме уменията. Необходимата енергия се пресмята по формулата Е = Ж/100 + А/20 (Е - необходима енергия, Ж - живот, А - атака). Класът трябва да притежава конструктор, копиращ конструктор и деструктор. Останалите методи и член-данни трябва да бъдат избрани от вас. Демонстрирайте използването на този клас в кратка main функция.
Имплементирайте нов клас Deck (тесте), които да съдържа масив от карти (произволен брой). За да кажем че тестето е валидно то трябва да съдържа поне 5 карти. Освен това трябва да поддържате добавяне на нова, изтриване или промяна на съществуваща карта от тестето .
За класа Card предефинирайте следните оператори:
operator= - присвояване на карата
operator== - проверява дали 2 карти имат еднакви параметри
operator!= - проверява дали поне едно поле на 2 карти е различно
operator<< - оператор за изход в поток
operator>> - оператор за вход от поток
operator+= - Приема десен аргумент число и увеличава текущата кръв на карата с него (кръвта не може да надминава максималната)
operator-= - Приема десен аргумент число и намалява текущата кръв на карата с него (кръвта не може да е отрицателно число)

За класа Deck предефинирайте следните оператори:
operator= - пристояване на дек
operator== - проверява дали два дека имат еднакви карти (без значение подредбата)
operator!= - проверява дали в два дека има поне 1 различна карата или различен брой карти
operator < > <= >= - сравнява сборът от атака и живота на картите в двата дека
operator<< - оператор за изход в поток
оператор+ - приема карта и дек и връша дек с картите от дека и новата карта(независимо дали отляво седи картата или дека)
оператор+ - приема два дека и връща нов дек с картите от двата
operator+= - приема десен аргумент карта и я прибавя в дека
operator+= - приема десен аргумент дек и прибавя всички карти от него в дека от ляво
opertaor[] - приема аргумент цяло число n и връща указател към n-тата карта в дека

За класа Deck предефинирайте поне още четири оператора по ваше усмотрение.
Домашно 5. Съставете подходяща йерархия от наследяване на следните класове:

- BattleField - клас който представя поле което съдържа изиграните карти на двамата играчи. Картите се слагат на полето винаги в най-лявата свободна позиция. На полето всеки играч може да има неограничено количество карти. Ако някоя карта умре трябва всички карти от дясната й страна да се преместят с една позиция на ляво (в масива не трябва да има дупки). Умрялата карта отива в общ масив наречен гробище където се съхраняват всички умрели карти до момента.
- Класа BattleField трябва да може да бъде сериализиран в масив от байтове, който не трябва да съдържа байт избран при конструирането на BattleField-а.

- Имплементирайте клас Skill който представлява умение на карта. Всяко умение може да въздейства на една или повече карти (свои или противникови) които се намират на полето или в гробището. Класа Skill трябва да има метод, който приема като аргумент бойното поле, списък с позиците на картите върху който ще въздейства (както и други методи който усигуряват нормалното му функциониране).
- Имплементирайте поне 2 класа, който наследяват Skill И въздействат на собствени карти. Като умението на поне един от класовете действа на повече от 1 карта.
- Имплементирайте поне 2 класа, който наследяват Skill и въздействат на вражеските картиКато умението на поне един от класовете действа на повече от 1 карта.
- Имплементирайте клас, който извиква от гробището, призовават, или забраняват на карта да се завърне от гробището.
- Имплементирайте 2 класа, който се случват с някаква вероятност.
- Имплементирайте 2 класа, който въздействат на поне 3 карти (противникови или собствени, като вие решете какво ще стане ако няма достатъчен брой карти на полето).

Атрибутите и въздействието на всички класове са по ваше усмотрение.
Документирайте добре как точно се очаква, да се държат вашите умения, за да не стават недоразумения.
Създайте два нови типа карта:

- Карта Guard - докато е на полето противниковите карти са задължени да атакуват нея преди всички останали. Изключение правят специалните умения който си действат по нормалния за тях начин.
- Карта Assasin - след като изпълни нормалния си ход нанася 150% щети на героя(или на карта Guard, ако има такава).

Променете класа Card по подходящ начин така че да е възможно да поддържа до 3 различни умения. Променете класа Deck така че да е възможно да се слагат карти от други типове.

Срока на това допълнително задание е до деня на защитата.

- Създайте клас Player който трябва да съдържа задължително Deck , точки живот и точки енергия.
- Допълнете класа BattelField така че да съдържа двама играчи.
- Имплементирайте механиката на битката при следните условия:

- Играчите трябва да имат избор, коя карта да сложат(спрямо това дали имат достатъчно енергия за нея), вие решете как играча получава точките енергия.
- Трябва да имат избор на всеки ход, кое умение на всяка карта да използват.(както и трябва да измислите начин не всички умения да са възможни за ползване на всеки ход).
- Когато карта удари по играча му намаля точките живот.
- Ако играча остане без точки живот умира и другия играч печели принцесата.

- За всичко това създайте подходящ потребителско интерфейс, който да е удобен и лесен за ползване.