**00 (рус)**

Создать класс **”Студент”** с несколькими полями.

Обязательные поля:

- строковое поле, хранящее имя студента

- 2 дробных поля (float, double), хранящие коэффициент лени (от 0 до 1) и IQ (от 100 до 200) студента

- целочисленное поле (byte), хранящее информацию о количестве экзаменов студента

- динамический вектор (int), хранящий инфу об оценке студента за каждый экзамен (от 0 до 10)

- статическое целочисленное поле, в котором будет храниться число зачисленных (созданных) студентов.

Необязательные поля: группа, номер паспорта, вредность, пол и пр.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей «подноготной» студента, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию подсчета среднего балла (среднее арифметическое оценок)
* Описать функцию подсчета стипендии студента (учитывая его лень и его IQ)
* Описать функцию «отличник», сравнивающую двух студентов (вызывающего и принятого как параметр) по среднему баллу, используя вышеописанную функцию. Имя того, кто учится лучше вывести на экран.
* Описать статическую функцию «умник», принимающую как параметры двух студентов, и считающую, на сколько первый студент умнее (в леях – по стипендии) или глупее второго (результат вывести на экран).

В функции **main()** понапринимать (понасоздавать) студентов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Студент**”, вывести на экран его поля методом класса. Создать динамический вектор студентов – «группа». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом гаврике из группы. Сравнить несколько пар студентов «отличником» и «умником». Найти и распечатать имя самого ленивого студента из группы. Последней строкой вывести число созданных студентов, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**00 (rom)**

De creat clasa “**Student**” cu câteva câmpuri.

Cîmpurile obligatorii:

- cîmpul de tip șir de caractere, care păstrează numele studentului.

- 2 cîmpuri fracționale (float, double), ce păstrează coeficienții lenii (0 - 1) și IQ (100 - 200) ai studentului.

- cîmp de tip număr intreg (byte), care păstrează informația despre numărul de examene ai studentului.

- vector dinamic (int), care păstrează informația despre notele studentului la fiecare examen (de la 0 la 10)

- cîmp static de tip întreg, care păstrează informația despre numărul studenților creați.

Cîmpurile neobligatorii: grupa, numărul pașaportului, nocivitatea, sexul ș.a.m.d.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).

De creat metoda pentru a afişa la ecran toată informaţia despre student.

De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.

De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.

De scris o funcţie care va calcula media aritmetică a notelor studentului.

De scris o funcţie ce calculеаză bursa studentului (luînd în considerație IQ și lenea lui).

De scris o funcţie ”eminent”care compară 2 studenți(unu care apelează și altul primit ca parametru) dupa media aritmetica a notelor, folosind funcția descrisă mai sus. De afișat la ecran numele celui care învață mai bine.

De scris o funcţie static ”desteptu”, care primeşte ca parametri doi studenți, şi care calculează cu cît primul student e mai deștept ca al doilea(în lei – după bursă) sau mai prost ca acesta (rezultatul de afișat la ecran).

În funcţia **main()** de creat o gașcă de studenți, folosind toţi constructorii descrişi. După crearea fiecărui obiect al clasei „Student”, de afişat la ecran cîmpurile lui folosind metoda clasei. De creat un vector dinamic de studenți -"grupa". Vectorul să fie iniţializat pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia despre fiecare student din grupă. De comparat cîțiva perechi de studenți după ”eminență” și ”deșteptăciune”. De găsit și de afișat numele celui mai lenos student din grupă. Ultimul rînd afişat să conţină numărul studenților creați, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**1 (рус)**

Создать класс “**Страус**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число потомства страуса;

- динамический вектор, в котором хранится информация о каждом потомке страуса (возможны 3 варианта: женский пол - значение **-1**, мужской - значение **1**, и - неизвестный - значение 0 :) т.е. это яйцо попросту съели, не успев определить его пол :))

- статическое целочисленное поле, в котором будет храниться число созданных страусов.

Необязательные поля: пушистость, скорость бега, сила вбивания головы в землю :), возраст и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всей подноготной одного конкретного страуса, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух страусов (двух экземпляров класса, принятых как параметры) по количеству потомства.
* Описать функцию, вычисляющую сумарное количество съеденных яиц у страуса :)

В функции **main()** понасоздавать страусов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Страус**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор страусов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом страусе из вектора. Сравнить плодовитость нескольких пар страусов соответствующими функциями класса. Результаты сравнения вывести на экран. Подсчитать сумму всех съеденных страусиных яиц у всех созданных страусов тоже при помощи метода класса, а также, при помощи функций-методов доступа к полям класса. Результат вывести на экран.

Последней строкой вывести число созданных страусов, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**1 (rom)**

De creat clasa “**Strutz**” cu câteva câmpuri.

Câmpurile obligatorii:

- un câmp de tip intreg (int), pentru păstrarea numărul de pui ai strutzului;

* un vector dinamic, în care se păstrează informaţia despre fiecare pui al strutzului (sunt posibile 3 variante: genul feminin cu valoare: -**1**, genul masculin: +**1**, şi necunoscut: **0** :) adicâ acest ou a fost mâncat :))
* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

Câmpuri opţionale: densitatea pufului, viteza de alergare, forţa aplicată la introducerea capului în pământ :), vârsta etc…

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre strutz, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda statică, ce compară doi strutzi (două instanţe (obiecte) a clasei – ambele obiecte primim ca parametrii) în baza numărului de pui
* De creat metoda, care calculează numărul sumar de ouă, care a fost mâncat la strutz :)

În metoda **main()** de creat nişte strutzi, utilizând toţi constructorii declaraţi. După creerea fiecărei instantze a clasei “**strutz**”, de afişat la ecran câmpurile lui cu ajutorul metodei respective (set() şi get()). De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia a tuturor strutzi in vector. De comparat câteva perechi de strutzi în baza numărului de pui cu ajutorul metodei respective. Rezultatele comparării de afişat la ecran. De calculat numărul de ouă, care au fost mâncate la toţi strutzii creaţi, de asemenea cu ajutorulmetodei corăspunzătoare precum şi cu metodele de acces la câmpurile clasei. Rezultatele de afişat la ecran.

De afişat la ecran numarul general de strutzi creaţi, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**2 (рус)**

Создать класс “**Коктейль**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее число ингридиентов коктейля;

- динамический вектор, где хранится инфа о каждом ингридиенте (его градус алкоголя)

- дробное поле, в котором хранится стоимость коктейля (что немаловажно ☺)

- статическое целочисленное поле, где будет храниться число созданных коктейлей.

Необязательные поля: общая масса коктейля, его цвет и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран рецептуры коктейля, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию сравнения двух коктейлей по числу ингридиентов.
* Описать функцию, вычисляющую общий градус коктейля (среднее арифметич. всех градусов ингридиентов).
* Описать статическую функцию void client(), принимающую два входных параметра: количество денег в кошельке клиента (double), и конкретный коктейль (объект класса). В зависимости от того, сколько таких коктейлей клиент может себе позволить, и от крепости коктейля в градусах, функция должна выводить на экран реплику клиента после употребления одного или нескольких таких коктейлей.

В функции **main()** понаделать коктейлей, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Коктейль**”, вывести на экран его поля методом класса, а также для каждого коктейля подсчитать и вывести его средний градус. Создать динамический вектор коктейлей. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом коктейле из вектора. Сравнить сложность (число ингридиентов) нескольких пар коктейлей соответствующей функцией класса. Результаты сравнения вывести на экран. Подсчитать общую стоимость всех выпитых коктейлей, используя методы get(). Результат вывести на экран. Вызвать для нескольких коктейлей функцию client. Последней строкой вывести число созданных коктейлей, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**2 (rom)**

De creat clasa **“Сokteil”** cu câteva câmpuri.

Câmpurile obligatorii:

- un câmp de tip intreg (int), pentru păstrarea numărul de ingrediente a cocteilului.

- un vector dinamic, în care se păstrează informaţia despre fiecare ingredient (procentul de alcool ingredientului respectiv).

- un câmp real, , în care se păstrează preţul cocteilului.

* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

Câmpuri opţionale: masa totală a kocteilului, culoarea lui si denumirea, etc…

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (metodele set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre cocleil (toate câmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda de comparare a 2 kocteiluri dupa numarul de ingrediente.
* De creat metoda care calculeaza procentul total de alcool din kocteil (medie aritmetică elementelor vectorului dinamic).
* De descris funţia statică void client(), care primeste 2 parametri: numar real ce indica cantitatea banilor clientului, si un coctail concret (obiectul clasei). In dependenţa de numarul de coctailuri ce clientul isi poate permite, si de gradul acestui coctail, funcţia trebuie sa afişeze pe ecran exclamaţia clientului după consumarea a unuia sau mai multor cocteiluri.

În metoda **main()** de creat diferite kocteiluri, utilizind toate tipurile de constructori. Dupa crearea fiecarui exemplar al clasei “Сocteil” de afişat la ecran cimpurile clasei cu ajutorul metodei respective si de afisat procentul de alcool al kocteilului respective. ()). De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector. De comparat numarul ingredientelor al citorva kocteiluri si rezultatul de afişat la ecran. De calculat preţul total de kocteiluri create, folosind metodele set() şi get(). De apelat pentru câteva cocteiluri metoda client(). De afişat la ecran numarul general de cocteiluri creaţi, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**3 (рус)**

Создать класс “**Фирменное блюдо**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число ингридиентов блюда;

- динамический вектор, в котором хранится инфа о каждом ингридиенте (его цена)

- дробное поле, показывающее общую массу блюда.

- статическое целочисленное поле, в котором будет храниться число созданных блюд.

- общая масса блюда

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения всех полей класса (set() и get()).
* Описать функцию вывода на экран рецептуры одного конкретного блюда, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух блюд (оба блюда принимаются как параметры) по числу ингридиентов
* Описать функцию, вычисляющую сумарную стоимость блюда
* Описать функцию client(), принимающую два входных параметра: число денег в кошельке клиента (double), и объем желудка клиента (float). Функция должна посчитать и вернуть целое значение текущих блюд (текущий объект класса), которое клиент может себе позволить в плане денег (анализируем общую стоимость блюда, используя соответствующую функцию ее подсчета) и в плане вместимости желудка (анализируем массу блюда).

В функции **main()** приготовить несколько разных блюд, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Фирменное блюдо**”, вывести на экран его поля с помощью соответствующего метода класса, для каждого блюда подсчитать его общую стоимость и вывести результат на экран. Создать динамический вектор блюд (меню). Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом блюде. Сравнить сложность (число ингридиентов) нескольких пар блюд (результаты сравнения вывести). Подсчитать и вывести стоимость всех съеденных блюд (всех созданных экземпляров) используя методы доступа к полям класса. Для нескольких блюд вызвать функцию client, и вывести на экран ее результат. Последней строкой вывести число созданных блюд, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**3 (rom)**

De creat clasa **"Bucata"** cu citeva cimpuri:

Câmpurile obligatorii:

* un câmp de tip intreg (int), care arata numarul ingredientelor.
* un vector dinamic, în care se păstrează informaţia despre fiecare ingredient (preţul ingredientului respectiv).
* masa totală a unei pizza.
* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (set() si get()).
* De creat metoda pentru a afişa reţeta unei anumite mincari cu toate câmpurile clasei.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda statică ce compară doua mincaruri dupa numarul ingredientelor (ambele obiecte primim ca parametri).
* De creat metoda care calculeaza pretul total al unei mincari.
* De descris funţia client(), care primeste 2 parametri: cantitatea banilor clientului (double), si volumul stomacului clientului (float). Metoda trebuie să returneze numarul de feluri de mincare, pe care clientul isi poate permite din punctul de vedere al pretului (luind in consideratie pretul felului), si din punct de vedere al volumului stomacului(luind in consideratie masa mincarii).

În metoda **main()** de pregătit diferite feluri de mincare, utilizând toate tipurile de constructor.Dupa crearea fiecarui exemplar al clasei "Bucata" de afişat la ecran câmpurile clasei cu ajutorul metodei respective, de afişat pretul fiecarei mincari. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector. De comparat numarul ingredientelor de la câteva feluri de mincare corespunzator câmpurilor clasei si rezultatul compararii sa se afişeze la ecran. De sumat numarul mincarurilor comestibile create. Rezultatele de afişat la ecran. De apelat pentru câteva feluri de mâncare metoda client(). De afişat la ecran numarul general de mâncări creaţi, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**4 (рус)**

Создать класс “**Автомобиль**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число аварий этого разнесчастного авто;

- динамический вектор, в котором хранится информация о каждой аварии (т.е. стоимость каждого ремонта)

- статическое целочисленное поле, в котором будет храниться число созданных автомобилей.

Необязательные поля: год выпуска, марка, максимальная скорость и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения всех полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию сравнения двух автомобилей по количеству аварий
* Описать функцию, вычисляющую сумарную стоимость всех ремонтов
* Описать статическую функцию, вычисляющую сумму, в которую обошлись все ремонты по двум машинам, принятым как входные параметры

В функции **main()** понасоздавать разных машин, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Автомобиль**”, вывести на экран его поля с помощью соответствующего метода класса, а также для каждого авто подсчитать его общую ремонтную стоимость и вывести результат на экран. Создать динамический вектор автомобилей. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом автомобиле из вектора. Сравнить аварийность (число аварий) нескольких пар авто соответствующими функциями класса. Результаты сравнения вывести на экран. Подсчитать сумму всех ремонтов всех созданных машин тоже при помощи метода класса, а также, при помощи функций-методов доступа к полям класса. Результат вывести на экран. Последней строкой вывести число созданных автомобилей.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

(и никогда собственнолично не попадать в аварии)

**4 (rom)**

De creat clasa **„ Automobil”** cu cîteva cîmpuri.

Câmpurile obligatorii:

* un câmp de tip intreg (int) pentru păstrarea numărului total de accidente ale automobilului;
* un vector dinamic în care se păstrează informaţia despre fiecare accident (costul fiecărei reparaţii);
* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

Câmpuri opţionale: anul producerii, marca, viteza maximă, etc.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (set() si get()).
* De creat metoda, ce afişează la ecran toate cîmpurile clasei.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda, ce compară 2 automobile după numărul de accidente.
* De creat metoda, ce calculează suma totală a reparaţiilor automobilului curent.
* De creat metoda statică care primeşte 2 parametri – obiecte clasei curente, si calculează suma totală a reparaţiilor pentru ambele automobile.

În metoda **main()** de creat nişte automobile, toate tipurile de constructor. Dupa crearea fiecarui exemplar al clasei "Automobil" de afişat la ecran câmpurile clasei cu ajutorul metodei respective, de afişat pretul fiecarei mincari. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector. Pentru fiecare automobil, de calculat şi de afişat la ecran suma totală a reparaţiilor. De comparat numarul de accidente a câtorva automobile cu metodele corespunzătoare. Rezultatele comparării de afişat la ecran. Pentru o pereche de automobile de calculat şi de afişat numarul total de accidente, utilizând metoda statică. De afişat la ecran numarul general automobile create, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRAAAAAA !!!**” şi de prezentat primul laborator profesorului. :)

**5 (рус)**

Создать класс “**Аквариум**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее число обитателей аквариума;

- динамический вектор, в котором хранится информация о каждом обитателе (возможны 3 варианта: хищники - значение **-1**, безобидные рыбки - значение **1**, и растительность - значение 0)

- длина, ширина и высота аквариума

- статическое целочисленное поле, в котором будет храниться число созданных аквариумов.

Необязательные поля: форма аквариума, цена, толщина стекла и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса(set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию сравнения двух аквариумов (двух экземпляров класса: текущего и принятого как входной параметр) по количеству их хищных обитателей.
* Описать функцию слияния двух аквариумов, т.е. добавление к одному аквариуму всех обитателей другого аквариума, полученного в качестве параметра данной функции.
* Описать статическую функцию, вычисляющий общий литраж двух аквариумов, принятых как входные параметры.

В функции **main()** понаделать разных аквариумов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Аквариум**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор аквариумов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом аквариуме. Сравнить заселенность (число обитателей) нескольких пар аквариумов соответствующими функциями класса. Результаты сравнения вывести на экран. Объединить несколько аквариумов. После каждого объединения вывести все измененные поля экземпляра класса на экран. Применить статическую функцию к двум аквариумам из вектора. Результат вывести на экран. Последней строкой вывести число созданных аквариумов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**5 (rom)**

De creat clasa **„ Acvariu”** cu câteva cîmpuri.

Câmpurile obligatorii:

* un câmp de tip intreg (int) pentru păstrarea numărului total de veţuitore in acvariu;
* un vector dinamic în care se pastrează informaţia destre fiecare veţuitoare (sunt posibile 3 variante : prădător – valoarea -1, peşti inofensivi – valoarea 1, vegetaţie – valoarea 0 )
* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

Câmpuri opţionale: forma lui acvariu, pretul lui, dimensiunea, volumul maximal, etc.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (metodele set() si get()).
* De creat metoda ce afişează la ecran toată informaţia despre “acvariu” (toate cîmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda ce compară două obiecte clasei (current si primit ca parametru la intrare) curente după numărul de prădătoare.
* De creat metoda care descrie procesul de fuziune a doua acvarii: la acvariu current se adauga toate veţuitoare acvariului, care este primit ca parametru la intrare.
* De creat metoda statică care primeşte 2 parametri – 2 obiecte clasei curente, si calculează suma totală de vegetaţie pentru ambele obiecte.

În metoda **main()** de creat nişte acvarii, utilizand toate tipurile de constructor. Dupa crearea fiecarui exemplar al clasei "Acvariu" de afişat la ecran câmpurile clasei cu ajutorul metodei respective. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector.

De comparat numarul de veţuitoare a câtorva acvarii cu metodele corespunzătoare. Rezultatele comparării de afişat la ecran. De fuzionat o pereche de acvarii şi de afişat datele schimbate. De apelat la metoda static pentru 2 acvarii din vector. În final de afişat la ecran numarul general de obiecte create, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat

primul laborator profesorului. :)

**6 (рус)**

Создать класс “**митинг**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число участников митинга;

- динамический вектор, в котором хранится информация о каждом участнике (возможны 3 варианта: активисты - значение **1**, милиция - значение **-1**, и журналисты - значение 0)

- статическое целочисленное поле, где хранится число созданных митингов.

Необязательные поля: число противников митинга, его разновидность и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух митингов (двух экземпляров класса, принятых как входные параметры) по количеству активистов
* Описать функцию слияния двух митингов, т.е. добавление к текущему митингу всех участников другого митинга, полученного в качестве параметра данной функции.

В функции **main()** понаделать митингов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**митинг**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор объектов класса. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом митинге из вектора.

Сравнить массовость (число участников) нескольких пар митингов соответствующими функциями класса. Результаты сравнения вывести на экран. Объединить несколько митингов в одну толпу. После каждого объединения вывести все новые поля экземпляра класса на экран. Последней строкой вывести число созданных объектов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**6 (rom)**

De creat clasa **„ Meeting”** cu câteva cîmpuri.

Câmpurile obligatorii:

* un câmp de tip intreg (int) pentru păstrarea numărului total a tuturor participantilor la meeting
* un vector dinamic în care se pastrează informaţia destre fiecare participant (sunt posibile 3 variante : activişti - valoarea 1, poliţişti 1 si jurnalişti- 0)
* un câmp (intreg, static), in care se păstrează № de obiecte create.

Câmpuri opţionale: numarul inamicilor, etc.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (metodele set() si get()).
* De creat metoda ce afişează la ecran toată informaţia despre “meeting” (toate cîmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda statică ce compară două obiecte (primite ca parametru la intrare) clasei curente după numărul de activişti.
* De descris metoda, care concateniază (fuzionează) 2 obiecte de tip Meeting: participantii Meeting-ului primit ca parametru la intrare se adaugă la Meeting current.
* De descris functia de afisare la ecran a tuturor cimpurilor clasei.
* De apelat functia de comparare ,de apelat functia de concatenare,

În metoda **main()** de creat nişte obiecte, utilizand toate tipurile de constructor. Dupa crearea fiecarui exemplar al clasei "Meeting" de afişat la ecran câmpurile clasei cu ajutorul metodei respective. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector.

De comparat numarul de activisti a câtorva acvarii cu metodele corespunzătoare. Rezultatele comparării de afişat la ecran. De fuzionat niste meeting-uri şi de afişat datele schimbate În final de afişat la ecran numarul general de obiecte create, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRAAAAAA !!!**” şi de prezentat primul laborator profesorului. :)

**7 (рус)**

Создать класс “**Усатый дворник**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число метелок у дворника;

- динамический вектор, в котором хранится информация о каждой метле (степень ее износа от 0 до 5, где 5- последняя степень, когда от метелки уже одна палка осталась :))

- статическое целочисленное поле, в котором будет храниться число созданных дворников.

Необязательные поля: возраст дворника, рост, вес, градус крови, длина усов и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух дворников (двух экземпляров класса, принятых как входные параметры) по трудолюбию, т.е. по общей изношенности метелок (по сумме элементов вектора)
* Описать функцию апгрейда метелок дворника, т.е. если износ метелки уже предельный (=5), то покупается новая метелка, т.е. значение этой ячейки меняется с 5 на 0.

В функции **main()** понанимать дворников, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Усатый дворник**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор дворников. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом дворнике из вектора.

Сравнить трудолюбие нескольких пар дворников соответствующими функциями класса. Результаты сравнения вывести на экран. Затем - апгрейдить некоторых из них и снова вывести на экран поля изменившегося дворника. После чего обновленного дворника опять сравнить с тем же аппонентом и результат опять вывести на экран. Последней строкой вывести число созданных дворников.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “УРРРРАААА ! ! ! !” и сдать первую лабораторную работу преподавателю. :)

**7 (rom)**

De creat clasa “**Măturatorul mustecios**” cu câteva câmpuri.

Câmpurile obligatorii:

- (int), pentru păstrarea numărului de mături a măturatorului;

- un vector dinamic, în care se păstrează informaţia despre coeficientul de uzare a fiecărei mături (coeficientul de uzare să fie de la 0 - 5, unde 5 - ultimul coeficient în cazul cînd matura sa uzat complet).

- un câmp de tip intreg static in care se pastreaza numarul de maturatori creati.

Câmpuri opţionale: vîrsta maturătorului, înălţimea, greutate(kg), nume, lungimea mustetii ş.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei.
* De creat metoda pentru a afişa la ecran toată informaţia despre măturător, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda statică, care compară doi măturători (2 obiecte ale clasei, primate ca parametrii la intrare) după hărnicie, după numarul de mature uzate (suma coificienţilor de uzare);
* De creat metoda, care reînoieste maturile uzate( adică dacă matura are coeficientul de uzare 5 se “arunca” şi se înlocueşte cu una nouă care are coeficientul de uzare 0);

În metoda **main()** de creat ciţiva măturători utilizând toţi constructorii declaraţi. După creerea fiecărei instantze a clasei “Măturător”, de afişat la ecran câmpurile cu ajutorul metodei respective. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia obiectelor din vector. De comparat câţiva măturători cu ajutorul metodei respective. Rezultatele comparării de afişat la ecran. De reactualizat inventariul măturătorului, de asemenea cu ajutorul metodei corăspunzătoare. Rezultatele de afişat la ecran.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**8 (рус)**

Создать класс “**Змей Горыныч**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее число голов у Горыныча;

- динамический вектор, в котором хранится IQ каждой головы :)

- статическое целочисленное поле, где будет храниться число созданных Горынычей.

Необязательные поля: возраст, рост, вес, максимальная температура пламени и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух Змеев (двух экземпляров класса) - кто умнее ? :), (т.е. среднее арифметическое IQ всех голов)
* Описать функцию сражения двух змеев, в которой они произвольно или по какому-то правилу (в зависимости от их ума, например) теряют одну или несколько голов.

В функции **main()** понапридумывать Змеев Горынычей, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Змей Горыныч**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор Змеев. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом Горыныче из вектора. Сравнить умственные способности нескольких пар Горынычей соответствующими функциями класса. Результаты сравнения вывести на экран. Затем - заставить несколько пар Змеев побиться и результат (т.е. уже измененные характеристики Горынычей) опять вывести на экран. Последней строкой вывести число созданных Горынычей.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

(и больше - близко не подходить к огнеопасным животным :))

**8 (rom)**

De creat clasa ”**Balaur**” cu cîteva câmpuri.

Câmpuri obligatorii:

* Cîmp întreg, care afișează numărul total de capete ale unui balaur.
* Vector dinamic în care se păstrează informația despre coeficientul de intelect al fiecărui cap balaurului.
* Cîmp static de numere întregi unde se va păstra numărul de balauri creați.

Plus cîteva câmpuri neobligatorii: vîrsta, greutatea, forța e.t.c...

De definit trei tipuri de constructori ai acestei clase(cu alocare dinamică de memorie):

* Constructor implicit(fără parametri).
* Cîțiva constructori diferiți cu parametri.
* Constructor de copiere.
* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (metodele set() si get()).
* De definit metoda de afișare a tuturor câmpurilor clasei.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De definit metoda statică de compareție a potențialului intelectual a doi balauri folosind media aritmetica a coeficienților de intelect al tuturor capetelor unui balaur.
* De definit metoda de lupta a doi balauri unde fiecare din ei aleator sau după o careva regulă pierde cîte un cap.

În funcția **main()** de creat diferiți balauri folosind toți constructorii descriși anterior. De creat un vector dinamic de balauri. De inițiat acest vetor pe părți folosind toate trei tipuri de constructori definiți. După crearea fiecărui nou obiect al clasei ”Balaur” de afișat toate câmpurile ”exemplarului” folosind metoda corespunzătoare din clasă. De comparat coeficienții intelectuali a cîtorva perechi de balauri. De afișat rezultatele comparării. Angajați cîteva perechi de balauri în lupte și apoi rezultatele luptelor(adică schimbările în datele fiecărui) deasemenea de afișat la ecran. În ciclu de afișat toată informația despre fiecare balaur din vector. În ultimul rînd de afișat numărul de balauri născuți.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

Dupa aceea - de spus “**URRRRAAAA ! ! !**” si de prezentat prima lucrare de laborator la profesor. :)

Și nicicînd mai mult nu de avut de furcă cu fiare foc-răsuflătoare.

**9 (рус)**

Создать класс “**Транспорт**” с несколькими полями.

Обязательные поля:

- целочисленное поле, хранящее общее число возможных пассажиров;

- динамический вектор, в котором хранится вес каждого пассажира

- статическое целочисленное поле, в котором будет храниться число созданных объектов класса.

Необязательные поля: марка, вид (самолет, троллейбус, маршрутка), максимальная скорость, возраст (период работы) и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию вычисления суммарной грузоподъемности (суммы максимальных весов всех пассажиров) текущего объекта класса Транспорт.
* Описать функцию сравнения двух видов транспорта (двух экземпляров класса: текущего и принятого как параметр) по количеству пассажиров
* Описать статическую функцию сравнения двух объектов класса (получаемых как параметры) по общей грузоподъемности (сумме весов всех пассажиров).

В функции **main()** понасоздавать разных видов общественного транспорта, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Транспорт**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор объектов класса. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом элементе (виде транспорта) из вектора. Сравнить вместимость (число пассажиров) нескольких пар транспортных средств соответствующими функциями класса. Результаты сравнения вывести на экран. Сравнить грузоподъемность нескольких пар транспортных средств соответствующими функциями класса. Результаты сравнения вывести на экран. Последней строкой вывести число созданных объектов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**9 (rom)**

De creat clasa “**Transport**” cu câteva câmpuri.

Câmpurile obligatorii:

- un cîmp de tip intreg (int), pentru păstrarea numărul de pasageri ai transportului;

- un vector dinamic, în care se păstrează informaţia despre greutatea fiecatui pasager;

- un câmp de tip intreg, static, se păstrează numărul de obiecte create ale clasei;

Câmpuri opţionale: tipul (avion, autocar etc), marca, viteza, virsta =perioda de funţionare a transportului.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (metodele set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre Transport, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda, care calculează greutate totală a pasagerilor unui Transport.
* De creat metoda, ce compară două Transporturi (două instanţe (obiecte) a clasei : obiectului curent cu obiectul primit ca parametru) în baza numărului de pasageri.
* De creat metoda statică, ce compară doi transporturi (două instanţe (obiecte) a clasei – ambele obiecte primim ca parametrii) după greutatea sumară a tuturor pasagerilor.

În metoda **main()** de creat cîteva Transporturi, utilizând toţi constructorii declaraţi. După creerea fiecărei instantze a clasei “Transport”, de afişat la ecran câmpurile cu ajutorul metodei respective. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia a tuturor transporturi în vector. De comparat câteva perechi de Transporturi în baza numărului de pasageri şi câteva perechi - în baza greutătii sumare a pasagerilor cu ajutorul metodei respective. Rezultatele comparării de afişat la ecran. De calculat greutatea medie a pasagerilor unui Transport. Rezultatele de afişat la ecran.

De afişat la ecran numarul general de Transporturi creaţi, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRAAAAAA !!!**” şi de prezentat primul laborator profesorului. :)

**10 (рус)**

Создать класс “**Звезда**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее общее число планет этой Звезды;

- динамический вектор (тип float), в котором хранится масса каждой планеты

- динамический вектор (тип String), в котором хранится название каждой планеты

- масса (тип double) Звезды

- название (тип String) Звезды.

- статическое целочисленное поле, в котором будет храниться число созданных звезд.

Необязательные поля: сила свечения, диаметр и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию подсчета общей суммы масс всех планет звезды.
* Описать функцию сравнения двух Звезд (текущего объекта класса и объектом, принятым как параметр) по суммарной массе сопровождающих планет каждой из них.
* Описать статическую функцию “черная дыра” (принимающую как параметры 2 объекта класса «Звезда»), которая добавляет к массе более тяжелой Звезды - массу более легкой, а также массу всех планет обеих звезд, после чего обнуляем массу более легкой Звезду и массы планет обоих звезд (более тяжелая звезда “затягивает” их массу на себя саму).

В функции **main()** понасоздавать Звезды, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Звезда**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор звезд (галактику). Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждой звезде из вектора. Сравнить несколько пар Звезд по массе их планет соответствующими функциями класса (результаты вывести). Вызвать функцию “черная дыра” для пары звезд. Новые характеристики звезды -“черной дыры” вывести на экран. Атакже вывести имя звезды, которую «засосало» в черную дыру. Вывести число созданных звезд.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**10 (rom)**

De creat clasa “**Stea**” cu câteva câmpuri.

Câmpurile obligatorii:

- un cîmp de tip intreg (int), pentru păstrarea numărul de planete a stelei;

- un vector dinamic (float), în care se păstrează informaţia despre masa fiecarei planete ;

- un vector dinamic (String), în care se păstrează informaţia despre numele fiecarei planete ;

- masa stelei (double);

- denumirea stelei (String);

- un câmp de tip intreg, static, se păstrează numărul de obiecte create ale clasei;

Câmpuri opţionale: diametru, traectoria, culoarea stelei, e t.c..

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (metodele set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre Stea (toate câmpurile clasăei)
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda care calculează suma maselor planetelor unei stele.
* De creat metoda ce compară doua stele (două instanţe (obiecte) a clasei, comparăm stea curentă cu stea, primită ca parametru) în dependenta de masa sumară a tuturor planetelor.
* De creat metoda statică “gaura neagra”, care primeşte 2 obiecte clasei Stea ca parametrii la intrare. Metoda adaugă la masa stelei mai grele - masa stelei mai uşoare, si înca masele tuturor planetelor ambelor stele, dupa aceea iniţializam cu zero masa stelei mai usoare si masele planetelor ambelor stelelor (Stea cea mai grea «atrage» masa lor la dânsa).

În metoda **main()** de creat nişte stele, utilizând toţi constructorii declaraţi. După crearea fiecărei instanţe a clasei **“Stea”,** de afişat la ecran câmpurile ei cu ajutorul metodei respective. De afişat suma maselor la câteva stele. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia a tuturor stelelor în vector. De comparat câteva perechi de stele în dependenta de masa planetelor ei. Rezultatele comparării de afişat la ecran. De apelat la metoda “gaura neagra” de la o pereche de stele. Caracteristicile noi ale stelei-“gaura neagra” de afişat la ecran. De afişat numele stelei destruse de catre gaura neagra. De afişat numarul de Stele create, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**11 (рус)**

*Создать класс “****Степенной многочлен****” вида*

y = a0x0 + a1x1 + a2x2 + a2x2 + … + an-1xn-1 + anxn

*с несколькими полями:*

*- целочисленное поле, показывающее степень многочлена (число коэффициентов ai);*

*- динамический вектор, в котором хранятся все коэффициенты ai*

*- статическое целочисленное поле, в котором будет храниться число созданных многочленов.*

*Описать три типа конструкторов этого класса (с динамическим выделением памяти):*

*- конструктор по умолчанию (без параметров)*

*- несколько разных конструкторов с параметрами*

*- конструктор копирования*

* *Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).*
* *Описать функцию вывода на экран всех полей класса.*
* *Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.*
* *Описать* функцию, заполняющую все поля вызывающего объекта случайными значениями.
* *Описать функцию сложения двух многочленов (складываются все соответствующие элементы динамических векторов вызывающего многочлена и многочлена, принятого как параметр). Результат хранится в вызывающем многочлене (функция работает только в случае совпадения степеней многочленов).*
* *Описать функцию вычитания двух многочленов (из элементов вектора вызывающего многочлена вычитаются все соответствующие элементы вектора многочлена, принятого как параметр). Результат хранится в вызывающем многочлене (функция работает только в случае совпадения степеней многочленов).*
* *Описать статическую функцию сравнения двух многочленов (сравниваются все соответствующие элементы динамических векторов обоих многочленов-операндов, принятых как параметры). (функция работает только в случае совпадения степеней многочленов).*

*В функции* ***main()*** *понасоздавать многочленов разных степеней, используя все описанные конструкторы. После создания каждого нового экземпляра класса “****степенной многочлен****”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор многочленов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом многочлене из вектора. Над несколькими парами многочленов произвести операции сложения, вычитания и сравнения. После каждой операции результаты выводить на экран. Последней строкой вывести число созданных многочленов.*

***На оценку******10*.** *Добавить ко всему вышеперечисленному:*

*- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;*

*- функцию сохранения всех полей класса в txt файл, имя которого принимается как параметр.*

*- в функции* ***main*** *сохранить поля всех созданных объектов в txt файлы, именами которых являются имена объектов.*

*После чего - сказать “****УРРРРАААА ! ! ! !****” и сдать первую лабораторную работу преподавателю. :)*

**11 (rom)**

*De creat clasa* ***“ Polinom”*** *de tipul*

y = a0x0 + a1x1 + a2x2 + a2x2 + … + an-1xn-1 + anxn

*cu următoarele câmpuri:*

*- un cîmp de tip intreg (int), descrie gradul polinomului (numărul coeficienţilor ai)*

*- un vector dinamic (float), în care se păstrează coeficienţii ai*

*- un câmp de tip intreg, static, se păstrează numărul de obiecte create ale clasei;*

*De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):*

*- constructor standard (fără parametri)*

*- câteva constructori cu parametri*

*- constructor de copiere*

* *De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).*
* *De creat metoda pentru a afişa la ecran toată informaţia despre Polinom (toate câmpurile clasăei)*
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* *De creat metoda de adunare a două polinoame (se adună coeficienţii polinomului care*
* *apelează funcţia cu coeficienţii respectivi ai polinomului obţinut ca parametru). Rezultatul se*
* *păstrează în polinomul apelant. Funcţia lucrează numai în cazul când coincid gradele*
* *polinoamelor.*
* *De creat funcţia de scădere a două polinoame (din coeficienţii polinomului apelant se scad*
* *coeficienţii respectivi ai polinomului obţinut ca parametru). Rezultatul se păstrează în polinomul*
* *apelant. Funcţia lucrează numai în cazul când coincid gradele polinoamelor.*
* *De creat funcţia statică de comparare a două polinoame (se compara coeficienţii la 2 polinoame,*

*obţinute ca parametrii). Funcţia lucrează numai în cazul când coincid gradele polinoamelor.*

*În funcţia* ***main()*** *să se creeze câteva polinoame de diferite grade, utilizând toţi constructorii descrişi. De creat un vector dinamic din obiectele clasei şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informatia a tuturor obiectelor în vector. După crearea fiecărui obiect din clasa „Polinom”, să se afişeze la ecran conţinutul obiectului cu ajutorul funcţiei membre descrise. Pentru câteva polinoame să se apeleze funcţiile de adunare, scădere şi comparare a polinoamelor. După fiecare operaţie să se afişeze rezultatul la ecran. De afişat la ecran numarul general de Polinoame create, folosind câmpul static clasei.*

***Pentru nota 10.*** *Adaugator pentru tot ce este descris mai sus:*

*- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.*

*- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.*

*- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.*

*După ce totul a fost terminat fără greşele — de spus “****URRRRRRRRRAAAAAAAAAA ! ! ! !****” şi de prezentat primul laborator profesorului. :)*

**12 (рус)**

Создать класс “**Кот**” с несколькими полями.

Обязательные поля:

- имя кота;

- целочисленное поле, показывающее общее число выпитых котом капель валерьяны;

- динамический вектор, в котором хранится информация о каждой капле (каждая капля задает одно из 4-х направлений: 1-вверх, 2-вниз, 3-вправо, 4-влево)

- статическое целочисленное поле, в котором будет храниться число созданных котов.

Необязательные поля: возраст, пушистость, цвет, вес, вредность, громкость мява в децибелах …

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию прорисовки траектории движения кота. После каждых 10-ти капель прерывать движение и выводить на экран текст “zzzzz-zzzzz-zzzzz… ”

В функции **main()** понаделать котов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Кот**”, вывести на экран его поля с помощью соответствующего метода класса, а также вызвать функцию траектории движения каждого созданного кота с указанием, какой именно кот ползет :). Создать динамический вектор котов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом коте из вектора. Затем с помощью метода доступа к полям класса подсчитать, сколько валерьянки (в каплях) выпили все созданные коты.

А также вывести на экран имя кота, который выпил валерьянки больше остальных. Если таковых несколько, вывести имена всех рекордсменов :) Последней строкой вывести число созданных котов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

(и больше никогда не поить котов валерьянкой :))

**12 (rom)**

De creat clasa “**Motan**” cu câteva câmpuri.

Câmpurile obligatorii:

- un cîmp de tip String, pentru păstrarea numelui motanului.

- un cîmp de tip intreg (int), care arată cîte picături de valeriană a băut motanul;

- vector dinamic, în care se conţine informaţia despre fiecare picătură (fiecare picătură indică una dintre 4 direcţii: 1- sus, 2- jos, 3- la stînga, 4- la dreapta);

- un câmp de tip intreg, static, se păstrează numărul de obiecte create ale clasei;

Cîmpuri opţionale: vîrsta, culoarea, greutatea, nocivitatea, pufositatea (ciufuleală), volumul acustic al meaunatului în decibeli…

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre Transport, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris o funcţie de desen a traectoriei motanului. După fiecare 10 picături de a întrerupe mişcarea şi de a afişa la ecran textul "zzzzz-zzzzz-zzzzz"

În funcţia **main()** de creat motani utilizînd toţi constructorii descrişi. După crearea fiecărui nou obiect al clasei a "**Motan**”: de a afişa la ecran cîmpurile lui cu ajutorul metodei corespunzătoare din clasă, si de a apela funcţia traectoriei mişcării fiecărui motan creat, cu indicaţia care anume motan se tîrîie. :)

Să se creeze un vector dinamic al motanilor, fiind iniţializat pe bucăţi, utilizînd constructori diferiţi. În ciclu de a afişat informaţia despre fiecare motan din vector. După, cu ajutorul metodei de acces la cîmpurile clasei, de a calcula cîtă valeriană (în picături) a fost băută de toţi motanii creaţi.

Deasemenea de a afişa la ecran numele motanului care a băut cel mai mult valerină. Dacă sunt cîţiva, să se afişeze numele inşilor :). Ultimul rînd să afişeze numărul pufoşilor creaţi.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

(şi mai mult niciodată de a nu îmbăta motanii cu valeriană :))

**13 (рус)**

Создать класс “**Дом**” с несколькими полями.

Обязательные поля:

- целочисленное поле, показывающее число этажей дома;

- целочисленное поле, показывающее число квартир на каждом этаже дома;

- динамическая матрица, в которой хранится информация о каждой квартире (число жителей в каждой квартире)

- статическое целочисленное поле, в котором будет храниться число созданных домов.

Необязательные поля: адрес, номер дома и т.д.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию переселения полностью одного дома в другой (оба дома – входные параметры). Т.е. из каждой квартиры дома, принятого как параметр, все жильцы переселяются в соответствующую квартиру дома, от имени которого вызывается функция. Функция работает только в случае полного совпадения обеих размерностей.
* Описать функцию, подсчитывающую общее количество жильцов в доме.

В функции **main()** понастроить домов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Дом**”, вывести на экран его поля с помощью соответствующего метода класса, а также вызвать функцию подсчета числа жителей для каждого дома. Результаты вывести на экран. Создать динамический вектор домов (улицу). Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом доме из вектора. Несколько домов переселить в другие дома. После чего вывести на экран поля пополненных домов. Затем подсчитать число всех жителей во всех построенных домах. Результат вывести на экран Последней строкой вывести число созданных домов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**13 (rom)**

De creat clasa **„ Bloc\_Locativ”** cu cîteva cîmpuri.

Câmpurile obligatorii:

* un câmp de tip intreg (int) pentru păstrarea numărului de etaje blocului curent;
* un câmp de tip intreg (int) pentru păstrarea numărului de apartamente la fiecare etaj;
* o matricie dinamică în care se păstrează informaţia despre fiecare apartament (numărul de locuitori);
* un câmp de tip intreg, static, in care se păstrează numărul de obiecte create.

Câmpuri opţionale: adresa, numărul propriu, etc.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi modificarea а tuturor câmpurilor clasei (metodele set() si get()).
* De creat metoda ce afişează la ecran toată informaţia despre bloc locativ (toate cîmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcţia statică ce reprezintă strămutarea a locuitorilor unui bloc in altul (ambele blocuri primim ca parametrii la intrare). Astfel inkât din fiecare apartament al blocului primit ca parametru, toţi locuitorii să se strămuta in apartamentele corespunzătoare a blocului din numele căruia este chemată funcţia. Fucţia se execută doar in cazul corezpunderii parametrilor ambelor blocuri de locuit.
* De creat metoda, care calculează numărul total de locuitorii in blocul locativ curent.

În metoda **main()** de construit nişte blocuri locative, utilizînd toţi constructorii descrişi. Dupa crearea fiecarui exemplar al clasei "Bloc\_Locativ" de afişat la ecran câmpurile clasei cu ajutorul metodei respective. De afişat numarul de locuitori a fiecărui bloc. De creat un vector dinamic din obiectele clasei (o stradă) şi de initializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informaţia obiectelor din vector. De strămutat oamenii dintr-un bloc – in altul. După ce de afisat datele schimbate ambelor blocuri locative. De calculat şi de afişat la ecran suma totală a locuitorilor in toate blocuri create. În final de afişat la ecran numarul general automobile create, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**14 (рус)**

Создать класс “**Султан**” с несколькими полями.

Обязательные поля:

- дробное поле, показывающее богатство султана;

- целочисленное поле, показывающее «населенность» гарема нашего султана :), т. е. общее число жен;

- динамический вектор, где хранится коэфициент красоты каждой жены (от 0 до 100)

- динамический вектор, в котором хранится коэфициент вредности каждой жены (от 0 до 100)

- статическое целочисленное поле, в котором будет храниться число созданных султанов.

Необязательные поля: коэфициент устойчивости султана к стрессам, скорость бега, возраст и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа к полям класса, и функции изменения полей класса (set() и get()).
* Описать функцию вывода на экран всей подноготной конкретного султана (всех полей класса).
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать статическую функцию сравнения двух султанов (двух экземпляров класса, принятых как параметры) по количеству жен.
* Описать функцию, вычисляющую коэфициент здоровья султана, обратно пропорциональный среднему арифметическому вредности и прямо пропорциональный среднему арифметическому красоты всех его жен.

В функции **main()** понасоздавать султанов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Султан**”, вывести на экран его поля с помощью соответствующего метода класса. Создать динамический вектор султанов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом султане из вектора. Сравнить парочку султанов по их любвеобильности (числу их жен) соответствующими функциями класса. Результаты сравнения вывести на экран. Найти самого здорового по коэффициенту султана из всех созданных. Подсчитать сумму всех жен всех созданных султанов при помощи функций-методов доступа к полям класса. Результат вывести на экран. Последней строкой вывести число созданных султанов.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**14 (rom)**

De creat clasa “**Sultan**” cu câteva câmpuri.

Câmpurile obligatorii:

- un câmp de tip real (double), care reprezintă bogăţia sultanului.

- un câmp de tip intreg (int), care reprezintă „populaţia” haremului sultanului nostru :), adică numărul soţiilor.

- vectorul dinamic, în care se păstrează coeficientul frumuseţii fiecărei soţii(de la 0 pîna la 100).

- vectorul dinamic, în care se păstrează coeficientul răutăciunii fiecărei soţii(de la 0 pîna la 100).

- câmpul static întreg, în care se va păstra numărul sultanilor creaţi.

Cîmpurile neobligatorii:

Coeficientul rezistenţei al sultanului la stres, viteza alergatului, vîrsta, etc.

De descris trei tipuri de constructori clasei date (cu alocarea dinamică a memoriei):

* constructor implicit(fără parametri)
* cîteva constructori diferiţi cu parametric
* constructor de copiere
* De descris metodele de acces la cîmpurile clasei, şi metodele de modificare a cîmpurilor clasei.
* De descris metoda de afişare la ecran toate informaţia privată a sultanului, adică a tuturor cîmpurilor clasei.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris o metoda statică de comparare a doi sultani (a exemplari ai clasei) dupa numărul de soţii.
* De descris o metodă de aflare a coeficientului de sănătate al sultanului, invers proporţional mediei aritmetice a răutaciunii soţiilor, şi direct proporţional mediei aritmetice a frumuseţii soţiilor.

În metoda **main()** de a crea un teanc de sultani, utilizînd toţi constructorii descrişi. După crearea fiecarui exemplar al clasei „**Sultan**”, de afişat la ecran cîmpurile lui cu ajutorul metodei corespunzătoare a clasei. De creat un vector dinamic cu sultani. De a-l iniţia pe bucăţi, utilizînd constructori diferiţi pe fiecare bucată. De afişat prin ciclu toate informaţia despre fiecare sultan din vector. De comparat cîţiva sultani dupa „potenţialul” de dragoste (numărul de soţii) folosind metodele corespunzătoare ale clasei. Rezultatele de afişat la ecran. De a găsi după coeficient sultanul cel mai sănătos. De a calcula suma tuturor soţiilor ale tuturor sultanilor cu ajutorul metodelor de acces la cîmpurile clasei. Rezultatele de afişat la ecran. În final de afişat numarul sultanilor creaţi.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare..

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**15 (рус)**

Создать класс “**Врач**” с несколькими полями.

Обязательные поля:

- 2 целочисленных поля, хранящие опыт врача и число его пациентов;

- целочисленный динамический вектор, в котором хранится инфа о каждом пациенте (№ полиса)

- 3 строковых поля, хранящих имя, фамилию и специализацию врача.

- статическое целочисленное поле, в котором будет храниться число созданных врачей.

Необязательные поля: сектор, в котором работает врач, число проведенных операций, возраст, пол и т.д…

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран личного дела врача, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию добавления пациента по номеру его полиса. Если такой номер у врача уже записан, вывести на экран сообщение, что номер введен неверно.
* Описать функцию удаления пациента из списка врача по номеру его полиса. Если такой номер у врача не записан, вывести на экран сообщение, что номер введен неверно.
* Описать статическую функцию сравнения двух врачей по популярности, т.е. по числу их пациентов (оба врача принимаются как входные параметры в функцию)

В функции **main()** понаделать врачей, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Врач**”, вывести на экран его поля методом класса. Создать динамический вектор врачей – «больница». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом враче из больницы. Удалить одного пациента из списка какого-нибудь частного врача, затем добавить этого пациента к другому врачу из больницы, после чего вывести личные дела обоих измененных врачей. Сравнить несколько пар врачей по популярности. Подсчитать и распечатать общее число пациентов всех созданных врачей. Последней строкой вывести число созданных коктейлей, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**15 (rom)**

De creat clasa **“Medic”** cu câteva câmpuri.

Câmpurile obligatorii:

- 2 câmpuri de tip întreg (int), pentru păstrarea experientei si nr.pacientilor;

- 3 câmpuri de tip şir de caractere (String) pentru păstrarea numelui medicului, prenumelui si specialitatii medicului;

- un vector dinamic de tip intreg, în care se păstrează informaţia despre fiecare pacient (nr.poliţei)

- o variabilă statică care numara ciţi medici sunt.

Câmpuri opţionale: sectorul, unde lucrează medic, numărul de operaţii, vârsta, sex etc…

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (metodele set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre medic, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda de adăugarea unui pacient (nr.poliţei pacientului – parametru la intrare). Dacă numărul introdus există în lista medicului dat – de afisat, ca numărul poliţei – este incorect.
* De creat metoda de scoaterea unui pacient (nr.poliţei pacientului – parametru la intrare). Dacă numărul introdus nu există în lista medicului dat – de afisat, că numărul poliţei – este incorect.
* De creat metoda statică, ce compară doi medici (două instanţe (obiecte) a clasei – ambele obiecte primim ca parametrii) în baza numărului de pacienţi.

În metoda **main()** de creat nişte medici, utilizând toţi constructorii declaraţi. După crearea fiecărei instanţe a clasei “medic”, de afişat la ecran câmpurile cu ajutorul metodei respective. De creat vector dinamic “spital” din obiectele clasei şi de iniţializat vectorul creat, folosind 3 tipuri de constructor. De afişat la ecran (in ciclu) informaţia a tuturor medicii în spital. De scos un pacient de la un medic un pacient şi de adăugat polita acestui pacient la alt medic. In sfirşit de afişat informaţia finală ale medicii schimbaţi. De comparat câteva perechi de medici în baza numărului de pacienti cu ajutorul metodei respective. De calculat si de afisat la ecran numarul total (sumar) de pacienti a tuturor medicilor creati. De afişat la ecran numarul general de Medicii creaţi, folosind câmpul static clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**16 (рус)**

Создать класс **”Книга”** с несколькими полями.

Обязательные поля:

- дробное поле, хранящее стоимость одного экземпляра копии книги.

- 2 строковых поля, хранящие название книги и имя-фамилию автора

- целочисленное поле, хранящее число копий этой книги

- динамический вектор, хранящий инфу о состоянии потрепанности каждой копии (от 0 до 10)

- статическое целочисленное поле, в котором будет храниться число созданных книг.

Необязательные поля: упитанность книги (число страниц), жанр, цвет обложки, язык и пр.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей инфы о книге, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию подсчета популярности книги (среднее арифметическое изношенности копий)
* Описать функцию подсчета общей стоимости книги (учитывая цену одной копии и число копий)
* Описать функцию, сравнивающую две книги (вызывающую и принятую как параметр) по степени их популярности, т.е. по среднему арифметическому среди коэффициентов износа всех копий книги. Чем выше коэффициент – тем популярнее книга.
* Описать статическую функцию, принимающую как параметры две книги, и считающую на сколько первая книга дороже второй (учитывая число копий книги).

В функции **main()** понапечатать книг, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Книга**”, вывести на экран его поля методом класса. Создать динамический вектор книг – «библиотека». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждой книге из библиотеки. Сравнить несколько пар книг по популярности и по стоимости. Подсчитать и распечатать общую стоимость всех созданных книг. Найти среди всех книг самую популярную, и напечатать название «чемпионки». Последней строкой вывести число созданных книг, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**16 (rom)**

De creat clasa “**Сarte**” cu câteva câmpuri.

Câmpurile obligatorii:

- cîmp cu numere reale, care păstrează preţul exemplarului unei cărţi.

- 2 şiruri de caractere, care păstrează denumirea cărţii şi numele-prenumele autorului.

- сîmp de numere întregi, care indică numărul de copii ai cărţii.

- мector dinamic, care păstrează informaţia despre cît de tare e uzată cartea(de la 0 la 10)

- o variabilă statică care numara ciţi сărţi sunt.

Cîmpuri opţionale : grosimea cărţii(numărul de pagini), gen, culoarea copertei, limbă ş.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre carte (toate câmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De scris o funcţie care va calcula preţul total al cărţii (luînd în consideraţie preţul unei copii şi numărul de copii).
* De scris o funcţie care va calcula cît e de întrebuinţată cartea (media aritmetică nivelului de uzură al copiilor).
* De scris o funcţie care compară 2 cărţi (una fiind de apel şi alta primită ca parametru) după nivelul de întrebuinţare, adică după media aritmetică dintre coeficienţii de uzură a tuturor copiilor cărţii. Cu cît e mai mare coeficientul, cu atît cartea e mai întrebuinţată.
* De scris o funcţie statică, care primeşte ca parametri două cărţi, şi care calculează cu cît prima carte e mai scumpă decît a doua(se ia în consideraţie numărul de copii).

În funcţia **main()** să se creeze cărţi, folosind toţi constructorii descrişi. După crearea fiecărui obiect al clasei „Carte”, de a afişa la ecran cimpurile lui folosind metodă clasei. Să se creeze un vector dinamic cu cărţi -"Biblioteca". Vectorul să fie iniţializat pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia despre fiecare carte din bilbiotecă. Să se compare cîteva perechi de cărţi după întrebuinţare şi după preţ. Să se calculeze şi să se afişeze costul total tuturor cărţilor create. Să se găsească cea mai întrebuinţată carte, şi să se afişeze denumirea "campioanei" Ultimul rînd afişat să conţină numărul cărţilor create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**17 (рус)**

Создать класс **”Топмодель”** с несколькими полями.

Обязательные поля:

- 2 дробных (float) поля для коэффициента спотыкания модели и коэффициента ее вредности.

- строковое поле, хранящее гламурный псевдоним модели.

- дробное поле (double) «кошелек», содержащее размер гонорара модели.

- 4 целых поля: рост, возраст, вес модели и число конкурсов, в которых она участвовала.

- динамический вектор, хранящий инфу о местах, занятых моделью на каждом конкурсе

- статическое целочисленное поле, в котором будет храниться число созданных топмоделей.

Необязательные поля: цвет глаз, IQ, число парней, с кем она одновременно встречается и пр… :)

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию «папарацци» вывода на экран всей личной инфы топмодели, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию «рейтинг» подсчета рейтинга топмодели (среднее арифметическое мест на конкурсах)
* Описать функцию «промоутер». В зависимости от коэффициента вредности, частоты спотыканий на подиуме, рейтинга (считается посредством предыдущей функции) и внешних физических параметров топмодели (рост, возраст, вес), функция изменяет размер гонорара модели.
* Описать статическую функцию «кастинг» (casting), принимающую как параметры двух гламурных красавиц, и считающую, на сколько первая топмодель дороже второй (более высокооплачиваема). Результат сравнения вывести на экран, указывая имена обоих участниц.

В функции **main()** понасоздавать топмоделей, используя все описанные конструкторы. После создания каждой из них, заставить «папарацци» вывести на экран ее личную инфу. Создать динамический вектор топмоделей – «model\_agency». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждой топмодели из агенства. Прогнать несколько пар красавиц через «кастинг». Найти самую популярную топмодель из всех созданных (у кого «рейтинг» максимальный), и для победительницы пересчитать гонорар с помощью «промоутера». Подсчитать и распечатать общую сумму, заплаченную «model\_agency» всем нанятым им топмоделям. Последней строкой вывести число созданных красавиц, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**17 (rom)**

De creat clasa **“Topmodel”** cu câteva cîmpuri.

Câmpurile obligatorii:

- 2 câmpuri cu numere fracţionale (float) pentru păstrarea informaţiei despre impiedicitatea modelei şi coeficientul de nocivitate.

- câmpul de tip string, care va păstra aliasul glamour al modelei.

- câmp cu număr fracţional (double) „Portmoneu”, în care va păstra mărimea venitului modelei.

- 4 câmpuri de numere întregi: înalţimea, vîrsta, greutatea modelei şi numărul de concursuri la care a participat.

- vectorul dinamic, care va păstra informaţia despre locurile ocupate de topmodel în fiecare concurs.

- cîmp static cu număr întreg în care se va păstra numărul modelelor create.

Cîmpuri opţionale: culoarea ochilor, IQ, numărul flăcăilor cu care se întâlneşte concomitent ş.a.m.d... :)

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei)

- constructor standard (fără parametri)

- câţiva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (metodele set() si get()).
* De creat metoda ”paparazzo”, care afişează la ecran toată informaţia despre Topmodel, adică toate câmpurile din această clasă.
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcţia „rating” de calcularea ratingului modelei (media aritmetica locurilor ocupate la concursuri).
* De descris funcţia „promoter”. În dependenţă de coeficientul de nocivitate, frecvenţa împiedicări pe podium, ratingului (se calculează cu ajutorul funcţiei descrise mai sus) şi parametrilor fizici externi ai topmodelului (înălţimea, vârsta, greutatea), funcţia schimbă mărimea venitului modelei.
* De descris o funcţie statică „casting”, care primeşte ca parametri două crăiese a la glamour, şi calculează cu cât prima modela e mai scumpă decât a doua (adică mai bine plătită). Rezultatul comparării să fie afişat la ecran, indicînd numele ambelor participante.

În funcţia **main()** de a crea modele, utilizând toţi constructorii descrişi. După crearea fiecărei modele, de a obliga pe „paparazzi” să afişeze la ecran informaţia personala a modelei.

De a crea un vector dinamic de topmodele – „model\_agency”, iniţializându-l pe părţi, folosind diferite tipuri de constructori. În ciclu de a afişa toată informaţia despre fiecare topmodel a agenţiei. De a pune câteva perechi de „zîne” prin „casting”. De a găsi cel mai popular topmodel dintre toate cele create (cu rating maxim), şi pentru câstigătoare de a recalcula onorariul cu ajutorul „promoterului”. De a calcula şi de a afişa suma totală plătită de „model\_agency” tuturor modelelor angajate.

În ultima linie să se afişeze numărul frumoaselor create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşeli — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**18 (рус)**

Создать класс **”Караван”** с несколькими полями.

Обязательные поля:

- строковое поле, хранящее имя владельца каравана.

- целочисленное поле, в котором хранится число верблюдов в караване.

- динамический вектор «дальнострельность», хранящий инфу о дальности плевка каждого верблюда.

- динамический вектор, хранящий скорость хода каждого верблюда.

-динамический вектор, хранящий информацию о «грузоподъемности» каждого верблюда.

- статическое целочисленное поле, в котором будет храниться число созданных верблюдов.

Необязательные поля: цвета верблюдов, их «скорострельность» :), дальность пути каравана и пр… :)

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

Описать функции доступа и изменения полей класса (методы set() и get()).

* Описать функцию вывода на экран всей инфы о караване, т.е. всех полей класса.
* Описать функцию вычисления скорости движения каравана (скорость самого медлительного верблюда).
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию подсчета общего веса груза каравана (сумму грузоподъемностей всех верблюдов).
* Описать функцию, возвращающую стоимость всего груза каравана (с использованием вызова предыдущей функции). Функция принимает как параметр цену груза за 1 кг.
* Описать функцию «снайпер», которая возвращает номер самого «дальноплюйного» верблюда.
* Описать статическую функцию «кто\_круче», принимающую как параметры два каравана и сравнивает их общую стоимость и скорость. Функция выводит на экран имя владельца более крутого каравана.

В функции **main()** понасоздавать караванов, используя все описанные конструкторы. После создания каждого из них, вывести на экран все его данные. Создать динамический вектор караванов. Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом караване из вектора. Найти и вывести на экран имя владельца каравана из вектора с самым дальноплюйным верблюдом. Сравнить несколько пар караванов на «крутизну». Подсчитать и распечатать общий вес груза, который можно перевезти за один раз, используя все созданные караваны. Последней строкой вывести число созданных караванов, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**18 (rom)**

De creat clasa **“Caravană”** cu câteva cîmpuri.

Câmpurile obligatorii:

- cîmp de tip șir de caractere, în care păstrează numele proprietarului caravanei.

- cîmp de tip număr întreg, în care se păstrează numărul cămilelor din caravană.

- vector dinamic, ”De tragere lunga”, ce păstrează informația despre cît de departe poate scuipa fiecare cămilă.

- vector dinamic, care păstrează viteza fiecărei cămile.

- vector dinamic, ce păstreaza informația despre ”sarcina maximă” cu care poate fi ”împovărată” fiecare cămilă.

- cîmp static de număr întreg, în care se va păstra numărul cămilelor create.

Cîmpuri opţionale: culorile cămilelor, ”rata scuipărilor”, lungimea traseului caravanei, etc…

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei)

- constructor standard (fără parametri)

- câţiva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces la toate câmpurile clasei şi de modificare a lor (metodele set() si get()).
* De creat metoda, care afişează la ecran toată informaţia despre Caravană (toate câmpurile din această clasă).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcția care va calcula viteza deplasării caravanei (viteza celei ami lente cămile).
* De descris funcțа ce calculеază masa totala a încărcăturii caravanei (suma ”însărcinării maxime” tuturor cămilelor).
* De descris o funcțite, care va returna ca valoare costul bunurilor transportate de caravană(folosind apelul funcției precedente). Funcția va primi ca parametru prețul încărcăturii pentru 1 kg.
* De descris funcția ”lunetist”, care va returna numărul de ordine al cămilei care scuipă cel mai departe.
* De descris funcția statica ”mare\_brînză”, care primește ca parametri 2 caravane și le compară dupa prețul total și viteza fiecărei. Funcția afișează la ecran numele proprietarului al caravanei cîștigătoare.

În funcția **main()** de creat caravane, folosind toți constructorii descriși. După crearea fiecărui, de afișat la ecran toate datele despre caravana creată. De creat un vector dinamic de caravane, inițializînd-ul pe bucăți și folosind constructori de tip diferit. Prin ciclu de a afișat toată informația despre fiecare caravană din vector. De găsit și de afișat numele proprietarului caravanei din vector cu cea mai lung-scuiptrăgătoare cămilă. De comparat căteva perechi de caravane aflînd care e ”mai beton”. De calculat și de afișat masa totală care poate fi transportată de toate caravanele create. În ultimul rînd de afișat numărul caravanelor create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşeli — de spus “**URRRRRRRRRAAAAAAAAAA ! ! ! !**” şi de prezentat primul laborator profesorului. :)

**19 (рус)**

Создать класс **”Препод”** с несколькими полями.

Обязательные поля:

- 2 строковых поля, хранящях имя преподавателя и название предмета, который он ведет.

- дробное поле (double), хранящее злобность преподавателя, т.е. процент его задолженников (от 0 до 100)

- дробное поле (float), хранящее максимальную громкость криков разозленного препода в Децибелах.

- целочисленно поле (short), хранящее число групп, где ведет преподаватель

- динамический вектор (long), хранящий инфу о числе студентов в каждой группе преподавателя.

- статическое целочисленное поле, в котором будет храниться число созданных преподов.

Необязательные поля: стаж, пол, средний балл оценок, выставляемых преподавателем и пр.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей «подноготной» препода, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию, возвращающую общее числа несчастных, обучаемых преподом (сумма эл-тов вектора)
* Описать функцию, сравнивающую двух преподавателей (текущего и принятого как параметр) по его «популярности» (по общему числу студентов, которые у него учатся). Имя и предмет наипопулярнейшего из этой двоки преподов вывести на экран.
* Описать статическую функцию «стресс студента», принимающую как параметры двух преподов, и считающую, насколько первый преподаватель лучше ( или хуже ) влияет на психику студентов, учитывая его злобность и громкость его криков в децибелах. Результат сравнения вывести на экран.

В функции **main()** понасоздавать преподов, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Препод**”, вывести на экран его поля методом класса. Создать динамический вектор преподов – «**вуз**». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом чудике из вуза. Сравнить несколько пар преподов по популярности и по состоянию психики их студентов. Найти и распечатать имена всех преподавателей, ведущих тот же предмет, что и первый созданный. Последней строкой вывести число созданных преподов, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРРАААА ! ! ! !**” и сдать первую лабораторную работу преподавателю. :)

**19 (rom)**

De creat clasa “**Prof**” cu câteva câmpuri.

Câmpurile obligatorii:

- 2 cîmpuri de tip șir de caractere, care păstrează numele profesorului și denumirea obiectului condus de el.

- cîmp fracțional (double), care păstrează răutăciunea profesorului (procentul restanțierilor lui) (de la 0 la 100)

- cîmp fracțional (float), care păstrează intensitatea maximă în decibeli a țipătului profesorului enervat.

- cîmp întreg (short), care păstrează numărul de grupe unde predă profesorul.

- vector dinamic (long), care păstrează informația despre numărul studenților în fiecare grupă.

- cîmp intreg static, în care se va păstra numărul profesorilor creați.

Cîmpuri neobligatorii: stagiul, sexul, media notelor puse de profesor s.a.m.d

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

De creat metodele pentru acces şi de modificare a campurilor clasei (set() si get()).

* De creat metoda pentru a afişa la ecran toată informaţia despre profesor (toate câmpurile clasei).
* De descris funcția, care returnează numărul total al sărmanilor, care au nimerit la acest prof (suma elementelor vectorului).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcția, care compară doi profesori (pe cel curent și cel primit ca parametru) dupa ”popularitate” (după numărul total de stundenți, care învață la el). De afișat numele și obiectul celui mai popular prof dintre cei comparați.
* De descris o funcție statică ”stresul studentului”, care va primi ca parametri doi profesori, și care calculează cu cît primul profesor influnențează mai bine (sau mai rău) starea de stres a studentului, luînd în considerație răutăciunea și strigăciunea lui în decibeli. Rezultatul comparării de afișat la ecran.

În funcția **main()** de creat nişte profi, folosind toți constructorii descriși. Dupa crearea fiecărui exemplar nou al clasei ”**Prof**”, de afișat la ecran valorile cîmpurilor lui printr-o metodă a clasei de afișare. De creat vector dinamic al profilor – ”univer”. Vectorul să fie iniţializat pe bucăţi, folosind constructori diferiţi. În ciclu de a afişa toată informaţia despre fiecare ciudat din univer. De comparat cîțiva profi după popularitate și dupa nivelul de stres al studenților. De găsit și de afișat numele tuturor profilor care duc același obiect ca și primul prof creat. Ultimul rînd afişat să conţină numărul profilor creați, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRAAAAAAA ! ! !**” şi de prezentat primul laborator profesorului. :)

**20 (рус)**

Создать класс **”Ювелирное украшение”** с несколькими полями.

Обязательные поля:

- строковое поле, хранящее вид украшения (браслет, колье, др)

- 2 целочисленных поля (byte, short), хранящие число металлов и число камней в украшении

- 3 динамических вектора (float, double, double), хранящие вес каждого металла, число карат камней и их стоимость соответственно

- статическое целочисленное поле, в котором будет храниться число созданных (купленных) украшений.

Необязательные поля: уникальный номер, имя владельца, и пр.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей инфы об украшении, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию, возвращающую общую стоимость драгоценностей в украшении (сумма эл-тов вектора)
* Описать функцию, возвращающую значение (в каратах) самого большого камня в украшении.
* Описать функцию, возвращающую общий вес всех металлов, используемых в украшении
* Описать функцию, сравнивающую два украшения (текущего и принятого как параметр) по общей стоимости его камней. Вид и найденную стоимость обоих украшений (в сравнении) вывести на экран.
* Описать статическую функцию, принимающую как параметры два украшения, и считающую, насколько первое украшение тяжелее (или легче) второго (по металлам). Результат сравнения вывести на экран.

В функции **main()** накупить украшений, используя все описанные конструкторы. После создания каждого нового экземпляра класса “**Ювелирное украшение**”, вывести на экран его поля методом класса. Создать динамический вектор украшений – «**клад**». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом чуде из клада. Сравнить несколько пар украшений по цене их камней и по весу их металлов. Найти и распечатать вид украшения, в котором использован самый крупный камень (в каратах). Последней строкой вывести число созданных украшений, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРААА ! ! !**” и сдать первую сверкающую лабораторную работу преподавателю. :)

**20 (rom)**

De creat clasa “**Bijuterie**” cu câteva câmpuri.

Cîmpurile obligatorii:

- cîmp de tip șir de caractere, care păstrează tipul bijuteriei(brățară, colier, ș.a).

- 2 cîmpuri întregi (byte, short), care păstrează numărul de metale și numărul pietrelor în bijuterie.

- 3 cîmpuri dinamice (float, double, double), care păstrează greutatea fiecărui metal, cîte carate au pietrele și respectiv valorarea lor.

- cîmp întreg static, in care se va păstra numărul bijuteriilor create.

Cîmpurile neobligatorii: numărul unic, numele proprietarului ș.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi de modificare a campurilor clasei (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia despre bijuterie (toate câmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcția care returnează prețul total al giuvaerelor din bijuterie(suma elementelor din vector).
* De descris funcția care returnează valoarea (în carate) a celei mai mari pietre din bijuterie.
* De descris funcția care returnează greutatea totală a tuturor metalelor folosite în bijuterie.
* De descris funcția care compară două bijuterii (cea curentă cu cea primită ca parametru) după prețul total al pietrelor. De afișat la ecran tipul și prețul găsit ale ambele bijuterii (din comparație).
* De descris funcția care primește ca parametri două bijuterii, și care calculează, cu cît prima e mai grea(mai ușoară) decît a doua (după metale). Rezultatul comparării de afișat la ecran.

În funcția **main()** de creat nişte bijuterii, folosind toți constructorii descriși. Dupa crearea fiecărui exemplar nou al clasei **”Bijuterie”**, de afișat la ecran valorile cîmpurilor lui printr-o metodă a clasei de afișare. De creat vector dinamic al bijuteriilor –”comoară”. De iniţializat vectorul pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia despre fiecare bijuterie din comoară. De comparat cîteva perechi de bijuterii după prețul pietrelor și după greutatea metalelor. De găsit și de afișat tipul bijuteriei, în care se folosește cea mai mare piatră (în carate). În ultimul rînd de afişat numărul bijuteriilor create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşele — de spus “**URRRRRRAAAAAAA ! ! !**” şi de prezentat primul laborator profesorului. :)

**21 (рус)**

Создать класс **”Комп”** с несколькими полями.

Обязательные поля:

- строковое поле, хранящее имя владельца компа

- целочисленное поле (byte), хранящее число деталей (составляющих) компа

- 4 динамических вектора (String, float, float, double), хранящие название (модель, изговтовитель), вес каждой детали, ее год выпуска и стоимость соответственно

- статическое целочисленное поле, в котором будет храниться число созданых (собраных) компов.

Необязательные поля: уникальный номер, цвет, мощность, ОС и пр.

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей технической инфы о компе, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию, возвращающую общую стоимость компа (сумма эл-тов вектора стоимостей)
* Описать функцию, возвращающую общий вес компа (сумма эл-тов вектора весов).
* Описать функцию, возвращающую средний возраст компа (среднее арифметическое годов выпуска)
* Описать функцию, сравнивающую два компа (текущего и принятого как параметр) по актуальности (по среднему году выпуска). Старейшего апгрейдить функциями set\_\*(). Данные измененного компа вывести на экран до и после изменения.
* Описать статическую функцию, принимающую как параметры два компа, и считающую, насколько один тяжелее (или легче) второго (по общему весу компов). Результат сравнения вывести на экран.

В функции **main()** насобирать компов, используя все описанные конструкторы. После создания каждого нового экземпляра класса **“Комп”**, вывести на экран его поля методом класса. Создать динамический вектор компов – «**сеть**». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом компе из сети. Сравнить несколько пар компов по актуальности и по весу. Найти и распечатать общую стоимость и владельца самого дорогого компа из всех созданных. Последней строкой вывести число созданных украшений, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРААА ! ! !**” и сдать первую сверкающую лабораторную работу преподавателю. :)

**21 (rom)**

De creat clasa “**Calculator**” cu câteva câmpuri.

Câmpurile obligatorii:

- câmp de tip șir de caractere, care păstrează numele proprietarului

- câmpuri întreg (byte), care păstrează numărul de detalii (componentelor) calculatorului.

- 4 câmpuri dinamice (String, float, float, double), care păstrează denumirea (model, producător), greutatea fiecărei componente, anul fabricării și respectiv prețul.

câmp întreg static, in care se va păstra numărul calculatoarelor create(asamblate).

Câmpurile neobligatorii: ID-ul, culoarea, puterea, SO ș.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi de modificare a campurilor clasei (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia tehnică despre calculator (toate câmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De descris funcția care returnează greutatea totală a calculatorului (suma elementelor vectorului greutății).
* De descris funcția care returnează prețul total al calculatorului (suma elementelor vectorului prețului).
* De descris funcția care returnează vechimea medie a calculatorului (media aritmetica anilor de fabricare a componentelor).
* De descris funcția care compară două calculatoare (cel curent cu cel primit ca parametru) dupa actualitate(după vechime). Celui mai vechi de făcut upgrade cu ajutorul funcțiilor set(). De afișat datele calculatorului modernizat înnainte și după upgrade.
* De descris funcția statica care primește ca parametri două calculatoare, și care calculează cu cît unul e mai greu (mai ușor) ca al doilea( dupa greutatea totală). De afișat la ecran rezultatele comparării

În funcția **main()** de asamblat niște calculatoare, folosind toți constructorii descriși. După crearea fiecărui exemplar nou al clasei **”Calculator”**, de afișat la ecran valorile câmpurilor lui printr-o metodă a clasei de afișare. De creat vector dinamic al calculatoarelor – ”rețea”. De iniţializat vectorul pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia despre fiecare calculator din rețea. De comparat câteva perechi de calculatoare după actualitate și după greutate. De găsit și de afișat prețul total și numele proprietarului celui mai scump calculator din cele create. În ultimul rând de afişat numărul calculatoarelor create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşeli — de spus “**URRRRRRAAAAAAA ! ! !**” şi de prezentat profesorului primul laborator. :)

**22 (rom)**

De creat clasa “**man\_for\_rent**” cu câteva câmpuri.

Câmpurile obligatorii:

- un câmp de tip caracter (char), pentru păstrarea numelui barbatului;

- un câmp de tip întreg (int), pentru păstrarea virstei barbatului;

- un câmp de tip şir de caractere (string) domeniului de activitate;

- un vector dinamic, în care se păstrează informaţia despre preturile allocate pentru fiecare comanda efectuata de barbat.

- o variabilă statică care va calcula cite obiecte au fost create.

Câmpuri opţionale: temperamental barbatului(extrovertit,introvertit),coeficientul aspectului fizic al barbatului(de la 0 pina la 10 puncte),tipul barbatului(brunet,blond,roscat,de culoare etc.),experienta si numarul clientilor(comenzi efectuate).ș.a.

De creat trei tipuri de constructori pentru această clasă (cu alocare dinamică a memoriei):

- constructor standard (implicit - fără parametri)

- câteva constructori cu parametri

- constructor de copiere

* De creat metodele pentru acces şi de modificare a campurilor clasei (set() si get()).
* De creat metoda pentru a afişa la ecran toată informaţia tehnică despre barbat (toate câmpurile clasei).
* De descries metoda, care completeaza toate campurile obiectului cu valori citite de la tastatura.
* De descries metoda, care completeaza toate campurile obiectului cu valori aleatoare.
* De creat metoda care calculează numărul mediu al preturilor comenzilor efectuate.
* De descris funcția care returnează prețul total al calculatorului (suma elementelor vectorului prețului).
* De descris funcția care returnează vechimea medie a calculatorului (media aritmetica anilor de fabricare a componentelor).
* De descris funcția statica, ce compară doi barbati (două instanţe (obiecte) a clasei) în baza valorii medii a preturilor tuturor comenzilor.

În funcția **main()** de asamblat niște barbati, folosind toți constructorii descriși. După crearea fiecărui exemplar nou al clasei, de afișat la ecran valorile câmpurilor lui printr-o metodă a clasei de afișare. De creat vector dinamic de obiecte – ”agentie”. De iniţializat vectorul pe bucăţi, folosind constructori diferiţi. În ciclu de afişat toată informaţia despre fiecare barbat din agentie. De comparat câteva perechi de barbati cu metoda statica. De găsit și de afișat numele barbatului cu cel mai mare numar de comenzi in vector. În ultimul rând de afişat numărul de obiecte create, folosind variabila statică a clasei.

**Pentru nota 10.** Adaugator pentru tot ce este descris mai sus:

- un constructor care primeşte ca parametru denumirea fişierului textual (String), de unde se incarcă valori pentru câmpurile obiectului creat.

- o funcţie care înscrie toate cîmpurile clasei în fişier, numele fisierului se indică ca parametru la intrare.

- În funcţia main pentru toate obiectele create de salvat datele în fişiere, denumirile cărora se preiau de la numele obiectelor.

După ce totul a fost terminat fără greşeli — de spus “**URRRRRRAAAAAAA ! ! !**” şi de prezentat profesorului primul laborator. :)

**2\* (рус)**

Создать класс **”Персонаж”** с несколькими полями.

Обязательные поля:

- строковое поле, хранящее имя игрового персонажа

- целочисленное поле (byte), хранящее число жизней игрового персонажа (XP)

- целочисленное поле (short), хранящее количество вооружения игрового персонажа

- 2 динамических вектора (String, float), хранящие название каждого оружия (лук, топор, меч, …) и урон, наносимый каждым из оружий

- целочисленное поле, хранящее число битв, проведенных игровым персонажем

- константное статическое целочисленное поле, хранящее максимально возможное число жизней любого персонажа

- статическое целочисленное поле, в котором будет храниться число созданых (собраных) компов.

*Необязательные поля: расса, пол, уровень, очки опыта и пр.*

Описать три типа конструкторов этого класса (с динамическим выделением памяти):

- конструктор по умолчанию (без параметров)

- несколько разных конструкторов с параметрами

- конструктор копирования

* Описать функции доступа и изменения полей класса (методы set() и get()).
* Описать функцию вывода на экран всей технической инфы о компе, т.е. всех полей класса.
* Описать функцию, заполняющую все поля объекта значениями, считанными с клавиатуры.
* Описать функцию, заполняющую все поля вызывающего объекта случайными значениями.
* Описать функцию, возвращающую суммарный урон всех оружий (сумма всех уронов из вектора)
* Описать функцию, возвращающую название самого смертоносного оружия вызывающего объекта-персонажа. (с промежуточной функцией нахождения номера самого смертоносного оружия)
* Описать функцию «излечение», восстанавливающую здоровье персонажа до максимума.
* Описать функцию, сравнивающую двух персонажей (текущего и принятого как параметр) по экипированности (числу оружия).
* Описать статическую функцию битвы двух игровых персонажей, принимающую как параметры два объекта, и считающую, у кого больше останется ХР после удара самым разрушительным оружием противника. Функция уменьшает ХР обоих противников, увеличивает число проведенных ими битв и возвращает победителя.

В функции **main()** насоздавать персонажей, используя все описанные конструкторы. После создания каждого нового экземпляра класса **“Персонаж”**, вывести на экран его поля методом класса. Создать динамический вектор персонажей – «**команда**». Инициализировать его по частям, используя разные типы конструкторов. В цикле вывести всю инфу о каждом персонаже из команды. Сравнить несколько пар персонажей по вооруженности. Самого вооруженного излечить. Найти и распечатать общую разрушительную силу всех созданных персонажей. Заставить сражаться несколько пар персонажей. Вывести на экран имя победителя каждой битвы. Последней строкой вывести число созданных персонажей, используя статическую переменную класса.

**На оценку** **10.** Добавить ко всему вышеперечисленному:

- конструктор, принимающий как параметр имя текстового файла (String), откуда берутся данные для полей создаваемого объекта;

- функцию сохранения всех полей класса в текстовый файл, имя которого принимается как параметр.

- в функции **main** сохранить данные всех созданных объектов в текстовые файлы, именами которых являются имена объектов.

После чего - сказать “**УРРРААА ! ! !**” и сдать первую сверкающую лабораторную работу преподавателю. :)