Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Зоопарк

Автор: Стела
Допълнителни изисквания/бележки важни за проекта: design pattern, RTTI, typeid (inheritance-of)
Връзки с други предмети: биология и география

Описание

Собствениците на местен зоопарк са ви възложили следната интересна задача – направете система за управление на техния бизнес. От системата се изисква следната функционалност:

1. Настаняване на ново животно:
   - При избиране на тази опция, трябва да излезе номериран списък с видове животни и да се изчака коректно въвеждане на номер или команда за отказ. Ако няма условия за настаняването му, да се изведе подходящо съобщение.
2. Хранене на животните:
   - Ако в склада няма достатъчно храна, да се изведе подходящо съобщение.
3. Попълване на склада:
   - Поръчка на ново зареждане с храна.
4. Строеж на нова клетка:
   - При избиране на тази опция, трябва да се изведе номериран списък с възможни клетки и да се изчака коректно въвеждане на номер или команда за отказ.
5. Извеждане на информация за животно:
   - При избиране на тази опция, трябва да излезе номериран списък с видове животни и да се изчака коректно въвеждане на номер.
6. Извеждане на информация за клетка:
   - При избиране на тази опция, трябва да излезе номериран списък с възможни клетки и да се изчака коректно въвеждане на номер.

Системата трябва да чака коректен потребителски вход или коректна команда за изход, както и да извежда въвеждащ и/или пояснителен текст на необходимите места.

Зоопаркът приема различни видове животни – птици, хищници, тревопасни, влечуги и риби. За собствениците е важно да знаят за всяко животно дали е хищник или тревопасно

(за да го хранят с правилната храна), както и към коя група принадлежи, за да го настанят в правилната клетка. Във всяка клетка може да се разполага само една група животни, като хищници и тревопасни не се настаняват заедно.

Собствениците са си изготвили следния списък с възможни животни[1], разделени по групи:

- кафява мечка (горски животни)
- вълк (горски животни)
- дива коза (горски животни)
- сова (горски животни)
- орел (горски животни)
- лъв (големи котки)
- тигър (големи котки)
- рис (големи котки)
- лемур (животни от джунглата)
- тукан (животни от джунглата)
- папагал (животни от джунглата)
- жираф (животни от саваната)
- слон (животни от саваната)
- зебра (животни от савана)
- полярна мечка (полярни животни)
- пингвин (полярни животни)
- тюлен (полярни животни)
- делфин (морски животни)
- акула (морски животни)
- риба клоун (морски животни)
- питон (влечуги)
- варан (влечуги)
- хамелеон (влечуги)

Информацията за него съдържа наименованието и групата му, вида му (бозайник, птица, влечуго, риба), колко и каква (месо, риба, растения) храна яде, какъв е звукът, който издава.

Клетките за животни имат много важна роля – стараят се да пресъздадат възможно най-близки до естествените условия на живот. Съответно всеки тип клетки има следните характеристики: влажност (100% за воден басейн), растителност в проценти, вид на терена (горист, скалист, пустинен, воден), локация в географска ширина (не важи за аквариум и терариум).

Клетките могат да бъдат няколко вида[2]:

- гора
- савана
- джунгла
- полярна зона
- аквариум
- терариум

Информацията за всяка клетка съдържа наименованието ѝ и посочените по-горе характеристики.

Складът съдържа запасите от месо, риба и растения на зоопарка. Едно зареждане е равно точно на това количество храна, което в момента наличните животни изяждат за едно хранене.

---

[1] списъкът е примерен – може да се използват други видове/групи животни, стига да е запазена структурата и функционалността.

[2] списъкът е примерен – варира според избора на разделение на животните
подсказка: приемаме, че големите котки живеят в саваната

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Шахмат

- Автор: Стела
- Допълнителни бележки важни за проекта: познания по шахмат, работа с файлове
- Описание:

Колегите от курса по изкуствен интелект имат за задача да обучат невронна мрежа да играе шах. Прекрасно! Но, понеже времето изобщо не им стига, нямат разработена игра на шах. И тук вие идвате на помощ.
Напишете игра на шахмат. За целта ще се нуждаете от шахматна дъска (8x8, бели и черни квадратчета), 32 фигури (16 собствени и 16 противникови) и кратки правила:
Пешка – 8 броя. Разположени на предна линия пред останалите фигури. Придвижват се с едно квадратче напред, а при първи ход имат възможност за 2. Атакуват фигура, премествайки се с едно квадратче по диагонал. Ако достигнат противоположния край на дъската, могат да бъдат заменени за коя да е друга фигура.
Топ – 2 броя. Разположени са в двата края на задната линия фигури. Движат се по вертикал и хоризонтал до достигане на фигура. Атакуват фигура по същия начин.
Кон – 2 броя. Разположени са на задната линия фигури, вътрешно от двата топа. Движат се Г-образно – 2 квадратчета по вертикал/хоризонтал и още едно по хоризонтал/вертикал. Единствени могат да “прескачат” фигури. Атакуват фигура по същия начин.
Офицер – 2 броя. Разположени са на задната линия фигури, вътрешно от двата коня. Движат се по диагоналите – един по белия и един по черния, до достигане на фигура. Атакуват фигура по същия начин.
Царица – 1 брой. Разположена на задната линия фигури, при белите (първи играч) е до левия офицер, а при черните – срещу нея, до десния. Придвижва се по вертикал, хоризонтал и диагонал до достигане на фигура. Атакува фигура по същия начин.
Цар – 1 брой. Разположен на задната линия фигури, при белите (първи играч) е до десния офицер, а при черните – срещу него, до левия. Придижва се с едно квадратче по вертикал, хоризонтал и диагонал. Атакува фигура по същия начин. Не може да бъде на едно квадратче разстояние от противниковия цар. Ако е заплашен от противникова фигура, тогава е обявен шах. Ако е атакуван от противникова фигура без възможност да премахне заплахата, се обявява мат и играта приключва.

На студентите от ФМИ обаче им стана много интересно да пробват играта на колегите си. Затова трябва да се измисли начин да се пази класиране между играчите. В началото на всяка игра се пита за имената на двама играчи и се отваря тяхното класиране. То се пази в текстови файлове с име, съвпадащо с името на играча, във формат:

<противник>, <резултат>

Резултата е 1, ако играчът е спечелил, и 0 – ако е изгубил играта. Резултатът на новата игра се записва на нов ред в края на файла.

Играта е конзолна, което значи, че на всеки ход в играта трябва да се визуализира игралното поле, да се изписва името на играча, който е на ход и да се чака валидна команда за преместване на фигура във формат:

<код на фигурата> <колона><ред>

Кодовете са съответно:

p – пешка
r – топ
b – офицер
k – кон
q – царица
k – цар

Колоните са номерирани с малки латински букви от a до h отляво надясно, а редовете – от 1 до 8 отдолу нагоре, гледано от страната на белите.

Да се извеждат подходящи съобщения и пояснения по време на играта.

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Недетерминиран краен автомат

- Автор : Ангел
- Допълнителни бележки важни за проекта: Важи само за колегите, изкарали курса по ЕАИ.
- Описание

Да се реализира програма, която поддържа операции с недетерминиран краен автомат над азбука, състояща латински букви и цифри. Автоматът да има следните функции:

1. Детерминира автомата (само ако е недетерминиран)
2. Минимализира автомата.
3. Прави автомата тотален.
4. Проверява дадена дума дали се разпознава от автомата.
5. Проверява дали езикът на автомата е празен.

Да се реализира и следната функционалност:

1. Функция, която прави обединение на 2 автомата.
2. Функция, която прави конкатенация на 2 автомата.
3. Функция, която прави звезда на клини на автомат.
4. Функция, която приема регулярен израз и връща автомат, който разпознава езика на този регулярен израз.
5. Функция, която приема автомат и връща регулярен израз за езика на автомата.

Пример за работа с проекта:

```
Automation  t("a.(b.a + b)*");
t.isEmptyLanguage(); //false
t.accept("aba"); //true;
t.accept("baa"); false;

Automation t2("b.(a)*");
Automation t3 = Union(t1,t2);
t3.accept("aba"); //true;
t3.accept("baa"); // true;
```

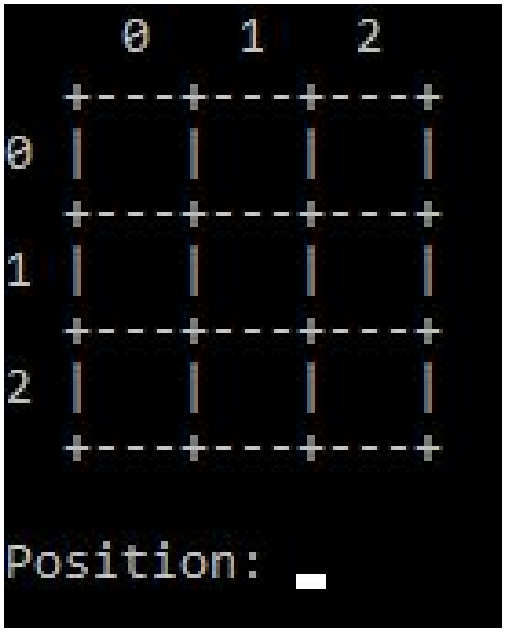
Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Морски шах

- Автор: Петър Събев
- Допълнителни бележки важни за проекта - познания по морски шах, работа с файлове
- Описание

Да се напише реализация на играта Морски шах. Потребителският интерфейс на играта трябва да е изцяло в текстов режим с изрисуване на дъската и фигурите на стандартния изход в конзолата. Играта трябва да поддържа възможност за multiplayer на една и съща машина. При стартиране на играта се определя фигурата на играчът, който ще бъде първи на ход. На стандартния изход трябва да се изведе следното съобщение **Enter X or O:** . След, което се очаква от първия играч да въведе избрания символ - възможните стойности са: **X** и **O** (при въвеждане на символи различни от посочените, на стандартния изход трябва да се изведе съобщението: **Only X or O are allowed.** И отново да се изведе **Enter X or O:** с изчакване за избор на символ - този процес се повтаря до въвеждане на един от разрешените символи). Следва изчистване на стандартния изход и изрисуване на дъската във следния вид:



Следва съобщението: **Position:** и изчакване за въвеждане на позиция на която да бъде поставена фигурата на първия играч. Позицията се задава по следния начин **xy**, където **x** и **y** са цели числа отговарящи на координатите съответно по x и y - например при избран от първия въвеждане на позиция 00 на стандартния изход трябва да се

изчисти и на него да се изрисува наново дъската. Преди всеки ход трябва да се извежда информация за играча на текущия ход, както и неговия символ.
Историята на играта следва да се запазва във файл.

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Динамичен масив и итератори

- Автор: Петър Събев
- Допълнителни бележки важни за проекта - оптимизация, шаблони, итератори и структури.
- Описание

Целта на проекта е да бъдат реализирани шаблонен клас - динамичен масив - Vector, подобен на std::vector и итератор от STL.

Класът Vector трябва да бъде шаблонен и да бъде реализиран в отделен header файл. В допълнение на това трябва да е изпълнено правилото на петте. Да се използва **operator new (или placement new – по желание)** и съответстващите им за освобождаване на паметта.

По отношение на именуването на методите **се спазва интерфейсът на вектор (std::vector) от стандартната библиотека и дефинираното там описание**. Вашият клас трябва да съдържа следните методи:

- **Конструктори, деструктори и оператори за присвояване**;
- **operator[]** - връща псевдоним към елемент по подаден индекс:
- **operator[]** - връща константен псевдоним към елемент по подаден индекс;
- **push_back** - добавя елемент в края на динамичния масив;
- **pop_back** - изтрива елемент в края на динамичния масив;
- **size()** - връща броя елементите;
- **resize(int newSize)** - променя размера на динаичния масив, така че да стане newSize;
- **empty()** - проверява дали има елементи;
- **back()** - връща стойността на елемента в края на Vector;
- **clear()** - изтрива цялото съдържание на Vector;

Наред с горе изброените методи, трябва да бъдат реализирани и още **поне 3 метода по Ваш избор**.

Итератор представлява описание на типове, които могат да бъдат използвани за обхождане на елементите на даден контейнер. Вашата задача е да реализирате поне два вида итератори: единият **RandomAccessIterator**, другият по Ваш избор. Наред с това е необходимо да приложите подходящи тип итератор в реализацията на динамични масив и да предоставите интерфейс за достъп.

Изисква се итераторите да бъдат направени по подобие на тези в STL и да бъдат подходящо приложени.

Да се покаже примерна употреба с написването на смислена програма, създаваща инстанции на вашия динамичен масив с голям брой елементи (поне 1000 елемента) и използваща поне веднъж минимум половината от методите на динамичния масив.

По желание (бонус):

- *Да се използва **placement new**;*
- *Да се подсигури правилно поведение при изключителни ситуации чрез сигнализиране и обработване на изключения;*
- *Опитайте се да минимизирате изискванията към шаблонните типове.*

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

База от данни за оценки на студенти

- Автор: Петър Събев
- Допълнителни бележки важни за проекта - файлове, оптимизация на бази от данни
- Описание

Да се напише система, предоставяща възможност за създаване, записване в постоянната памет (във вид на файл във файловата система) и преглед на записи с оценки на студенти. **Основно изискване е записаната информация да се съхранява за постоянно и да е достъпна след изход от програмата и последващото и стартиране.**

При стартиране програмата очаква на стандартния вход една от следните команди една от тези команди:

1. **create** – след въвеждане на тази команда от потребителя се очаква да въведе информация за оценка на студент по програмиране в следния формат: *FN FirstName LastName Grade*, където:
    a. *FN* е факултетен номер на студент (цяло положително число с максимална стойност);
    b. *FirstName* е първото име на студента (низ);
    c. *LastName* е фамилия на студента (низ);
    d. *Grade* е оценка на студента (цяло положително число в интервала);

    След въвеждане на информацията и натискане на Enter, програмата автоматично записва въведената информация в текстов файл с името **StudentsGrades.db**. В случай, че файлът **StudentsGrades.db** не съществува, програмата трябва да го създаде автоматично и след това да запише информацията в него. В случай, че файлът **StudentsGrades.db** вече съществува, програмата трябва да го отвори и да допълни съдържанието му с въведената информация.

    Всеки запис за студент във файла трябва да бъде отделен на нов ред. Факултетния номер *FN* за всеки запис за студент трябва да е уникален за файла (във файла не трябва да присъстват записи с дублиращи се стойности за *FN*).

    Основно изискване е действията (приложени с описаните по-долу

команди за обновяване и изтриване) да се прилагат върху информация в посочения файл за постоянно и да не се губят след изход от програмата и последващото и стартиране.

След записване на информацията, програмата извежда съобщението „**Record saved!**“ и се връща в начално състояние.

2. **sequentialSearch** – след въвеждане на тази команда от потребителя се очаква да въведе *FN*. След въвеждане на *FN*, програмата извършва последователно търсене във файла и в случай на точно съвпадение с въведения *FN* на стандартния изход се извежда пълната информация за студента във следния формат: *FN FirstName LastName Grade*.

   В случай, че запис за студент с въведения *FN* не е намерен, то на стандартния изход се извежда съобщението: „**Record not found!**“. След извеждане на информацията, програмата се връща в начало състояние.

3. **update** – след въвеждане на тази команда от потребителя се очаква да въведе новата информация за оценка на студент по програмиране в следния формат FN *Grade*. След въвеждане на факултетен номер, оценка и натискане на Enter:
   - В случай на съществуващ запис за студент с посочения *FN*, програмата автоматично обновява въведената оценка *Grade* в файлът с име **StudentsGrades.db**, извежда съобщението „**Record saved!**“ и се връща в начално състояние.
   - В случай, че не съществува запис за студент с посочения *FN* програмата извежда съобщение „**Record not found!**“ и се връща в начално състояние.

4. **delete** – след въвеждане на тази команда от потребителя се очаква да въведе FN. След въвеждане на факултетен номер и натискане на Enter:
   - В случай на съществуващ запис за студент с посочения *FN*, програмата автоматично изтрива записът за студент с въведения *FN* от файлът с име **StudentsGrades.db**, извежда съобщението „**Record deleted!**“ и се връща в начало състояние.
   - В случай, че не съществува запис за студент с посочения *FN* програмата извежда съобщение „**Record not found!**“ и се връща в начално състояние.

- **exit** – след въвеждане на тази команда се излиза от програмата.

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

World of Warcraft

- Автор: Теодор Кънев
- Допълнителни бележки важни за проекта: игра, време за изпълнение
- Описание:

Ще направим версия на играта “World of Warcraft”. За целта трябва да се реализират следните класове:

1. **Hero** - Този клас представлява основните функционалности за героя. Всеки герой има:
    - Name (име) - име, което притежава героят.
    - HP (жизнени точки) - всеки герой започва с 100 такива точки. Щом тези точки станат 0, героят се смята за мъртъв.
    - Strength (сила) - точките, отговарящи за физическите способности на героя.
    - Intellect (интелект) - точките, отговарящи на магическите способности на героя.
    - Level (ниво) - ниво на героя.
    - Attack (атака) - функция, с която всеки играч нанася щети на противника си.
    - Defend (защита) - защита от противниците.
    - LevelUp (вдигане на ниво) - нивата се вдигат автоматично след убиване на 2^(текущото ниво) противника. При вдигане на ниво, максималните жизнени точки на героя се вдигат с 10% и текущите стават новия максимум, а силата и интелекта се повишават по формулата (началната стойност на съответните точки) / 3. Всеки герой започва от 0 ниво.
2. **Monster** - Този клас представлява основните функционалности за противниците. Всяко чудовище има:
    - Name (име) - име на чудовището.
    - HP (жизнени точки) - оставащ живот в него.
    - Strength (сила) - физическата му сила.
    - Intelligence (интелект) - точки за магическите способности.
    - Attack (атака) - функция, с която звяра напада.
    - Defend (защита) - функция, с която се защитава.

Всеки герой може да е един от следните:

1. **Warrior** (Войн) - Разчита на физическата сила. Всеки войн започва с 13 сила и 2 интелект. Всеки войн има точки гняв, които могат да достигнат до 100. Гнева се генерира след всяка атака +2 точки и след всяка защита +3 точки. Войнът се бие така: В началото изразходва цялата си ярост като тя му дава (ярост / 5) процента атака за цялата битка. Атаката, която има се изчислява по формулата (сила) + 0.3*(интелект).
2. **Mage** (Магьосник) - Разчита на магиите. Всеки магьосник започва с 4 сила и 11 интелект и 100 мана. Това са точки, които отговарят за силата на магиите. Атакува по следната формула (интелект + (мана/100)*3). След всяка атака магьосника губи 10 от маната си и започва всяка битка с 100.
3. **Paladin** (Паладин) - Паладините започват с 9 сила и 6 интелект. Те нямат специални точки, но пък всеки трети удар е с 50% по-силен. Всяка атака се пресмята по формулата 0.5*(сила) + 0.5*(интелект).

Всяко чудовище може да е:

1. **Goblin** - Има 10 точки живот и атакува с 3 + 0.1*(интелекта на врага) щета.
2. **Dragonkin** - Има 30 точки живот и атакува с 8 щета. Всяка трета защита намаля щетите, които получава с 100.
3. **Death Knight** - Това е чудовище породено от Goblin и Dragonkin. Редува атаките на двете чудовища, а се защитава като Dragonkin.

Да се имплементира интерактивен режим на играта, който включва:

- Избор на герой;
- Карта, по която са разположени разнообразни противници;
  Когато се опитаме да стъпим върху поле на противник, то героят го атакува и след това, ако противникът е все още сред живите,той отвръща с атака. Ако се опитаме да стъпим на невалидно поле - да се връща пояснително съобщение защо това е “невалиден ход” и да се дава възможност за повторно въвеждане.
- Стандартен изход на актуална информация за картата и последиците от последния ход на играча.
- Запазване на героя в база данни и възможност за стартиране с вече запазен такъв. Извеждане на списък от запазени герои.

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Магазин

- Автор: Теодор Кънев
- Допълнителни бележки важни за проекта - изграждане на системи
- Описание

Реализирайте система на **Магазин**. Системата трябва да разполага със списък от наличните му продукти и списък от продавачи и клиенти. Всеки продукт представлява абстрактен клас, който съдържа уникално име на продукта, цената му и брой артикули в магазина. Всеки продукт може да е един от следните видове: дреха, храна, козметика и техника. Всяка дреха има размер и дължина. Храната има калории, грамаж и съдържание. Козметиката има низ, отговарящ на държава на производство, състав, а всяка техника има тегло и години гаранция.

Трябва ни и клас **Човек**, съдържащ име и години. Всеки човек е или клиент или продавач. Продавачите служат единствено да обслужват клиентите на даден магазин. Клиентът на даден магазин може да извършва различни дейности:

- Да пита за цена на даден продукт.
- Да получи цялостна информация за даден продукт.
- Да купува продукт от магазин.
- Да получи информация за налични продукти в магазина от дадена категория или всички.

Системата на магазина може да има най-много до 100 служителя и 5000 артикула и има следните функционалности:

- Да продава на своите клиенти само налични продукти и при наличие на продавач.
- Да добавя нови продукти.
- Да премахва продукт.
- Да пази всички свои продукти в база данни (не е необходимо тя да се обновява при всяка промяна).
- Да назначава нови продавачи.

Важно е дори при затваряне на програмата базата данни да е коректна и при ново стартиране, да не губим информацията за продуктите в него.

Напишете програма, която да симулира създаването и използването на системата за един такъв магазин.

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Генериране на схема на турнир

- Автор: Михаил Атанасов
- Допълнителни бележки важни за проекта
    - **Да се използват поне 2 pattern-а от шаблоните за дизайн(https://bg.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%B7%D0%B0_%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD).**
    - **Системата да е обектно моделирана посредством ООП парадигмите.**
    - **Да върне коректни данни спрямо зададените входни данни.**

- Описание
  Всички входни данни се получават от база данни под формата на файл.
  Входните данни съдържат:
    1. Списък с отборите (до 32 отбора).
    2. Типа схема (единична елиминация или всеки срещу всеки)
    3. Времетраене на мачовете
    4. Брой на свободните кортове
    5. Дата и час на започване на събитието.
    6. Тип на изходен файл

  Изход:
  Схема на всички предстоящи мачове. Мачовете трябва да се разпределени по кортове и по време във файл под формата на разписание.
  Да се поддържа лесно разширение на функционалността и входните данни и изход към различни формати (json, excell , txt)

Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Курс по Обектно-ориентирано програмиране
на специалност Информатика
Летен семестър на учебната 2018/2019 година

Студентски дневник

- Автор: Николай Атанасов
- Допълнителни бележки важни за проекта
    - **Да се използват поне 2 pattern-а от шаблоните за дизайн(https://bg.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%B7%D0%B0_%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD)**
    - **Системата да е обектно моделирана посредством ООП парадигмите.**
    - **Да върне коректни данни спрямо зададените входни данни.**
    - **Работа с множество от критерии.**
- Описание
  Всички входни данни се получават от база данни под формата на файл.
  Входните данни съдържат:
    1. Списък от студентите.
    2. Списък с критериите за оценяване, които ще бъдат включени

  Пример: [„Участия в час“, „Домашни работи“, „Контролни Работи“, „Проект“ и др,]

  Всеки критерий съхранява:
    1. Специфична информация (За критерий „Контролни работи“ се пазят оценките за проведените контролни).
    2. Тежест в крайната оценка.
    3. Начинът по-който се формира оценката за Критерия. (Пример: за критерий „Контролни Работи“ се взима средно аритметичното от всички контролни на студента)

  В Студента се пазят стойностите за различните критерии. След завършване на всеки час, оценяване на контролни, домашни или проект и др. информацията за съответния студент трябва да се попълва. Информацията за всички студенти се пази в бинарен файл.

  След завършване на семестъра се изчисляват финалните оценки.

  Изходни данни:
  Финални оценки на всеки студент и оценките за всеки критерий записани в подходящо форматиран текстов файл.