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

ДОМАШНА РАБОТА №2

курс Обектно-ориентирано програмиране
специалност Софтуерно инженерство
летен семестър 2017/2018 г.

СРОК ЗА ПРЕДАВАНЕ: посочен в Moodle

Изисквания за предаване:

- Предаване на домашното в указания срок от всеки студент като .zip архив със следното име: (номер_на_домашно)_SI_(курс)_(група)_(факултетен_номер), където:
  - (номер_на_домашно) е цяло число, отговарящо на номера на домашното за което е отнася решението (например 2);
  - (курс) е цяло число, отговарящо на курс (например 1);
  - (група) е цяло число, отговарящо на групата Ви (например 1);
  - (факултетен_номер) е цяло число, отговарящо на факултетния Ви номер (например 63666);
- Архивът да съдържа само изходен код (.cpp и .h файлове) с решение отговарящо на условията на задачите, като файловете изходен код за всяка задача трябва да са разположени в папка с име (номер_на_задача), където (номер_на_задача) е номера на задачата към която се отнася решението;
- Не е разрешено да ползвате класове от библиотеката STL като std::string, std::vector, std::stack и др.
- Качване на архива на посоченото място в Moodle;

Пример за .zip архив за домашно: 2_SI_1_1_63666.zip

Задача 1. Електрически устройства и електрическа мрежа 3 точки

Да се реализира клас **ElectricDevice**, който описва електрическо устройство и съхранява данни за името му (символен низ в динамичната памет) и консумирана мощност в киловати (цяло число). Да се реализира клас **ElectricNet**, който описва електрическа мрежа със зададена максимална консумация на електроенергия. Да се реализират следните оператори:

- + и += с аргумент **ElectricDevice**, който добавя устройство към мрежа, ако тя няма да се претовари, т.е. сумарната консумация на енергията на всички включени в нея устройства няма да надхвърли максималната консумация за мрежата;
- - и -= с аргумент **ElectricDevice**, който премахва устройство от мрежата, ако то е включено в нея. Устройствата се разпознават по името си;
- [] с аргумент символен низ, който дава възможност да се достъпи дадено устройство, включено в мрежата;
- !, който проверява дали има някакво включено в мрежата устройство;
- ++, който удвоява максималната консумация на енергия в мрежата;
- --, който намалява два пъти максималната консумация на енергия в мрежата, ако тя няма да се претовари след тази операция;

Задача 2. Шаблони на класове Pair и Settings 4 точки

a. Да се реализира шаблон на клас **Pair** (двойка) с параметър **T**, който има член-данни **key** (ключ) от тип низ от знаци и **value** (стойност) от тип **T**. Да се реализират методите от голямата четворка, и методи **getKey**, **getValue** и **setValue**.
b. Да се реализира шаблон на клас Settings (настройки) с параметър **T**, който е контейнер на двойки от клас **Pair**. Да се реализират:
    - методите от голямата четворка;
    - метод **count**, който връща броя на елементите в контейнера;
    - метод **set** с 2 параметъра: **key** (от тип низ) и **value** (от тип **T**). В случай, че в контейнера няма двойка с ключ **key**, тогава в контейнера се добавя нова двойка със съответните ключ и стойност. Ако в контейнера има двойка с ключ **key**, тогава стойността на двойката се замества с **value**;
    - метод **get** с 2 параметъра: **key** (от тип низ) и **value** (от тип **T &**). Методът проверява дали в контейнера има двойка с ключ **key** и ако има – връща в параметъра **value** намерената стойност на двойката и връща **true**. В противен случай – връща **false**.
c. Да се напише главна програма за тестване на класовете.

Задача 3. Превозни средства 3 точки

a. Създайте клас **Vehicle** (превозно средство), който има следните член-данни:
   - **make** - марка
   - **model** - модел
   - **color**- цвят
   - **year** – година на производство
   - **mileage** – пробег в мили

   и реализирайте следните методи:

   - методи за достъп (**get**) и мутатори (**set**) за всички член данни
   - подразбиращ се конструктор, конструктор с параметри и необходимите методи (ако използвате динамична памет)
   - (виртуална?) функция **details**, който не прави нищо.

b. Създайте клас **Car** (автомобил), който наследява клас **Vehicle** и няма допълнителни член-данни. Реализирайте нужните методи и метод **details**, който припокрива метода от базовия клас и извежда по подходящ начин информацията за автомобила.

c. Създайте клас **Truck** (камион), който наследява клас **Vehicle** и има собствена член-данна **size** – размер на камиона в метри. Реализирайте нужните методи и метод **details**, който припокрива метода от базовия клас и извежда по подходящ начин информацията за камиона.

d. Създайте клас **Motorcycle** (мотоциклет, който наследява клас **Vehicle** и има собствена член-данна **type** – тип на мотоциклета (например *chopper*, *cruiser*, *dirt bike*, *touring*). Реализирайте нужните методи и метод **details**, който припокрива метода от базовия клас и извежда по подходящ начин информацията за камиона.

e. Напишете главна програма, в която да илюстрирате по подходящ начин използването на дефинираните класове.