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

ПОПРАВИТЕЛЕН ИЗПИТ

курс Обектно-ориентирано програмиране

специалност Софтуерно инженерство

27.08.2019 г.

Времетраене: 2 ч. и 30 мин. (3 ч. и 30 мин.)

Изисквания за предаване:

- Предаване на решенията от изпита става като .zip архив със следното име: Exam_SI_(курс)_(група)_(факултетен_номер), където:
  - (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);
- Качване на архива на посоченото място в Moodle;

Пример за .zip архив за изпита: Exam_SI_1_1_63666.zip

Задача 0. (Задача за допускане) Албум с песни 40 т.

1. Да се реализира клас **Song**, който описва музикално изпълнение със следните свойства:
    - заглавие (низ до 100 символа)
    - изпълнител (низ до 100 символа)
    - година на издаване (цяло число)
    - дължина на песента (цяло число секунди)
2. За класа **Song** да се реализират:
    - подходящи конструктори, селектори и мутатори
    - голяма четворка, ако е нужна
    - операция за извеждане на информация за песен на стандартния изход
3. Клас **Album**, който описва албум със следните полета:
    - списък от песни с произволна дължина
    - име на албума (низ до 100 символа)
4. За класа **Album** да се реализират
    - подходящи конструктори, селектори и мутатори
    - голяма четворка, ако е нужна
    - метод за извеждане на информация за албум на стандартния изход
    - метод **getLength**, който намира дължината на албума в секунди
    - метод **getArtist**, който връща името на изпълнителя на албума, ако всички песни са на един и същ изпълнител, или “Various Artists”, ако албумът се състои от песни на повече от един изпълнител
    - метод **findSong**, който връща указател към първата песен в албума с дадено име, или NULL, ако такава не е намерена
    - метод **deleteDuplicates**, който изтрива от албума всички дублиращи се песни. Две песни се считат за дублиращи се, ако за тях съвпадат заглавието, изпълнителя, годината на издаване и дължината
5. Да се реализира примерна програма, която демонстрира всички гореизброени операции

Забележка: Не е разрешено да ползвате класове от STL библиотеката. Да се осигури правилното функциониране на методите от “голямата четворка”.

Задача 1. Контейнери **40 т.**

1. Да се реализира шаблон на абстрактен базов клас `Container<T>`, дефиниращ следните методи за работа с контейнери:

- `bool member (const T&) const` : проверка за принадлежност на елемент
- `bool add (const T&)`: добавяне на елемент към контейнера, върнатата стойност показва дали добавянето е било успешно
- `bool remove (const T&)`: премахване на (първото срещане на) елемент от контейнера, върнатата стойност показва дали премахването е било успешно
- `int size () const`: брой елементи в контейнера
- `T& operator [] (int i)`: достъп до i-тия елемент на контейнера

2. Да се реализира шаблон на клас `ArrayVector<T>`, наследник на `Container<T>`, реализиращ контейнер с ограничен капацитет чрез динамичен масив, позволяващ по няколко екземпляра от една и съща стойност. Капацитетът на контейнера да се задава по време на конструиране, като по подразбиране да е 10 елемента.

3. Да се реализира шаблон на клас `Set<T>`, наследник на `Container<T>`, реализиращ контейнер, който позволява най-много по един екземпляр от всяка стойност (например, числото 5 може да се среща най-много един път в множество от числа).

Забележка: Не е разрешено да ползвате класове от STL библиотеката. Да се осигури правилното функциониране на методите от “голямата четворка”.

Задача 2. Магазин **40 т.**

Да се реализира софтуер за магазин. В магазина ще се продават следните типове артикули - **тениски** и **обувки**

Да се напишат следните класове и техните член-данни/методи:

Клас `StoreItem` (абстрактен):

- Член-данни:
  - Тип на продукта (enum)
  - Име на продукта (низ)
  - Цена (реално число) (0 по подразбиране)
- Методи:
  - get() за всяка член-данна
  - set() за всяка член-данна
  - Конструктор с параметри

Клас `TShirt` (тениска): (наследява `StoreItem`)

- Член-данни:
  - Тип на продукта: тениска
  - Цвят на тениската (низ)
  - Размер на тениската (низ – S, M, L, XL, XXL)
- Методи:
  - get() за всяка член-данна
  - set() за всяка член-данна
  - Конструктор с параметри
  - Оператор == - сравнява два продукта по цвят и размер
  - print() - извежда информацията на продукта на конзолата

Клас `Shoes` (обувки): (наследява `StoreItem`)

- Член-данни:
  - Тип на продукта: обувки
  - Цвят на обувките (низ)
  - Вид на обувките (низ – детски, дамски, мъжки)
  - Размер на обувките (цяло число)
- Методи:
  - get() за всяка член-данна
  - set() за всяка член-данна
  - Конструктор с параметри
  - Оператор > - сравнява два продукта по размер
  - print() - да изведе информацията на продукта на конзолата

Клас `Shop`:

Служи като контейнер за артикулите в магазина.

- Член-данни:
  - Име ва магазина (низ)
  - Хетерогенен контейнер за артикулите в магазина
- Методи:
  - Добавяне на продукт (приема продукта като аргумент)
  - Премахване на продукт (приема индекс)
  - Промяна на цена на продукт (приема индекс и цена)
  - Промяна на име на продукт (приема индекс и ново име)
  - Извежда информация за всички продукти на конзола

Да се реализира примерна програма, която демонстрира всички гореизброени операции

Забележка: Разрешено е да ползвате класове от STL библиотеката. Да се осигури правилното функциониране на методите от “голямата четворка”.