Упражнение 9-10

Наследяване

Видове наследяване

- в c++
  - public
  - protected
  - private
- семантично
  - is-a
  - has-a
  - is-implemented-in-terms-of

Overloading vs. Overriding

***Следващите примери са само с цел пояснение на всяка една от изложените концепции.

Асоциация - всеки обект отговаря за собствения си “life-cycle”, няма зависимости

Пример:

```
class Person
{
…
        void drive(Car car);
...
};
```

Агрегация - има родителски клас, който взима назаем обект, ако родителя умре, обекта взет назаем **продължава да живее независимо.** Родителският клас **НЕ** отговаря за създаване/унищожаване на обекта.

Пример:

```
class Client
{
public:
        Client(Car* car) : car(car)
        {}
        ...
private:
        Car* car;
        ...
```

```
};
```

Композиция - има родителски клас, който притежава обект и **отговаря** за неговия живот.

Пример:

```
class Tree
{
public:
        Tree(int initial)
        {
                Fruits = new Fruit[initial];
                ...
        }

private:
Fruit* fruits;
...
};
```

Задачи:

1.Да се дефинира клас Person, който описва човек по съответно:

- име - символен низ, разположен в динамичната памет
- рождена дата във формат dd/mm/yyyy (година на раждане)

Да се дефинира клас Student, който наследява Person и добавя към наследените характеристики:

- наименование на университета
- наименование на специалността
- факултетен номер

Да се реализира функция за извеждане на информацията за студент.

2.Да се дефинира клас Person, който описва човек по съответно:

- име - символен низ, разположен в динамичната памет
- година на раждане

Да се дефинира клас Teacher, който наследява Person и добавя към наследените характеристики:

- наименование на университета
- наименование на предмета, по който преподава .

Да се реализира функция за извеждане на информацията за преподавател.

Да се реализират функции за промяна на университета/(предмета), в който/(по който) преподава учителя.

3.Да се дефинира клас *Appointment*, който описва уговорена среща със следните характеристики:

- описание – символен низ, разположен в динамичната памет
- начален и краен час – низове с дължина точно 4 символа във формат ччмм, които съответстват на началния и крайния час на срещата. Валидирането на стойностите може да се пропусне.

Да се дефинира клас *Meeting*, който наследява Appointment и добавя към наследените характеристики име на контакта, с който ще бъде проведена срещата. Името на контакта да е символен низ, разположен в динамичната памет.Да се дефинират подходящи конструктори, селектори и мутатори за всеки от класовете.

Нека крайно множество от срещи от тип Appointment или Meeting се разглежда като дневен график. Графикът е представен като масив. Да се избере подходящ тип за елементите на масива.

Да се реализират следните фунции:

- bool hasAppointmentWith([подходящ тип] schedule,..., const char* constact), която получава като аргументи график и име на контакт и проверява дали има включена среща с посочения контакт в графика;

Забележка: Функциите могат да имат допълнителни параметри, извън описаните.