

ЗАДАЧИ ЗА ЗАДЪЛЖИТЕЛНА
САМОПОДГОТОВКА
ПО
Обектно-ориентирано програмиране
Сериализация

email: kalin@fmi.uni-sofia.bg

12 април 2017 г.

1. Да се сериализира и десериализира масив от масиви от числа:
`vector<vector<int> >`
Да дефинират необходимите за целта оператори. Да се тества програмата!
2. Да се сериализира и десериализира масив от масиви от символни низове (`char*`):
`vector<vector<char*> >`
Да дефинират необходимите за целта оператори. Да се тества програмата!
3. Към задачата 2.4.25. за софтуерна фирма да се реализира сериализация и десериализация на масив от служители.
Да се тества програмата!
4. Да се реализира абстрактен клас `NetworkDevice`, който дефинира следните (абстрактни, чисто втируални) операции:
 - `bool attachTo(NetworkDevice* device)`: свързва устройството с друго устройство `device`.
Реализациите на метода в наследниците на `NetworkDevice` ще връщат `true`, ако свързването е възможно и `false`, ако

свързването не е възможно. Правилата, по които се определя дали може или не може да се свърже устройството, зависят от конкретния наследник на `NetworkDevice`.

При дефиниране на метода в наследените класове осигурете, че създадената връзка е двупосочна. Т.е. счита се, че ако устройството `A` е свързано с устройството `B`, то и устройството `B` е свързано с устройството `A`. Не допускайте дадено устройство да може да се свърже повече от веднъж с едно и също устройство или пък да се свърже със себе си.

- [попълнете правилния тип] `getAttachedDevice(int i)`. Връща (*Упътване: указател към*) `i`-тото поред устройство, към което даденото устройство е свързано и `NULL`, ако индексът `i` не е валиден.

Класът `NetworkDevice` също да съдържа и уникален идентификатор от тип `int` за всяко устройство. Идентификаторът може да се задава чрез конструкторите на наследниците.

Да се реализират производните класове `EndDevice` и `Switch`.

- За `EndDevice` е характерно, че устройството може да е свързано най-много с още едно устройство. Т.е. във всеки момент от времето `EndDevice` или не е свързан с нищо, или е свързан с точно едно устройство. Веднъж свързано, `EndDevice` не може да бъде свързано отново с друго устройство.
- За `Switch` е характерно, че устройството може да е свързано с максимум 8 други устройства. При достигане на броя на свързаните устройства до 8, устройството `Switch` не може да бъде свързано с повече устройства.

Промяна на вече създадена връзка не е възможна и при двата вида устройства.

За така дефинираните класове да се решат следните задачи:

- (a) Да се реализира функция `void printConnections(NetworkDevice* devices[], int n)`, която отпечатва на екрана информация за връзките на всяко устройство в масив от устройства. *Упътване: добавете нова виртуална функция `printConnnection` в базовия клас. Връзките можете да печатате във формата `<идентификатор 1> -- <идентификатор 2>`.*

- (б) Да се създаде примерна програма, в която се инициализират няколко устройства, добавят се в масив, създават се връзки между тях и се използва функцията `printConnections` за отпечатване на връзките.
- (в) Да се реализира функция `bool connected([попълнете правилния тип] d1, [попълнете правилния тип] d2)`, която проверява дали има връзка (пряка или косвена) между устройствата `d1` и `d2`. За улеснение приемоте, че няма циклични връзки между устройствата. *Упътване: използвайте рекурсивна функция по подобие на функциите за търсене на път.*
- (г) *Допълнителна задача:* Решете горната задача и при случая на възможни циклични връзки.
- (д) *Допълнителна задача:* Напишете функция, която по масив от устройства търси дали има циклична връзка между някои две от тях.
- (е) *Допълнителна задача с повишена трудност:* Сериализирайте и десериализирайте масив от свързани устройства.