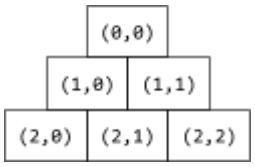
Второ контролно по Обектно-ориентирано програмиране

спец. Компютърни науки, 22.05.2019 г. *Вариант 1*

Задача 1. Да се създаде клас `Triangle`, представящ равностранен "триъгълник" от елементи от произволен тип `T`. Максималният брой редове на триъгълника е фиксиран и се задава като параметър при конструиране на обект от класа. Всеки елемент на триъгълника се указва с двойка индекси `(i,j)` съгласно фигурата вляво.

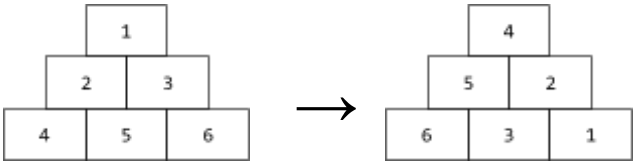


За класа да се реализират:

- конструктор, създаващ триъгълник с един елемент и фиксиран максимален брой редове
- необходимите канонични методи от голямата четворка
- метод `getRows`, връщащ броя на редовете на триъгълника
- операция `t += a`, която добавя нов ред в основата на триъгълника `t`, състоящ се от елементите на масива, към който сочи указателят `a`. Може да се приеме, че в подадения масив са записани достатъчно на брой елементи, за да запълнят новия ред на триъгълника.
- метод `getAt(i,j)`, който връща елемента намиращ се на индекси `(i,j)`, като при некоректни индекси връща елемента на позиция `(0,0)`
- булев метод `isStable`, който проверява дали е вярно, че всеки елемент на триъгълника е по-голям от елементите над него. Считаме, че типът `T` поддържа операциите `<` и `>`.

Бонус: да се напише метод `rotate`, който "завърта" триъгълника на 120° по часовниковата стрелка.

Пример:



Задача 2. В малък зоопарк има само два вида животни: папагали и маймуни. Всяко животно има име, представено чрез низ. Когато са гладни, папагалите свирукат, а маймуните крещят. В зоопарка има инсталирано устройство `Zooawei` с изкуствен интелект, което знае имената на всички животни и може да разпознава техните звуци и да извежда информация за тях на стандартния изход. За целта устройството поддържа следните класове:

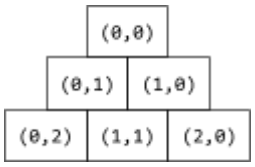
- клас `Animal`, който представя животно като пази името му. Класът има подходящи конструктори и канонични методи, където са необходими и селектор `getName`. Класът дефинира чисто виртуалната операция `asksForFood.`
- клас `Parrot`, представящ папагал, който разширява `Animal` и реализира операция `asksForFood`, която извежда на стандартния изход низа `"parrot <name> whistles for food"`, където `<name>` е името на папагала
- клас `Monkey`, представящ маймуна, който разширява `Animal` и реализира операция `asksForFood`, която извежда на стандартния изход низа `"monkey <name> cries for food"`, където `<name>` е името на маймуната
- клас `Zoo`, представящ списък от животните в зоопарка (не повече от 100 на брой), и реализиращ:
    - операция `add(Animal*)`, добавящ ново животно в зоопарка
    - операция `morningSounds`, която извежда на стандартния изход информация за звуците от всички животни на сутринта, когато всички са гладни. За всяко животно се извежда на отделен ред в последователността, в която животните са добавени
    - Бонус: конструктор за копиране и операция за присвояване и деструктор, който освобождава паметта за всички добавени в зоопарка животни

Да се напишат реализации на описаните горе четири класа.

Второ контролно по Обектно-ориентирано програмиране

спец. Компютърни науки, 22.05.2019 г. *Вариант 2*

Задача 1. Да се създаде клас `Triangle`, представящ равностранен "триъгълник" от елементи от произволен тип `T`. Максималният брой редове на триъгълника е фиксиран и се задава като параметър при конструиране на обект от класа. Всеки елемент на триъгълника се указва с двойка индекси `(i,j)` съгласно фигурата вляво.

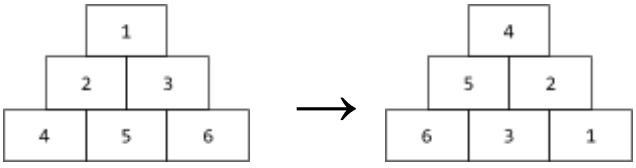


За класа да се реализират:

- конструктор, създаващ триъгълник с един елемент и фиксиран максимален брой редове
- необходимите канонични методи от голямата четворка
- метод `getRows`, връщащ броя на редовете на триъгълника
- операция `t++`, която добавя нов ред в основата на триъгълника `t`, като всеки елемент се получава като сума на двата елемента над него. Считаме, че типът `T` поддържа бинарната операция `+`, която по два дадени елемента от тип `T` връща елемент от тип `T`.
- метод `getAt(i,j)`, който връща елемента намиращ се на индекси `(i,j)`, като при некоректни индекси връща елемента на позиция `(0,0)`
- операция `t1 + t2`, която по два триъгълника връща трети триъгълник `t3`, чиито елементи се получават от поелементното събиране на елементите на `t1` и `t2` на едни и същи позиции. Ако единият триъгълник има повече редове от другия, допълнителните редове на по-големия триъгълник се игнорират и не се записват в `t3`. Считаме, че типът `T` поддържа двуместната операция `+`, която по два дадени елемента от тип `T` връща елемент от тип `T`.

Бонус: да се напише метод `rotate`, който "завърта" триъгълника на 120° по часовниковата стрелка.

Пример:



Задача 2. Професор Х се забавлява в свободното си време като разглежда редици от цели числа с интересни свойства. Той иска да има програма на C++, която да му помогне да проверява бързо и лесно дали редица има дадена комбинация от свойства, която той сам си е измислил.

Програмата трябва да поддържа следните класове:

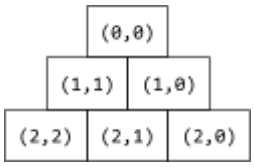
- клас `Property`, който представя свойство на редица от цели числа. Класът дефинира чисто виртуалната операция `test(a,n)`.
- клас `DivisibleBy`, който разширява `Property` и реализира булев метод `test(a,n)`, който проверява дали е вярно, че всички елементи на масива `a` с дължина `n` се делят на фиксирано число `x`, което се задава при конструиране на обект от класа
- клас `CountNumbers`, който разширява `Property` и реализира булев метод `test(a,n)`, който проверява дали е вярно, че в масива `a` с дължина `n` числото `x` се среща точно `k` пъти, като `x` и `k` се задават при конструиране на обект от класа
- клас `Properties`, представящ списък от свойства (не повече от 100 на брой) и реализиращ:
  - операция `add(Property*)`, добавящ ново свойство в списъка
  - операция `testAll(a,n)`, която по подаден масив от цели числа a с дължина n проверява дали всички свойства в списъка са изпълнени за него
  - Бонус: конструктор за копиране и операция за присвояване и деструктор, който освобождава паметта за всички добавени в списъка свойства

Да се напишат реализации на описаните горе класове.

Второ контролно по Обектно-ориентирано програмиране

спец. Компютърни науки, 22.05.2019 г. *Вариант 3*

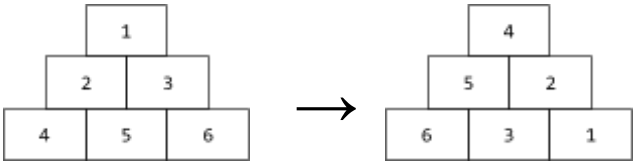
Задача 1. Да се създаде клас `Triangle`, представящ равностранен "триъгълник" от елементи от произволен тип `T`. Максималният брой редове на триъгълника е фиксиран и се задава като параметър при конструиране на обект от класа. Всеки елемент на триъгълника се указва с двойка индекси `(i,j)` съгласно фигурата вляво.



За класа да се реализират:

- конструктор, създаващ триъгълник с един елемент и фиксиран максимален брой редове
- необходимите канонични методи от голямата четворка
- метод `getRows`, връщащ броя на редовете на триъгълника
- операция `s >> t`, която добавя нов ред в основата на триъгълника `t` и прочита стойностите на елементите му от потока `s`. Считаме, че типът `T` поддържа операцията `>>`.
- метод `getAt(i,j)`, който връща елемента намиращ се на индекси `(i,j)`, като при некоректни индекси връща елемента на позиция `(0,0)`
- булев метод `isSymmetric`, който проверява дали е вярно, че триъгълникът е симетричен относно височината към основата му. Считаме, че типът `T` поддържа операциите `==` и `!=`.

Бонус: да се напише метод `rotate`, който "завърта" триъгълника на 120° по часовниковата стрелка.
Пример:



Задача 2. Изобретателят Минчо К. иска да построи прост калкулатор, използвайки кристали от специален червен минерал. Преди да построи калкулатора, Минчо иска да го симулира с помощта на обектно-ориентирана програма, написана на езика C++. Той е измислил как да строи кристали от червен минерал, които извършват трансформации над входа си, като всяка една от тях поддържа операция `transform`, която приема цяло число `n` и връща като резултат съответното трансформирано число. Трите вида трансформации, за които Минчо е изобретил кристали, са:

- `Id`, която връща числото `n` без да го променя
- `Sum`, която добавя към `n` някакво цяло число `x`, което е предварително зададено при конструиране на обект от класа
- `Product`, която умножава `n` по някакво цяло число `x`, което е предварително зададено при конструиране на обект от класа

Пример: Ако `Sum s(3);` то `s.transform(4)` връща `7`, а `s.transform(10)` връща `13`.
Всички горни класове имат общ базов клас `NumberTransformation`, който реализира чисто виртуалната операция `transform.`
Калкулаторът на Минчо работи като той подреди няколко (не повече от 100 на брой) кристала в редица, зададе число на входа на първия кристал и погледне какъв е резултата в последния.
Помогнете на Минчо да реализира клас `Calculator`, който предоставя:

- операция `add(NumberTransformation*)`, добавяща нова трансформация в редицата
- операция `calculate(int n)`, която изпълнява последователно всички зададени трансформации над дадено цяло число `n` и връща резултата
- засега калкулаторът на Минчо не поддържа конструктор за копиране, операция за присвояване и деструктор, който освобождава паметта за всички добавени в калкулатора трансформации, но планира да го реализира в бъдеще като бонус.

Задача 2. [полиморфизъм]

Идея:

XMenTeam
.add(SuperHero *)
.fight() // calls useSuperPower() to all heroes

SuperHero
- name

Wolverine
.useSuperPower() // "$name: Using my claws
Storm
.useSuperPower() // "$name: Manipulate weather