Декларативно програмиране в процедурните езици

Курсова задача №2: Итератори, множества и поточни линии

Задача 1:
Да се реализира итератор, който обхожда подаден граф с числа по възлите, чрез търсене в лъч. Пироритета на обхождане на възлите се определя от числата в тях (по-големите се обхождат първи). Широчината на лъча се подава като параметър.
Вход: Граф с числа по възлите и широчина на лъча.
Изход (на екран): Итерира възлите, през които се минава по време на обхождането.

Задача 2:
Дадени са вече реализирани (чрез Map обекти) функции f_1, f_2, ... f_n. Да се намери максимална по дължина редица f_{i_1}, f_{i_2}, ... f_{i_k} (където i_1, i_2, ... i_k са рзлични индекси измежду от 1 до n) такава, че композицията $f_{i_1} \bullet f_{i_2} \bullet \cdots \bullet f_{i_k}$ е непразно множество (Map).
Вече реализирани методи: Map обекти f_1, f_2, ... f_n.
Изход (на екран): Индексите i_1, i_2, ... i_k.

Задача 3:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида $(f_{i_1}|f_{i_2}| \ldots |f_{i_k})\{n,m\}$, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, | обозначава алтернативност, а {n,m} е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.

Задача 4:
Да се реализира итератор, който изброява ребрата от даден граф с тежести по ребрата, по които се сформира минимално покриващо дърво по алгоритъма на Крускал.
Вход: Граф с тежести по ребрата.
Изход (на екран): Итерира ребрата, по които се построява минималното покриващо дърво.

Задача 5:
Дадени са вече реализираните итератори I_1, I_2, ... I_n на рационални числа. Да се реализира итератор, който изброява стойностите $\lambda_1.I_1[i]+\lambda_2.I_2[i]+\cdots+\lambda_n.I_n[i]$, където λ_1, λ_2 ... λ_n са дадени коефициенти. Чрез итератора да се намери $max\{\ \lambda_1.I_1[i]+\lambda_2.I_2[i]+\cdots+\lambda_n.I_n[i]\ \}$. Ако някой от итераторите завърши генерирането на стойности преди другите, се счита че той генерира нули.
Вход: Коефициенти λ_1, λ_2 ... λ_n.
Вече реализирани методи: Итератори/генератори I_1, I_2, ... I_n.
Изход: Максималната стойност за $\lambda_1.I_1[i] + \lambda_2.I_2[i] + \cdots + \lambda_n.I_n[i]$.

Задача 6:
Да се реализира итератор на всички ациклични пътища в даден ориентиран граф и чрез него да се реализира метод за транзитивно затваряне на графа.
Вход: Ориентиран граф.
Изход: Транзитивното затваряне на дадения граф.

Задача 7:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида f_{i_1}+?.f_{i_2}+?. ... .f_{i_k}+?, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, . обозначава

конкатенация, а +? е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.

Задача 8:
Да се реализира итератор, който обхожда подадено дръво с числа по възлите, чрез α-β отсичане.
Вход: Дърво с числа по възлите.
Изход (на екран): Итерира възлите, през които се минава по време на обхождането.

Задача 9:
Имаме пространство (екран) с размер NxM (пиксела). В него са разположени няколко квадрата (с дадени координати на центъра и размер на страната). Да се реализира итератор, който изброява възможните позиции на нов квадрат (размера на страната му се подава като параметър) така, че той да не се пресича с вече съществуващи квадрати и да е възможно най-близко до центъра на пространството, но не по-близко от дадено ограничение (минимално разстояние от центъра). Използват се само целочислени координати.
Вход: N, M, списък от съществуващите квадрати, минимално разстояние от центъра и размер на страната на новия квадрат.
Изход (на екран): Итерира възможните позиции на центъра на новия квадрат.

Задача 10:
Да се реализира итератор, който изброява матриците от редица от генетично получени квадратни матрици с рационални числа. За всяка матрица е определено число, което представлява нейния живот (в брой итерации на итератора). Редицата започва с две матрици с еднакъв размер, а всяка следваща матрица се получва, като на случаен принцип се изберат две все още живи (по време на текущата итерация) матрици и се образува нова метрица от средно-аритметичното им. Живота на новата матрица е произволно число от 0 до 5. Една матрица живее в продължение на толкова итерации, колкото е размера на живота ѝ.
Вход: Двете начални матрици и техните животи (числа от 2 до 5).
Изход (на екран): Итерира матриците от получаващата се редица.

Задача 11:
Да се реализира итератор на елементарните цикли в даден граф с тежести по ребрата. Итератора да се използва за намиране на сумите от ребрата по циклите, които са по-дълги от половината от броя на върховете на графа.
Вход: Граф с тежести по ребрата.
Изход (на екран): Итерира сумите на дългите цикли.

Задача 12:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида ($f_{i_1}|f_{i_2}|$ ... $|f_{i_k}$){n,}?, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, | обозначава алтернативност, а {n,}? е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.

Задача 13:
Да се реализира итератор, който изброява всички подграфи на даден граф.
Вход: Граф.
Изход (на екран): Итерира подграфите.

Задача 14:
Дадени са вече реализираните итератори I_1, I_2, ... I_n на цели числа. Да се реализира итератор, който изброява следните стойности: $min(I_1[0], I_2[0] \dots I_n[0])$, $max(I_1[1], I_2[1] \dots I_n[1])$, $min(I_1[2], I_2[2] \dots I_n[2])$, ... и т.н. Ако някой от итераторите завърши генерирането на стойности преди другите, се счита че той генерира нули.
Вече реализирани методи: Итератори/генератори I_1, I_2, ... I_n.
Изход (на екран): Итерира първите 20 числа от реализирания итератор.

Задача 15:
Даден е файл и размер N на страница. Да се реализира итератор на страниците на файла (една страница съдържа N поредни реда от файла) и чрез него да се намерят тези страници, в които броя на символите е по-голям или равен от 50N и за всяка от тези страници да се изведат броя на символите и броя на думите.
Вход: Файл и число N.
Изход (на екран): Итерира размера (брой символи) и броя думи на достатъчно големите страници.

Задача 16:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида (f_{i_1}&f_{i_2}& ... &f_{i_k}){n,m}?, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, & обозначава едновременност, а {n,m}? е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.

Задача 17:
Да се реализират итератори на редове и колони на подадена матрица и чрез тях да се реализира умножение на матрици.
Вход: Матрица $N \times M$ и матрица $M \times K$.
Изход: Произведението на двете матрици.

Задача 18:
Даден е монотонен оператор Φ (оператор Φ, който преобразува множества и за всяко множество X имаме, че $X \subseteq \Phi(X)$). Да се намери момента на насищане на прилагането на Φ към подадено множество X (насищане се получава, като във $\Phi^{n+1}(X)$ не се срещат нови елементи спрямо $\Phi^n(X)$).
Вход: Множество X.
Вече реализирани методи: Оператор Φ (метод, който приема множество като параметър и връща множество като резултат).
Изход: Резултата (множество) $\Phi^n(X)$ при който се постига насищането.

Задача 19:
Дадени са вече реализираните итератори I_1, I_2, ... I_n на монотонно растящи редици от числа. Да се реализира итератор на тези числа, които придналежат на поне половината от редиците.
Вече реализирани методи: Итератори/генератори I_1, I_2, ... I_n.
Изход (на екран): Итерира първите 20 числа от реализирания итератор.

Задача 20:
Да се реализира итератор, който изброява ребрата от даден граф с тежести по ребрата, по които се сформира минимално покриващо дърво по алгоритъма на Прим.
Вход: Граф с тежести по ребрата.
Изход (на екран): Итерира ребрата, по които се построява минималното покриващо дърво.

Задача 21:
Да се използват итератора за подмножества и операцията за проверка на $\subseteq$ за да се реализират наново операциите за сечение, обединение и разлика на множества.
Вход: Няколко множества.
Изход (на екран): Техни сечения, обединения и разлики.

Задача 22:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида $f_{i_1}*.f_{i_2}*. \ \ldots \ .f_{i_k}*$, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, . обозначава конкатенация, а * е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.

Задача 23:
Да се реализира итератор, който по подаден връх v от даден граф с тежести по ребрата, извършва обхождане на всички върхове, достижими от v, такива, че пътя от v до върха е с дължина най-много N и е възходящ (т.е. всяко следващо ребро е с по-голяма тежест).
Вход: Граф с тежести по ребрата, връх v и число N.
Изход (на екран): Итерира достижимите върхове.

Задача 24:
Да се използва итератора за подмножества (с пермутиране) за да се реализира итератор на всички частични функции от дадено множество A в дадено множество B (функциите са Map обекти).
Вход: Множества A и B.
Изход (на екран): Итерира първите 20 частични функции от A в B.

Задача 25:
Да се реализира метод, който проверява дали подадена итерация/поток от обекти отговаря на "функционален" регулярен израз. "Функционалните" регулярни изрази са от вида $(f_{i_1}\&f_{i_2}\& \ \ldots \ \&f_{i_k})\{n,\}$, където f_i са вече реализирани статични булеви функции, които могат да се прилагат за конкретен обект, & обозначава едновременност, а {n,} е стандартното означение от регулярните изрази.
Вход: Списък от обекти.
Вече реализирани методи: static bool f_1(object), static bool f_2(object), ... static bool f_n(object).
Изход: True/False.