

Programare Orientată pe Obiecte

Propuneri de proiecte în C++

1. Sa se definească o clasă generică pentru fracții cu numărător și numitor de un tip T neprecizat (parametru), în care operatorii $=$, $==$, operatorii aritmetici, $+$, $-$ (unar și binar), $*$ și $/$ să fie supraîncărcați pentru operațiile obișnuite cu fracții, iar operatorul (tip) de conversie a tipurilor să fie supraîncărcat pentru a efectua conversia unui obiect de tipul T la o fracție care are ca numitor "unitatea" din tipul T (element neutru la $*$), care poate fi elementul construit de un anumit constructor al clasei T , având argument de tip int , care să dea sens declarației cu inițializare $T \ a=1$ (ca și declarației cu inițializare $T \ a=0$, din care va rezulta elementul "zero" din tipul T , neutru la $+$). Se vor da exemple de creare și utilizare de obiecte pentru diferite tipuri ale numărătorului și numitorului : int , întregi Gauss, adică numere complexe cu părțile reală și imaginară de tip int (după definirea acestora ca o clasă separată), etc.
2. Sa se definească o clasă generică pentru polinoame de o nedeterminată cu coeficienți de un tip neprecizat (parametru), în care operatorii $=$, $==$, operatorii aritmetici, $+$, $-$ (unar și binar), $*$ și, eventual $/$, $\%$, să fie supraîncărcați pentru operațiile obișnuite cu polinoame, iar operatorul (tip) de conversie a tipurilor să fie supraîncărcat pentru a efectua conversia unui obiect de tipul coeficienților la un polinom de grad 0 și invers. Șirul coeficienților unui polinom se poate reprezenta ca un vector (adică pointer la tipul coeficienților) sau ca o listă simplu/dublu înlănțuită (la alegere). Se vor da exemple de creare și utilizare de obiecte pentru diferite tipuri ale coeficienților: int , float , complex (după definirea acestui tip ca o clasă separată), etc.
3. Idem pentru matrici dreptunghiulare cu elemente de tip neprecizat, cu supraîncărcarea operatorilor $=$, $==$, $+$, $-$ (unar și binar), $*$, $[\]$ (indexare) și conversie de la un obiect de tipul neprecizat dat la o matrice cu 1 linie și 1 coloană. O matrice cu m linii și n coloane se va reprezenta ca un vector (adică pointer la tipul elementelor) format din liniile matricii așezate succesiv, adică $a_{11}, a_{12}, \dots a_{1n}, a_{21}, a_{22}, \dots a_{2n}, \dots \dots a_{m1}, a_{m2}, \dots, a_{mn}$. Operatorul

[] va întoarce un pointer la începutul liniei al cărei număr este parametru, numărându-se începând cu 0, în reprezentarea, dată mai sus, adică, dacă a este un obiect a clasei, a[i] va fi pointer la tipul elementelor având ca valoare adresa elementului $a_{(i+1)1}$ în notația de mai sus, iar a[i][j] va fi elementul $a_{(i+1)(j+1)}$ din aceeași notație, a doua utilizare a operatorului [] fiind cea predefinită pentru pointeri la tipul elementelor matricii.

4. Idem pentru liste simplu (sau dublu) înlanțuite cu elemente de tip neprecizat, cu supraîncărcarea operatorilor =, ==, <, <=, >, >= (care întorc 1 sau 0 după cum primul operand este sau nu listă inițială/finală strictă/nestrică al celui de al doilea), + (concatenare de liste), - (binar, care întoarce lista obținută prin ștergerea listei "scazator" din "descăzut", dacă "scăzătorul" este prefix al "descăzutului", altfel întoarce "descăzutul" neschimbat), * (care pentru cei doi operanzi liste întoarce lista elementelor care ocupă aceeași poziție și coincid în cele două liste), <<, >> reprezentând deplasările la stânga/ dreapta ale listei invocante (primul operand) cu un număr întreg de poziții care este parametru (al doilea operand), [] (indicarea unui element al unei liste prin numărul său de ordine = indice), conversie de la un obiect de tipul neprecizat dat la o listă cu un element și conversia inversă de la o listă la primul/ultimul său element. O listă se va reprezenta ca un pointer la primul său element.
5. Idem pentru șiruri de elemente de un tip neprecizat, cu supraîncărcarea operatorilor =, ==, <, <=, >, >= (care întorc 1 sau 0 după cum primul operand este sau nu subșir/suprașir strict/nestrică al celui de al doilea), + (reprezentând concatenarea), - (binar, care întoarce șirul obținut prin ștergerea șirului "scazator" din "descăzut", dacă "scăzătorul" este prefix al "descăzutului", altfel întoarce "descăzutul" neschimbat), * (care pentru cei doi operanzi șiruri întoarce șirul elementelor care ocupă aceeași poziție și coincid în cele două șiruri), <<, >> reprezentând deplasările la stânga/ dreapta ale șirului invocant (primul operand) cu un număr întreg de poziții care este al doilea operand, (aceasta însemnând ștergerea de la începutul/sfârșitul șirului al unui număr de elemente egal cu al doilea operand, eventual ștergerea întregului șir dacă lungimea sa este mai mică sau egală cu al doilea operand), conversia de la un obiect de tipul elementelor la un șir de lungime 1 și conversia inversă de la un șir la primul său element. Șirurile se vor reprezenta ca vectori (adică pointeri la tipul elementelor), deci operatorul [] va fi cel predefinit pentru pointeri, nu trebuie supraîncărcat

- .
6. Idem pentru șiruri ordonate de elemente de un tip neprecizat, prevăzut cu o ordine totală (și cu operatorii relaționali, operatorii de ordine fiind corespunzători respectivei ordini totale), cu supraîncărcarea operatorilor: =, ==, <, <=, >, >= (operatorii de ordine întorc 1 sau 0 conform ordonării lexicografice a operanzilor); + (reprezentând interclasarea); - (binar, care întoarce șirul obținut prin scoaterea din "descăzut" ale elementelor șirului "scazator" care sunt în "descăzut"); * (care pentru cei doi operanzi șiruri întoarce șirul ordonat al elementelor comune celor două șiruri, indiferent de poziție); <<, >> ca la tema 5; conversia de la un obiect de tipul elementelor la un șir de lungime 1 și conversia inversă de la un șir la primul său element. Se va programa construirea unui șir prin adăugarea succesivă la el (cu operatorul supraîncărcat +) a unor elemente citite, ceea ce revine la ordonarea prin inserție simplă a elementelor citite. Șirurile se vor reprezenta ca vectori (adică pointeri la tipul elementelor), deci operatorul [] va fi cel predefinit pentru pointeri, nu trebuie supraîncărcat .
 7. Idem pentru mulțimi finite de elemente de un tip neprecizat, cu supraîncărcarea operatorilor =, ==, <, <=, (incluziunea strictă/nestricată), >, >= (relațiile duale); + (reuniunea); * (intersecția); - (binar, diferența de mulțimi); <<, >> (adăugarea/ extragerea unui element, care este al doilea operand, la/dintr-o mulțime, reprezentată de primul operand) și conversia unui element la mulțimea care îl are ca unic element. Multimile se reprezintă printr-un întreg fără semn care este numărul elementelor (pentru mulțimea vidă acest număr va fi 0) și printr-un vector (adică pointer la tipul elementelor) care conține elementele. Metodele de creare și modificare a mulțimilor vor asigura ca fiecare element să apară o singură dată în acest vector.
 8. Idem pentru arbori binari de căutare stricți cu contor, având chei de un tip neprecizat, prevăzut cu o ordine totală (și cu operatorii relaționali, operatorii de ordine fiind corespunzători respectivei ordini totale), cu supraîncărcarea operatorilor indicați mai jos.
 =, ==; <, <=, >, >= (operatorii de ordine întorc 1 sau 0 conform ordonării lexicografice a șirurilor ordonate care rezultă prin parcurgerea în ordine a operanzilor, cu repetarea fiecărei chei conform contorului);
 + se supraîncarcă în 3 moduri: a) primul operand cheie, al doilea operand arbore binar de căutare strict cu contor; b) primul operand arbore binar de

căutare strict cu contor, al doilea operand cheie; c) ambii operanzi arbori binari de căutare stricți cu contor; rezultatul operațiilor a) b) este arborele obținut prin inserarea cheii în arbore; rezultatul operației c) este arborele obținut prin inserarea în primul arbore a cheilor arborelui al doilea, parcurse în inordine, cu repetarea fiecărei chei conform contorului;

– se supraîncarcă în 2 moduri: a) primul operand arbore binar de căutare strict cu contor, al doilea operand cheie; b) ambii operanzi arbori binari de căutare stricți cu contor; rezultatul operației a) este arborele obținut prin ștergerea cheii din arbore; rezultatul operației b) este arborele obținut prin ștergerea din primul arbore a cheilor arborelui al doilea, parcurse în inordine, cu repetarea fiecărei chei conform contorului.

9. Idem pentru ansamble, reprezentate ca vectori, având chei de un tip neprecizat, prevăzut cu o ordine totală (și cu operatorii relaționali, operatorii de ordine fiind corespunzători respectivei ordini totale), cu supraîncărcarea operatorilor indicați mai jos.

=, ==;

<, <=, >, >= (operatorii de ordine întorc 1 sau 0 conform ordonării lexicografice a șirurilor ordonate care rezultă din operanzi);

+ se supraîncarcă în 3 moduri: a) primul operand cheie, al doilea ansamblu b) primul operand ansamblu, al doilea operand cheie; c) ambii operanzi ansamble; rezultatul operațiilor a) b) este ansamblul obținut prin inserarea cheii în ansamblu; rezultatul operației c) este ansamblul obținut prin inserarea în primul ansamblu a cheilor ansamblului al doilea, extrase prin decapități succesive;

– se supraîncarcă în 2 moduri: a) primul operand ansamblu, al doilea operand cheie; b) ambii operanzi ansamble; rezultatul operației a) este ansamblul obținut prin ștergerea unei apariții a cheii din ansamblu, dacă există (aceasta revine la reasamblarea elementelor rămase după ștergerea cheii din vectorul care reprezintă ansamblul); rezultatul operației b) este ansamblul obținut prin ștergerea din primul arbore a cheilor arborelui al doilea, extrase prin decapități succesive.

10. Structură ierarhică de clase privitoare la figuri plane, care conține o clasă abstractă, din care derivă clase care reprezintă: puncte (date prin perechi de coordonate exprimate în pixeli, perechile fiind date cu ajutorul unei structuri definite separat); segmente orizontale (date prin 3 numere reprezentând ordonata comună și cele două abscise ale capetelor, exprimate în pixeli);

segmente verticale (date prin 3 numere reprezentând abscisa comună și cele două ordinate ale capetelor, exprimate în pixeli); dreptunghiuri (date printr-un punct reprezentând vârful din stânga sus și două numere reprezentând laturile dreptunghiului, exprimate în pixeli). Operatorii $+$, $*$ sunt supraîncărcați cu operațiile de translație, respectiv de scalare (adică modificarea fiecărei coordonate a punctelor figurii prin înmulțirea cu un număr real numit factor de scalare atașat respectivei coordonate). Ambele operații au primul operand o figură, iar al doilea operand o pereche de numere reprezentând vectorul de translație, respectiv perechea de factori de scalare, rezultatul fiind o figură. Operațiile se dau ca metode virtuale pure în clasa abstractă și se definesc pentru fiecare clasă reprezentând figuri particulare. Se vor scrie metode de afișare a figurilor, prin supradefinirea unei metode virtuale pure din clasa abstractă, utilizându-se funcții din bibliotecile grafice. Clasa abstractă are o dată membră de tip `int` reprezentând tipul de figură, iar fiecare constructor al unei clase derivate dă acestui câmp o valoare proprie clasei: 0 pentru puncte, 1 pentru segmente orizontale, 2 pentru segmente verticale, 3 pentru dreptunghiuri.

11. Structură ierarhică de clase privitoare la figuri sau grupuri de figuri, unde structura ierarhică pentru figuri este cea de la tema 11, iar grupurile de figuri formează o clasă separată în care un grup se reprezintă ca un vector (pointer) de pointeri la clasa abstractă pentru figuri, construit prin alocare dinamică de memorie corespunzător numărului de elemente ale grupului, acest număr fiind, alături de vector, o dată membră a clasei. Operatorul $+$ se supraîncarcă în 3 moduri: formarea unui grup din două figuri (funcție prietenă); reuniunea dintre două grupuri (funcție prietenă); adăugarea unei figuri la un grup (funcție membră). Operatorul $*$ se supraîncarcă în 2 moduri: intersecția a 2 figuri; intersecția tuturor figurilor dintr-un grup cu toate figurile din alt grup (este vorba de intersecția contururilor), rezultatul fiind întotdeauna un grup (intersecția a două segmente este grupul vid sau un grup cu un punct sau cu un segment; a unui segment cu un dreptunghi este grupul vid sau un grup cu un segment sau cu 1 sau 2 puncte; a două dreptunghiuri este grupul vid sau un grup cu un dreptunghi, 1, 2, 3 segmente, un număr de segmente și puncte, sau 1, 2, 4 puncte; a unui punct cu altă figură este grupul vid sau punctul respectiv). Se vor scrie metode de afișare a figurilor și grupurilor de figuri, utilizându-se funcții din bibliotecile grafice.

Cerințe comune

Toate clasele vor trebui să aibă constructor fără argumente, constructor de copiere, cel puțin încă un constructor, destructor și metode de citire a datelor de la consolă.

Se vor putea propune și alte teme de către studenți, îndeplinind următoarele condiții:

- cerințele comune de mai sus;
- supraîncărcarea mai multor funcții și operatori, dintre care operatorul de atribuire și cel puțin încă 2 operatori;
- se utilizează clase generice sau clase abstracte și moștenire.

La o temă pot lucra până la 2 studenți, urmând ca examinarea și verificarea cunoștințelor să se facă individual.