Specificații proiecte structuri de date 2007 - 2008

Autor	
Cristian Ioniță	cristian.ionita@softmentor.ro

Revizii		
Data	Versiunea	Comentarii
Dec 14 2006	1.0	Versiunea inițială (condiții generale).
Dec 17 2006	1.1	Adăugare specificații pentru teme şi alocare.
Nov 18 2007	1.2	Simplificare teme, corecții minore, alocare teme, actualizare termene.

Informații generale

Obiectivul proiectului este exersarea capacităților în proiectarea şi utilizarea structurilor de date în condițiile unor restricții date. Pentru toate temele se presupune că volumul de date este suficient de mic pentru ca datele să fie reținute integral în memorie. Colecțiile care trebuie prelucrate şi tipurile de operații sunt specificate pentru fiecare temă în parte. Frecvența operațiilor şi dimensiunea relativă a datelor trebuie stabilită de către student.

Toate aplicațiile vor avea o interfață similară, în mod consolă, care va prelua de la intrarea standard comenzi de tipul:

$$nume\_comanda\ param_1\ param_2\ param_3\ \ldots\ param_n$$

Numele comenzii este un identificator care poate conține litere mici şi caracterul „_”. O comandă poate avea un număr oarecare de parametri sau poate să nu aibă nici un parametru (numărul şi tipul parametrilor este specificat în descrierea fiecărei teme). Parametrii pot avea unul din următoarele trei tipuri:

- *Întreg (I)*: o succesiune de k cifre (k ≥ 1) cu prima cifră diferită de 0 precedată opțional de semnul „-”;
- *Double (D)*: o succesiune de k cifre (k ≥ 1) cu prima cifră diferită de 0 precedată opțional de semnul „-”; partea zecimală este delimitată prin ‘.’;
- *Caracter (C)*: o succesiune înconjurată de ghilimele duble.

Fiecare comandă se scrie pe un rând. O comandă nu poate fi scrisă pe mai multe rânduri şi nu se pot scrie mai multe comenzi pe un singur rând.

Comenzile se pot introduce manual sau printr-un fişier de comenzi. În primul caz se execută aplicația şi se introduc comenzile linie cu linie. În cazul folosirii unui fişier de comenzi se creează un fişier text care conține comenzile de executat, după care se foloseşte facilitatea de redirectare a fişierelor standard din sistemul de operare (pentru Windows se porneşte programul folosind comanda ***program.exe <comenzi.txt***). Pentru mediul de dezvoltare Visual Studio 2005 se poate folosi proprietatea proiectului (click dreapta şi selectare *Properties* în *Solution Explorer*) *Debugging → Command Arguments* cu o valoare de forma *<$(TargetPath)\..\1000_0_Ionita_Cristian.txt* pentru a executa un fişier de comenzi inclus în proiect la pornirea executabilului. Un exemplu de fişier de comenzi se află în fişierul *Exemplu\1000_0_Ionita_Cristian.txt*.

Toate proiectele trebuie să suporte următoarele două comenzi (în plus față de cele specificate în cadrul descrierii temei):

salvare numeSalvare(C)

Salvează toate datele problemei în fişiere pe disc în directorul curent. Datele pot fi salvate într-unul sau mai multe fişiere binare sau text.

incarcare numeSalvare(C)

Încarcă de pe disc datele salvate anterior folosind comanda *salvare*. La încarcarea de pe disc se pierd toate datele aflate în acel moment în memorie.

Structura aplicației

Codul sursă al aplicației va fi conținut într-un unic fișier sursă cpp cu denumirea *grupa_nrtema_nume_prenume.cpp*. Fișierul sursă trebuie să se compileze fără erori folosind compilatorul *Microsoft 32-bit C/C++ Optimizing Compiler* versiunea 14.00.

Fișierul sursă va conține următoarele secțiuni:

I. **Declarații generale**: directive *include*, alte declarații generale

II. **Constante**: definirea constantelor utilizate în cadrul aplicației

III. **Definire structuri**: definițiile pentru toate structurile de date utilizate în cadrul proiectului. Toate datele care compun starea aplicației vor fi conținute într-o singură structură denumită *DateProiect*.

IV. **Funcții diverse**: diverse funcții utilizate pentru implementarea mecanismului de citire pentru comenzi și a comenzilor

V. **Implementare operații**: implementarea operațiilor specificate pentru fiecare temă (câte o funcție pentru fiecare operație)

VI. **Programul principal**: mecanismul de citire și apelare comenzi conform modelului.

Pentru implementarea proiectului se va folosi șablonul din fișierul *ModelProiect.cpp*. Pentru un exemplu de completare vezi fișierul sursă corespunzător temei 0 (fișierul *Exemplu\1000_0_Ionita_Cristian.cpp)* și fișierul de diferențe *Exemplu\DiferenteProiect.pdf*.

Definirea structurii de date

Alegerea structurii de date utilizate se va face în funcție de datele și operațiile specificate în descrierea temei, precum și a ipotezelor referitoare la dimensiunile relative ale datelor și frecvenței operațiilor specificate de către student. Nu este necesară folosirea altor structuri de date în afara celor predate în cadrul cursului și seminarului.

Toate datele prelucrate de către aplicație vor fi grupate într-o singură structură denumită *DateProiect*. Această structură va fi transmisă ca parametru în toate funcțiile care implementează operațiile. Nu se acceptă variabile globale.

Implementarea comenzilor

Fiecare comandă din specificația temei va fi implementată sub forma unei funcții în cadrul secțiunii *V. Implementare operații*. În cazul în care este necesară folosirea unor funcții ajutătoare, acestea vor fi adăugate în secțiunea *IV. Funcții diverse*.

Funcțiile care implementează comenzi vor avea ca parametri o referință la structura *DateProiect* și ceilalți parametri de tip *int* / *double* / *char** specificați în descrierea proiectului. În cazul în care comanda acceptă un număr variabil de parametri funcția va avea un parametru de tip vector (în plus față de *DateProiect*).

Valoarea returnată din de către funcții va fi de tipul *CodEroare* și va avea valoarea *Ok* în cazul în care operația s-a executat corect sau un cod de eroare corespunzător în cazul în care operația a eșuat. Cazurile de eroare care trebuie tratate obligatoriu sunt precizate în descrierea comenzii.

Apelarea comenzilor

Apelarea funcțiilor care implementează comenzile se va face din programul principal.

Programul principal trebuie să aibă structura prezentată în model. Singura modificare permisă este adăugarea de secvențe de cod pentru apelarea comenzilor. Conversia parametrilor se va face utilizând funcțiile *ConvertireLaIntreg* si *ConvertireLaDouble*. Erorile întoarse de către aceste funcții trebuie tratate. În cazul în care apare o eroare la conversie aceasta trebuie afișată folosind funcția *AfisareEroare* și oprită execuția comenzii respective.

Structura documentației

Documentația asociată proiectului va fi creată prin completarea formularului din fișierul *ModelProiect.doc* și salvată sub numele *grupa_nrtema_nume_prenume.doc*. Pentru un exemplu de completare vezi fișierul *Exemplu\1000_0_Ionita_Cristian.pdf*.

Cerințe şi modalitatea de predare

Proiectele se predau în format electronic și se prezintă personal la seminar.

Termenul limită pentru predarea proiectelor în format electronic este duminică, 6 ianuarie 2008. Predarea în format electronic se va face prin email la adresa: cristian.ionita@softmentor.ro, cu subiectul de forma *[ProiectSD][Grupa][NrTema] Nume Prenume* (exemplu: [ProiectSD][1069][3] Popescu Ion). Mesajul trebuie să conțină atașate următoarele trei fișiere:

- un fișier sursă C++ cu numele de forma *grupa_nrtema_nume_prenume.txt*
- fișierul cu documentația cu numele de forma *grupa_nrtema_nume_prenume.doc* sau *grupa_nrtema_nume_prenume.pdf*
- un fișier de comenzi cu numele *grupa_nrtema_nume_prenume.txt*

Prezentarea proiectului se va face în cadrul seminarului în data de 11 ianuarie 2008.

Cerințe minime pentru evaluarea proiectului (în cazul în care nu este respectată una dintre aceste cerințe proiectul nu va fi luat în considerare):

- trimitere versiune electronică pe email până la data de 6 ianuarie 2007
- prezentare proiect personal în cadrul seminarului pe data de 11 ianuarie
- respectare format stabilit pentru documentația proiectului
- respectare structură indicată pentru proiect
- codul sursă se compilează și se execută corect cel puțin trei tipuri de comenzi

Punctaj:

- 5p – implementarea comenzilor specificate
- 5p – documentație
- 5p – calitatea implementării

Tema 0: Stocuri (exemplu)

Descriere

Se consideră o gestiune aferentă unui depozit de produse finite. se dorește realizarea unei aplicații pentru evidența stocurilor aflate în depozit. Aplicația trebuie să gestioneze nomenclatorul de produse aflate în depozit, precum și tranzacțiile aferente.

Pentru un produs de cunosc următoarele date: codul produsului, denumirea și prețul de catalog.

Pentru fiecare tranzacție se cunosc următoarele date: tipul (0 - ieșire, 1 - intrare), codul produsului, cantitatea și data operației (un întreg care reprezintă numărul de zile de la data 01 ianuarie 1990). Fiecare tranzacție va primi un număr la momentul înregistrării.

În cazul în care avem atât tranzacții de intrare cât și tranzacții de ieșire în aceeași zi, se consideră că tranzacțiile de ieșire au fost efectuate după tranzacțiile de intrare.

Comenzi

adaugare_produs `CodProdus(I) Denumire(C) Pret(D)`

Adaugă un produs în lista de produse. În cazul în care mai există un produs cu același cod se va afișa un mesaj de eroare.

modificare_produs `CodProdus(I) Denumire(C) Pret(D)`

Modifică datele asociate produsului cu codul specificat. În cazul în care nu există nici un produs cu acest cod se va afișa un mesaj de eroare.

stergere_produs `CodProdus(I)`

Șterge un produs din listă. În cazul în care nu există nici un produs cu acest cod sau există tranzacții înregistrate pentru produsul respectiv se va afișa un mesaj de eroare.

adaugare_tranzactie `Tip(I) CodProdus(I) Cantitate(I) Data(I)`

Adaugă o tranzacție nouă. Se va genera un mesaj de eroare în cazul în care produsul nu există sau stocul la data specificată nu este suficient. Comanda va afișa numărul cu care a fost înregistrată tranzacția.

anulare_tranzactie `NumarTranzactie(I)`

Șterge o tranzacție din lista de tranzacții. Se va afișa un mesaj de eroare în cazul în care nu există nici o tranzacție cu codul respectiv sau anularea tranzacției ar genera un stoc insuficient.

stoc_produs `CodProdus(I) Data(I)`

Afișează pe ecran stocul pentru produsul respectiv la data indicată. În cazul în care nu există nici un produs cu acest cod se va afișa un mesaj de eroare.

stoc_curent

Afișează un tabel cu toate produsele și stocul corespunzător la data curentă. Tabelul va conține următoarele coloane: Cod, Denumire, Preț, Stoc.

fisa_produs `CodProdus(I) DataInceput(I) DataSfarsit(I)`

Afișează datele produsului și istoricul stocului pentru perioada indicată. În cazul în care nu există nici un produs cu acest cod se va afișa un mesaj de eroare.

Tema 1: Angajați

Descriere

O companie dorește să implementeze un sistem informatic pentru gestiunea angajaților organizați în departamente. Pentru fiecare angajat trebuie reținute următoarele date: marca, numele, salariul și codul departamentului din care face parte (sau 0 dacă nu este alocat încă unui departament).

Departamentele companiei formează o structură ierarhică. Datele asociate unui departament sunt: codul departamentului, denumirea, codul departamentului părinte (sau 0 dacă este departament rădăcină) și marca șefului de departament (sau 0 daca nu are un șef desemnat).

Comenzi

adaugare_angajat codAngajat(I) nume(C) salariu(D) codDepartament(I)

Adaugă în listă un angajat cu datele specificate.

modificare_angajat codAngajat(I) nume(C) salariu(D) codDepartament(I)

Caută angajatul pe baza codului și actualizează celelalte date. În cazul în care angajatul nu există se va afișa un mesaj de eroare.

stergere_angajat codAngajat(I)

Caută angajatul pe baza codului și îl șterge din lista de angajați. În cazul în care angajatul nu există se va afișa un mesaj de eroare.

adaugare_departament codDep(I) denumire(C) codDepParinte(I) marcaSef(I)

Adaugă un departament nou cu datele specificate în lista de departamente.

modificare_departament codDep(I) denumire(C) codDepParinte(I) marcaSef(I)

Caută departamentul pe baza codului și îi actualizează celelalte informații. În cazul în care departamentul nu există se va afișa un mesaj de eroare.

stergere_departament codDep (I)

Caută departamentul pe baza codului și îl șterge din lista de departamente. În cazul în care departamentul nu există se va afișa un mesaj de eroare. La stergerea unui departament angajații din interior vor primi codul de departament 0.

top_angajati n(I)

Afișează un tabel cu *n* angajați în (cei care au cele mai mari salarii).

arbore_departamente

Afișează sub formă de arbore departamente din cadrul firmei cu următoarele date pentru fiecare department: Cod, Denumire, Nume Șef (dacă există)

lista_departamente

Afișează lista de departamente cu următoarele informații: cod, denumire, nume șef, număr angajați, valoarea totală a salariilor

fisa_departament

Afișează toate datele despre departament și lista de angajați.

Tema 2: Bibliotecă

Descriere

O bibliotecă dorește o aplicație pentru gestiunea fondului de carte, al cititorilor și al împrumuturilor. În cadrul aplicației trebuie gestionate următoarele colecții de date:

- Cărți: cota, titlu, autori, număr bucăți în total și număr bucăți disponibile
- Cititori: cod, nume
- Împrumut: număr, cota carte, cod cititor, data împrumut, data returnare.

Câmpurile de tip dată sunt stocate ca întreg care reprezintă numărul de zile de la data 01 ianuarie 1990. Cărțile împrumutate trebuie restituite în maxim 15 zile. Un cititor poate avea maxim trei împrumuturi active (au data returnare 0) simultan.

Comenzi

adaugare_carte `cota(I) titlu(C) autori(C) bucTotal(I)`

Adaugă o carte cu informațiile specificate. Numărul de bucăți disponibile inițial este egal cu numărul total de bucăți.

stergere_carte `cota(I)`

Caută cartea pe baza cotei și o elimină din sistem. În cazul în care cartea nu există sau cartea este împrumutată se va afișa un mesaj de eroare.

adaugare_cititor `cod(I) nume(C)`

Adaugă un cititor nou în sistem.

stergere_cititor `cod(I)`

Caută cititorul pe baza codului și îl elimină din sistem. În cazul în care cititorul nu există sau are împrumuturi active se va afișa un mesaj de eroare.

imprumuta `cota(I) codCititor(I) dataImprumut(I)`

Înregistrează împrumutul în sistem și afișează numărul împrumutului. În cazul în care cartea sau cititorul nu pot fi găsiți, cartea nu este disponibilă sau nu sunt respectate regulile de împrumut se va afișa un mesaj de eroare.

restituie `numar(I) dataRestituire(I)`

Inregistrează restituirea împrumutului. Se va genera o eroare în cazul în care împrumutul nu există sau a fost deja restituit (are câmpul data restiruire completat).

lista_carti

Afișează o listă cu toate cărțile existente și informațiile despre ele.

fisa_cititor `codCititor(I)`

Afișează datele despre cititor, lista de împrumuturi active și cea de împrumuturi inactive.

fisa_carte `cota(I)`

Afișează datele despre carte și istoricul împrumuturilor.

Tema 3: Compensare

Descriere

Un grup de companii doreşte să implementeze un sistem pentru compensarea creanțelor. Sistemul trebuie să permită introducerea datelor referitoare la companii şi la facturile existente, să determine circuitele de compensare posibile şi să permită executarea acestora.

Un circuit de compensare reprezintă un lanț de forma C_1 datorează n_1 companiei C_2, care datorează suma n_2 companiei C_3, …, care datorează suma n_{k-1} companiei C_k, care datorează suma n_k companiei C_1. Valoarea circuitului(n) este minimul dintre n_1, n_2, …, n_k. Executarea unui circuit presupune anularea sau diminuarea facturilor între participanți în valoare totală n.

Datele vor fi stocate în două colecții de date: Companii (cod, denumire), Facturi(cod furnizor, cod client, valoare).

Comenzi

adaugare_companie `cod(I) denumire(C)`

Adaugă în sistem o companie nouă cu datele specificate.

stergere_companie `cod(I)`

Şterge compania specificată din sistem. Dacă nu există compania se va genera un mesaj de eroare. La ştergerea unei companii trebuie şterse şi toate facturile în care este implicată.

înregistrare_factura `codF(I) codC(I) valoare(D)`

Adaugă o factură nouă în sistem. După înregistrare se va afişa numărul alocat.

anulare_factura `numarFactura(I)`

Şterge o factură din sistem. În cazul în care factura nu poate fi găsită pe baza numărului se va genera un mesaj de eroare.

compensare_circuit `k(I) cod1(I) cod2(I) cod3(I) ... codk(I)`

Verifică faptul că şirul de coduri formează un circuit (valoarea este mai mare ca 0) şi execută compensarea diminuând sau eliminând facturile corespunzătoare. În cazul în care una dintre companii nu poate fi găsită sau nu există un circuit format din companiile specificate (în ordinea specificată) se va afişa un mesaj de eroare.

fisa_companie `cod(I)`

Afişează datele despre companie şi lista de datorii şi creanțe. În cazul în care compania nu poate fi găsită se va afişa un mesaj de eroare.

lista_companii

Afişează un tabel cu toate companiile din sistem cu următoarele coloane: cod, denumire, total creanțe, total datorii.

Tema 4: Transporturi

Descriere

O companie de transport rutier interurban dorește să-și creeze o aplicație pentru gestionarea curselor între orașe. Aplicația trebuie să permită introducerea informațiilor despre orașe, legături și curse, precum și afișarea rapidă a informațiilor despre acestea.

Colecțiile de date care trebuie gestionate de către aplicație sunt: Orașe (cod oraș, denumire), Legături (cod oraș sursă, cod oraș destinație) și Curse (cod cursă, ora de plecare, lista de orașe).

Se presupune că toate cursele pleacă la oră fixă și distanța dintre două orașe este parcursă într-o oră. Orele sunt în intervalul 0 – 23 și nici o cursă nu trece peste miezul nopții (toate cursele trebuie să se încheie în aceeași zi cu ziua de plecare).

Comenzi

adaugare_oras `cod(I) denumire(C)`

Adaugă în sistem informațiile despre un oraș. În cazul în care mai există un oraș cu același cod se va genera o eroare.

stergere_oras `cod(I)`

Șterge un oraș din sistem. În cazul în care există curse care conțin orașul sau orașul nu există se va genera o eroare. La ștergerea unui oraș se șterg și toate informațiile referitoare la legăturile asociate.

adaugare_legatura `sursa(I) destinatia(I)`

Adaugă o legătură între două orașe în sistem. În cazul unul dintre orașe nu poate fi găsit pe baza codului se va genera o eroare. Dacă mai există o legătură definită între cele două orașe se va ignora comanda. În cazul în care există curse care foloses legătura se va genera o eroare.

stergere_legatura `sursa(I) destinatia(I)`

Șterge o legătură existentă între două orașe. În cazul unul dintre orașe nu poate fi găsit pe baza codului se va genera o eroare. Dacă nu există o legătură definită între cele două orașe se va ignora comanda.

adaugare_cursa `cod(I) oraPlecare(I) k(I) oras1(I) oras2(I) ... orask(I)`

Adaugă în sistem informațiile despre o cursă. În cazul în care mai există o cursă cu același cod sau unul dintre orașe nu poate fi găsit pe baza codului se va genera un mesaj de eroare. Se presupune că există o legătură directă între orașele i și i+1, i=1..k-1; în caz contrar se va genera un mesaj de eroare.

stergere_cursa `cod(I)`

Șterge o cursă din sistem. În cazul în care cursa nu există se va afișa un mesaj de eroare.

tabel_oras `cod(I)`

Afișează o un tabel cu datele orașului, sosirile și plecările de curse și legăturile cu alte orase.

afisare_ruta `codPlecare(I) codDestinatie(I)`

Se va afișa o rută între cele două orașe sau mesajul "Nu există curse între orasele i si j.". În cazul în care unul dintre orașe nu poate fi găsit pe baza codului se va afișa un mesaj de eroare. Se vor lua in consiretare doar cursele directe.

Alocare teme – Grupa 1065

Nr	Nume şi prenume	Tema
1	COSMA A ANITA GEORGIANA	1
2	CRACIUN I MIHAIL MARIO	2
3	CRAITA E MIHAI COSMIN	3
4	CRISMARU P IRINA NICOLETA	4
5	CRISTEA I ROXANA DANIELA	1
6	CURCULESCU G I BOGDAN IONUT	2
7	CURTEANU M MIHAI	3
8	DAVID C LIVIA ANCA	4
9	DESPA F MIHAI LIVIU	1
10	DIACONU I IONELA SIMONA	2
11	DIMA C CATALIN	3
12	DIMIAN M RAZVAN MARIAN	4
13	DINA I MARIA TEODORA	1
14	DINIASI C LUCIAN ALEXANDRU	2
15	DOBRE A VLAD GABRIEL	3
16	DOBRE D DRAGOS DANIEL	4
17	DOBRE P FLORIN CEZAR	1
18	DOBRIN V IONUT MADALIN	2
19	DOMNITEANU I LIANA	3
20	DOROBANTU M LIVIU CATALIN	4
21	DRAGAN I SIMONA CLAUDIA	1
22	DRAGAN T DAN ALEXANDRU	2
23	DRAGOMIR C STEFAN	4
24	DRANGU E MIHAI VALENTIN	1

Alocare teme – Grupa 1075

Nr.	Nume şi prenume	Tema
1	THEODORESCU S MIHAELA	1
2	TOADER A L ELENA ALEXANDRA	2
3	TOADER D CONSTANTIN MARIAN	3
4	TOADER D D ADINA DIANA	4
5	TOADER M ECATERINA LACRAMIOARA	1
6	TOLBARU A ELENA VERONA	2
7	TOMA M MARIAN STEFAN	3
8	TOMA S DIANA ALEXANDRA	4
9	TOMITA C IGOR	1
10	TOMULESCU A ALEXANDRA	2
11	TONCA O GEANINA ELENA	3
12	TRIF V RALUCA MIHAELA	4
13	TRIFAN D GEORGE MIHAI	1
14	TROGMAER N CRISTIAN ALIN	2
15	TUDOR I IONELA	3
16	TUDOR M DANIEL IONUT	4
17	TUDOR S OANA DANIELA	1
18	TUDORA M ISABELA ANDREEA	2
19	TUREAC G LIVIU	3
20	UNTILA V ANDREI	4
21	URSEI P MARIA ELENA	1
22	URSU V ADRIANA ANDREEA	2