

proces sau funcție a întreprinderii, date de detaliu cu un anumit grad de volatilitate (pot suferi actualizări) și prezentând interes în primul rând pentru cei ce le introduc. Datele stocate în depozite sunt date pentru asistarea deciziei, referitoare la subiecte de interes decizional, sunt date centralizate sau derivate din datele operaționale, nu se schimbă în timp și sunt orientate către utilizatorii finali - managerii de nivel tactic și strategic. Putem spune că bazele de date utilizate de sistemele operaționale sunt orientate spre tranzacții și reflectă situația curentă în timp ce depozitele de date utilizate de sistemele de asistare a deciziei sunt orientate spre subiectele analizelor și reflectă situații globale, cu caracter istoric.

- Performanțele cerute în cazul sistemelor tranzacționale se referă în special la integritate, siguranță, confidențialitate, trasabilitate și timp de răspuns, având în vedere faptul că un număr foarte mare de utilizatori introduc date primare în sistem. Concurența în utilizarea sistemelor de asistare a deciziei este foarte redusă, numărul de manageri - utilizatori finali fiind foarte mic. De asemenea, securitatea și siguranța în exploatare nu sunt expuse unor riscuri majore, procedurile de salvare și recuperare fiind mult mai relaxate față de cazul sistemelor tranzacționale.
- Procesarea datelor în sistemele tranzacționale se aplică unui set mic de date- de regulă introduse recent și stocate compact în cel mult câteva tabele- fiind în consecință foarte rapidă, în timp ce fundamentarea unei decizii necesită procesarea unui volum foarte mare de date stocate dispersat, fiind în consecință foarte lentă.
- Bazele de date ale sistemelor tranzacționale sunt proiectate și realizate pe baza unor cerințe cunoscute în prealabil, adaptarea sistemului la cerințe ulterioare necesită reluări ale unor faze din ciclul de viață și de regulă, sistemul o dată dat în exploatare funcționează fără modificări majore o lungă perioadă. Sistemele de asistare a deciziei evoluează în timp într-o manieră incrementală, cerințele nu sunt cunoscute în totalitate în momentul proiectării și realizării sistemului. În consecință, depozitul de date va trebui să se adapteze mereu cerințelor. Datele stocate în sisteme tranzacționale sunt gestionate ca un întreg pe când datele stocate în sistemele de asistare a deciziei pot fi gestionate și pe secțiuni întrucât sunt organizate distinct pe subiecte de analiză.
- Sistemele tranzacționale urmăresc fluxul datelor din activitatea curentă, sunt orientate spre procese ca de exemplu vânzări, achiziții, încasări, plăți, producție, etc. Depozitele de date sunt organizate și gestionate având în vedere scopul final al analizelor, sunt orientate spre subiecte ca de exemplu clienți, furnizori, resurse, produse, etc.. Afacerea propriu-zisă, procesele întreprinderii care stau la baza modelării și proiectării sistemelor informatice operaționale, nu influențează designul depozitului de date, nu sunt reflectate în structura sau comportamentul acestuia.

STRUCTURI DE DATE

Arbori. Reprezentare, parcurgere.

(Bădică, A., Structuri de date, Note de curs, pag. 173-177)

Listele sunt potrivite pentru reprezentarea datelor organizate liniar. Dacă însă dorim să descriem date structurate ierarhic, simpla enumerare a obiectelor componente cu ajutorul unor liste este insuficientă.

Organizarea datelor sub formă ierarhică este frecvent întâlnită în cele mai diverse domenii aplicative. Câteva exemple sunt: organizarea administrativă sau managerială a unei

societăți, planificarea meciurilor unei competiții sportive de tip turneu, structurarea unei cărți sau a directorului de fișiere dintr-un sistem de operare, reprezentarea expresiilor aritmetice și logice într-un compilator, în vederea evaluării lor. Generalizând, orice entitate poate fi descrisă la un nivel abstract printr-un obiect primitiv sau la un nivel detaliat sub forma unei ierarhii de obiecte componente.

Pentru definirea structurilor arborescente o atenție importantă trebuie acordată relației de "ramificare" sau echivalent "subordonare".

Se numește arbore o mulțime A de unul sau mai multe noduri astfel încât:

- i) există un nod special $\text{rad}(A) \in A$ numit rădăcina arborelui A
- ii) mulțimea celorlalte noduri din A cu excepția rădăcinii este partiționată în $n \geq 0$ mulțimi nevide și disjuncte $A_i; 1 \leq i \leq n$, care sunt la rândul lor arbori. $A_1 \dots A_n$ se numesc subarborii lui A . Dacă ordinea relativă a subarborilor $A_1 \dots A_n$ în punctul ii) al definiției este importantă, spunem că A este un arbore ordonat. În acest caz A_i este al i -lea subarbore al lui A pentru $1 \leq i \leq n$.

Dacă pentru a distinge între doi arbori se consideră că ordinea subarborilor este nerelevantă, atunci arborii se numesc orientați pentru a sugera faptul că are importanță doar orientarea arcelor (de la rădăcină spre rădăcinile subarborilor,) nu și ordinea relativă a acestora. Arborii orientați sunt un caz particular de grafuri orientate. Ei trebuie deosebiți de arborescențe, care sunt un caz particular de grafuri neorientate.

Structurile arborescente pot fi reprezentate grafic sau alfanumeric în diverse moduri care sugerează în fapt o aceeași structură.

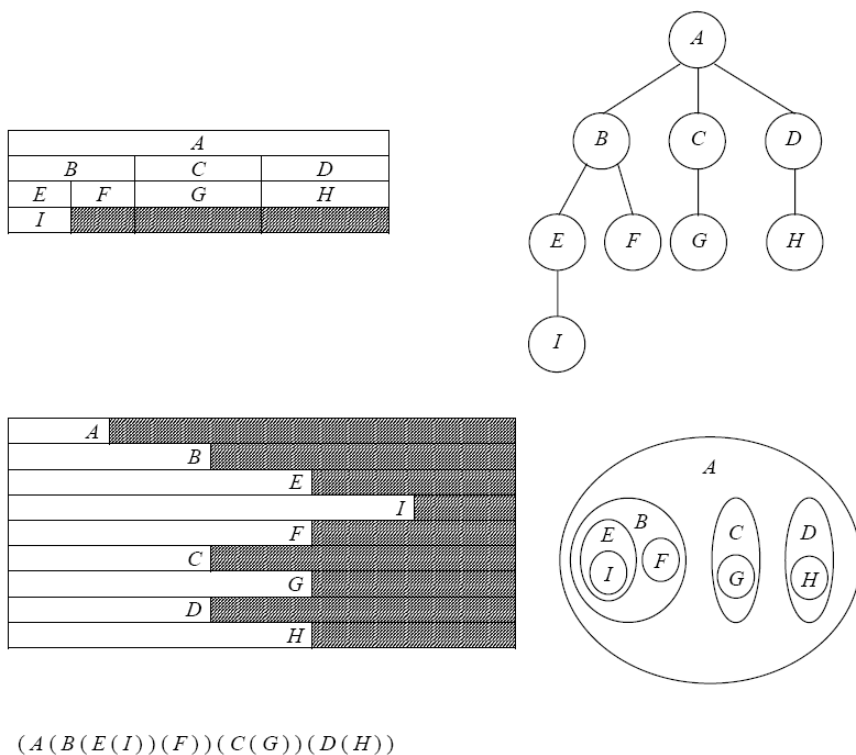


Figura 5.5: Reprezentări posibile ale structurilor arborescente

Maximul gradelor subarborilor corespunzători fiecărui nod al unui arbore se numește aritatea arborelui. Nodurile cu gradul egal cu 0 se numesc noduri frunză sau noduri terminale. Celelalte noduri se numesc noduri neterminale.

Rădăcinile subarborilor unui arbore cu rădăcina X se numesc copiii sau fiii nodului X . Orice nod este tatăl sau părintele fiilor săi. Copiii unui aceluiași nod părinte se numesc frați.

Pentru orice nod X al unui arbore există o cale unică ce unește rădăcina R a arborelui cu nodul X . Toate nodurile de pe această cale, exceptându-l pe X , formează mulțimea strămoșilor lui X . Numărul strămoșilor lui X plus 1 reprezintă nivelul nodului X .

Prin înălțimea sau adâncimea unui arbore vom înțelege nivelul maxim al nodurilor sale.

O mulțime de $n \geq 0$ arbori distincți se numește pădure. Dacă ordinea arborilor unei păduri este relevantă atunci avem o pădure ordonată.

Un caz special de arbore frecvent întâlnit în diverse domenii aplicative este arborele binar. Într-un astfel de arbore fiecare nod are maxim doi subarbori; mai mult, când este prezent un singur subarbore, se face distincție între subarborii stâng și subarborii drept. Se numește arbore binar o mulțime finită de noduri care fie este vidă, fie constă dintr-o rădăcină și elementele a doi arbori binari disjuncți numiți subarborii stâng și subarborii drept ai rădăcinii.

Prelucrarea informațiilor dintr-un arbore implică parcurgerea arborelui. Se numește parcurgere o metodă de examinare sistematică a nodurilor unui arbore astfel încât fiecare nod să fie vizitat exact o singură dată. Parcurgerea arborilor ne oferă o aranjare liniară a nodurilor, astfel încât în orice moment vom ști precis care este următorul nod care va fi vizitat și prelucrat.

În cazul parcurgerii în lărgime se prelucrează mai întâi informația din rădăcini, în ordinea în care arborii apar în cadrul pădurii, după care sunt prelucrate, de la stânga la dreapta, nodurile aflate pe primul nivel în arborii pădurii, apoi cele aflate pe al doilea nivel, ș.a.m.d. Parcurgerea în lărgime presupune folosirea unei cozi. Inițial coada este vidă. Dacă atât coada cât și pădurea sunt vide nu se efectuează nici o prelucrare. Dacă pădurea este nevidă, se depune primul arbore al pădurii în coadă și parcurgerea continuă cu restul pădurii și noua coadă. Altfel (pădurea este vidă), dacă coada este nevidă, se extrage un arbore din coadă, se vizitează rădăcina și se continuă parcurgerea cu pădurea subarborilor arborelui extras din coadă și cu restul cozii.

În cazul parcurgerii în adâncime copiii unui nod sunt vizitați tot de la stânga la dreapta, însă trecerea de la nodul curent la fratele din dreapta are loc numai după ce a fost parcurs tot subarborii cu rădăcina în nodul curent. Pentru a memora informațiile relative la punctul de revenire (nodul tată și următorul fiu neprelucrat al său) se folosește o stivă. Operația de parcurgere în adâncime decurge absolut la fel ca și cea de parcurgere în lărgime, cu deosebirea că în locul operațiilor cu cozi se folosesc operații cu stive. În plus, la parcurgerea în preordine rădăcina este prelucrată înaintea parcurgerii subarborilor săi, iar la parcurgerea în postordine rădăcina este prelucrată după parcurgerea subarborilor săi.

În cazul arborilor binari este consacrată parcurgerea în adâncime. Datorită faptului că un arbore binar este compus din 3 componente: rădăcina R, subarborii stâng S și subarborii drept D, rezultă că se pot defini $3! = 6$ parcurgeri în adâncime, simbolizate sugestiv prin următoarele mnemonice: SRD, RSD, SDR, DSR, DRS, RDS. Dintre acestea, doar primele 3 au denumiri consacrate și anume: inordine, preordine și respectiv postordine. Prefixele in, pre și post sugerează ordinea relativă de vizitare a rădăcinii în raport cu cei doi subarbori.

Arborii binari se reprezintă fie utilizând alocarea secvențială, fie utilizând alocarea înlanțuită.

Tipuri de date, clasificare, tip abstract de date, tipuri primitive (Bădică, A., Structuri de date, Note de curs, pag 4-7)

Orice constantă, variabilă, expresie sau funcție se caracterizează printr-un anumit tip de date. Prin tip de date se înțelege o mulțime de elemente numite valori sau obiecte, care pot clar distinge între ele. Într-un limbaj de programare, orice constantă aparține unui anumit tip de date, orice variabilă poate stoca valori de un anumit tip, orice expresie se evaluează la o valoare de un anumit tip și orice funcție întoarce o valoare de un anumit tip.

După modul de definire în raport cu un anumit limbaj de programare, tipurile de date se clasifică în:

- tipuri predefinite - limbajul permite folosirea variabilelor de tipul respectiv și furnizează o mulțime de operații predefinite pentru manipularea acestor variabile în cadrul unui program.
- tipuri definite de utilizator

Obiectele unui limbaj de programare sunt:

- constante: tipul rezultă din forma sintactică (modul de scriere) a lor
- variabile: tipul rezultă prin specificarea în program a unui enunț special – declarație
- expresii: tipul rezultă din tipurile operanzilor și modul lor de compunere cu ajutorul operatorilor.
- funcții: tipul rezultă din declarația funcției (în C - prototip sau semnătură).

Declarația este un enunț care asociază o variabilă sau o funcție cu tipul său. Definiția este un enunț care asociază o variabilă sau funcție cu tipul său și alocă spațiu de memorie variabilei sau corpului funcției respective.

Dupa modul de definire în termenii altor tipuri de date, tipurile de date se clasifică în:

- tipuri primitive
- tipuri derivate

Un obiect poate fi:

- compus: se poate descompune în mai multe componente
- simplu sau scalar.

La rândul său, un tip de date poate fi:

- compus: valorile sale sunt compuse
- simplu sau scalar

Un tip de date este ordonat dacă pe mulțimea valorilor sale se poate defini o relație de ordine. În caz contrar, se spune că este neordonat. Un tip de date este ordinal dacă pe mulțimea elementelor sale se poate defini o relație de ordonare liniară (este posibilă definirea predecesorului și succesorului unui element). Un tip ordinal modelează o mulțime finită sau numărabilă.

Cardinalitatea unui tip T reprezintă numărul de valori distincte aparținând tipului T.

O structură de date se definește ca mulțimea de obiecte care alcătuiesc un anumit tip, proprietățile obiectelor, relațiile dintre obiecte și operațiile asupra obiectelor. Definiția unei structuri de date trebuie să conțină trei elemente:

- numele structurii
- operațiile asupra structurii
- proprietățile structurii

Operațiile de bază aplicabile obiectelor unei structuri de date sunt:

- constructori – se referă la modul de generare a noi obiecte aparținând structurii respective;
- destructori – se utilizează atunci când gestiunea obiectelor se face explicit de către programator; eliberează memoria alocată obiectelor;
- selectori - se referă la modul de accesare a elementelor care compun un obiect compus;
- modificatori – se referă la modul de actualizare a elementelor care compun un obiect compus;
- recunoscători – se utilizează pentru recunoașterea diverselor proprietăți ale obiectelor aparținând structurii respective;
- testori – se utilizează pentru compararea obiectelor aparținând structurii respective

Proprietățile structurii nu descriu modul în care trebuie implementate operațiile structurii. Cu toate acestea, proprietățile unei structuri de date exprimă maniera în care trebuie să funcționeze operațiile structurii, indiferent de modul lor de implementare. Cu alte cuvinte, proprietățile abstractizează operațiile de modul lor de implementare. Din acest motiv, noțiunea de structură de date este cunoscută și sub numele de tip abstract de date, iar această manieră de definire a structurilor de date se numește abstractizarea datelor.

Formal, o structură de date poate fi definită ca un tuplu $S = \{D, d, O, P\}$, unde:

- D este o mulțime de tipuri de date implicate în definirea lui S
- $d \in D$ este tipul de date al obiectelor structurii S
- O este mulțimea operațiilor definite pe structura S
- P este mulțimea proprietăților structurii S

O implementare a unei structuri de date S este o transformare care definește modul de

reprezentare a obiectelor aparținând tipului de date al structurii în funcție de obiectele altei structuri de date T. Fiecare operație a lui S trebuie definită în termenii operațiilor lui T

Cele mai întâlnite tipuri primitive în limbajele moderne de programare sunt:

a) tipuri numerice: ele sunt reprezentări ale unor mulțimi de numere cunoscute din matematică, precum N , Z sau R . Vom avea în consecință date întregi fără semn, întregi cu semn sau reali. Numerele întregi se reprezintă intern în calculator sub forma unei succesiuni de biți, cu ajutorul codului complement (cod complement față de 2). Numerele reale se reprezintă intern în calculator cu ajutorul reprezentării în virgulă mobilă (flotantă), care cuprinde semnul, mantisa și exponentul. Pentru a reprezenta corect și numerele subunitare, se utilizează reprezentarea cu semn, caracteristică și exponent.

b) tipul caracter: este destinat reprezentării caracterelor alfanumerice. Tipul caracter modelează mulțimea finită a tuturor caracterelor afișabile. Din nefericire, nu există un set standard de caractere specific tuturor sistemelor de calcul. Din acest motiv, termenul "standard" trebuie interpretat în acest context ca referindu-se la sistemul de calcul particular pe care este executat programul în cauză. Cel mai folosit set "standard" de caractere este cel definit de Organizația Internațională pentru Standarde (ISO) și anume versiunea sa americană (codul ASCII), care constă din 128 de caractere, dintre care primele 33 sunt caractere de control și celelalte 95 sunt caractere afișabile. Caracterele sunt codificate intern prin valori întregi numite coduri. Deoarece codul ASCII are 128 de caractere, pentru codificarea caracterelor sale sunt suficienți 7 biți. Cu toate acestea se folosește pentru codificare un octet, deoarece memoria internă a sistemelor de calcul este structurată de obicei sub forma unei secvențe de locații cu dimensiunea de un octet. Pentru seturile cu peste 256 de caractere, a fost introdusă specificația standard Unicode (cum este cazul limbii chineze sau japoneze), pentru reprezentarea căreia se folosesc doi octeți.

c) tipul logic: este destinat reprezentării mulțimii valorilor logice fals și adevărat.

d) tipul enumerare: este destinat reprezentării unei mulțimi finite de valori, fiecare valoare fiind desemnată printr-un nume simbolic. Se utilizează de obicei atunci când este necesară definirea unui tip de date prin indicarea explicită a elementelor sale. Elementele unui tip enumerare se reprezintă intern prin codificare cu ajutorul unor constante întregi.

e) tipul subdomeniu: este destinat reprezentării unui interval al unui tip ordonat liniar. Tipurile subdomeniu se utilizează de obicei pentru reprezentarea indicilor tablourilor.

f) tipul referință: este destinat reprezentării adreselor altor obiecte. Tipul referință se utilizează de obicei pentru implementarea unor obiecte abstracte complexe cum sunt listele și arborii. Numeroase prelucrări se referă la relațiile dintre obiectele prelucrate și nu doar la valorile lor. În astfel de cazuri poate fi necesară reprezentarea explicită a acestor relații. Pentru aceasta anumite obiecte trebuie să se poată referi la alte obiecte. Este posibil chiar ca un același obiect să fie referit din două sau mai multe locuri. Întrucât copierea valorii obiectului în locul de unde a fost referit nu este o soluție acceptabilă din cauza consumului mare de memorie și dificultăților de menținere a consistenței, rezultă că trebuie apelat la o altă tehnică și anume de a defini în program obiecte care să poată referi alte obiecte. Pentru aceasta se folosesc tipul referință și obiectele pointer. O referință la un obiect se implementează ca adresă a locației de memorie, corespunzătoare obiectului respectiv.

INGINERIA PROGRAMĂRII

Sabloane de proiectare. Definiții, categorii, utilizare.
(Bădică, A., Ingineria programării, Note de curs, pag. 201-204)

Proiectarea orientată pe obiecte a software-ului presupune identificarea de obiecte, abstractizarea lor în clase de granularitate potrivită, definirea interfețelor și ierarhiilor de moștenire, stabilirea relațiilor între aceste clase. Soluția trebuie să rezolve problema și să fie în același timp suficient de flexibilă pentru a rezista la noi cerințe și probleme ce pot apărea în timp.