JOHDANTO

Tähän uusittuun painokseen on lisätty useita uusia algoritmeja ja ongelmanratkaisuja sekä STL-kirjaston hyödyntämismalleja. Liitteeksi on nyt lisätty toivomusten mukainen C++-ohjelmointikielen opas, josta lukija voi palauttaa mieleen kielen syntaksiin ja rakenteisiin liittyviä seikkoja.

Tämän kirjan tarkoituksena on toimia ohjelmoijien ja ohjelmoinnista kiinnostuneiden aarrearkkuna, josta voi poimia suoraan ratkaisuja moninaisiin ohjelmointiongelmiin. Kirja antaa vinkkejä ja menettelytapoja sekä ohjelmoinnin että muiden aineiden opettajille. Jotkut ongelmaratkaisut on toteutettu aina valmiisiin ohjelmiin saakka. Joihinkin ongelmiin on annettu pelkästään algoritmi, josta voi johtaa ohjelman millä ohjelmointikielellä tahansa. Mukaan on otettu sekä monimutkaisia että suppeita ohjelmanratkaisuja.

Kirja sisältää satakunta erilaista algoritmia, ongelman ratkaisua ja ohjelmaesimerkkiä. *Algoritmilla* tarkoitetaan yleensä yksikäsitteisten lauseiden muodostamaa virtauskaaviota, jolla ratkaistaan jokin ongelma. Lauseet voivat olla esimerkiksi loogisia tai matemaattisia operaatioita. Algoritmit ja tietorakenteet muodostavat ohjelman. Algoritmia voidaan pitää ohjelmoinnin ytimenä; sen nopeus, tarkkuus ja luotettavuus sekä kyky ratkaista ongelma ovat koko ohjelman avainasioita. Algoritmit kuvaavat myös tietokoneen toimintatapaa.

Algoritmeja muodostetaan usein vaikeimpien ja monimutkaisimpien osaongelmien ratkaisuihin. Tärkeää on kuvata ongelman ratkaisu siinä muodossa, jossa tietokone voi sen ymmärtää ja täten kykenee sen ratkaisemaan. Tietokoneen avulla voidaan suorittaa etenkin raskaita iterointeja, jotka usein sisältävät toistuvia laskentarutiineja.

Ennen algoritmin kehittämistä tulee ongelma kuvata tarkasti. Kuvaamisessa kaavio helpottaa ongelman osittelua pienempiin osaongelmiin. Monimutkaiselta tuntuva ongelma tulee kaavion avulla jäsenneltyä, jolloin saadaan hyvä pohja algoritmin kehittämiselle. Ongelmaa voi lähestyä myös suppean esimerkin valossa, jolloin päästään heti ikäänkuin käytännöllisemmälle tasolle. Algoritmiinhan liittyy yleensä aina tietty abstraktisuus. Ensimmäinen algoritmiversio on harvoin aivan täydellinen. Kehittämistä tukee algoritmin testaus esimerkiksi manuaalisesti pienellä aineistolla simuloimalla.

Saman ongelman ratkaisuun kehitetyt algoritmit ovat harvoin myöskään aivan samanlaisia: toinen on ehkä kirjoitettu suppeammalla koodilla ja nopeammaksi kuin joku toinen. Esimerkiksi viivan piirtämiseen tietokoneen näytölle on kehitetty useita erilaisia algoritmeja, joissa on etunsa ja haittansa: jotkut algoritmit muodostavat

tarkan viivan, mutta ovat tehottomia, kun taas toiset ovat yksinkertaisia, mutta nopeita. Eri käyttöön tarvitaankin saman ongelman kohdalla erilaisia algoritmeja.

Algoritmi voidaan kuvata joko *ohjelmointikielellä* itsellään, *vuokaaviolla* tai *pseudokielellä* (luonnoskielellä, joka on selväkielinen ja kuvaa samalla ohjelman kulkua). Tässä teoksessa on käytetty algoritmien kuvaamiseen sekä pseudokieltä että C++ -kielisiä lauseita, mutta ohjelmoinnin ilo on joskus jätetty ohjelmoijalle itselleen. Sen sijaan on laadittu johdatus algoritmin kehittämiseen. Monen ongelman kohdalla on esitetty myös manuaalisia simulointeja, jotka selventävät ja havainnollistavat ongelma-alueen ratkaisua ja algoritmin muodostamista. Täten erityisesti C++-ohjelmointikielellä laaditut algoritmit tulee sovittaa lukijan käyttämälle ohjelmointikielelle. Tämä ei tuottane aloittelevallekaan ohjelmoijalle vaikeuksia.

Tietotekniikan parissa työskentelevien on tunnettava ohjelmointikieliä sekä osattava löytää ja kehittää erilaisia algoritmeja ongelmien ratkaisemiseksi. Tämä on perusta, jonka avulla opitaan paremmin ymmärtämään uusia työympäristöjä ja työkaluja. Vaikka käytössä olisikin esimerkiksi valmis, pitkälle kehitetty sovelluskehitin, tarvitaan tietämystä, jolla kehittimellä generoitua koodia voidaan tarkastella. Samoin on helpompi ymmärtää erilaisten valmiiden työvälinekirjastojen käyttöä ja hyödyntämistä, kun pohjalla on ohjelmointikielen ja algoritmien osaaminen. Algoritmit kuvaavat lisäksi usein tietokoneen tapaa toimia ja suorittaa tehtäviä; siksi niiden tärkeyttä ei voida aliarvioida.

Teos on tarkoitettu kaikille ohjelmoinnista kiinnostuneille: sekä ohjelmoinnin opiskelijat että ammattiohjelmoijat saanevat tukea teoksen algoritmeista. Teoksesta löytää nopeasti yleisimmin tarvittavat algoritmit, jolloin aikaa säästyy runsaasti itse ohjelmointityöhön. Ohjelmoinnin opettajalle teos tarjoaa hyvän pohjan. Opettaja voi esimerkiksi löytää hyviä esimerkkejä harjoitustöihin. Yksi erinomainen menettely on antaa oppilaiden kehittää algoritmi jollain muulla ohjelmointikielellä kuin kirjassa käytetyllä. Esimerkiksi C++-kielellä laadittuja algoritmeja voidaan antaa oppilaiden tehtäväksi java-kielellä ja pseudokielisiä algoritmeja voidaan taas antaa oppilaiden muunnettavaksi mille tahansa opetettavalle ohjelmointikielelle. Tärkeää on oppia kehittämään ja ymmärtämään algoritmeja.

Kun alat ratkaista ohjelmointiongelmaa, mieti ensin, kuinka ongelma ratkaistaisiin manuaalisesti, ilman tietokonetta; ehkäpä siitä poikii ajattelutapa, joka sopii tietokoneelle. Listausten hyödyntäjän on useimmiten räätälöitävä peruslistauksia, jotta ne palvelisivat kunkin omia tarpeita.

Teos kuuluu jokaisen tietotekniikasta kiinnostuneen ja jo kokeneen ammattilaisen käyttöön. Teoksen kirjoittaja on valinnut algoritmien aiheet laajahkolta sektorilta ajatellen sekä ohjelmoijia että muita ohjelmoinnista ja tietotekniikan hyödyntämisestä kiinnostuneita.