hyväksymispäivä arvosana

arvostelija

Ohjelmointikielien kehittyminen

Hannu-Pekka Rajaniemi

Helsinki 28.4.2006

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Hannu-Pekka Rajaniemi

Työn nimi — Arbetets titel — Title

Ohjelmointikielien kehittyminen

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	28.4.2006	13 sivua

Tiivistelmä — Referat — Abstract

Ilman kunnollisia ohjelmointikieliä tämän päivän huipputietokoneillakaan ei kykenisi tekemään kovin monimutkaisia asioita. Ohjelmointikielet ovat tulleet jo varsin kauas sitten 1940-luvun ja 1950-luvun alun konekieliohjelmoinnin. Kääntäjät ja tulkit vapauttivat ohjelmat laitteistoriippuvuuden kahleista ja samalla paransivat huomattavasti ohjelmoijien tuottavuutta. Ohjelmointikielten kehitys ei kuitenkaan ole ollut kovin suoraviivaista tai selkeää. Tämän paperin tarkoituksena onkin esitellä ohjelmointikielten kehittymistä.

ACM Computing Classification System (CCS):

K.2.c [History of Programming: Software] D.3.0 [Programming Languages: General]

Avainsanat — Nyckelord — Keywords
Ohjelmointikielet, ohjelmointikielten historia

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Viimeisen vähän yli puolen vuosisadan aikana tietokoneet ovat mullistaneet maailman. Kun vielä 40- ja 50-luvuilla tietokoneet olivat hallin kokoisia massiivisia monesti yhteen alaan erikoistuneita aparaatteja, ja niitä löytyi vain muutamasta harvasta paikasta maailmalla ovat nykyiset tietokoneet pieniä ja niitä on kaikkialla. Tietokoneita löytyy nykyisin niin autoista kuin vaatteistakin. Tietokoneet hoitavat nykyisin puolestamme lähes kaiken laskentaa vaativan toiminnan, valvovat kriittisiä järjestelmiä ja pystyvät sisältämään taskukoossa enemmän tietoa kuin ihminen pystyy elinikänsä aikana oppimaan.

Kuitenkin tietokoneet itsessään osaavat suorittaa vain mekaanisia varsin yksinkertaisia binääriaritmeettisia operaatioita. Ilman hyviä ohjelmia maailman tehokkaimmastakaan tietokoneesta ei olisi kuin laskimeksi. Ohjelmointi onkin tästä näkökulmasta eräs maailman tärkeimmistä tietotaidoista.

Vielä UNIVAC-koneiden aikana ohjelmointi tehtiin puhtasti konekielellä, joka on parhaimmillaankin hidasta, kallista, virhealtista ja suunnattoman monimutkaista vähänkään suurempaa ohjelmaa tehtäessä. Nykypäivänä ohjelmointi tapahtuu useimmiten korkean tason ohjelmointikielillä, jotka ovat parhaimmillaan täysin alustariippumattomia ja verrattuna konekieliin äärimmäisen ilmaisuvoimaisia.

Korkean tason ohjelmointikielten kehittyminen on ollut monimutkainen prosessi. Suurin osa innovaatioista on syntynyt, kun ihmiset ovat turhautuneet vanhojen kielten ja niiden käyttämien menetelmien heikkouksiin tai suoranaisiin suunnitteluvirheisiin. Tästä syystä ohjelmointikielien kehittyminen ei ole ollut suoraviivaista ja selkeää, sillä mielipiteet paremmista menetelmistä ovat menneet monesti ristiin. Tämä selittää osittain myös sen, miksi viimeisen 60 vuoden aikana on kehitetty tuhansia ja taas tuhansia erilaisia ohjelmointikieliä.

Tässä seminaaripaperissa pyritään käymään yleisellä tasolla läpi ohjelmointikielten kehittymistä konekielistä tämän päivän uusiin korkean tason kieliin. Samalla paperissa pyritään tuomaan esille kirjoittajan mielestä ohjelmointikielten kehitykseen suuresti vaikuttaneita innovaatioita.

2 Kääntäjät ja tulkit

Viime vuosisadan puolessa välissä tietokoneet olivat kalliita, epäluotettavia, hitaita ja erittäin pienimuistisia. Koneita valmistettiin vielä 50-luvulla yksittäisinä kappaleina, kunnes UNIVAC-koneet tulivat markkinoille ja loivat ajatuksen massatuotantona valmistettavista tietokoneista. [Tun06a]

Tähän aikaan ohjelmointi tapahtui konekielillä. Konekielillä ohjelmoiminen oli hankalaa varsinkin jos kyseessä oli laajempi ohjelmisto, ja tutkijat kokivat nopeasti tarpeelliseksi kehittää korkeamman tason kieliä, joilla pystyisi ilmaisemaan algoritmeja enemmän luonnollisia kieliä muistuttavilla ohjelmointikielillä. Ensimmäinen tällainen kieli oli Short Code, jonka kehitti John Mauchly 1949. Short Coden kääntäminen tehtiin aluksi käsin, mutta myöhemminse käännettiin myös UNIVAC 1-alustalle jossa sillä luotua koodia ajettiin tulkin avulla. Short Coden ongelmana oli kuitenkin se, että se oli tulkittuna n. 50 kertaa hitaampi kuin konekielisenä kirjoitettu vastaava ohjelma. [Mal98]

Short Code toi esille konkreettisesti idean ohjelmakoodin tulkkaamisesta ja kääntämisestä konekielelle sen sijaan, että ohjelma kirjoitettaisiin puhtaasti konekielellä. Se, että Short Code oli erittäin hidas ei kuitenkaan lannistanut muita kehittämästä vastaavanlaisia järjestelmiä. Toisaalta näin saattoi käydä myös sen vuoksi, että Short Code ei ollut koskaan erityisen suosittu, eikä siitä edes ole jäänyt suuremmin tietoa jälkipolville.

Koska korkeamman tason kielien tulkitseminen oli niin hidasta, oli vain luonnollista, että tulkkauksen sijasta joku kehittäisi kääntäjän. Kääntäjän etuna tulkkiin on valmiiksi korkeamman tason kielestä konekieliseksi käännetty ohjelmakoodi. Näin ohjelman nopeus riippuu enimmäkseen kääntäjän tuottaman konekielisen koodin laadusta.

Jo vuonna 1951 Grace Hopperin johtama ryhmä loi A-0 - nimisen ohjelmointikielen matemaattisten ongelmien ratkaisuun. Se oli ensimmäinen kieli, jolle luotiin kääntäjä. [Sam69] A-0 oli kuitenkin varsin hankalalukuista, ja se sekä sen jatkoversiot implementoitiin ainoastaan UNIVAC-1 ja -2-koneille eikä se saavuttanut koskaan erityisen suurta suosiota.

Kääntäjien ongelmana oli niiden luoman koodin hitaus. Koska koneet olivat hitaita, koodin nopeudella oli suuri merkitys kääntäjien suosioon. Ensimmäiset kääntäjät olivat varsin kömpelöitä ja tuottivat varsin hidasta koodia. Tästä syystä kääntäjiä opittiin jäsentämään nopeasti ja varsin nopeasti niiden toimintaa saatiin optimoitua

riittävästi, jotta ne kykenivät tuottamaan nopeudeltaan kilpailukykyistä koodia. [Knu62]

Yksi suurimmista kääntäjiä eteenpäinvieneistä kielistä on John Backuksen johtaman ryhmän kehittämä FORTRAN-kieli. FORTRANin ensimmäinen määrittely julkaistiin vuonna 1954 ja se oli tarkoitettu matemaattisten ongelmien tehokkaaseen ratkaisemiseen. Kielen luvattiin pystyvän kääntämään korkean tason ohjelmakoodi yhtä nopeaksi kuin käsin kirjoitettu konekielinen vastaava. Kun FORTRAN 0 julkaistiin vihdoin vuonna 1957 se pitikin lupauksensa nopeudesta erinomaisen hyvin. [Bac78]

Vaikka FORTRANin ensimmäinen versio olikin nykypäivän näkökulmasta varsin rajoittunut, se oli aikalaiseksi varsin edistynyt. FORTRAN sisälsi aritmeettisen If-lauseen, Do-lauseen sekä samalla käännettyjen alirutiinien käytön. Myöhemmin FORTRAN salli myös erikseen käännettyjen alirutiinien käytön ja loi perustan nykyisten datatyyppien kuten integer ja float käyttöön. [Mal98]

FORTRANIsta tuli nopeasti suosittu, ja sen suosio muutti koko ohjelmistoalan. Ennen sitä käytiin pitkään kiistaa siitä, voivatko käännetyt korkean tason ohjelmat koskaan korvata suoraan konekielellä koodattuja ohjelmia. FORTRANin suosion jälkeen kiista väistyi ja pian suurin osa kehittäjistä siirty käyttämään korkean tason ohjelmointikieliä kun vain mahdollista.

3 1950-luvun loppu ja 1960-luku

1950-luvun lopulla ja 1960-luvun alussa ohjelmointikielet kehittyivät suurin harppauksin. Yritykset kiinnostuivat pikkuhiljaa tietokoneiden tuomista mahdollisuuksista ja tätä lisäsi entisestään COBOL-kielen julkaiseminen vuonna 1959. Siinä missä Fortran oli varsin hyvä kieli numeroiden pyörittelemiseen se ei soveltunut hyvin yritystoimintoihin sen alkeellisten I/O-toimintojen vuoksi. [Fer04] COBOL oli sen sijaan kehitetty yhdysvaltain puolustusministeriössä nimenomaan yritysmaailmaa silmälläpitäen.

COBOL-kielen määrittelyssä vaadittiin, että sen tuli olla helppo kieli oppia. Tämä toteutuikin, sillä sen syntaksi muistutti varsin paljon luonnollista kieltä. Määrittelyssä jopa todetaan, että luettavuutta ja ymmärrettävyyttä pitää kehittää jopa kielen tehokkuudenkin kustannuksella. [Sam78] COBOLin etuna oli varsinkin monipuoliset I/O-toiminnot, desimaalilukujen hyvä käsittely ja monipuoliset merkkijonotoiminnot. COBOLin suosiosta saattaa kertoa se, että joidenkin arvioiden mukaan

se on edelleen yksi eniten käytetyistä ohjelmointikielistä kun lasketaan kirjoitettuja koodirivejä.

Vuonna 1958 julkaistiin ensimmäinen versio ALGOL-kielestä. Kieli määriteltiin kansainvälisessä hengessä saksalaisten ja yhdysvaltalaisten kehittäjien yhteisprojektina. ALGOL määriteltiin olevan syntaksiltaan mahdollisimman lähellä matemaattista notaatiota ja siten sen haluttiin pysyvän helposti luettavana. Yksi kauaskantoisimpia ALGOLin tuomia uutuuksia oli ohjelman jakaminen osiin aaltosuluilla. ALGOL ei nauttinut koskaan suurta suosiota varsinkaan Yhdysvalloissa, mutta se innoitti mm. Pascal- ja C-ohjelmointikielten kehittymisen. [Fer04]

Eräs erikoisemmista vielä tänäkin päivänä käytössä olevista ohjelmointikielistä on LISP. Sen kehitti John McCarthy vuonna 1958. LISP on ensimmäinen suosion saavuttanut funktionaalinen ohjelmointikieli. LISPin lähtökohtana oli luoda elegantti matemaattinen järjestelmä, jolla oli symbolinen esitys ja laskenta numeerisen sijasta. [McC78] LISPin selkein erikoisuus on sen ainoa tietotyyppi, lista. Itse asiassa jokainen LISPin ohjelma on itsessään vain joukko listoja. Tämän vuoksi se on niinsanotusti reflektiivinen eli itseään muokkaamaan kykeneviä ohjelmia mahdollistava ohjelmointikieli. [Tun06b] LISP on myös ensimmäinen kieli, jolle toteutettiin kääntäjä sen omalla kielellään. Lisäksi LISP-kielessä oli ensimmäisenä automaattinen roskienkeruu.

Vuonna 1963 IBM aloitti PL/I:n kehityksen. Kehitys sai alkunsa siitä, että IBM näki ohjelmointimaailman hajonneen useisiin eri kieliryhmiin. Tällaisia olivat tieteelliseen laskentaan keskittyvät (esim. FORTRAN), yritysmaailmaan keskittyvät (esim. COBOL) ja erikoistuneisiin toimintoihin keskittyvät (esim. LISP ja ALGOL-variantti JOVIAL).

Alunalkujaan jokainen erillisryhmä muodostui, koska he tarvitsivat erilaisia toimintoja ohjelmointikieliltään. Tieteellistä laskentaa tarvinneet halusivat kieliltään erityisesti mm. liukulukuaritmetiikkaa, taulukoita ja alirutiineja, yritysmaailman edustajat tarvitsivat mm. desimaaliaritmetiikkaa, merkkijonojen tehokasta käsittelyä ja asynkronista I/O:ta ja erityistyneet ryhmät mm. vaihtelevanmittaisten bittijonojen aritmetiikkaa, makrokirjastoja, listankäsittelymenetelmiä jne. [**?**]

Kuitenkin myös jokainen ryhmä tarvitsi jotain ominaisuuksia toisten ryhmien kielistä, joten PL/I:n tarkoituksena oli olla ensimmäinen yleiskäyttöinen rikkaan ominaisuuslistan omaava ohjelmointikieli. Kieli saatiinkin valmiiksi ja toteutettua vuonna 1964, mutta siitä ei tullut koskaan niin suosittua kuin IBM toivoi. Sen oli tarkoitus syrjäyttää mm. COBOL ja FORTRAN, mutta näin ei koskaan käynyt. Tähän

syynä oli kielen monimutkaisuus ja siitä johtuneet alkuaikojen heikkotehoiset kääntäjät. [Ros72] Monimutkaisuutta kritisoi varsin voimakkaasti mm. Dijkstra. [**?**] Kieli oli laaja ja siten vaikeasti opittava. Huolimatta siitä, että PL/I:stä ei tullut koskaan suurinta ja kauneinta, se oli kuitenkin aktiivisessa käytössä sekä tieteellisessä- että yritysmaailmassa.

Ominaisuuksiltaan PL/I oli erinomainen. Siihen lisättiin olemassaolevista kielistä parhaita puolia kuten ALGOLin aaltosulkujen käyttö ja rekursiiviset kutsut, FORTRANin osien erillinen kääntäminen ja COBOLin I/O:n muotoilu. Lisäksi PL/I oli ensimmäinen kieli joka toteutti mm. osoittimet ja poikkeuksien käsittelyn.

Myös olio-ohjelmointi pääsi alkuun 1960-luvulla. Vuonna 1964 norjalaiset tutkijat Kristen Nygaard ja Ole-Johan Dahl julkaisivat Simula I:n, joka oli nimensä mukaisesti simulointikieli. Kuitenkin suurin innovaatio Simulassa oli luokkien ja olioiden kehittyminen. Luokat kehittyivät, kun Nygaard ja Dahl havaitsivat hyödyn, joka saataisiin kun tietorakenteisiin lisättäisiin operaattoreita. Simula I ei itsessään sisältänyt vielä luokkia ja olioita, mutta Simula 67 sisälsi julkaistaessaan jo suoran viittauksen olioihin hyvin nykyisen kaltaisella viittausmenetelmällä. [ND78]

Huolimatta Simula I:n ja Simula 67:n kohtuullisesta suosiosta olio-ohjelmointiparadigma ei saanut vielä suuremmin tuulta purjeisiin. Tämä saattoi johtua uuden ajattelutavan erilaisuudesta suhteessa tavalliseen proseduraaliseen ohjelmointitapaan. Vasta vuonna 1972 julkaistussa Hoaren paperissa tehdään selväksi olion sisäpuolen ja ulkopuolen käsitteiden perustavanlaatuiset erot ja hyödyt. [Hoa72]

Vuonna 1964 lähes vuosikymmenen kestäneen ajatus- ja kehitysprosessin jälkeen Kurtz ja Kemeny julkaisivat ensimmäisen versionsa tulkatusta BASIC-ohjelmointikielestä. Sen lähtökohtana oli houkutella muitakin kuin luonnontieteen ja varsinkin tietojenkäsittelytieteen ja matematiikan opiskelijoita ohjelmoinnin pariin. Täten kielen tuli olla niin yksinkertainen ja helposti ymmärrettävä, että kuka tahansa pystyi opettelemaan sen. Alkuperäinen BASIC ja sen monet myöhemmät variantit olivat rivinumeropohjaisia proseduraalisia ohjelmointikieliä. [Kur78]

Rivinumeropohjaisuus aiheutti myös GOTO-käskyn käytön välttämättömyyden, joka on aiheuttanut paljon harmaita hiuksia. BASIC-kielestä julkaistiin 1964 ja 1971 yhteensä kuusi eri versiota ja sen lisäksi useat eri tahot julkaisivat omia BASIC-murteita. Murteiden suuren lukumäärän vuoksi myöhemmät standardointiyritykset tuottivat huomattavia vaikeuksia. Tämä toisaalta todistaa, että BASIC onnistui siinä mihin sillä pyrittiinkin: BASIC oli 1970- ja 1980-luvuilla suosituimpia ohjelmointikieliä varsinkin tavallisten kotitietokoneiden käyttäjien joukossa.

4 1970- ja 1980-luku

Vuonna 1970 Niklaus Wirth kehitti ALGOL-varianttinsa, Pascalin. Eräs suurimpia syitä Pascalin kehittämiseen oli yksinkertaisen opetuskieleksi kelpaavan proseduraalisen ohjelmointikielen puute. Pascal siis kehitettiin opetustarkoitukseen, ja olikin erityisesti opetuskielenä varsin suosittu aina pitkälle 90-luvulle.

Suurimmat Pascaliin liittyvät innovaatiot olivat useiden erilaisten tietotyyppien- ja -rakenteiden lisääminen ja tyyppimäärittelyn ja muuttujan määrittelyn välille. Huolimatta siitä, että Pascalin kehitysaikaan ALGOLin suurimmat ongelmat olivat jo tunnettuja, Pascaliin eksyi myös nopeasti tunnistettuja heikkouksia. Ehkä suurin vika kielessä oli dynaamisten taulukkoparametrien puute, joka häiritsi varsinkin numeeristen algoritmien ohjelmoijia. [Wir93]

Ranskalainen Alain Colmerauer ja yhdysvaltalainen Robert Kowalski kehittivät vuonna 1972 vaihtoehdon LISPille. Prolog on LISPin tavoin matemaattiseen logiikkaan pohjautuva ohjelmointikieli jota käytetään pääasiassa tekoälyn ja tietokonelingvistiikan tutkimukseen ja sovelluskehittämiseen. Itseasiassa Prolog kehitettiin nimenomaan tietokonelingvistiikkaa varten. Tästä syystä Prolog ei ole sanan varsinaisessa merkityksessä tietotyypitetty kieli, vaan tietotyypit ovat enemmänkin kieliopillisia elementtejä. Tämä on ymmärrettävää, sillä Prologilla ohjelmoiminen on hyvin erilaista verrattuna tavallisiin proseduraalisiin ohjelmointikieliin. Kielen perusideana on luoda tietokanta faktoista ja säännöistä jolle sitten suoritetaan kyselyitä.

C, eräs maailman käytetyimmistä ohjelmointikielistä, sai alkunsa Dennis Ritchien toimesta vuonna 1974. C:n edeltäjä B ja sen edeltäjä BCPL olivat tyypittömiä kieliä. Tämä tarkoitti sitä, että kaikki datan muoto oli kiinni alla olevan koneen sanan koossa. Tämä muodostui nopeasti ongelmaksi, kun Ritchie siirtyi käyttämään tavupohjaisia koneita. Turhautuessaan B:n rajoitteisiin Ritchie koodasi vuonna 1971 B:n laajennoskielen joka tunnisti char- ja int-tyyppisiä muuttujia. Hän nimesi tämän kielen New B:ksi (NB). NB oli kuitenkin varsin lyhytikäinen, sillä se kehittyi nopeasti varsin erilaiseksi kuin B, ja Ritchie uudelleennimesi sen nopeasti C:ksi. Ensimmäinen virallinen versio julkaistiin kuitenkin vasta vuonna 1974, ja kieli jatkoi kehittymistä varsinkin siirrettävyyden ja tyyppitarkastusten osalta pitkälle 80-luvun puolelle.

B-kieli oli suunniteltu alunperin systeeminkehityskieleksi ympäristöihin, joissa oli aikalaisittain rajatut resurssit. B:llä ohjelmoitiinkin Unixin ensimmäinen toimiva ydin PDP-7-koneelle. Tämä tapahtui jo 1969 Ken Thompsonin toimesta. Kuitenkin

pian C:n kehittymisen jälkeen eräs ensimmäisistä sillä tehdyistä suurista projekteista oli Unixin ytimen uudelleenkirjoitus. Siten C ja Unix kehittyivät käsi kädessä, ja C:stä tulikin todennäköisesti suosittu juuri Unixin suosion vuoksi.

C oli jossain määrin radikaali kieli. Sen kääntäjä oli innovatiivinen, siinä on B:hen kehitetyt ++ ja – - operaattorit, structit jotka pystyivät sisältämään pointtereita ja siten myös dynaamisen mittaisia taulukoita. Lisäksi C:ssä oli aikalaiseksi hyvin omaperäinen taulukoiden käsittelytapa. Siinä missä normaalit taulukot olivat perättäisiä soluja, joiden ensimmäinen solu määräsi taulukon pituuden C:ssä taulukot (ja siten myös merkkijonot) päättyvät erilliseen lopetusmerkkiin. C ei ole kuitenkaan täydellinen kieli, sillä sen taaksepäinyhteensoapivuudesta johtuvat syntaksiomituisuudet, heikko tyyppitarkastus johtuen kielen lähtökohdista ja monimutkainen käännettävyys on aiheuttanut monille kehittäjille vuosien saatossa harmaita hiuksia. [Rit93]

Yhdysvaltain puolustusministeriö suunnitteli ja toteutti Jean Ichibachin johdolla strukturoidun ja staattisesti tyypitetyn kielen nimeltään Ada. Ada tarkoitettiin alunperin sulautettuihin ja reaaliaikajärjestelmiin. Sitä kehitettiin aktiivisesti vuosien 1977-1983 välillä ja se sai päivitetyn version vuonna 1995.

Adan kehitys aloitettiin, sillä yhdysvaltain puolustusministeriö huolestui 70-luvun alkupuolella sen alaisissa projekteissa käytettyjen eri ohjelmointikielten nopeaan kasvuun. Tästä syystä siellä päätettiin kehittää yleiskäyttöinen kieli, joka korvaisi mahdollisimman suuren osan eri käytetyistä kielistä. [Sam86] Tämä onnistuikin varsin hyvin ko. puolustusministeriön sisällä. Adasta ei kuitenkaan tullut koskaan niin suosittua kieltä muualla maailmassa. Ongelmana siinä oli sen valtaisa laajuus ja tästä johtuneet epäluotettavuudet kääntäjissä. Toisaalta myös C++ osoittautui suositummaksi, todennäköisesti siksi, että C:llä oli jo valtaisa kehittäjäpohja valmiiksi kun Ada oli taas täysin uusi kieli.

Adassa on varsin monipuolinen ominaisuuslista. Se on vahvasti tyypitetty, modulaarinen pakettijärjestelmällä varusteltu kieli. Lisäksi Ada suorittaa ajonaikaisia virhetarkistuksia, poikkeuksienkäsittelyä ja tukee geneerisyyttä. Lisäksi Adassa on dynaaminen turvallinen korkean tason muistinhallinta kuten Javassa, sekä automaattinen roskienkerääjä. [Sam86] Vuodesta 1995 eteenpäin Ada on tukenut myös luokkia ja olioita.

Olio-ohjelmat jatkoivat kehittymistään 70-luvulla lähinnä Smalltalkin muodossa. Smalltalk-71 julkaistiin vuonna 1971 Palo Alto Research Centerissä Alan Kayn toimesta. Ensimmäinen kääntäjä kirjoitettiin vedonlyönnin tuloksena, jossa väiteltiin siitä olisiko Simulan inspiroima viestinvälitysmalli implementoitavissa muutamalla

koodisivulla. Smalltalk oli monipuolinen ja innovatiivinen kieli. Se tuki jo 70-luvun alussa graafisia käyttöliittymiä ja ennusti ominaisuuksillaan pienten pöytäkoneiden esiintuloa. Smalltalk kehittyi aina vuoteen 1980, jolloin se julkaistiin suuremmalle yleisölle. Tällöin kielessä oli toimintoina mm. perintä ja metaluokat. [Tun06d]

Kuitenkin suosituimmaksi olio-ohjelmointikieleksi nousi Bjarne Stroustrupin kehittämä C++. Alkuperäinen nimi, C with classes, kuvaa hyvin kielen tarkoitusta. C++ onkin sekä proseduraalista- että olioparadigmallista ohjelmointia tukeva kieli. Stroustroup aloitti kielen kehittämisen vuonna 1979, ja nimenvaihdos tapahtui vuonna 1983.

C++ on monipuolinen staattisesti tyypitetty kieli, jossa on mm. poikkeuksien käsittely, nimiavaruudet, operaattoreiden uudelleenkäyttäminen ja geneerisyyden luovat Template-luokat. Lisäksi C++ tukee moniperintää. C++ ei kuitenkaan sisällä roskienkerääjää johtuen sen C-tyylisestä muistinkäsittelystä. [Str93] Se on myöskin varsin hankala ensimmäiseksi ohjelmointikieleksi, sillä C++ on todella laaja, monimutkainen ja paikoin hankalasti tajuttava.

C++ onkin säilyttänyt suosionsa hyvin huolimatta sen korkeasta oppimiskynnyksestä. Yritysmaailma on ottanut kielen vastaan varsin hyvin, mutta sitä käytetään aktiivisesti myös akateemisessa maailmassa.

Funktionaaliset ohjelmointikielet kehittyivät edelleen 1970- ja 1980-luvuilla. Vuonna 1975 julkaistu Scheme pyrki yksinkertaistamaan LISPiä, eikä ollut varsinainen uusi kieli vaan enemmänkin LISPin variantti. Scheme tehtiin minimalistiseksi. Sen ideologiana on tehdä niin minimalistinen mutta monipuolinen järjestelmä, ettei siihen tarvi lisätä uusia ominaisuuksia uusien versioiden myötä. Scheme on oliopohjainen kieli. [Kri94]

1970-luvun lopulla Robin Milner kehitti ML-kielen, jonka alkuperäinen tarkoitus oli toimia alustana LCF-teoreeman todistamistekniikoille. ML:n ei kuitenkaan katsota olevan aito funktionaalinen ohjelmointikieli, sillä se sallii nk. sivuvaikutukset. Tällaisia ovat esimerkiksi funktiot, jotka ovat suoriuduttuaan aiheuttaneet muutakin muutoksia ohjelman tilaan kuin mitä sen paluumuuttuja tekee, esim. muokanneet globaaleja staattisia muuttujia. [Tun06c]

Myös tulkitut kielet jatkoivat elämää 70- ja 80-luvuilla. Vuonna 1987 Larry Wall julkaisi Perlin ensimmäisen version. Se on yleiskäyttöinen lähinnä tekstiä manipuloiva kieli, joka on vuosien varrella laajentunut monille muillekin ohjelmistokehityksen saralle. Perl ei ole kuitenkaan tarkoitettu korvaamaan olemassaolevia kieliä, vaan

Wall tarkoitti sen enemmänkin liimaksi ja purkaksi sinne, missä muut kielet tekevät asiat huonommin. Perl ei ole ollut koskaan erityisen elegantti kieli, sen sijaan sen kehityksessä on keskitytty helppokäyttöisyyteen ja ilmaisuvoimaan. [Cra94]

5 1990- ja 2000-luku

1900-luvun viimeinen vuosikymmen näki vähemmän uusia edes kohtuullista suosiota saaneita kieliä. Tällaisia kuitenkin löytyy, esimerkiksi Haskell, Python ja Ruby ovat jokainen keränneet kohtuullisen suuren kehittäjämäärän taakseen. Kuitenkin suurin uutuus 1990-luvulla oli selkeästi Sun Microsystemsin kehittämä Java. Java oli alunperin kehitetty sulautettuun *7-nimiseen kämmentietokonetta hieman suurempaan multimedialaitteeseen. Kieltä kutsuttiin vielä tuolloin nimellä Oak, mutta sen perustavanlaatuisin idea alustavapaudesta oli jo olemassa. Huolimatta alkuperäisestä käyttötarkoituksesta kielestä tuli suosittu vasta aivan toisenlaisessa ympäristössä, Internetissä. Vuonna 1993 alkanut ja 1995 räjätänyt WWW-buumi kiitos Tim Berners-Leen kehittämän HTML-kielen.

Javan tärkein piirre, sen alustariippumattomuus, mahdollisti kielen käyttöönoton nopeasti monissa eri ympäristöissä. Javalla riitti vain se, että sen virtuaalikone oli käännetty toimimaan alustalle. Tämä mahdollisti Java-pohjaisten ohjelmien käytön Web-selainten yhteydessä. WWW, joka oli ollut aiemmin staattinen ja liikkumaton muuttui yhtäkkiä eläväksi ja animoiduksi. Tämän jälkeen Javan suosio oli taattu. [Byo98]

Java kykenee alustariippumattomuuteen mielenkiintoisella tekniikalla. Java on sekä käännetty että tulkattu kieli. Ohjelmakoodi käännetään tavukoodiksi, jota koko järjestelmän mielenkiintoisin aspekti eli Javan virtuaalikone sitten tulkkaa alla olevalle alustalle ymmärrettäväksi. Näin siirtyminen alustalta toiselle oli mahdollista muuttamatta riviäkään koodia, ja toiminnan piti ainakin periaatteessa olla samanlaista.

Kielenä Java on olio-ohjelmointikieli. Siinä on automaattinen muistinhallinta sekä roskienkeruu, tuki geneerisyydelle, vahva tyypitys, laajat kirjastot ja valmiudet erilaisiin asioihin kuten verkon yli kommunikointiin, grafiikkaan, säikeisiin. Javassa on myös valmiina suuri määrä erilaisia tietorakenteita. Lisäksi, varsinkin alottelijoiden helpotukseksi, Java on piiloittanut osoittimet ohjelmoijalta.

Etuna Javalla verrattuna esimerkiksi C-kieleen on standardikirjastojen yhtäläisyys kaikissa alustoissa. Tämä tarkoittaa sitä, että esimerkiksi siinä, missä C-ohjelmat

käyttävät POSIX-säiekirjastoa yhdellä alustalla ja System V-säikeitä toisaalla Java-ohjelmat käyttävät ohjelmoijan näkökulmasta kaikkialla täsmälleen samoja kirjastoja ja toteutuksia.

Java ei kuitenkaan ole täydellinen kieli, sillä samanlaisen ajettavuuden, varsinkin graafisissa ympäristöissä on erittäin vaikeaa. Javaa pidetään myös hitaana kielenä, sillä varsinkin alkuaikana sen suorittaminen oli huomattavasti hitaampaa kuin C:n tai C++:n. Tällä saralla Java on kuitenkin parantanut huomattavasti, ja Javan potentiaalista nopeutta on todisteltu myös tieteellisissä piireissä. [Rei00]

Kun Sun ja Microsoft lopettivat yhteistyönsä Javan osalta vuonna 1998 Microsoft aloitti oman kilpailevan kielen kehityksen. C#-kieli julkaistiin vuonna 2000 ja se on kasvattanut suosiotaan tasaisesti. C#-kieli kuuluu Microsoftin kehittämään .NET-kehykseen. Kehyksen ideana on yhdistää useiden eri osa-alueiden ja kielten käyttäminen ja tuottaminen yhteen tiiviiseen ryhmään. .NET-yhteensopivaa koodia pystyy tuottamaan millä tahansa siihen liittyvistä kielistä siten, että jokainen kielistä tuottaa käännettynä CLR-nimistä välikieltä. Erona siis Javan ja .NET-kehyksen välillä on se, että Java on alustariippumaton kun taas .NET on kieliriippumaton.

C# on ominaisuuksiltaan hyvin Javan kaltainen. Se on vahvasti tyypitetty kieli jolla on mm. automaattinen roskienkeruu ja geneerisyys. C- ja C++-kieliin verrattuna suurin ero on siinä, että kuten Javassa, C#:ssa osoittimet ovat piiloitettu käyttäjältä. C#:ssä osoittimet saa kuitenkin käyttöön erillisen unsafe-määrittelyn avulla.

6 Yhteenveto

Vain vähän yli 60 vuoden aikana sekä tietokoneet että ohjelmointikielet ovat kehittyneet valtavasti. Varsin pian tietokoneiden kehityttyä käsin ohjelmoitaviksi ns. Von Neumann-arkkitehtuurisiksi koneiksi kehittivät koodaajat tapoja muuttaa ohjelmien kehitys helpommaksi. Kirjoitettavan ohjelmakoodin muuttuminen lähemmäksi luonnollista kieltä käännettävien ja tulkittavien ohjelmointikielten avulla mullisti ohjelmoinnin lopullisesti.

Ohjelmointikielten kehitys on ollut eräällä tavalla loogista. Vaikka historian saatossa on kehitetty lukuisia toisistaan täysin erilaisia ohjelmointikieliä, eräs piirre kehittämisessä on vallinnut lähes aina. Vaikka ohjelmointikieli perustuikin uuteen ajatukseen, se on silti aiempien kielien ongelmista ja virheistä oppimisen tuotosta. Alkuaikojen koneet olivat hitaita ja vähämuistisia, mutta koneiden kehittyes-

sä kykenivät myös ohjelmointikielet kehittymään ja monipuolistumaan. Kuitenkin myös ohjelmointikielten kehittyminen tuki koneiden kehittymistä. Esimerkiksi 1950-luvulla liukulukujen laskenta oli toteutettu puhtaasti ohjelmallisesti ilman suoraa laitetukea. Laitetukea kehitettäessä liukulukujen tehokkaaseen käsittelyyn saatiin kuitenkin varmasti hyviä ideoita jo toteutetuista ohjelmallisista ratkaisuista.

1950-luvun lopulla ja sen jälkeen ohjelmointikielten kehitys käynnistyi todella. Ennen vuotta 1960 sekä COBOL, FORTRAN, LISP kuin ALGOL oli kehitetty ja ainakin jossain määrin myös toteutettu. Suurin osa tämän päivän kielistä on ainakin jollain tavalla niin ajatusmallisesti kuin syntaktisestikin näiden kielten jälkeläisiä.

Ohjelmien siirrettävyys alustalta toiselle nousi oleelliseksi asiaksi ohjelmien kokojen kasvaessa. Uudelleenkoodaamisen kustannukset olivat kollektiivisesti nopeasti suuremmat kuin saman kielen hyvien kääntäjien kehittäminen eri alustoille oli. Kielten kehitys seurasi myös nopeasti perässä, ja esimerkiksi alunperin Unix-käyttöjärjestelmään kiinteästi liittyneen C-ohjelmointikielen tuotosten siirrettävyyteen panostettiin huomattavasti 1974-1978-välisenä aikana.

Tämän päivän suuret ohjelmointikielet ovat selkeitä historiansa tuotteita. Jokaisessa niistä on nähtävissä aiempien sukupolvien hyvistä puolista ja virheistä oppiminen. Ohjelmointikielet ovat kehittyneet valtavan paljon laajemmaksi, ja vaikka tänäkin päivänä kielten katsotaan olevan erikoistuneita johonkin, voi monilla niistä tehdä myös paljon muuta kuin tiettyä erikoisalaa. Toisaalta myös erikoisalan käsite on laajentunut huomattavasti. Myös alustariippumattomuus ja helppo siirrettävyys käyttöjärjestelmästä ja alustasta toiseen on tänä päivänä erittäin tärkeä ominaisuus.

Lähteet

Bac78 Backus, J., The history of fortran i, ii, and iii. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*. ACM Press, 1978, sivut 165–180.

Byo98 Byous, J., Java technology: The early years, 1998. `http://java.sun.com/features/1998/05/birthday.html`

Cra94 Cranor, L. F., Programming perl: an interview with larry wall. *Crossroads*, 1,2(1994), sivut 10–11.

Fer04 Ferguson, A., The history of computer programming languages,

2004. http://www.princeton.edu/~ferguson/adw/programming_languages.shtml

Hoa72 Hoare, C., Proof of correctness of data representation. sivut 271–281.

Knu62 Knuth, D. E., Invited papers: History of writing compilers. *Proceedings of the 1962 ACM national conference on Digest of technical papers*. ACM Press, 1962, sivu 43.

Kri94 Krishnamurthi, S., An introduction to scheme. *Crossroads*, 1,2(1994), sivut 19–27.

Kur78 Kurtz, T. E., Basic. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*. ACM Press, 1978, sivut 103–118.

Mal98 Malik, M. A., Evolution of the high level programming languages: a critical perspective. *SIGPLAN Not.*, 33,12(1998), sivut 72–80.

McC78 McCarthy, J., History of lisp. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*. ACM Press, 1978, sivut 217–223.

ND78 Nygaard, K. ja Dahl, O.-J., The development of the simula languages. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*. ACM Press, 1978, sivut 245–272.

Rei00 Reinholtz, K., Java will be faster than c++. *SIGPLAN Not.*, 35,2(2000), sivut 25–28.

Rit93 Ritchie, D. M., The development of the c language. *HOPL-II: The second ACM SIGPLAN conference on History of programming languages*. ACM Press, 1993, sivut 201–208.

Ros72 Rosen, S., Programming systems and languages 1965-1975. *Commun. ACM*, 15,7(1972), sivut 591–600.

Sam69 Sammet, J., *Programming Languages: History and Fundamentals*. Prentice-Hall, 1969.

Sam78 Sammet, J. E., The early history of cobol. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*. ACM Press, 1978, sivut 121–161.

Sam86 Sammet, J. E., Why ada is not just another programming language. *Commun. ACM*, 29,8(1986), sivut 722–732.

Str93 Stroustrup, B., A history of c++: 19791991. *HOPL-II: The second ACM SIGPLAN conference of History of programming languages*. ACM Press, 1993, sivut 271–297.

Tun06a Tuntematon, History of computers, 2006. `http://en.wikipedia.org/wiki/History_of_computers`

Tun06b Tuntematon, Lisp, 2006. `http://en.wikipedia.org/wiki/LISP`

Tun06c Tuntematon, Ml programming language, 2006. `http://en.wikipedia.org/wiki/ML_programming_language`

Tun06d Tuntematon, Smalltalk, 2006. `http://en.wikipedia.org/wiki/Smalltalk_programming_language`

Wir93 Wirth, N., Recollections about the development of pascal. *HOPL-II: The second ACM SIGPLAN conference on History of programming languages*. ACM Press, 1993, sivut 333–342.