hyväksymispäivä arvosana

arvostelija

Ohjelmointikielten varhaishistoria

Esa-Matti Miettinen

Helsinki 10.5.2005

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Esa-Matti Miettinen

Työn nimi — Arbetets titel — Title

Ohjelmointikielten varhaishistoria

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaariesitelmä	10.5.2005	13 sivua

Tiivistelmä — Referat — Abstract

Tämä seminaarityö käsittelee korkean tason ohjelmointikielten syntyä ja varhaishistoriaa. Nykyiset korkean tason käännettävät kielet syntyivät monimutkaisen prosessin tuloksena. Koska viime vuosisadan puolivälissä ei ollut olemassa mitään kaikkia ohjelmoijia yhdistävää tiedotuskanavaa, ei tutkijoiden välillä aina ollut todellista yhteistyötä. Kääntäjiä, tulkkeja ja itse korkean tason kieliä kehiteltiin varsin itsenäisesti eri puolilla maailmaa.

Ensimmäisistä korkean tason kielistä erityisen huomion kohteeksi tässä työssä on otettu Zusen jo 1940-luvulla luoma Plankalkül, sekä Mauchlyn Short Code, ensimmäinen korkean tason kielen toteutus. Kääntäjien syntyä esitellään Hopperin, Glennien sekä Laningin ja Zierlerin työn kautta. Lopuksi selvitetään korkean tason kielten läpimurron aikaan saaneen Fortranin syntyolosuhteita.

ACM Computing Classification System (CCS):
K.2.c [Software]
D.3 [Programming Languages]

Avainsanat — Nyckelord — Keywords

Ohjelmointikielet, historia

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Tietotekniikan kehitys on viime aikoina ollut huimaavan nopeaa. Tietokoneohjelmien tuottajat ovat olleet tärkeässä asemassa ohjaamassa tätä muutosta. Samalla ohjelmoijien oma maailma on kokenut kenties vieläkin suuremman mullistuksen. Viime vuosisadan puolivälissä ohjelmoija-sanaa ei edes ollut olemassa. Kahdeksankantaluvuista koostuvien ohjelmien tekijöitä kutsuttiin silloin vielä koodaajiksi [Hop81].

Uusia menetelmiä ohjelmointiin kehittäneet tiedemiehet ja ammattilaiset kohtasivat jo tuolloin samoja ongelmia kuin nykyään. Aina ei ollut helppoa vakuuttaa yritysten johtoa tai rahoittajia siitä, että tietokone voisi joskus tuottaa yhtä tehokasta konekieltä kuin ihminen. Jo pelkkä ohjelmien kirjoittaminen sanoilla numeroiden sijaan saattoi olla liian mullistava ajatus [Hop81].

Ensimmäinen korkean tason ohjelmointikieli syntyi jo 1940-luvulla Konrad Zusen suunnitellessa Plankalkülin. Tieto kielestä ei kuitenkaan kulkeutunut muiden tutkijoiden korviin, ja samoja asioita keksittiin myöhemmin uudelleen. Parempien välineiden puuttuessa ohjelmointia pyrittiin toisaalla helpottamaan kehittämällä korkeamman tason tulkattavia kieliä ja toisaalla ohjelmoimalla erilaisia enemmän tai vähemmän kehittyneitä kääntäjiä.

Pitkään ohjelmoijan todellisina vaihtoehtoina oli joko tuottaa tehokkaita ohjelmia hitaalla ja virhealttiilla konekieliohjelmoinnilla, tai käyttää jotakin helpompaa tulkattavaa kieltä suoritusnopeuden kustannuksella. Korkean tason kielet löivät itsensä toden teolla läpi vasta 1950-luvun lopulla Fortranin ja sen erityisen tehokkaan kääntäjän myötä. Vasta tuolloin oli mahdollista yhdistää nopea suoritus ja silloisesta näkökulmasta tarkasteltuna helppo ohjelmointi.

Tämän seminaarityön on tavoitteena on tuoda esiin kehityksen suuntauksia ja merkittäviä edistysaskelia. Samalla tarkoituksena on käsitellä hieman myös tehtyjen ratkaisujen taustatekijöitä ja seurauksia. Esitelmässä käsiteltävät kielet listataan taulukossa 1.

Kieli	Vuosi	Merkitys
Plankalkül	1945	Ensimmäinen korkean tason ohjelmointikieli
Short Code	1950	Ensimmäinen korkean tason tulkattava kieli
Autocode	1952	Ensimmäinen aito käännetty kieli
Laningin ja Zierlerin kieli	1952-1953	Ensimmäinen käännetty selvästi korkean tason kieli
A-2	1953	Ensimmäinen makrojen käsittelijä.
Fortran I	1956	Ensimmäinen suuren suosion saavuttanut korkean tason kieli

Taulukko 1: Käsiteltävien kielten aikajana. Vuosiluvut ovat suuntaa antavia arvioita valmistumisajankohdista.

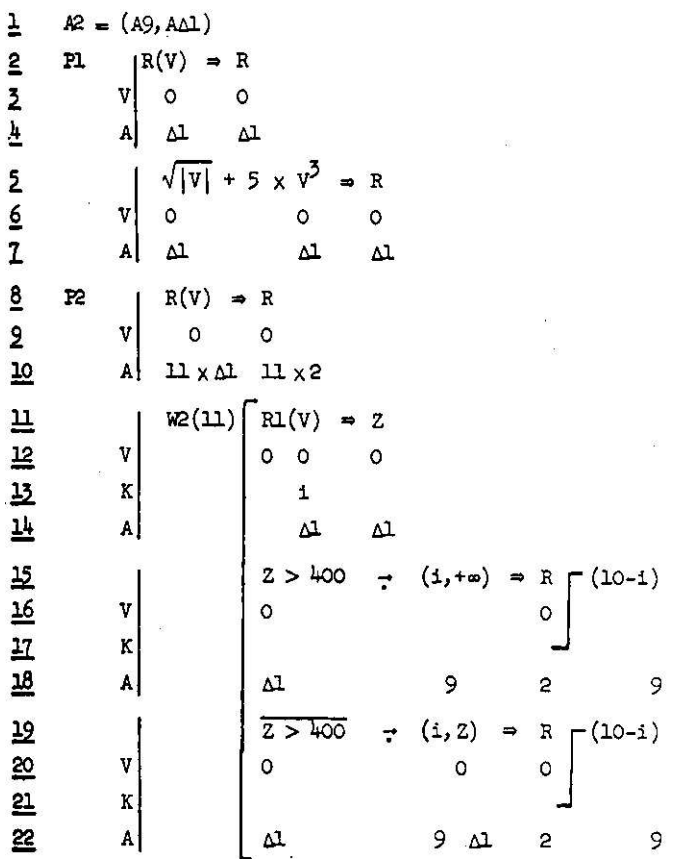
2 Plankalkül, ensimmäinen korkean tason kieli

Konrad Zusen 1940-luvulla kehittämää Plankalkül-kieltä [Zus72] pidetään yleisesti ensimmäisenä korkean tason kielenä. Kieli kuitenkin julkaistiin vasta vuonna 1972. Toteutusta saatiin lisäksi odottaa vuoteen 2000.

Plankalkül sisältää useita nykykielten ominaisuuksia. Siitä löytyvät taulukot, fortyyppinen toistolause, ehtolause ja eräänlainen Eiffel-kieltä muistuttava invarianttien tarkistus [Seb99]. Ohjelmointikielissä käytettävälle sijoitusoperaatiolle (esimerkiksi Javan "=") käytettiin Plankalkülissä ensimmäistä kertaa omaa symbolia. Zuse kehitti myös useita ohjelmia luomallaan kielellä. Niihin kuului jakoalgoritmeja, kokonaislukuaritmetiikkaa ja jopa shakkiohjelma.

Kuva 1 esittää esimerkkialgoritmin Plankalkül-kielellä. Ohjelma lukee joukon lukuja, ja tulostaa sitten käänteisessä järjestyksessä funktion $f(x) = \sqrt{|x|} + 5 * x^2$ arvot syötteenä annetuilla luvuilla. Ohjelma ei tee mitään kovin hyödyllistä, mutta tuo hyvin esiin kielen piirteitä. Selvyyden vuoksi ja vertailun helpottamiseksi kuva 2 esittää saman algoritmin Javalla kommenttien kera.

Vaikka Plankalkül-kieli saattaakin näyttää vaikeaselkoiselta, on merkintätapa totuttelun jälkeen melko helppolukuinen. Useimmista nykykielistä poiketen Plankalkülissä yksi nykyohjelman riviä vastaava kokonaisuus voi koostua jopa neljästä rivistä. Kuvassa 1 neljän rivin rakenne näkyy kirjaimin V, K ja A merkittyjen ja merkitsemättömien rivien vuorotteluna. Ylimpänä esiintyvä merkitsemätön rivi on lähimpänä nykyohjelman riviä, ja kolme muuta riviä täydentävät sen sisältöä. Rivillä V voidaan antaa ylimmällä rivillä esiintyville muuttujilla indeksejä. A-rivi puolestaan



Kuva 1: TPK-algoritmi Plankalkül-kielellä [KnP76].

kertoo muuttujien tyypit ja K-riviä käytetään muuttujien komponentteihin viittaamiseen. Jos ylimmällä rivillä esimerkiksi on muuttuja R, V-rivillä 0 ja rivillä K 1, on kyse muuttujan R_0 komponentista 1.

TPK-algoritmin ensimmäisellä rivillä määritellään tyyppiä A2 oleva muuttuja, joka koostuu kokonaisluvun ja liukuluvun muodostamasta järjestetystä parista. Tietotyyppien joustavuus onkin yksi Plankalkülin edistyksellisimmistä piirteistä. Kaikkien kielen tyyppien lähtökohtana on yhtä bittiä vastaava tyyppi, ja muut tyypit on rakennettu tyyppejä yhdistelemällä. Uusi tyyppi voidaan luoda mitä tahansa tyyppejä yhdistelemällä [BW72]. Samoin voidaan luonnollisesti käyttää taulukoita, jotka voivat myös olla muuttuvan pituisia. Kaikkien tyyppien pohjana olevan yhdestä bitistä koostuvan perustyypin lisäksi kieleen valmiiksi määriteltyihin tyyppeihin sisältyvät muun muassa kompleksi-, liuku-, kokonais-, rationaali ja binääriluvut. Joitakin mahdollisia tyyppejä esitetään taulukossa 2.

Plankalkülin tärkeimmät tyypit ovat listat ja parilistat [Gil97]. Listoille on määri-

```
// Lukee 11 lukua ja tulostaa funktion f(x) = SQRT((ABS(x)) + 5 * x^3
// arvot kullakin syötetyllä luvulla käänteisessä järjestyksessä.
// Jos funktion arvo syötteellä on liian suuri, tulostetaankin
// ilmoitus liian isosta luvusta.
public static void tpk(){
  int[] input=new int[11];
  double value;
  for (int i=0; i<11; ++i) input[i] = Lue.kluku();  // Lue 11 lukua.

  for(int i=10; i>=0; --i) {  // Käsittele luvut käänteisessä järjestyksessä.
    // Laske funktion f(x) = SQRT((ABS(x)) + 5 * x^3 arvo.
    value=Math.sqrt(1.0*Math.abs(input[i]))+5*Math.pow(input[i],3);
    if (value > 400)  // Tulos ei saa olla > 400!
      System.out.println(i+" TOO LARGE");
    else  // Tulosta funktion arvo.
      System.out.println(i+'' ''+value);
  }
}
```

Kuva 2: TPK-algoritmi Java-kielellä.

telty operaatiot, joilla voi selvittää listan alkioiden lukumäärän tai pyytää seuraavan, ensimmäisen, viimeisen, suurimman tai pienimmän alkion. Listasta voi myös pyytää kaikki tietyn ehdon täyttävät alkiot. Listan alkuun tai loppuun voi valmiilla operaatioilla lisätä alkion, ja listoja saattoi myös liittää toistensa perään. Ainoa operaatio taulukoille oli alkion valinta indeksoimalla. Zuse tosin suunnitteli myös muun muassa matriisioperaatioiden toteuttamista.

Kuvassa 1 esitetyn listauksen riveillä 2-4 määritellään P1-niminen proseduuri, joka saa syötteekseen liukulukutyyppisen muuttujan V_0, ja palauttaa samaa tyyppiä olevan muuttujan R_0. Yleisemminkin ohjelmissa käytettiin yleensä V-, Z- ja R-muuttujia. Ohjelma sai syötteensä V-muuttujien välityksellä, tallensi tulokset R-muuttujiin ja käytti apuna Z-muuttujia. Ohjelmassa saattoi myös esiintyä C-muuttujia vakioina [Gil97].

Seuraava rivien 5-7 muodostama yksikkö sijoittaa muuttujaan R_0 kaavan $\sqrt{|t|}+5*t^2$ arvon. Rakenne ei sisällä mitään nykyohjelmoijaa hämmästyttävää, mutta sijoitusmerkin käyttö tällä tavoin sai alkunsa juuri Plankalkülissä. Erona nykyisiin ohjelmointikieliin on, että Plankalkülissä muuttuja, jolle arvo sijoitetaan, kirjoitettiin sijoitussymbolin oikealle puolelle. Samaa kirjoitustapaa käytettin myös joissakin muissa varhaisissa kielissä.

Tyyppi	Selitys
S0	Boolean-arvo
S1.n (=n x S0)	N:n elementin bittivektori
n x S1.4	N:stä numerosta koostuva kokonaisluku BCD-esityksenä
m x n x S1.32	M x n-kokoinen matriisi, jonka alkiot ovat 32-bittisiä kokonaislukuja
σ . τ	Määrittelemättömiä rakenteita, määritellään ohjelmaa suoritettaessa
$\square$ x σ	Muuttuvankokoinen ($\square$) lista, jonka alkiot ovat määrittelemättömiä (σ)
$\square$ x 2 σ	Lista samaa tyyppiä olevista pareista
$\square$ x (σ.τ)	Lista erityyppisistä muuttujista muodostettuista pareista

Taulukko 2: Esimerkkejä Plankalkülin mahdollisista tietotyypeistä [Gil97].

Rivi kahdeksan aloittaa toisen proseduurin. Riveillä 9-10 kerrotaan proseduurin saavan syötteenään tiettyä tyyppiä olevan muuttujan V_0, ja tallentavan tuloksen saamaa tyyppiä olevaan muuttujaan R_0. Algoritmin pääsilmukka on riveillä 11-22. Merkintä W2(n) vastaa for-toistolausetta n-1:stä nollaan. Seuraavia rivejä siis toistetaan kymmenen kertaa. Rivien 11-14 merkintä tarkoittaa, että muuttujaan V_i sovelletaan operaatiota P1. Kyse on siis eräänlaisesta aliohjelmakutsusta.

Silmukan sisällä riveillä 15-18 esiintyvä merkintä toimii ehtolauseena. Nykynotaatiolle tulkattuna se olisi "if $z_0 > 400$ then R_0[10-i] = (i, $+\infty$)". Viiva rivin 19 yläpuolella tarkoittaa ehdon negaatiota, eli vastaa esimerkiksi Javan huutomerkkioperaatiota. Kieli ei sisältänyt varsinaista else-operaatiota. Ehtolauseen lopusta kannattaa myös huomata symboli $+\infty$. Plankalkülissa on näet tavallisten arvojen lisäksi määritelty erityisarvot "ääretön", "hyvin pieni" ja "määrittelemätön".

Vaikka Plankalkül oli hyvin kehittynyt kieli, ei sen merkitys ohjelmointikielten käytännön toteutusten kannalta ollut kovin merkittävä. Zuse harkitsi kääntäjän toteuttamista, mutta hänellä ei ollut riittäviä resursseja. Ohjelmointikielten kehityksen varhaisvaiheessa juuri toteutusten laadukkuus usein ratkaisi kielen menestyksen. Koska kieli julkaistiinkin vasta kolme vuosikymmentä kehittämisen jälkeen, jäi se suurelta osin syrjään kehityksestä edistyksellisenä mutta ehkä myös aikaansa edellä olleena kielenä.

3 Short Code, ensimmäinen toteutettu korkean tason kieli

Jo ennen varsinaisia korkean tason kieliä ohjelmointia pyrittiin helpottamaan luomalla konekielen päälle helppokäyttöisempiä virtuaalisia tietokoneita. Tällaisia tulkattavia konekielten lähisukulaisia oli eri koneille toteutettuina useitakin.

		Symbolinen esitys	Koodiesitys
00		i = 10	00 00 00 W0 03 Z2
01	0:	y = ($\sqrt{}$abs t)+ 5 cube t	T0 02 07 Z5 11 T0
02			00 Y0 03 09 20 06
03		y 400 if≤to 1	00 00 00 Y0 Z3 41
04		i print, 'TOO LARGE' print-and-return	00 00 Z4 59 W0 58
05		0 0 if=to 2	00 00 00 Z0 Z0 72
06	1:	i print, y print-and-return	00 00 Y0 59 W0 58
07	2:	TO UO shift	00 00 00 T0 U0 99
08		i = i-1	00 W0 03 W0 01 Z1
09		0 i if≤to 0	00 00 00 Z0 W0 40
10		stop	00 00 00 00 ZZ 08

Kuva 3: TPK-algoritmi Short Codella [KnP76]. Ohjelma annettiin kääntäjälle oikean puoleisessa muodossa. Vasemman puoelisessa esityksessä koodimerkinnät on korvattu symbolisilla esityksillä.

Koska korkean tason kielet kehittyivät konekielistä vähitellen pienin askelin, vaatii ensimmäisen toteutetun konekielen nimeäminen rajanvetoa korkean ja matalan tason kielten välillä. Knuth ja Pardo [KnP76] antavat kunnian John Mauchlyn vuonna 1949 kehittämälle Short Codelle. Ensimmäisen toteutuksen kielestä koodasi William Schmitt BINAC:lle samana vuonna, ja myöhemmin edelleen UNIVAC:lle Albert Tonikin avustuksella. Koska näiden koneiden käyttäjiä oli melko vähän, ei kieli saavuttanut heti suurta suosiota. Osaltaan vähäiseen käyttöön vaikutti myös Short Coden hitaus. Ohjelman tulkkaaminen oli noin viisikymmentä kertaa hitaampaa kuin vastaavan konekielellä kirjoitetun ohjelman suoritus [Mal98].

Suosion vähyyden vuoksi kielestä on säilynyt melko vähän tietoa. Knuthin ja Pardon näkemys TPK-algoritmin toteutukselle Short Codella on kuvassa 3. Algoritmi on esitetty kahdessa muodossa, joista oikean puoleisessa ohjelma annettiin kääntäjän käsiteltäväksi. Vasemman puoleisessa esityksessä koodimerkinnät on korvattu vas-

taavilla symbolisilla tunnuksilla selkeyden vuoksi. Kuvasta puuttuvat muun muassa tarvittavat muistipaikkojen alustukset.

Ohjelman symbolinen esitysmuoto on suhteellisen helppolukuinen muutamaa poikkeusta lukuun ottamatta. Ehtolause riveillä 03, 05 ja 09 poikkeaa rakenteeltaan hieman nykyisistä. Ehtolauseessa annetaan ensin kaksi vertailtavaa arvoa. Tämän jälkeen annetaan vertailuoperaatio, ja rivi jolle hypätään, jos ehto täyttyy. Short Codessa ei ole indeksoituja muuttujia, joten tavanomaisen taulukoinnin sijaan tässä on käytetty muuttujia muistipaikoista siirtävää shift-käskyä. Koodiesitys vastaa rakenteeltaan symbolista esitystä, joskin joidenkin koodirivien alkuun on lisätty nollia täytteeksi.

4 Ensimmäiset kääntäjät

Kääntäjien kehityksessä konkretisoituvat 1950-luvun tutkijoita vaivanneet tiedonkulun ongelmat. Ainakin kolmea eri kehittäjäryhmää voi näkökulmasta riippuen pitää ensimmäisten kääntäjien ohjelmoijina. A-2, Autocode sekä Glennien ja Zierlerin kääntäjä tuotettiin suunnilleen samaan aikaan itsenäisesti ja toisista projekteista tietämättä.

Yksi tärkeimmistä kääntäjien ja korkeamman tason kielten varhaisista puolestapuhujista oli joka tapauksessa Grace Murray Hopper [Hop81]. Kohtaamastaan suuresta skeptisyydestä huolimatta hän pyrki muuttamaan tapaa jolla ohjelmia tuotetaan. Ohjelmointikielten kääntäjien kehitys lähti kuitenkin liikkeelle hyvin vaatimattomista tavoitteista.

GMI000	000002	AA0040	038038	YT0036	038000	4RG000	000007
ITEM01	WS.000	AS0004	038040	GMM000	000001	5RG000	000006
SERV02	BLOCKA	OWNΔCO	DEΔ003	000194	200220	6RG000	000007
1RG000	000000	K00000	K00000	1RG000	011000	1CN000	000002
GMM000	000001	F00912	E001RG	GMM000	000001	2CN000	000014
000180	020216	000000	Q001CN	000222	200196	1RS000	000036
1RG000	001000	1RG000	008040	1RG000	012000	2RS000	000037
AM0034	034040	1CN000	000010	ALL012	F000TI	OWNΔCO	DEΔ002
RNA040	010040	GMM000	000001	1RG000	013036	810000	820000
APN034	012038	000188	020238	2RG000	000037	900000	900000
AM0002	038038	1RG000	009000	3RG000	000006	1RG000	014000
						RØENDΔ	INFO.R

Kuva 4: TPK-algoritmi A-2:lla [KnP76].

Univac-tietokoneella Eckert-Mauchly Computer Corporationissa työskennellessään Hopper sai vuonna 1951 tehtäväksi koota usein toistuvia ohjelmakoodin rakenteita

esimerkiksi matemaattisten laskutoimitusten käsittelyyn, jotta niiden käyttöä voitaisiin standardisoida. Käsin tehtävässä kopioinnissa esiintyvien virheiden välttämiseksi ja konekielisissä ohjelmissa käytettävien absoluuttisten osoitteiden automaattiseksi korjaamiseksi kopiointi päätettiin antaa tietokoneen tehtäväksi. Näin syntyi ohjelmointikielten kääntäjien esi-isä A-0 vuonna 1952, ja myöhemmin vuonna 1953 sen seuraaja A-2.

Kuva 4 esittää TPK-algoritmin A-2:n ymmärtämässä muodossa. Koodi on muihin ensimmäisen käännetyn kielen asemasta kilpaileviin kieliin verrattuna ilman lisäohjeita varsin vaikeasti ymmärrettävää. Hopperin kääntäjiä muistuttavat apuvälineet toivat varsinkin alkuun lähinnä makroja muistuttavan toiminnallisuuden ohjelmoijan ulottuville. Ne eivät myöskään varsinaisesti kääntäneet kokonaista uutta kieltä konekielelle, vaan A-2:n käskyjen seassa saattoi suunnilleen vuoteen 1955 asti kuvan 4 tapaan olla OWNCO-symbolilla merkittyjä, koneen omalla konekielellä kirjoitettuja jaksoja.

Hopperin ryhmän varhaiset tuotokset eivät ehkä täysin vastaa nykyistä kääntäjän käsitettä. Englannin kielen muun muassa ohjelmointikielen kääntäjää nykyään tarkoittava sana compiler otettiin kuitenkin ohjelmoijien käyttöön juuri näiden apuvälineiden kehityksen yhteydessä, ja sen nykyinen merkitys kehittyi vasta myöhemmin. A-0 saattoi siten tietyssä mielessä olla ensimmäinen “compiler”.

Toinen ehdokas ensimmäiseksi kääntäjäksi on vähemmän tunnettu. Alick Glennie kehitti 1950-luvun alussa brittien ydinaseprojektin parissa työskennelleessään lähinnä vapaa-ajallaan noin kolmessa kuukaudessa kääntäjän kehittämästään Autocode-kielestä Manchester Mark I-koneen konekielelle [KnP76].

Mark I-koneen konekieli oli varsin hankalakäyttöinen, joten helpommalle kielelle oli selvästi tarvetta. Autocode oli kuitenkin suhteellisen koneenläheinen kieli, joten se ei ratkaissut kaikkia koneen hankalaan arkkitehtuuriin liittyviä ongelmia. Vaikka edistysaskel konekielestä korkean tason kieleen olikin ohjelmoinnin kehityksen kannalta merkittävä, ei Autocode parantanut riittävästi juuri kyseisellä koneella työskentelevien ohjelmoijien työskentelyä. Siten kielestä ei tullut koskaan niin suosittua kuin sen historiallinen asema olisi voinut antaa olettaa.

Kuvassa 5 esitetään TPK-algoritmi Autocodella. Koko algoritmin läpikäynti ei liene tarkoituksenmukaista, mutta esimerkistä saa ehkä jonkinlaisen kuvan kielen rakenteesta. Kiinnostavana yksityiskohtana kannattaa huomata, että rivien 4 ja 22 ehtolauseissa viitataan suoraan alla olevan koneen tiettyihin rekistereihin. Samoin rivien 1 ja 12 muuttujien tallennusoperaatiossa viitataan suoraan koneen muistiosoitteisiin.

```
1   c@VA t@IC x@1/2C y@RC z@NC
2   INTEGERS +5 -> c
3   ->t
4   +t TESTA Z
5   -t
6   ENTRY Z
7   SUBROUTINE 6-><
8   +tt->y->x
9   +tx->y->x
10  +z+cx CLOSE WRITE 1
11  a@1/2 b@MA c@GA d@OA e@PA f@HA i@VE x@ME
12  INTEGERS +20 ->b +10 ->c +400 ->d +999 ->e +1 ->f
13  LOOP 10n
14  n->x
15  +b-x->x
16  x->q
17  SUBROUTINE 5->aq
18  REPEAT m
19  +c->i
20  LOOP 10n
21  +an SUBROUTINE 1-y
22  +d-y TESTA z
23  +i SUBROUTINE 3
24  +e SUBROUTINE 4
25  CONTROL X
26  ENTRY Z
27  +i SUBROUTINE 3
28  +y SUBROUTINE 4
29  ENTRY X
30  +i-g->i
31  REPEAT n
32  ENTRY A CONTROL A WRITE 2 START2
```

Kuva 5: TPK-algoritmi Autocodella [KnP76].

```
    v|N = ⟨input⟩,
    i = 0,
1   j = i+1,
    a|i = v|j,
    i = j,
    e = i−10.5,
    CP 1,
    i = 10,
2   y = F¹(F¹¹(a|i))+5(a|i)³,
    e = y−400,
    CP 3,
    z = 999,
    PRINT i,z.
    SP 4,
3   PRINT i,y.
4   i = i−1
    e = −0.5−i,
    CP 2,
    STOP
```

Kuva 6: TPK-algoritmi Laningin ja Zierlerin kehittämällä kielellä [KnP76].

Kolmas ehdokas ensimmäiseksi korkean tason käännetyksi kieleksi on Laningin ja Zierlerin Whirlwind-tietokoneelle vuonna 1953 kehittämä kieli [Bac78]. Se oli Au-

tocodeen verrattuna huomattavasti riippumattomampi alla toimivasta tietokoneesta. Kuten kuvasta 6 näkyy, kieli on myös selvästi helppolukuisempi, ja siinä on Autocodesta puuttuneet liukuluvut. Kielen suoritus tapahtuu tulkkaamalla käännettyä kieltä.

5 Fortran ja korkean tason kielten läpimurto

Tulkattavien virtuaalisten tietokoneiden hitaus nousi viimeistään ongelmaksi, kun yleisesti käytettyihin tietokoneisiin saatiin lisätyksi laitteistotasolla toimiva liukulukujen käsittely. Tällöin ohjelmien toiminta nopeutui suuresti, ja hidas tulkkaus nousi suorituskyvyn pullonkaulaksi [Bac78]. Ratkaisua lähdettiin hakemaan ohjelmien kääntämisestä ennakkoon käytettävän koneen konekielelle.

Monet alkuaikojen kääntäjistä olivat melko kömpelöitä sekä tuotetun konekielen suoritusnopeuden että käännettävän kielen ominaisuuksien puolesta. Kääntäjien tuottama koodi saattoi olla jopa 10-15 kertaa hitaampaa kuin taitavan ohjelmoijan käsin kirjoittama ohjelma.

John Backus kokosi vuonna 1954 IBM:llä joukon ihmisiä projektiin, joka asetti kunnianhimoiseksi tavoitteekseen tuottaa kääntäjän, joka kykenisi lähes yhtä tehokkaan ohjelmakoodin tuottamiseen kuin tavanomainen ohjelmoija. Ohjelmointikielen oli myös määrä olla niin yksinkertainen, että tavallinen ohjelmoija ymmärtäisi sillä ohjelmoidun proseduurin toiminnan ilman lisäohjeita tunnin koulutuksen jälkeen. Lisäksi ryhmän väliraportin sanoin "...FORTRAN should virtually eliminate coding and debugging..." [IBM54].

Fortranin ensimmäinen toteutus julkaistiin vuonna 1957. Fortran oli tuolloin reilusti myöhässä aikataulustaan, ja suurelta osin yhä toimimaton [Ros64]. Vähitellen virheiden määrää saatiin vähennettyä, ja kaikkien yllätykseksi Fortranin lupaama tehokkuus todella saavutettiin. Ensimmäiseen versioon kielestä oli myös lisätty vuoden 1954 suunnitelmien jälkeen lisäominaisuuksia, kuten ensimmäistä kertaa korkean tason kielessä esiintyvä kommentointimahdollisuus sekä syötteen ja tulostuksen muotoilukomennot. Kuvassa 7 annetaan esimerkki Fortran-ohjelmasta TPK-algoritmin muodossa.

Seuraavina vuosina Fortranin suosio lähti valtavaan nousuun. Erään raportin [Bac58] mukaan heinäkuussa 1958 puolet 26:n 704-tietokoneen käyttäjistä käytti Fortrania yli puolessa tapauksista, ja melkein kaikki käyttivät sitä ainakin joskus. Samana

```
C     TPK-algoritmi Fortranilla
      FUNF(T)=SQRTF(ABSF(T))+5.0*T**3
      DIMENSION A(11)
1     FORMAT(6F12.4)
      READ 1, A
      DO 10 J=1,11
      I=11-J
      Y=FUNF(A(I+1))
      IF (400.0-Y)4,8,8
4     PRINT 5,I
5     FORMAT(I10, 10H TOO LARGE)
      GO TO 10
8     PRINT 9,I,Y
9     FORMAT(I10,F12.7
10    CONTINUE
      STOP 52525
```

Kuva 7: TPK-algoritmi Fortranilla [KnP76].

syksynä 66:n tutkitun 704-tietokeen suorittamista konekäskyistä puolet tuotettiin Fortranilla.

6 Yhteenveto

Ohjelmoijia kutsuttiin koodaajiksi jo viime vuosisadan puolivälissä. Silloin sanan merkitys oli kuitenkin hieman toinen kuin nykyään. Ohjelmointi tarkoitti aluksi ohjelmien kirjoittamista numeromuodossa, usein esimerkiksi oktaalijärjestelmässä. Ohjelmointi oli hyvin koneenläheistä, hidasta ja virhealtista.

Konekielten käyttöön liittyviä ongelmia ratkaisemaan alettiin kehittää korkeamman tason ohjelmointimenetelmiä. Niiden kehitys ei kuitenkaan ollut aivan suoraviivaista. Heikon tiedonkulun vuoksi tutkijat eivät aina hyötyneet toistensa keksinnöistä, ja kehitys kohti todellisia korkean tason kieliä eteni useita sivupolkuja pitkin.

Ensimmäisenä korkean tason ohjelmointikielenä pidetään yleisesti Konrad Zusen jo 1940-luvulla kehittämää Plankalkül-kieltä. Kieli sisälsi monia nykyisten kielten piirteitä, kuten rakenteiset tietotyypit, indeksointi, liukuluvut ja for-tyyppinen toistosilmukka. Kieltä ei kuitenkaan julkaistu ennen 1970-lukua, joten sen vaikutus ohjelmoinnin kehitykseen jäi lopulta vähäiseksi.

Käytännön toteutukset korkeamman tason kielistä lähtivät liikkeelle suurelta osin

erilaisten tulkattavien, tietokoneiden oman kielen päälle rakennettavien apukielten muodossa. Tulkattavat kielet sisälsivät usein kaivattua lisätoiminnallisuutta, kuten liukuluvut ja indeksoinnin. Noin vuonna 1950 kehitetty Short Code oli luultavasti ensimmäinen tulkattava korkean tason kieli. Kuten muutkin tulkattavat kielet, senkin suoritus oli kuitenkin varsin hidasta.

Ensimmäisen käännettävän korkean tason kielen nimeäminen ei ole aivan ongelmatonta. Vastaus riippuu siitä mitä pidetään korkean tason kielenä. Varhaisten kääntäjien yhteydessä mainitaan usein Grace Hopper ja hänen A-2 "compiler". A-2 toimi aluksi hieman kuten nykyiset makrojen käsittelijät, ja sen hyväksymää kieltä saatettiin täydentää koneen omalla konekielellä. Korkeamman tason käännettäviä kieliä olivat Glennien Autocode sekä Laningin ja Zierlerin kehittämä kieli. Ensiksi mainittu valmistui ensimmäisenä, mutta oli jälkimmäiseen verrattuna vielä melko koneenläheinen eikä sisältänyt liukuluja.

Todellinen korkean tason kielten läpimurto tapahtui kuitenkin vasta Fortranin myötä. Kieli ei itsessään ehkä ollut merkittävästi helppokäyttöisempi kuin muut yrittäjät. Vastoin odotuksia sen kääntäjä oli kuitenkin erittäin tehokas. Käännetyt ohjelmat toimivat suunnilleen yhtä nopeasti kuin ohjelmoijan suoraan konekielellä kirjoittamat. Nopeuden ansiosta Fortran syrjäytti useimmat muut kielet, ja on monin paikoin yhä vieläkin käytössä.

Lähteet

Bac58 Backus, J. W., Automatic programming: properties and performance of Fortran systems I and II. *Proceedings of the Symposium on the Mechanisation of Thought Processes*, marraskuu 1958.

Bac78 Backus, J., The history of Fortran I, II, and III. *HOPL-1: The first ACM SIGPLAN conference on History of programming languages*, New York, USA, 1978, ACM Press, sivut 165–180.

BW72 Bauer, F. L. ja Wössner, H., The Plankalkül of Konrad Zuse: a forerunner of today's programming languages. *Commun. ACM*, 15,7(1972), sivut 678–685.

Gil97 Giloi, W. K., Konrad Zuse's Plankalkül: The First High-Level non von Neumann Programming Language. *IEEE Annals of the History of Computing*, 19,2(1997), sivut 17–24.

Hop81 Hopper, G. M., Keynote address. *History of Programming Languages*, USA, 1981, sivut 7–20.

IBM54 Specifications for The IBM Mathematical FORmula TRANslating System, FORTRAN. Preliminary report, I.B.M. Applied Science Division Programming Research Group, New York, 1954.

KnP76 Knuth, D. E. ja Pardo, L. T., The early development of programming languages. Tekninen raportti, Stanford University, Stanford, 1776.

Mal98 Malik, M. A., Evolution of the high level programming languages: a critical perspective. *SIGPLAN Notices*, 33,12(1998), sivut 72–80.

Seb99 Robert W. Sebesta, *Concepts of Programming Languages.* Addison-Wesley, Boston, 1999.

Ros64 Rosen, S., Programming systems and languages, a historical survey. *Proceeding of the Spring Joint Computer Conference*, 1964, sivut 1–16.

Zus72 Zuse, K., Der Plankalkül. *Berichte der Gesellschaft fur Mathematik un Dataenverarbeitung*, 63.