

Luento 4 Aliohjelmien toteutus

Tyypit
Parametrit
Aktivointitietue (AT)
AT-pino
Rekursio

31/10/2001

Copyright Teemu Kerola 2001

1

Aliohjelmatyypit ⁽²⁾

- Korkean tason ohjelmointikielen käsitteet
 - aliohjelma, proseduri
 - parametrit
 - funktio
 - parametrit, paluuarvo
 - metodi
 - parametrit, ehkä paluuarvo
- Konekielen tason vastaavat käsitteet
 - aliohjelma
 - parametrit ja paluuarvo(t)

31/10/2001

Copyright Teemu Kerola 2001

2

Parametrit ja paluuarvo ⁽²⁾

- Muodolliset parametrit
 - määritelty aliohjelmassa **ohjelmointihetkellä**
 - tietty järjestys ja tyyppi
 - paluuarvot
 - käsittely hyvin samalla tavalla kuin parametreillekin
- Todelliset parametrit ja paluuarvo
 - tod. parametrit sijoitetaan muodollisten parametrien paikalle kutsuhetkellä suoritusaikana
 - paluuarvo saadaan paluuhetkellä ja sitä käytetään kuten mitä tahansa arvoa

```
Tulosta (int x, y)
Laske(int x): int
```

```
Tulosta (5, apu);
x = Laske( y+234);
```

31/10/2001

Copyright Teemu Kerola 2001

3

Parametryypit ⁽³⁾

- Arvoparametri
 - välitetään parametrin arvo kutsuhetkellä
 - arvoa ei voi muuttaa
- Viiteparametri
 - välitetään parametrin osoite
 - arvo voidaan lukea, arvoa voi muuttaa
- Nimiparametri
 - välitetään parametrin nimi
 - nimi (merkkijono) kuvataan arvoksi kutsuhetkellä
 - semantiikka määräytyy vasta kutsuhetkellä

31/10/2001

Copyright Teemu Kerola 2001

4

Arvoparametri ⁽¹⁰⁾

- Välitetään todellisen parametrin arvo
 - muuttuja, vakio, lauseke, pointeri, olioviite
- Aliohjelma ei voi muuttaa mitenkään todellisen parametrina käytettyä muuttujaa
 - muuttujan (esim. y) arvo
 - olioviitteen arvo
 - lausekkeen arvo
 - muuta arvoparametrin arvoa aliohjelmassa
⇒ muutetaan todellisen parametrin arvon kopiota!
 - todellisen parametrin ptrX arvoa ei voi muuttaa
 - osoitinmuuttujan osoittamaa arvoa voidaan muuttaa
- Javassa ja C:ssä vain arvoparametreja

```
Tulosta (A+3, B)
```

```
arvon kopio
```

```
Laske (int y, *ptrX);
{
  ...
  y = 5;
  *ptrX = 10
}
```

31/10/2001

Copyright Teemu Kerola 2001

5

Viiteparametri ⁽⁵⁾

- Välitetään todellisen parametrin osoite
 - muuttujan osoite
- Aliohjelma voi muuttaa parametrina annettua muuttujan arvoa
- Pascalin *var* parametri

```
Summaa (54, Sum)
```

```
pointeri
```

```
Summaa (x: int; var cum_sum: int)
vrt. C:ssä arvoparametrina välitetyn osoitinmuuttujan osoittaman arvon (PtrX, ed. kalvo) muuttaminen
{
  ...
  cum_sum = cum_sum + x;
  ...
}
Summaa (6, Kok_lkm)
```

31/10/2001

Copyright Teemu Kerola 2001

6

Nimiparametri ⁽⁶⁾

- Välitetään todellisen parametrin nimi
 - merkkijono!
 - Algol 60
 - yleensä makrot
 - sivuvaikutuksia
 - nimiparametri korvataan todellisella parametrilla joka viittauskohdassa tekstuaalisesti

```
void swap (name int x, y)
{
    int t;
    t := x; x := y; y := t;
}
```

```
swap(i,j)
t := i, i := j; j := t;
```

Ei käsitellä enää jatkossa. **STOP**

entä: swap (n, A[n]) % n ↔ A[n]
 t := n; n := A[n]; A[n] := t;

"väärä" n

31/10/2001 Copyright Teemu Kerola 2001 7

Aliohjelmien toteutuksen osat ⁽⁵⁾

- Paluuosoite
 - kutsukohtaa seuraava käskyn osoite
- Parametrien välitys
- Paluuarvon välitys
- Paikalliset muuttujat
- Rekistereiden allokointi (varaus)
 - kutsuvalla ohjelman osalla voi olla käytössä rekistereitä, joiden arvon halutaan säilyä!
 - pääohjelma, toinen aliohjelma, sama aliohjelma, metodi, ...
 - käytettyjen rekistereiden arvot pitää aluksi tallettaa muistiin ja lopuksi palauttaa ennalleen

31/10/2001 Copyright Teemu Kerola 2001 8

Aktivointitietue ⁽⁷⁾

(activation record, activation frame)

```
int funcA (int x,y);
```

- Aliohjelman toteutusmuoto (ttk-91)
 - funktion paluuarvo (tai kaikki paluuarvot)
 - kaikkien (sisäänmeno- ja ulostulo-) parametrien arvot
 - paluuosoite
 - kutsukohdan aktivointitietue
 - kaikki paikalliset muuttujat ja tietorakenteet
 - aliohjelman ajaksi talletettujen rekistereiden alkuperäiset arvot

31/10/2001 Copyright Teemu Kerola 2001

Aktivointitietueiden hallinta ⁽⁴⁾

- Aktivointitietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
 - SP (=R6) osoittaa pinon pinnalle
- Aktivointitietuepino
 - FP (R7) osoittaa voimassa olevan AT:n sovitettuun kohtaan (ttk-91: vanhan FP:n osoite)
- Pinossa olevaa AT:tä rakennetaan ja puretaan käskyillä:
 - PUSH, POP, PUSHR, POPR
 - CALL, EXIT

Talleta R0-R5 pinoon

kasvava muistiosoitte

31/10/2001 Copyright Teemu Kerola 2001 10

Aliohjelman käytön toteutus ⁽¹²⁾

- Toteutus jaettu eri yksiköille
 - Kutsuva rutiini**
 - varaava tilaa paluuarvolle pinosta
 - laita parametrit (arvot tai osoitteet) pinoon
 - talleta vanha PC ja FP, aseta uudet PC ja FP
 - CALL käsky**
 - varaava tilaa paikallisille muuttujille
 - talleta käytettävien rekistereiden vanhat arvot pinoon
 - Kutsuttu rutiini**
 - (itse aliohjelman toteutus)
 - palauta rekistereiden arvot
 - vapautaa paikallisten muuttujien tila
 - EXIT käsky**
 - palauta PC ja FP
 - vapautaa parametrien tila
 - Kutsuva rutiini**
 - ota paluuarvo pinosta

prolog

epilog

31/10/2001 Copyright Teemu Kerola 2001 11

Aliohjelmaesimerkki ⁽¹³⁾

käyttö:

```
int fB (int x, y)
{
    int z = 5;
    z = x * z + y;
    return (z);
}
```

...
 T = fB (200, R);

tämän-hetkinen, nykyinen FP

R DC 24
 ...
 PUSH SP, =0; tila paluuarvolle
 PUSH SP, =200 ← muistista muistiin!!
 PUSH SP, R
 CALL SP, fA
 POP SP, R1 ← 2. operandi aina rekisteri
 STORE R1, T
 ...

talleta PC, FP aseta PC, kutsu & paluu palauta FP, PC

31/10/2001 Copyright Teemu Kerola 2001 12

Aliohjelmaesimerkki (ei anim)

```

int fB (int x, y)
{
    int z = 5;
    z = x * z + y;
    return (z);
}
...
T = fB (200, R);
    
```

käyttö:

```

R      DC 24
...
PUSH SP,=0 ; ret. value space
PUSH SP,=200 ← muistista muistiin!!
PUSH SP, R ←
CALL SP, fA ← talleta PC, FP
                aseta PC,
                kutsu & paluu
                palautaa FP, PC
POP  SP, R1 ← 2. operandi
                aina rekisteri
STORE R1, T
    
```

tämänhetkinen, nykyinen FP

...
paluuarvo
par x=200
par y=24

31/10/2001 Copyright Teemu Kerola 2001 13

Aliohjelmaesimerkki (11)

```

int fA (int x, y)
{
    int z = 5;
    z = x * z + y;
    return (z);
}
...
T = fA (200, R);
    
```

aliohjelman toteutus:

```

retfA EQU -4 # params
parX  EQU -3
parY  EQU -2
locZ  EQU 1 # local vars
fA    PUSH SP,=0 ; alloc Z
      PUSH SP, R1 ; save R1
      LOAD R1,=5; init Z
      STORE R1, locZ (FP)
      LOAD R1, parX (FP)
      MUL  R1, locZ (FP)
      ADD  R1, parY (FP)
      STORE R1, locZ (FP)
      STORE R1, retfA (FP)
      POP  SP, R1; recover R1
      SUB  SP,=1 ; free Z
      EXIT SP,=2 ; 2 param.
    
```

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo

prolog

epilog

31/10/2001 Copyright Teemu Kerola 2001 14

Aliohjelmaesimerkki (ei anim)

```

int fA (int x, y)
{
    int z = 5;
    z = x * z + y;
    return (z);
}
...
T = fA (200, R);
    
```

aliohjelman toteutus:

```

retfA EQU -4
parX  EQU -3
parY  EQU -2
locZ  EQU 1
fA    PUSH SP,=0 ; alloc Z
      PUSH SP, R1 ; save R1
      LOAD R1,=5; init Z
      STORE R1, locZ (FP)
      LOAD R1, parX (FP)
      MUL  R1, locZ (FP)
      ADD  R1, parY (FP)
      STORE R1, locZ (FP)
      STORE R1, retfA (FP)
      POP  SP, R1; recover R1
      SUB  SP,=1 ; free Z
      EXIT SP,=2 ; 2 param.
    
```

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo
param x
param y
vanha PC
vanha FP
paik z
vanha R1

FP

SP

prolog

epilog

31/10/2001 Copyright Teemu Kerola 2001 15

Viiteparametri esimerkki (2)

(Pascal)

```

procB (x, y: int, var pZ:int)
{
    pZ = x * 5 + y;
    return;
}
...
procB (200, R, T);
    
```

käyttö:

```

...
PUSH SP,=200
PUSH SP, R
PUSH SP,=T ; T's address!
CALL SP, procB
; T has new value
    
```

Ei välitetä arvoa T, vaan T:n osoite.
Ainoa tapa monisanaiselle parametrille (taulukko, tietue) tai ulostuloparametreille

Ero C-kieleen: *pZ = x * 5 + y; ?????

31/10/2001 Copyright Teemu Kerola 2001 16

Viiteparam. (jatk) (1)

```

procB (x, y: int, var pZ:int)
{
    pZ = x * 5 + y;
    return;
}
...
procB (200, R, T);
    
```

aliohjelman toteutus:

```

parX EQU -4 ; relative to FP
parY EQU -3
parpZ EQU -2
procB PUSH SP, R1 ; save R1
      LOAD R1, parX (FP)
      MUL  R1, =5
      ADD  R1, parY (FP)
      STORE R1, @parpZ (FP)
      POP  SP, R1; restore R1
      EXIT SP,=3 ; 3 param.
    
```

param x
param y
vparam pZ
vanha PC
vanha FP
vanha R1

FP

SP

prolog

epilog

ks. procB.k91

31/10/2001 Copyright Teemu Kerola 2001 17

Aliohjelma kutsuu funktiota (1)

```

procC (x, y: int, var pZ:int)
{
    pZ = fA(x,y);
    return;
}
...
procC (200, R, T);
    
```

itse aliohjelman käyttö kuten ennen:

```

...
PUSH SP,=200
PUSH SP, R
PUSH SP,=T ; T's address
CALL SP, procC
; T has new value
    
```

31/10/2001 Copyright Teemu Kerola 2001 18

Aliohjelma kutsuu funktiota (2)

```

procC (x, y: int, var pZ:int)
{
    pZ = fA(x,y);
    return;
}
...
procC (200, R, T);
    
```

aliohjelman toteutus:

```

parXc EQU -4 ; relative to FP
parYc EQU -3
parpZ EQU -2      ks. procC.k91

procC PUSH SP, R1 ; save R1
      ; call fA(parXc, parYc)
      PUSH SP,=0 ; ret. value
      PUSH SP, parXc(FP)
      PUSH SP, parYc(FP)
      CALL SP, fA
      POP SP, R1
      STORE R1, @parpZ (FP)

      POP SP, R1; restore R1
      EXIT SP,=3 ; 3 param.
    
```

AT kuten ennen:

param x
param y
vparam pZ
vanha PC
vanha FP
vanha R1

FP →
SP →

31/10/2001 Copyright Teemu Kerola 2001 19

KJ-palvelun kutsu (7)

- Samalla tavalla kuin aliohjelman kutsu
 - CALL käskyn asemesta SVC
- Tila paluuarvolle?
- Parametrit pinnoon
- SVC kutsu
- IRET paluu
- Paluuarvo (OK, virhe) pois pinosta tarkistusta varten

```

fOK = ReadBlock (fp, 64)
...
PUSH SP, =0 ;paluuarvo
PUSH SP, =FileBuffer
PUSH SP, CharCnt
PUSH SP, FilePtr

SVC SP, =ReadFile

POP SP, R1
JNZER R1, =FileTrouble
...
    
```

31/10/2001 Copyright Teemu Kerola 2001 20

Rekursiivinen aliohjelma (5)

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla
- Rekursio ei onnistu, jos paikallisten muuttujien tilanvaraus aliohjelman ohjelmakoodin yhteydessä
 - jotkut Fortran versiot
- Joka kutsukerralla suoritetaan sama koodi-alue (aliohjelman koodi), mutta dataa varten on käytössä oma aktivointitietue

31/10/2001 Copyright Teemu Kerola 2001 21

Rekursio esimerkki (1)

```

fPow (n: int)
{
    if (n=1)
        return (1);
    else
        return (n * fPow (n-1));
}
...
k = fPow (4);
    
```

kutsu:

```

K DC 0
; k = fPow (4)
PUSH SP, =0
PUSH SP, =4
CALL SP, fPow
POP SP, R1
STORE R1, K
    
```

31/10/2001 Copyright Teemu Kerola 2001 22

Rekursiivisen toteutus (2)

```

parRet EQU -3      ks. fPow.k91
parN EQU -2

fPow PUSH SP, R1 ; save R1

LOAD R1, parN(FP)
COMP R1,=1
JEQU One ; return 1 ?

; return fPow(N-1) * N
SUB R1, =1 ; R1 = N-1
PUSH SP, =0 ; ret. value space
PUSH SP, R1
CALL SP, fPow
POP SP, R1 ; R1 = fPow(N-1)

MUL R1, parN(FP)
STORE R1, parRet(FP)

One POP SP, R1; restore R1
EXIT SP,=1 ; 1 param.
    
```

```

fPow (n: int)
{
    if (n=1)
        return (1);
    else
        return (n * fPow (n-1));
}
...
k = fPow (4);
    
```

paluuarvo
param n
vanha PC
vanha FP
vanha R1

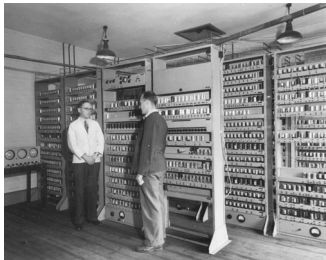
FP →
SP →

31/10/2001 Copyright Teemu Kerola 2001 23

-- Luennon 4 loppu --

M.Wilkes:
EDSAC I (1949)

- rekisterit (6W), tyhjiöputkilla
- käsky- ja datamuisti, 32 elohopea -viiveputkea, 512W à 36b
- kertolasku 5.4ms, 650 IPS
- ensimmäinen ”stored program” –tietokone
- 3000 tyhjiöputkea, sähkökulutus 12 kW, tila 5x4m



http://www.cl.cam.ac.uk/Relics/archive_photos.html

31/10/2001 Copyright Teemu Kerola 2001 24