

## Luento 4 Aliohjelmien toteutus

Tyypit  
Parametrit  
Aktivointitietue (AT)  
AT-pino  
Rekursio

25/03/2004

Copyright Teemu Kerola 2003

1

## Aliohjelmatyypit <sup>(2)</sup>

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

25/03/2004

Copyright Teemu Kerola 2003

2

## Parametrit ja paluuarvo <sup>(2)</sup>

- Muodolliset parametrit
  - määritelty aliohjelmassa ohjelmointihetkellä
  - tietty järjestys ja tyyppi
  - paluuarvot
    - käsittely hyvin samalla tavalla kuin parametreillekin
- Todelliset parametrit ja paluuarvo
  - todelliset parametrit sijoitetaan muodollisten parametrien paikalle kutsuhetkellä suoritusajan
  - 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);
```

25/03/2004

Copyright Teemu Kerola 2003

3

## Parametryypit <sup>(4)</sup>

- Arvoparametri
  - välitetään parametrin arvo (eli sen kopio) kutsuhetkellä
  - arvo voidaan lukea
  - alkuperäistä arvoa ei voi muuttaa, mutta arvon kopiota voi muuttaa
- Viiteparametri
  - välitetään parametrin osoite
  - arvo ja osoite voidaan lukea, arvoa voi muuttaa
- Nimiparametri
  - välitetään parametrin nimi
  - nimi (merkkijono) kuvataan arvoksi kutsuhetkellä
  - semantiikka määräytyy vasta kutsuhetkellä

```
Swap(i, k); Swap(i, T[i]);
```

25/03/2004

Copyright Teemu Kerola 2003

4

## Arvoparametri (10)

Tulosta (A+3, B)

- Välitetään todellisen parametrin arvo
  - muuttuja, vakio, lauseke, pointteri, 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 (osoitinmuuttuja ptrX on siis välitetty arvoparametriina)
- Javassa ja C:ssä vain arvoparametreja

arvon  
kopio

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

25/03/2004

Copyright Teemu Kerola 2003

5

## Viiteparametri (5)

Summaa (54, Sum)

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

pointteri

vt. C:ssä arvoparametrina välitetyn osoitinmuuttujan osoittaman arvon (PtrX, ed. kalvo) muuttaminen

```
Summaa (x: int; var cum_sum: int)
{
  ...
  cum_sum = cum_sum + x;
  ...
}
```

Summaa (6, Kok\_lkm)

25/03/2004

Copyright Teemu Kerola 2003

6

## Nimiparametri <sup>(6)</sup>

- 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.



```
entä: swap (n, A[n]) % n ↔ A[n]
```

```
t := n; n := A[n]; A[n] := t;
```

”väärä” n

25/03/2004

Copyright Teemu Kerola 2003

7

## Aliohjelmien toteutuksen osat <sup>(5)</sup>

- Paluusoite
  - 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

25/03/2004

Copyright Teemu Kerola 2003

8

## Aktivoititietue <sup>(7)</sup>

(activation record,  
activation frame)

`int funcA (int x,y);`

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

25/03/2004 Copyright Teemu Kerola 2003

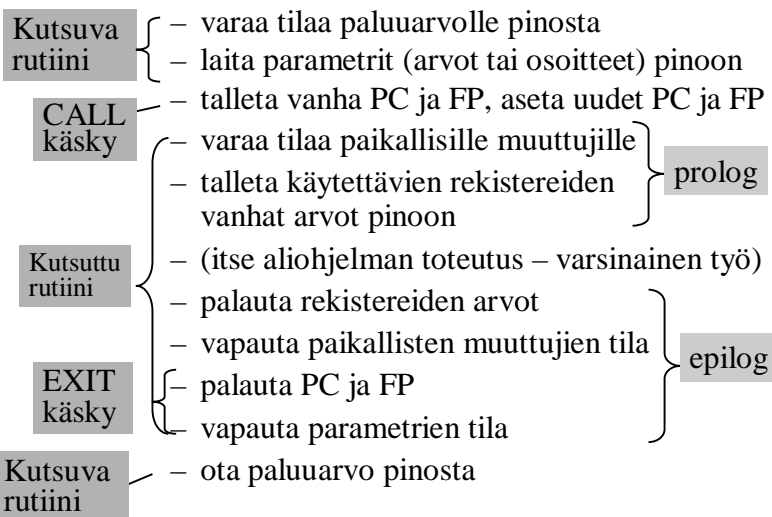
## Aktivoititietueiden hallinta <sup>(4)</sup>

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

25/03/2004 Copyright Teemu Kerola 2003

## Aliohjelman käytön toteutus (12)

- Toteutus jaettu eri yksiköille



25/03/2004

Copyright Teemu Kerola 2003

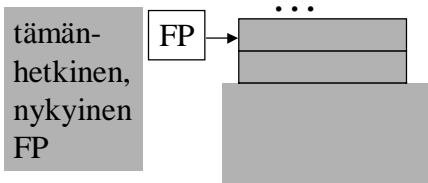
11

## Aliohjelmaesimerkki (13)

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

### käyttö:

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



25/03/2004

Copyright Teemu Kerola 2003

12

## Aliohjelmaesimerkki (ei anim)

```

int fA (int x, y)
{
    int z = 5;

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

tämänhetkinen, nykyinen FP

...
paluarvo
par x=200
par y=24

**käyttö:**

```

R    DC 24
...
PUSH SP,=0 ; ret. value space
PUSH SP,=200
PUSH SP, R
CALL SP, fA
POP  SP, R1
STORE R1, T
...
        
```

muistista muistiin!!

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

2. operandi  
aina rekisteri

25/03/2004
Copyright Teemu Kerola 2003
13

## Aliohjelmaesimerkki (11)

```

int fA (int x, y)
{
    int z = 5;

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

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluarvo
----------

**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.
        
```

prolog

epilog

25/03/2004
Copyright Teemu Kerola 2003
14

## Aliohjelma- esimerkki (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

paluarvo
param x
param y
vanha PC
vanha FP
paik. z
vanha R1

FP → vanha FP  
SP → paik. z

prolog

epilog

25/03/2004
Copyright Teemu Kerola 2003
15

## Viiteparametri esimerkki <sup>(2)</sup>

(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; /\* vain arvoparametrejä \*/

25/03/2004
Copyright Teemu Kerola 2003
16



## Viiteparam. (jatk) <sup>(1)</sup>

```

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

param x

param y

**vparam pZ**

vanha PC

vanha FP

vanha R1

FP

→

vanha FP

SP

→

vanha R1

aliohjelman toteutus:

```

parX EQU -4 ; relative to FP
parY EQU -3
parpZ EQU -2 ; call-by-ref

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.
        
```

prolog

ks. procB.k91

epilog

25/03/2004
Copyright Teemu Kerola 2003
17

## Aliohjelma kutsuu aliohjelman <sup>(1)</sup>

```

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

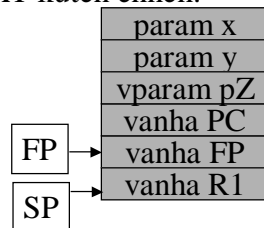
; T has new value
...
        
```

25/03/2004
Copyright Teemu Kerola 2003
18

## Aliohjelma kutsuu funktiota (2)

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

AT kuten ennen:



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.
```

25/03/2004

Copyright Teemu Kerola 2003

19

## Rekursiivinen aliohjelma (5)

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla (uusi AT 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

25/03/2004

Copyright Teemu Kerola 2003

20

## Rekursio esimerkki (1)

```
int fPow (n: int)
{
  if (n=1)
    return (1);
  else
    return (n * fPower (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
```

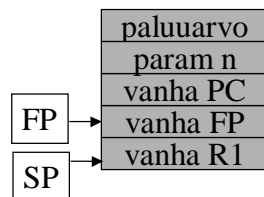
25/03/2004

Copyright Teemu Kerola 2003

21

## Rekursion toteutus (2)

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



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

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)

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

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

25/03/2004

Copyright Teemu Kerola 2003

22

## KJ-palvelun kutsu (proseduraalisesti) <sup>(7)</sup>

- Samalla tavalla kuin aliohjelman kutsu
  - CALL käskyn asemesta SVC
- Tila paluuarvolle?
- Parametrit pinoon
- 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
...

```

25/03/2004

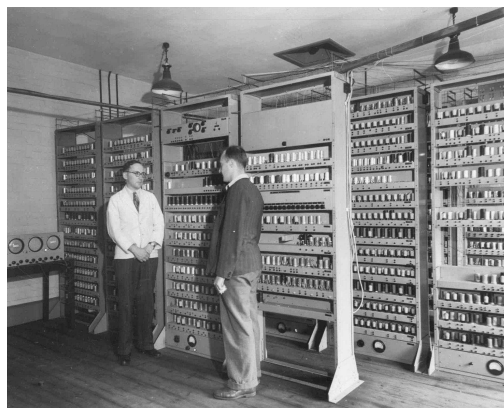
Copyright Teemu Kerola 2003

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

25/03/2004

Copyright Teemu Kerola 2003

24