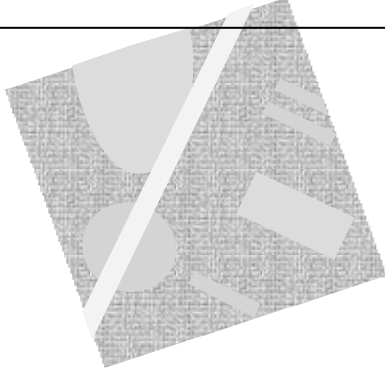


Jakso 4

Aliohjelmien toteutus



- Tyypit
- Parametrit
- Aktivointitietue (AT)
- AT-pino
- Rekursio

11.8.2000Teemu Kerola, K20001

Aliohjelmatyypit ⁽²⁾

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

11.8.2000Teemu Kerola, K20002

Parametrit ja paluuarvo (2)

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

11.8.2000

Teemu Kerola, K2000

3

Arvoparametri (10)

- Välitetään todellisen parametrin arvo
 - muuttuja, vakio, lauseke, pointeri, olioviite
- Aliohjelma ei voi muuttaa mitenkään todellisen parametrina käytettyä muuttujaa
 - muuttujan X tai B arvo
 - olioviitteen arvo
 - lausekkeen arvo
 - muuta muodollisen parametrin arvoa aliohjelmassa ⇒ muutetaan todellisen parametrin arvon kopiota!
 - osoitinmuuttuja parametrin ptrX arvoa ei voi muuttaa
 - osoitinmuuttujan osoittamaa arvoa voidaan muuttaa
- Javassa ja C:ssä vain arvoparametreja

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

```
arvon
kopio
```

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

11.8.2000

Teemu Kerola, K2000

4

Viiteparametri ⁽⁴⁾

Summaa (54, Sum)

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

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

11.8.2000
Teemu Kerola, K2000
5

Nimiparametri ⁽⁴⁾

- Välitetään todellisen parametrin nimi
 - merkkijono!
 - Algol 60
 - C:n makrot
 - sivuvaikutuksia
 - nimiparametri korvataan todellisella parametrilla joka viittauskohdassa

Ei käsitellä enää jatkossa.

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

swap (n, A[n]) % n ↔ A[n]

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

väärä n

11.8.2000
Teemu Kerola, K2000
6

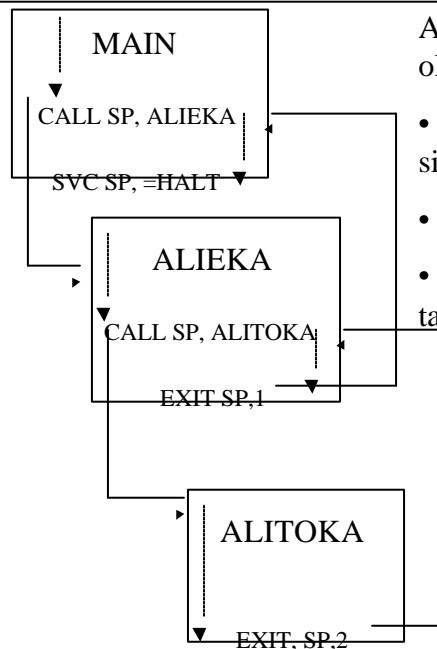
Aliohjelmien toteutuksen osat ⁽⁵⁾

- 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

11.8.2000

Teemu Kerola, K2000

7



Aliohjelmat voivat olla sisäkkäisiä

- aliohjelmaan siirtyminen
- aliohjelmasta paluu
- aliohjelman tarvitsemat tiedot



Aliohjelman ympäristö eli aktivointitietue

11.8.2000

Liisa Marttinen, Kesä 2000

8

Aktivointitietue (6)

int funcA (int x,y);

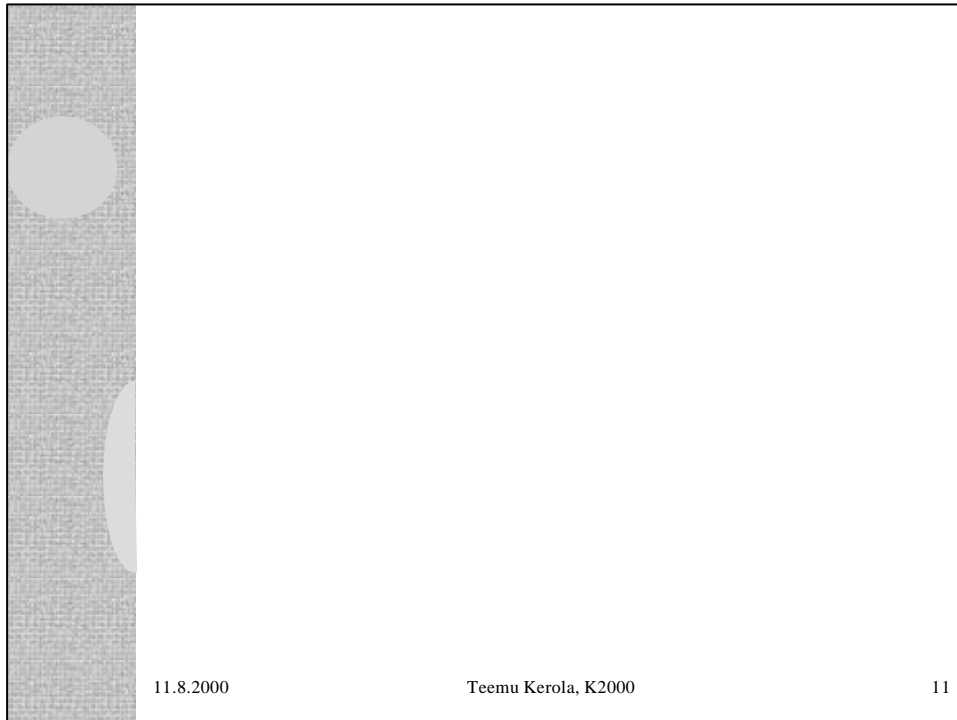
- Aliohjelman toteutusmuoto (ttk-91)
 - kaikkien ulostuloparametrien arvot
 - kaikkien (sisäänmeno) parametrien arvot
 - paluuosoite
 - kutsukohdan aktivointitietue
 - aliohjelman ajaksi talletettujen rekistereiden arvot
 - kaikki paikalliset muuttujat ja tietorakenteet

11.8.2000
Teemu Kerola, K2000
9

Aktivointitietueiden hallinta (4)

- Aktivointitietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta
 - SP (=R6) osoittaa pinon pinnalle
- Aktivointitietuepino
 - 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

11.8.2000
Teemu Kerola, K2000
10



Aliohjelman käytön toteutus ⁽¹²⁾

- Toteutus jaettu eri yksiköille

Kutsuva rutiini	<ul style="list-style-type: none"> - varaa tilaa paluuarvolle pinosta - laita parametrit (arvot tai osoitteet) pinoon 	
CALL käsky	<ul style="list-style-type: none"> - talleta PC ja FP 	
Kutsuttu rutiini	<ul style="list-style-type: none"> - talleta käytettyjen rekistereiden arvot pinoon - varaa tilaa paikallisille muuttujille - (itse aliohjelman toteutus) - vapauta paikallisten muuttujien tila 	} prolog
EXIT käsky	<ul style="list-style-type: none"> - palauta rekistereiden arvot - palauta PC ja FP 	} epilog
Kutsuva rutiini	<ul style="list-style-type: none"> - vapauta parametrien tila - ota paluuarvo pinosta 	

11.8.2000
Teemu Kerola, K2000
12

Aliohjelmaesimerkki (13)

```

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
PUSH SP, R
        
```

muistista muistiin!!

```

CALL SP, fA
        
```

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

```

POP  SP, R1
STORE R1, T
        
```

2. operandi
aina rekisteri

tämänhetkinen, nykyinen FP

11.8.2000
Teemu Kerola, K2000
13

Aliohjelmaesimerkki (ei animm)

```

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
PUSH SP, R
        
```

muistista muistiin!!

```

CALL SP, fA
        
```

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

```

POP  SP, R1
STORE R1, T
        
```

2. operandi
aina rekisteri

tämänhetkinen, nykyinen FP

11.8.2000
Teemu Kerola, K2000
14

Aliohjelma-esimerkki (11)

```

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

aliohjelman toteutus:

retfA	EQU -4	ks. fA.k91
parX	EQU -3	
parY	EQU -2	
locZ	EQU 2	

```

fA  PUSH SP, R1 ; save R1
    PUSH SP, =0 ; alloc Z

    LOAD R1, =5; init Z
    STORE R1, locZ (FP)

    LOAD R1, parX (FP)
    MUL  R1, locZ (FP)
    ADD  R1, parY (FP)

    STORE R1, retfA (FP)
    SUB  SP, =1 ; free Z
    POP  SP, R1; recover R1
    EXIT SP, =2 ; 2 param.
        
```

Kaikki viitteet näihin tehdään suhteessa FP:hen

paluuarvo

11.8.2000
Teemu Kerola, K2000
15

Aliohjelma-esimerkki (ei)

```

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

aliohjelman toteutus:

retfA	EQU -4	ks. fA.k91
parX	EQU -3	
parY	EQU -2	
locZ	EQU 2	

```

fA  PUSH SP, R1 ; save R1
    PUSH SP, =0 ; alloc Z

    LOAD R1, =5; init Z
    STORE R1, locZ (FP)

    LOAD R1, parX (FP)
    MUL  R1, locZ (FP)
    ADD  R1, parY (FP)

    STORE R1, retfA (FP)
    SUB  SP, =1 ; free Z
    POP  SP, R1; recover R1
    EXIT SP, =2 ; 2 param.
        
```

Kaikki viitteet näihin tehdään suhteessa FP:hen

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

11.8.2000
Teemu Kerola, K2000
16

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 taulukkoa T (ja sen kaikkia alkioita), vaan ainoastaan T:n osoite (yksi arvo)

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

11.8.2000
Teemu Kerola, K2000
17

Viiteparam. (jatk) (1)

```

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

```

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.
        
```

ks. procB.k91

11.8.2000
Teemu Kerola, K2000
18

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
...
        
```

11.8.2000
Teemu Kerola, K2000
19

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:

11.8.2000
Teemu Kerola, K2000
20

Rekursiivinen aliohjelma ⁽⁴⁾

- Aliohjelma, joka kutsuu itseään
- Ei mitään erikoista muuten
- Aktivointitietue hoitaa tilanvarauksen automaattisesti paikallisille muuttujille joka kutsukerralla
- Joka kutsukerralla suoritetaan sama koodi-alue (aliohjelman koodi), mutta dataa varten on käytössä oma aktivointitietue

11.8.2000

Teemu Kerola, K2000

21

Rekursio esimerkki ⁽¹⁾

```
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
```

11.8.2000

Teemu Kerola, K2000

22

Rekursion toteutus (2)

```

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

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)	
One	MUL R1, parN(FP)	
	STORE R1, parRet(FP)	
	POP SP, R1; restore R1	
	EXIT SP, =1 ; 1 param.	

FP	→	naluuarvo
		param n
		vanha PC
		vanha FP
SP	→	vanha R1

11.8.2000
Teemu Kerola, K2000
23

-- Jakson 4 loppu --

The diagram illustrates the flow of control between two procedures. On the left is the 'Calling procedure' and on the right is the 'Called procedure'.

- The 'Calling procedure' starts with 'A called from main program'.
- It makes a 'CALL' to the 'Called procedure'.
- Inside the 'Called procedure', there is a 'CALL' to another sub-procedure.
- That sub-procedure returns ('RETURN') to the 'Called procedure'.
- The 'Called procedure' then returns ('RETURN') to the 'Calling procedure'.
- Finally, the 'Calling procedure' returns ('RETURN') to 'A returns to main program'.

Figure 5-42. When a procedure is called, execution of the procedure always begins at the first statement of the procedure.

[Tane99]

11.8.2000
Teemu Kerola, K2000
24