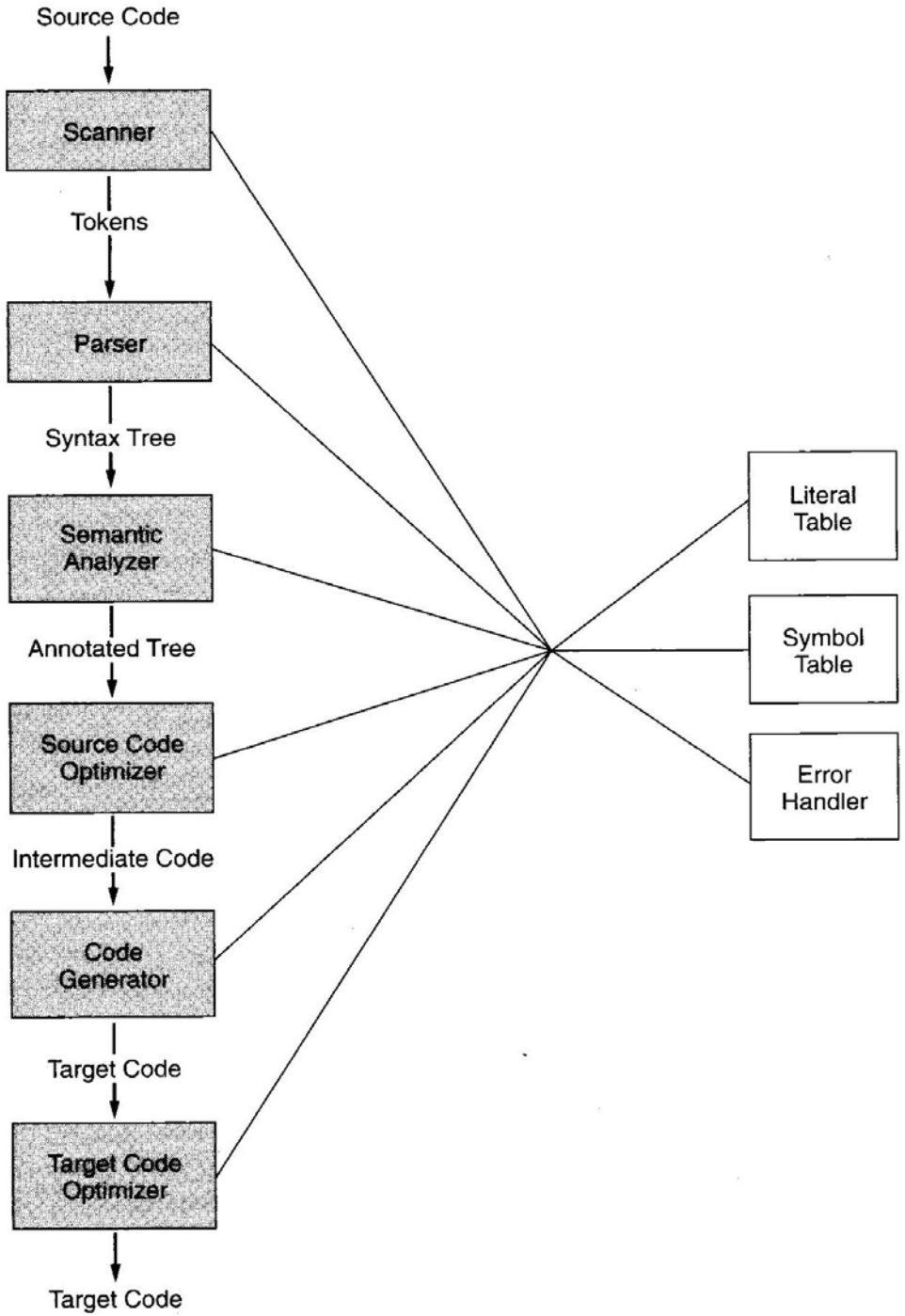
Runtime-systemer del II

Parameteroverføring

Kapittel 7.5



INF 5110 - 2013

Oversikt

- Call by value
- Call by reference
- Call by value-result
- Call by name

'by value' parameteroverføring (verdioverføring)

```
void inc2( int x)
/* incorrect! */
{ ++x;++x; }
```

```
void inc2( int* x)
/* now ok */
{ ++(*x);++(*x); }
```

Kall: inc2(&y)

```
void init(int x[],int size)
/* this works fine when called
   as init(a), where a is an array */
{ int i;
  for(i=0;i<size;++i) x[i]=0;
}
```

Kall: init(a)

- Hver formell parameter blir implementert som en lokal variabel i prosedyren
- Ved kall blir disse variable initialisert slik:
  formell_var = aktuelt uttrykk
- I noen språk kan den formelle variablen ikke forandres
- I C er dette eneste overføringsmåte. Man kan dog overføre pekere 'by value'

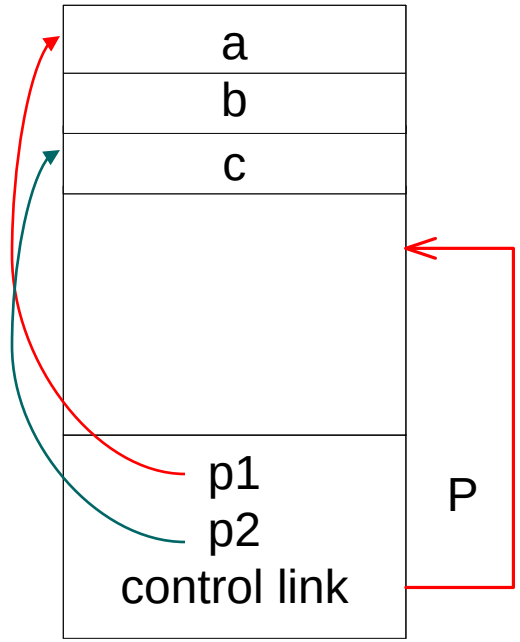
’by reference’ parameteroverføring

- Overfører en peker/adresse til den aktuelle variabel
- Bare lov med variable som aktuelle parametere
- Fortran tillater imidlertid
  - P(5,b)
  - P(a+b,c)
- Passer til store datastrukturer

```
void P(p1,p2) {
  ...
  p1 = 3
}
var a,b,c;
P(a,c)
```

```
void inc2( int & x)
/* C++ reference parameter */
{ ++x;++x; }
```

Kall: inc2(y)



INF 5110 - 2013

’by value-result’ parameteroverføring

- Bare variable kan være aktuelle parametere
- Det allokeres en lokal variabel, som ved ’by value’
- Ved kallet gjøres
  formell_var = aktuell_var
- Ved retur utføres
  aktuell_var = formell_var
- Kan gi effekt forskjellig fra ’by reference’ (Men f.eks. Ada godkjenner også dette som en implementasjon)

```
void p(int x, int y)
{  ++x;
   ++y;
}

 main()
 {  int a = 1;
    p(a,a);
    return 0;
 }
```

Eksempel på at det kan være forskjell på ’by reference’ og ’by value-result’

’by name’ parameteroverføring

- Den aktuelle parameteren blir substituert inn for den formelle (’nesten’ rent tekstlig: den aktuelle parameteren beholder sitt skop, så altså ikke makro-ekspansjon)
- Om den aktuelle parameteren er et uttrykk blir det ikke beregnet før man bruker parameteren inne i prosedyren (lazy evaluering)
- Men: Uttrykket blir beregnet om igjen hver gang
- Implementasjon
  - Se den aktuelle parameteren (f.eks. et uttrykk) som en liten prosedyre (’thunk’)
  - Må optimaliseres for det tilfellet hvor parameteren er en enkel variabel (da er effekten som ved ’by reference)

```
void p(int x)
{ ++x; }
```

p(a[i]) ++a[i]

```
int i;
int a[10];

void p(int x)
{ ++i;
  ++x;
}

main()
( i = 1;
  a[1] = 1;
  a[2] = 2;
  p(a[i]);
  return 0;
}
```

’by name’ eksempel

```
procedure P(par); name par; int par;
begin
  int x,y;
  ...
  par := x + y;        a)
  ...
  x := par + y;        b)
  ...
end;
...
P(v);
P(r.v);
P(5);
P(u+v);
```

	v	r.v	5	u + v
a)	OK	OK	feil	feil
b)	OK	OK	OK	OK

OPPGAVE 7.15 OG 7.16

7.15 Give the output of the following program (written in C syntax) using the four parameter passing methods discussed in Section 7.5:

```
#include <stdio.h>
int i=0;

void p(int x, int y)
{ x += 1;
  i += 1;
  y += 1;
}

main()
{ int a[2]={1,1};
  p(a[i],a[i]);
  printf("%d %d\n",a[0],a[1]);
  return 0;
}
```

by value	by reference
? ?	? ?
by value-result	**by name**
? ?	? ?

7.16 Give the output of the following program (in C syntax) using the four parameter passing methods of Section 7.5:

```
#include <stdio.h>
int i=0;

void swap(int x, int y)
{ x = x + y;
  y = x - y;
  x = x - y;
}

main()
{ int a[3] = {1,2,0};
   swap(i,a[i]);
   printf("%d %d %d %d\n",i,a[0],a[1],a[2]);
   return 0;
}
```

by value	by reference
? ? ? ?	? ? ? ?
by value-result	**by name**
? ? ? ?	? ? ? ?

OPPGAVE 1 EKSAMEN 2006

Neste forelesning: Fredag 12. april (Sed)

Runtimesystemer del III