

Ukeoppgaver i INF3110/4110

Uke 45 (5.-7.11.2003)

Oppgave 1

Løsningsforslag ligger her: (<http://www.ifi.uio.no/studinf/eksamen/eksoppg/in211/>)

Oppgave 2

Løsningsforslag ligger her: (<http://www.ifi.uio.no/studinf/eksamen/eksoppg/in211/>)

Oppgave 3

Det greieste er vel å se på hvordan programmet (i store trekk) må se ut:

```
PROCEDURE A;  
BEGIN  
  PROCEDURE C;  
  BEGIN  
    ...  
  END;  
  B;  
END;  
  
PROCEDURE B;  
BEGIN  
  C;  
END;  
  
A;
```

Her kan altså ikke prosedyren B kalle C. Det kan derfor umiddelbart virke som om en slik run-time stakk som gitt i oppgaven ikke er mulig. Hvis vi imidlertid tillater prosedyrer som parametre vil en slik run-time stakk være fullt mulig:

```

PROCEDURE A;
BEGIN
  PROCEDURE C; ...;

  B(C);
END;

PROCEDURE B(Q); PROCEDURE Q;
BEGIN
  Q;    !** Her kalles C. ;
END;

A;

```

Oppgave 4

Språket L har ingen metoder, og vi kan derfor benytte statisk minneallokering (som for C1). Siden L-program maksimalt kan benytte ti variable (v0-v9), vil det aller enkleste være å alltid sette av plass til alle variablene — uansett om de blir brukt eller ikke. v0 vil da ligge i D[0], v1 i D[1], osv...

Oppgave 5 Forberedelse til hjemmeksamen

<i><program></i>	→ <i><var decl></i> ? <i><proc decl list></i> ? begin <i><statement list></i> end
<i><var decl></i>	→ var <i><name list></i> ;
<i><proc decl list></i>	→ <i><proc decl></i> [; <i><proc decl></i>]*
<i><proc decl></i>	→ procedure <i><proc name></i> : <i><var decl></i> ? begin <i><statement list></i> end
<i><name list></i>	→ <i><name></i> [, <i><name></i>]*
<i><statement list></i>	→ <i><statement></i> [; <i><statement></i>]*
<i><statement></i>	→ <i><proc call></i> <i><assignment></i> <i><input></i> <i><output></i> <i><if-statement></i>
<i><proc call></i>	→ call <i><proc name></i>
<i><proc name></i>	→ <i><name></i>
<i><assignment></i>	→ assign <i><expression></i> > <i><variable></i>
<i><variable></i>	→ <i><name></i>
<i><expression></i>	→ <i><term></i> [<i><ar-op></i> <i><term></i>]*
<i><ar-op></i>	→ + - * /
<i><term></i>	→ <i><number></i> <i><variable></i>
<i><input></i>	→ ? <i><variable></i>
<i><output></i>	→ ! <i><variable></i>
<i><if-statement></i>	→ if <i><expression></i> then <i><statement list></i> ? fi

```

1 % Reads two numbers and returns the maximum.
2
3 var a,b;
4 begin
5   ?a;
6   ?b;
7   if a - b then !a fi;
8   if b - a then !b fi
9 end

```

Figur 1: maks2.l2

```

1 % Reads three numbers and returns the maximum.
2
3 var a,b,c,m;
4 begin
5   ?a;
6   ?b;
7   ?c;
8
9   if a - b then          % if a > b
10    if a - c then !a fi; % if a > c
11    if c - a then !c fi % if c > a
12    fi;
13
14   if b - a then          % if b > a
15    if b - c then !b fi; % if b > c
16    if c - b then !c fi % if c > b
17    fi
18 end

```

Figur 2: maks3.l2

```

1 % Reads a number and prints the absolute value of this number.
2
3 var a, result;
4 begin
5   ?a;
6   if a then !a fi;
7   if ~1*a then
8     assign ~1*a > a;
9     !a fi
10 end

```

Figur 3: abs.l2

```

1 % Reads a number; checks if it is an odd number.
2
3 var a;
4
5 procedure minus2:
6 begin
7   if a - 2 then % if a >= 2
8     assign a-2 > a;
9     call minus2
10    fi
11  end
12
13
14 begin
15 ?a;
16 if a*~1 then assign a*~1 > a fi; % convert to positive
17 call minus2;
18 !a
19 end

```

Figur 4: odd.l2

```

1 % Leser N tall og skriver ut summen.
2
3 var N, Sum;
4 procedure SumN: % Leser N tall og legger summen i Sum.
5   var X;
6   begin
7     ?X;
8     assign Sum+X > Sum;
9     assign N-1 > N;
10    if N-1 then call SumN fi
11  end
12 begin
13 ?N;
14 assign 0 > Sum;
15 if N-1 then call SumN fi;
16 !Sum
17 end

```

Figur 5: sumn.l2