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 hjemmeeksamen

⟨*program*⟩ → ⟨*var decl*⟩$^?$ ⟨*proc decl list*⟩$^?$ **begin** ⟨*statement list*⟩ **end**
⟨*var decl*⟩ → **var** ⟨*name list*⟩ **;**
⟨*proc decl list*⟩ → ⟨*proc decl*⟩ ⟦**;** ⟨*proc decl*⟩⟧*
⟨*proc decl*⟩ → **procedure** ⟨*proc name*⟩ **:**
⟨*var decl*⟩$^?$ **begin** ⟨*statement list*⟩ **end**
⟨*name list*⟩ → ⟨*name*⟩ ⟦**,** ⟨*name*⟩⟧*
⟨*statement list*⟩ → ⟨*statement*⟩ ⟦**;** ⟨*statement*⟩⟧*
⟨*statement*⟩ → ⟨*proc call*⟩ | ⟨*assignment*⟩ | ⟨*input*⟩ | ⟨*output*⟩ |
⟨*if-statement*⟩
⟨*proc call*⟩ → **call** ⟨*proc name*⟩
⟨*proc name*⟩ → ⟨*name*⟩
⟨*assignment*⟩ → **assign** ⟨*expression*⟩ **>** ⟨*variable*⟩
⟨*variable*⟩ → ⟨*name*⟩
⟨*expression*⟩ → ⟨*term*⟩ ⟦⟨*ar-op*⟩ ⟨*term*⟩⟧*
⟨*ar-op*⟩ → **+** | **-** | ***** | **/**
⟨*term*⟩ → ⟨*number*⟩ | ⟨*variable*⟩
⟨*input*⟩ → **?** ⟨*variable*⟩
⟨*output*⟩ → **!** ⟨*variable*⟩
⟨*if-statement*⟩ → **if** ⟨*expression*⟩ **then** ⟨*statement list*⟩$^?$ **fi**

```
% Reads two numbers and returns the maximum.

var a,b;
begin
?a;
?b;
if a - b then !a fi;
if b - a then !b fi
end
```

Figur 1: maks2.l2

```
% Reads three numbers and returns the maximum.

var a,b,c,m;
begin
?a;
?b;
?c;

if a - b then         % if a > b
 if a - c then !a fi; % if a > c
 if c - a then !c fi % if c > a
fi;

if b - a then         % if b > a
 if b - c then !b fi; % if b > c
 if c - b then !c fi % if c > b
fi
end
```

Figur 2: maks3.l2

```
% Reads a number and prints the absolute value of this number.

var a, result;
begin
 ?a;
 if a then !a fi;
 if ~1*a then
  assign ~1*a > a;
  !a fi
end
```

Figur 3: abs.l2

```
% Reads a number; checks if it is an odd number.

var a;

  procedure minus2:
  begin
  if a - 2 then    % if a >= 2
   assign a-2 > a;
   call minus2
  fi
  end


begin
?a;
if a*~1 then  assign a*~1 > a fi;  % convert to positive
call minus2;
!a
end
```

Figur 4: odd.l2

```
% Leser N tall og skriver ut summen.

var N, Sum;
procedure SumN: % Leser N tall og legger summen i Sum.
  var X;
  begin
    ?X;
    assign Sum+X > Sum;
    assign N-1 > N;
    if N-1 then call SumN fi
  end
begin
  ?N;
  assign 0 > Sum;
  if N-1 then call SumN fi;
  !Sum
end
```

Figur 5: sumn.l2