INF5110

Oblig 2 presentasjon

Oversikt

- Informasjon
- Oppgaven
- Semantikksjekk
- Kodegenerering
- Bytecode-biblioteket
- Ant-targets
- Oppsummering

Informasjon

- Oblig 2 tilgjengelig på kurssiden
  - Patch med testfiler og filer for kodegenerering
    - Nye versjoner av Compiler.java og build.xml
    - Ny kode i runtime.* og bytecode.*
- Frist mandag 2 mai
- Orakeltime onsdag 27 april

Oppgaven

- Kontrollere at syntakstreet (programmet) oppfyller alle kravene i språknotatet (statisk semantiske regler)
- Gi en kort feilmelding dersom det er feil
- Kompilatoren skal gi følgende svar:
  - (0) Godkjent
  - (1) Syntaksfeil
  - (2) Semantikkfeil
- Generere bytecode vha bibliotekklasser
- Teste semantikksjekken mot programmene i testsuiten
- Teste bytecode-genereringen mot eksemplet RunMe.comp
- Rapport som beskriver løsningen

Semantikksjekk

- Multiple deklarasjoner av et navn på samme nivå må detekteres
- Typekonvertering
  - int is-a float
- Main-funksjonen
  - Må være deklarert på øverste blokkniva
  - Ingen argumenter eller returverdi
- Funksjonsdeklarasjoners returverdi
  - ikke noe (void)
  - type i symboltabellen (predefinert eller brukerdefinert struct)

Semantikksjekk - Stmts

- AssignStmt
  - Variabelen på venstresiden må være deklarert
  - Typene på venstre side og høyre side må være like, etter evt. typekonvertering
- ReturnStmt
  - Typen til uttrykket skal stemme overens med funksjonens deklarerte type
- WhileStmt
  - Må ha en betingelse av typen bool
- IfStmt
  - Må ha en betingelse av typen bool

Semantikksjekk – Exps

- NewExp
  - Argumentet må være deklarert som en strukt-type
- Literals
  - Må være av en de forhåndsdefinerte typene i språket
    - float, int, string, bool eller null
- Binære uttrykk
  - Lik type, etter evt. typeforfremming, på begge sider av operatoren
- Aritmetiske uttrykk
  - Begrenses til aritmetiske typer (int eller float)
- Logiske uttrykk (med "&&", "||" eller "!")
  - Begge operander må være av typen bool

Semantikksjekk – Exps (2)

- CallStmt
  - Navnet som kalles må være deklarert som funksjon
  - Kallet må ha likt antall aktuelle og formelle parametre
  - De aktuelle parametrene må ha lik type (eller mulig å tilordne) som de formelle parametrene
  - Bruk av referanser ("ref") må stemme overens med funksjondeklarasjonen
- Var
  - Evt. strukt-type må være deklarert
  - Navnet må være deklarert

Kodegenerering

- Benyttebytecode-bibliotekettilålagebyte-kode
  - Beskrevet i notat om bytecode og interpreter (bytecode-interpreter.pdf)
- Det er en begrenset semantikk for kodegenereringen
  - RunMe.comp skal benyttes for å teste kodegenereringen
  - Testprogrammene i katalogen *test* skal ikke benyttes

Kodegenerering (2)

- Forskjeller i semantikken
  - Ikke blokkstruktur
    - Kun et globalt og et lokalt nivå
  - Ikke referanseparametere
  - Litt annen bruk av navn
    - Funksjoner kalles Procedure

Bytecode-biblioteket

```
CodeFile codeFile = new CodeFile();

// Her bygges byte-koden opp ...
byte[] bytecode = codeFile.getBytecode();
DataOutputStream stream =
  new DataOutputStream(
    new FileOutputStream("eks.bin"));
stream.write(bytecode);
stream.close();
```

Bytecode-biblioteket (2)

```
// Definerer først deklarasjonen codeFile.addProcedure
("Main");

// Lager metoden, variablen eller strukten
CodeProcedure main =
  new CodeProcedure("Main", VoidType.TYPE, codeFile);
// Legger til dens egenskaper
main.addInstruction(new RETURN());
// Oppdaterer Code-File-objektet codeFile.updateProcedure
(main);
```

Ant-targets

- test
  - Kjører testene (ca. 40 stk)
- compile-runme
  - Kompilerer eksemplet RunMe.comp
- list-runme
  - Lister innholdet i bytekoden laget for RunMe.comp
- run-runme
  - Kjører RunMe.bin på den virtuelle maskinen

Oppsummering

- Kontrollere statisk semantikk
  - Utvide syntakstre-nodene
  - Lage en context-klasse som håndterer symboltabell etc.

Kjøre testene i katalogen *test*

- Generere byte-kode
  - Utvide syntakstre-nodene
  - Holde orden på stack etc.
  - Kjøre RunMe.comp
- Dokumentere løsningen i en rapport
- Frist: Mandag 2. mai