

Grundlagen Programmiersprachen und Compilerbau

Wintersemester 2005/2006

6. Übungsblatt

16.1.2006

Dieses Übungsblatt wird, voraussichtlich :-), am 25,26,27.01.2006 besprochen.

Aufgabe 6.1

Die folgende Grammatik für Statements ist mehrdeutig (Dangling-Else-Problem):

```
Statement → if Expression then Statement
Statement → if Expression then Statement else Statement
Statement → other
Expression → value
```

- Geben Sie eine Eingabe an, für die die Grammatik zwei verschiedene Parse-Bäume erlaubt.
- Wie kann prinzipiell die Mehrdeutigkeit aufgelöst werden?
- Überlegen Sie sich, wie bei einem LR-Parser die Mehrdeutigkeit aufgelöst werden kann.

Aufgabe 6.2

In Aufgabe 1.2 wurde diskutiert, welche Phasen eines Pascal-Compilers folgende Fehler entdecken:

- `(* ein fehlerhafter Kommentar *)`
- `TYPE Tag = (Mo, Di, Mi, Do, Fr, Sa, So);`
- Der Tippfehler `UNTLI Ende;`
- Die Verwendung einer nicht deklarierten Variable `i`
- ```
VAR x: REAL;
 y: RECORD a, b: INTEGER; c: REAL; END;
...
IF x > y THEN BEGIN ... END;
```
- Unterprogrammaufruf mit einer falschen Anzahl von Parametern
- Ein fehlendes `END`
- `TYPE Farbe = (rot, gelb gruen, blau);`
- `VAR 5x : Real;`

Wie könnte oder sollte ein guter Compiler auf diese Fehler reagieren? Welche der Programmierfehler verursachen Folgefehler? Kann man diese durch eine geeignete Fehlerbehandlung vermeiden?

### Aufgabe 6.3

Die Einhaltung der in Deklarationen gegebenen Zusicherungen über Wertebereiche und der Schutz gegen fehlerhaftes Überschreiben des Speichers kann durch entsprechende Übersetzungs- und Laufzeitprüfungen erreicht werden.

Duplizieren Sie das eingegrenzte Ada-Programmstück („von hier - bis hier“), ggf. mit semantikerhaltenden Umformungen, und fügen Sie alle expliziten Prüfungen ein, die den notwendigen impliziten Laufzeitprüfungen entsprechen, um derartige Sicherheit zu garantieren.

```
Vorgang_Max: constant Integer := Read(from_somewhere);
type Transaktions_Art is (Einzahlung, Auszahlung);
type Waehrung is range 0 .. 10*6; -- ein Bereichstyp
type Konto_Nummer is range 0 .. 10**9;
type Buchungseintrag(Art: Transaktions_Art) is
 record -- ein record mit zwei Varianten;
 -- die Diskriminante "Art" unterscheidet, welche
 -- Variante gerade vorliegt; auf "Art" kann wie auf
 -- Komponenten lesend zugegriffen werden
 Konto: Konto_Nummer; -- in beiden Varianten enthalten
 case Art is --
 when Einzahlung => -- erste Variante
 Gutschrift: Waehrung;
 ...
 when Auszahlung => -- zweite Variante
 Abzug: Waehrung;
 ...
 end case;
 end record;

type Buchung is access Buchungseintrag; -- ein Zeigertyp
type Buchungs_Liste is array(0 .. Vorgang_Max) of Buchung;

procedure Verarbeite
 (Tagesliste: Buchungs_Liste;
 Vorgang : Integer;
 Konto : Konto_Nummer;
 Betrag : Waehrung;
 Kosten : Waehrung) is
begin
 -- beachte: Ada führt implizite Dereferenzierung von Zeigern durch
 ----- von hier....
 Tagesliste(Vorgang).Konto := Konto;
 Tagesliste(Vorgang).Gutschrift := Betrag - Kosten;
 ----- bis hier

end Verarbeite;
```

Die statische Typprüfung sei erfolgreich gewesen; nehmen Sie an, dass alle beteiligten Variablen Werte besitzen, die ihren deklarierten Wertebereichen entsprechen, aber diese Werte sind zur Übersetzungszeit nicht bekannt. Die Prüfungen sollen von der folgenden Form sein:

```
if Verletzungsbedingung then raise Error; end if;
eigentliche Ausführung
```

Geben Sie nun Ihren Programmtext an.

(Klausuraufgabe)

#### Aufgabe 6.4

Gegeben sei das folgende Programm in einer Ada-ähnlichen Sprache:

```
procedure Mysterium
 I : Integer;
 A : array (1..3) of Integer := (1, 2, 3);

 procedure Was (X : Integer) is
 begin
 I := 3;
 declare
 I : Integer := 2;
 begin
 X := 20;
 A[1] := A[1] + I;
 end;
 end Was;
begin
 I := 1;
 Was(A[I]);
 Put(A[1]); Put(A[2]); Put(A[3]);
end Mysterium;
```

Welche Ausgabe liefert dieses Programm, wenn als Parameterübergabemechanismus

- a. call-by-reference
- b. call-by-value
- c. call-by-value/result
- d. call-by-name
- e. call-by-name mit Algol-Regel

benutzt wird?

(Nach einer alten Klausuraufgabe)